Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig

Implementing a Bayes Filter in a Neural Circuit: The Case of Unknown Stimulus Dynamics

by

Sacha Sokoloski

Preprint no.: 30 2017

Implementing a Bayes Filter in a Neural Circuit: The Case of Unknown Stimulus Dynamics

Sacha Sokoloski
Max Planck Institute for Mathematics in the Sciences

Abstract

In order to interact intelligently with objects in the world, animals must first transform neural population responses into estimates of the dynamic, unknown stimuli which caused them. The Bayesian solution to this problem is known as a Bayes filter, which applies Bayes' rule to combine population responses with the predictions of an internal model. The internal model of the Bayes filter is based on the true stimulus dynamics, and in this paper we present a method for training a theoretical neural circuit to approximately implement a Bayes filter when the stimulus dynamics are unknown. To do this we use the inferential properties of linear probabilistic population codes to compute Bayes' rule, and train a neural network to compute approximate predictions by the method of maximum likelihood. In particular, we perform stochastic gradient descent on the negative log-likelihood of the neural network parameters with a novel approximation of the gradient. We demonstrate our methods on a finite-state, a linear, and a nonlinear filtering problem, and show how the hidden layer of the neural network develops tuning curves which are consistent with findings in experimental neuroscience.

1 Introduction

Whether its concerns the location of distant food or the presence of a lurking predator, animals must reason about the world based on uncertain beliefs. The Bayesian brain is the hypothesis that the brain represents these beliefs with probability distributions, and reasons about the world based on the principles of Bayesian inference (Knill and Pouget, 2004; Doya, 2007). The Bayesian brain is supported by both theoretical arguments (Jaynes, 2003) and experimental evidence (Ernst and Banks, 2002; Fischer and Peña, 2011; Fetsch et al.,

2011; Coen-Cagli et al., 2015), yet many open questions remain in how exactly the brain implements Bayesian inference in populations of neurons (Bowers and Davis, 2012; Pouget et al., 2013).

The goal of this work is to understand how neural circuits compute accurate beliefs about dynamic stimuli with Bayesian inference. To do this we first assume that the neural populations in a given circuit encode probability distributions in their activity with probabilistic population codes (Zemel et al., 1998). A key property of probabilistic population codes for Bayesian theories of the brain is that theoretical neural circuits based on probabilistic population codes can trivially implement Bayes' rule (Ma et al., 2006).

Bayes' rule is the most fundamental equation in Bayesian inference, and describes how to compute optimal beliefs about an unknown stimulus by combining prior beliefs with observations of the stimulus. Nevertheless, Bayes' rule alone can only be applied to independent observations of static stimuli, and is not sufficient for explaining how animals compute beliefs about dynamic stimuli. The dynamic extension of Bayesian inference is known as Bayesian filtering, which complements Bayes' rule with an equation for computing optimal predictions (Thrun et al., 2005; Särkkä, 2013). By combining prediction and online inference, it is possible to define the Bayes filter, which is an algorithm for computing optimal beliefs about unknown, dynamic stimuli.

The Kalman filter is an example of a Bayes filter for when the stimulus dynamics are known and linear. Beck et al. (2011) show how to implement a form of Kalman filter in a theoretical neural circuit by combining the inferential properties of linear probabilistic population codes with a recurrent neural network which computes predictions. In order to define the neural network, they derive a closed-form expression for the network parameters based on the parameters of the linear stimulus dynamics. However, animals do not generally have built-in knowledge of the dynamics of stimuli, nor can they assume that these dynamics are linear. As such, this work is not sufficient for explaining how animals maintain accurate beliefs about dynamic stimuli.

In this paper we generalize the approach of Beck et al. (2011) to arbitrary dynamical systems where the stimulus dynamics are unknown. To do this, we begin by replacing the derived, linear network of Beck et al. (2011) with a general network with tunable parameters. By taking advantage of the exponential family structure of linear probabilistic population codes (Welling et al., 2004; Beck et al., 2007), we then show how to minimize the negative log-likelihood of the parameters of the network with stochastic gradient descent. Finally, we develop a novel algorithm for approximating the true stochastic gradient, which we compare against contrastive divergence minimization (Hinton, 2002). Although in our demonstrations we define the recurrent neural network as a form of multilayer perceptron, the theory we present can be applied to any param-

eterized network architecture, in particular ones which satisfy more realistic biological constraints.

The theory we develop in this paper is related to two additional approaches in the machine learning and computational neuroscience literatures. Firstly, the theoretical neural circuit that we construct can be roughly interpreted as a form of the model presented in Boulanger-Lewandowski et al. (2012) for approximate filtering. In our approach, however, the neural circuit satisfies a set of equations which ensure that Bayes' rule is exactly implemented, which is not present in the work of Boulanger-Lewandowski et al. (2012). Secondly, Makin et al. (2015) present an alternative approach to implementing a Bayes filter in a neural circuit based on probabilistic population codes. The form of the circuit in Makin et al. (2015), however, is markedly different from our own, and although it lacks some of the theoretical advantages of our neural circuit, it makes use of a more biologically realistic learning rule. We discuss these related methods in more detail at the end of this paper.

For the purposes of demonstration we apply our methods in three simulated experiments, each of which models how a particular neural circuit learns to maintain accurate beliefs about some unknown stimulus. In the first simulation the stimulus is a set of colours in a sequence learning task, which we model with a 3-state Markov chain, and the neural circuit is composed of colour recognition and sequence learning neural populations along the ventral stream. In the second simulation the stimulus is the position of a mouse on a track, which we model as a linear dynamical system, and the neural circuit is part of the self-localization system of the hippocampus. In the third simulation the stimulus is the angular position and velocity of a human arm, which we model as a stochastic pendulum, and the neural circuit is part of the proprioceptive system in the cerebellum.

In the first and second simulations it is possible to compute the optimal beliefs of the Bayes filter based on the true stimulus dynamics, whereas in the third simulation, approximate beliefs can be computed with a form of extended Kalman filter. These filters provide ground truth for our theoretical neural circuits, and in all cases, we find that our circuits are able to maintain good approximate beliefs about the stimulus. Moreover, by analyzing the hidden layer of the multilayer perceptron in the third experiment, we show how the network uses gain-fields to represent position and velocity information in a manner that is consistent with theory and experiment (Sejnowski, 1995; Salinas and Thier, 2000; Paninski et al., 2004; Herzfeld et al., 2015). At the end of this paper we consider additional features of our work which are relevant for neuroscience, and present ways in it can be extended in the future.

2 Inference

Statistical inference is the process of estimating unknown quantities through observation. There are many paradigms for formalizing statistical inference, but Bayesian inference is arguably the most appropriate framework for describing how an agent maintains subjective beliefs which correspond to the world. Two of the most well-known theoretical arguments for Bayesian inference are Cox's theorem and the class of Dutch book arguments (Jaynes, 2003; Talbott, 2015). On one hand, Cox's theorem demonstrates that the consistent extension of propositional logic on binary truth values to continuous probabilities is given by Bayesian inference. On the other hand, Dutch book arguments demonstrate that failing to follow the principles of Bayesian inference can lead gamblers to make wagers which they are guaranteed to lose.

Both of these arguments begin with the assumption that subjective beliefs are represented by probabilities, and lead irrevocably to Bayesian inference. In the context of neuroscience, this implies that if populations of neurons represent beliefs with encoded probability distributions, then populations of neurons must extract information from observations in accordance with the principles of Bayesian inference. Nevertheless, how populations of neurons implement Bayesian inference remains an open question, because there are many proposals for how populations of neurons might encode probability distributions (Pouget et al., 2013).

Each coding scheme has its own advantages, and greatly simplify particular operations on encoded information. Linear probabilistic population codes (LPPCs) have the invaluable property that they allow Bayes' rule to be trivially implemented on encoded beliefs (Ma et al., 2006). Evaluating Bayes' rule is the most fundamental operation in Bayesian inference, and involves combining the likelihood of an observation with prior beliefs in order to compute posterior beliefs. By using LPPCs, Ma et al. (2006) demonstrate how to construct a neural circuit which computes encodings of posterior beliefs by taking a linear combination of the response of a neural population and encoded prior beliefs.

A concrete example of such a neural circuit is the self-localization system in the hippocampus and entorhinal cortex (Moser et al., 2008). Place cells in the hippocampus encode information about the position of the animal, and many studies suggest that place cells optimize this encoding through a combination of local activity in the limbic system and incoming sensory information (McNaughton et al., 2006). In the Bayesian picture, place cells encode posterior beliefs about the position of the animal, and update these beliefs with Bayes' rule given the responses of sensory neurons.

We begin this section by introducing Bayes' rule, and we then formally define populations of neurons which generate Poisson-distributed spikes in re-

sponse to stimuli. We then formally introduce LPPCs, and demonstrate their relationship with the family of machine learning models known as exponential family harmoniums (Smolensky, 1986; Welling et al., 2004). This relationship allows us to easily rederive the results of Ma et al. (2006), and later allows us to apply machine learning algorithms to gather learning statistics from LPPCs and thereby model how neural systems learn to solve the filtering problem.

2.1 Bayes' Rule

Assuming it exists, the joint density p_{XN} provides a complete description of the relationship between the pair of random variables X and N. Given the joint density p_{XN}, we may derive the marginal densities p_X and p_N, and the conditional densities $p_{X|N}$ and $p_{N|X}$. With a bit of algebra, we may also derive Bayes' rule

$$p_{X|N}(\mathbf{x} \mid \mathbf{n}) = \frac{p_{N|X}(\mathbf{n} \mid \mathbf{x}) p_X(\mathbf{x})}{p_N(\mathbf{n})}. \tag{1}$$

Bayes' rule itself is a simple equation, but its interpretation is the basis for a general approach to statistical inference. Since we are applying Bayesian inference in the context of neuroscience, let us refer to $\mathbf{n}$ as the response and $\mathbf{x}$ as the stimulus. In Bayesian inference, p_X is the prior, which represents our current beliefs about the stimulus, and $p_{N|X}$ is the likelihood, which describes how responses are generated to the stimulus. The density $p_{X|N}$ is the posterior, which represents our new beliefs about the stimulus after observing a response, and all of these densities may be derived from the generative model p_{XN}, which is a complete description of the probabilistic relationship between the stimulus and response.

For a given response $\mathbf{n}$, $p_N(\mathbf{n})$ is constant. Since the posterior $p_{X|N=\mathbf{n}}$ is a probability density and must therefore integrate to 1, knowing the posterior up to a constant factor is sufficient for determining the posterior. For this reason, Bayes' rule is often defined as the proportionality relation

$$p_{X|N}(\mathbf{x} \mid \mathbf{n}) \propto p_{N|X}(\mathbf{n} \mid \mathbf{x}) p_X(\mathbf{x}), \tag{2}$$

which emphasizes that knowing the prior and the likelihood is sufficient for determining the posterior.

2.2 Poisson Neurons

Poisson neurons are a simple, yet theoretically rigorous approach to modelling how neurons respond to stimuli (Dayan and Abbott, 2005), and we make use of them throughout this paper. Given a population of d_N Poisson neurons and

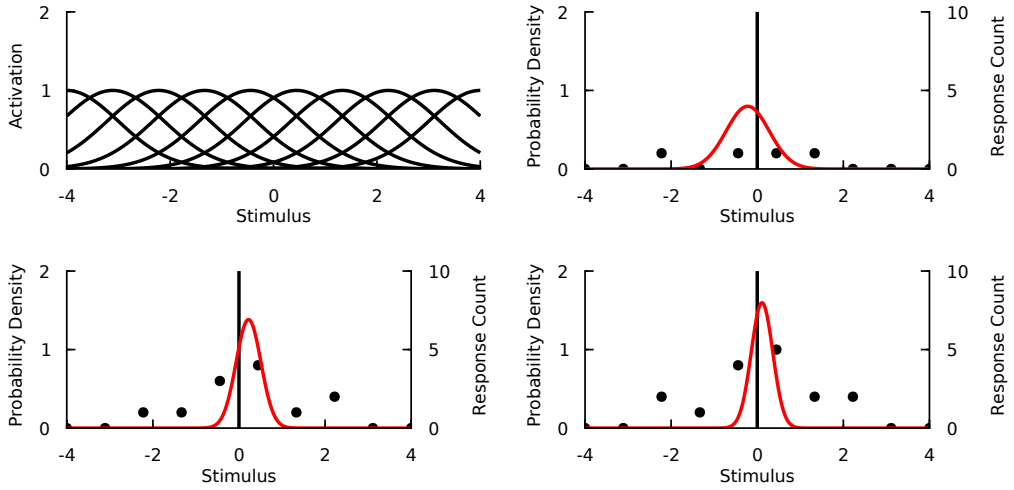

Figure 1: **Linear Probabilistic Population Codes**: These plots demonstrate encoding and decoding with linear probabilistic population codes. *Top Left*: The rates of ten Gaussian tuning curves with uniformly distributed preferred stimuli, and gain $\gamma = 1$. *Top Right*: The components of the response $\mathbf{n}_1$ (black dots) to the stimulus 0 (black line) generated with $\gamma = 2$, and the resulting posterior $p_{X|N=\mathbf{n}_1}$ (red line) based on an approximately flat prior. *Bottom Left*: Response $\mathbf{n}_2$, generated with $\gamma = 4$. *Bottom Right*: The encoded posterior of the rate of the posterior population $\mathbf{z} = \mathbf{n}_2 + \mathbf{y}$, where $\mathbf{A} = \mathbf{B} = \mathbf{I}$, and the rate of the prior population is $\mathbf{y} = \mathbf{n}_1$. Note that $\mathbf{z}$ is a more accurate encoding of the stimulus than either $\mathbf{y}$ or $\mathbf{n}_2$ alone.

a stimulus $\mathbf{x}$, the ith Poisson neuron generates a Poisson-distributed number of spikes with rate $\gamma f_i(\mathbf{x})$, where $f_i(\mathbf{x})$ is the tuning curve of the neuron, and γ is the gain. The component neurons of the population are conditionally independent of each other given the stimulus, such that the likelihood with respect to the entire population response N may be written

$$p_{N|X}(\mathbf{n} \mid \mathbf{x}) = \prod_{i=1}^{d_N} p_{N_i|X}(n_i \mid \mathbf{x}) = \prod_{i=1}^{d_N} \frac{e^{-\gamma f_i(\mathbf{x})}(\gamma f_i(\mathbf{x}))^{n_i}}{n_i!}. \tag{3}$$

When example tuning curves are called for in this paper, we consider the 1-dimensional Gaussian tuning curve

$$f_i(x) = e^{\frac{-(x-x_i^0)^2}{2\sigma^2}}, \tag{4}$$

with preferred stimuli x_i^0 (figure 1: Top Left). The theory we develop may nevertheless be generalized to a variety of tuning curves, as we later demonstrate in our simulations.

2.3 Linear Probabilistic Population Codes

A population code is a description of how to encode information about a stimulus in the combined activity of a population of neurons, as well as how to decode this information from the population. For a random stimulus X and random response N, a probabilistic population code (PPC) stochastically encodes information in neural populations by sampling from the likelihood $p_{N|X}$, and decodes this information by computing the posterior $p_{X|N}$ (Zemel et al., 1998; Beck et al., 2007). Finally, a linear probabilistic population code (LPPC) is a PPC where both the likelihood and posterior may be expressed as log-linear functions (Ma et al., 2006; Pouget et al., 2013). In this section we formally define LPPCs by using the theory of exponential families.

An exponential family is a set of probability densities with a specific form (Amari and Nagaoka, 2007; Wainwright and Jordan, 2008; Nielsen and Garcia, 2009). Each exponential family $\mathscr{M}$ is defined by a sufficient statistic $\mathbf{s}$ and a base measure ν. In turn, each density q in the exponential family $\mathscr{M}$ is given by

$$q(\mathbf{x}) \propto e^{\boldsymbol{\theta} \cdot \mathbf{s}(\mathbf{x})} \nu(\mathbf{x}), \tag{5}$$

where $\boldsymbol{\theta}$ are the natural parameters which specify the particular density. A wide variety of families of probably densities such as the normal, von Mises, and categorical families are in fact exponential families, and working with exponential families is typically much easier than generic families of densities.

Recall that at every stimulus $\mathbf{x}$, the likelihood $p_{N|X=\mathbf{x}}$ of a population of Poisson neurons (3) is a product of Poisson densities. The set of all products of Poisson densities is an exponential family with sufficient statistic equal to the identity function and base measure $\nu(\mathbf{n}) = (n_1! \cdots n_{d_N}!)^{-1}$. As such, where $\mathscr{M}_N$ is the exponential family of products of Poisson densities, the likelihood $p_{N|X=\mathbf{x}} \in \mathscr{M}_N$ for any $\mathbf{x}$.

Suppose that we also wish for the posterior $p_{X|N=\mathbf{n}}$ at any response $\mathbf{n}$ to be an element of some exponential family $\mathscr{M}_X$, where $\mathscr{M}_X$ is defined by the sufficient statistic $\mathbf{s}$, and a base measure which, for simplicity, we assume to be constant. Then it can be shown (Besag, 1974; Arnold and Press, 1989; Arnold et al., 2001) that the only generative model p_{XN} consistent with the likelihood $p_{N|X}$ of a Poisson population and a posterior $p_{X|N}$ which satisfies these assumptions is another exponential family density with the log-linear form

$$p_{XN}(\mathbf{x}, \mathbf{n}) \propto \frac{e^{\mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\Theta}_N \cdot \mathbf{n} + \mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\theta}_X + \mathbf{n} \cdot \boldsymbol{\theta}_N}}{n_1! \cdots n_{d_N}!}. \tag{6}$$

In the machine learning literature, models with this form are known as exponential family harmoniums (Welling et al. (2004), figure 2).

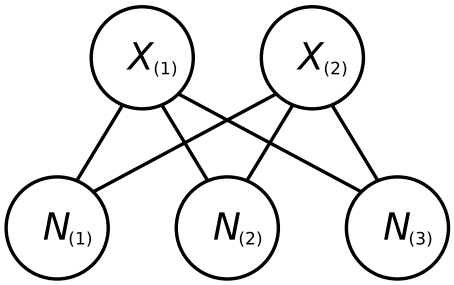

Figure 2: **Exponential Family Harmoniums**: Here we depict the graphical representation of an exponential family harmonium. From this graph we may infer that the component random responses $N_{(1)}$, $N_{(2)}$, and $N_{(3)}$ are mutually independent given the component random stimuli $X_{(1)}$ and $X_{(2)}$, and that $X_{(1)}$ is independent of $X_{(2)}$ given $N_{(1)}$, $N_{(2)}$, and $N_{(3)}$.

It follows from the theory of exponential family harmoniums (Welling et al., 2004) that the posterior $p_{X|N}$ and likelihood $p_{N|X}$ of a generative model defined in this way may be expressed in the log-linear forms

$$p_{N|X}(\mathbf{n} \mid \mathbf{x}) \propto \frac{e^{\mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\Theta}_N \cdot \mathbf{n} + \mathbf{n} \cdot \boldsymbol{\theta}_N}}{n_1! \cdots n_{d_N}!}, \tag{7}$$

and

$$p_{X|N}(\mathbf{x} \mid \mathbf{n}) \propto e^{\mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\Theta}_N \cdot \mathbf{n} + \mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\theta}_X}, \tag{8}$$

such that the likelihood $p_{N|X=\mathbf{x}} \in \mathscr{M}_N$ has natural parameters $\mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\Theta}_N + \boldsymbol{\theta}_N$, and the posterior $p_{X|N=\mathbf{n}} \in \mathscr{M}_X$ has natural parameters $\boldsymbol{\Theta}_N \cdot \mathbf{n} + \boldsymbol{\theta}_X$. Since a linear probabilistic population code is a probabilistic population code where the likelihood and posterior have log-linear forms, the conditional distributions of an exponential family harmonium define an LPPC.

Although it is perhaps surprising that the likelihood of a Poisson population (3) can be expressed in the log-linear form of relation 7, the results of Arnold et al. (2001) imply that this must be the case. For example, if $p_{N|X}$ is defined by the set of Gaussian tuning curves $f_i(x)$ with preferred stimuli x_i^0 and shared tuning width σ, then we can express it in the form of relation 7 by setting the elements of the matrix $\boldsymbol{\Theta}_N$ equal to

$$\boldsymbol{\Theta}_{N,(1,i)} = \frac{x_i^0}{\sigma^2}, \quad \boldsymbol{\Theta}_{N,(2,i)} = -\frac{1}{2\sigma^2}, \tag{9}$$

setting the elements of the vector $\boldsymbol{\theta}_N$ equal to

$$\boldsymbol{\theta}_{N,(i)} = \log \gamma - \frac{(x_i^0)^2}{2\sigma^2}, \tag{10}$$

and by letting $\mathbf{s}(x) = (x, x^2)$, which is the sufficient statistic of the family of normal distributions (figure 1: Top Right, Bottom Left).

2.4 Stimulus-Independent Total Rate

In many applications of Poisson neurons, the total rate $\gamma \sum_{i=1}^{d_N} f_i(X)$ of the population is designed to be approximately independent of the stimulus X itself (Sejnowski, 1995; Pouget et al., 2003; Beck et al., 2011). For the total rate to be independent of the stimulus, the sum of the tuning curves must satisfy

$$\sum_{i=1}^{d_N} f_i(\mathbf{x}) = \lambda \tag{11}$$

for some constant λ. It turns out that for many families of tuning curves, satisfying this relation is relatively straightforward. In the case of Gaussian tuning curves, for example, if one distributes the preferred stimuli uniformly over the space of the stimulus, then the sum of the tuning curves converges to a constant as the number of preferred stimuli is increased (Ma et al., 2006).

In addition to providing expressions for the LPPC prior and posterior, it also follows from the theory of exponential family harmoniums that the LPPC prior may be expressed as

$$p_X(\mathbf{x}) \propto e^{\mathbf{s}(\mathbf{x}) \cdot \boldsymbol{\theta}_X + \gamma \sum_{i=1}^{d_N} f_i(\mathbf{x})}. \tag{12}$$

Observe that when equation 11 is satisfied, the sum of the tuning curves in the LPPC prior is constant and can be absorbed into the constant of proportionality. This forces the LPPC prior to be an element of the same exponential family as the LPPC posterior. A prior is known as a conjugate prior when the prior and posterior share the same form. Conjugate priors have numerous computational advantages, and we return to equation 11 throughout this paper. One more advantage of LPPCs which satisfy this equation is that by setting $\boldsymbol{\theta}_X = \mathbf{0}$, we define the prior over the stimulus to be flat, which is a practical prior for many applications of Bayesian inference.

2.5 Neural Bayes' Rule

Let us refer to the population of Poisson neurons which generate the responses N as the observation population. In order to implement Bayesian inference in a theoretical neural circuit, let us define two further neural populations, called the prior population and the posterior population, with firing rates given by the random variables Y and Z, and number of neurons equal to d_Y and d_Z,

respectively. Our goal is to define Y and Z such that they encode the prior and posterior densities in an application of Bayes' rule.

Now theoretically, by defining the random variables Y and Z, we imply the existence of a generative model p_{NXYZ}, as well as implicit decoding densities $p_{X|Y}$ and $p_{X|Z}$. However, there are two reasons why we avoid relying on the implicit probabilistic structure of the random variables in order to define our decoders. Firstly, we ultimately know the form that we wish for the decoders to have, and ensuring that $p_{X|Y}$ and $p_{X|Z}$ have the desired forms requires fairly careful and rather abstruse analysis. Secondly, when it comes to the task of learning to approximate a Bayes filter, which we undertake in section 4, the implicit decoders are anyway inaccessible.

Therefore, in this section, and throughout this paper, we instead consider the prior and posterior decoders $q_{X|Y}$ and $q_{X|Z}$. We define these decoders to have forms of our choosing, and then demonstrate how to define Y and Z to encode the relevant densities in terms of these decoders. In those cases where we can perform exact inference, then these decoders are equal to the implicit decoders of the generative model. When we cannot perform exact inference, then we may still use these decoders to implement good approximations.

Let us define the density encoded by the prior rate vector $\mathbf{y}$ as

$$q_{X|Y}(\mathbf{x} \mid \mathbf{y}) \propto e^{\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_Y\cdot\mathbf{y}+\gamma\sum_{i=1}^{d_N} f_i(\mathbf{x})}, \tag{13}$$

where $\boldsymbol{\Theta}_Y$ is a matrix which we refer to as the decoding matrix of the prior population. Observe that $q_{X|Y=\mathbf{y}}$ is equal to the prior in relation 12 where $\boldsymbol{\theta}_X = \boldsymbol{\Theta}_Y \cdot \mathbf{y}$. Therefore, we may ensure that $q_{X|Y=\mathbf{y}} = p_X$ by choosing rates of the prior population $\mathbf{y}$ which satisfy $\boldsymbol{\theta}_X = \boldsymbol{\Theta}_Y \cdot \mathbf{y}$.

Let us then define the density encoded by the posterior rate vector $\mathbf{z}$ as

$$q_{X|Z}(\mathbf{x} \mid \mathbf{z}) \propto e^{\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_Z\cdot\mathbf{z}}, \tag{14}$$

where $\boldsymbol{\Theta}_Z$ is the decoding matrix of the posterior population. Let us also assume that $\boldsymbol{\Theta}_Z$ is related to the matrix $\boldsymbol{\Theta}_N$ of the likelihood by

$$\boldsymbol{\Theta}_N = \boldsymbol{\Theta}_Z \cdot \mathbf{A}, \tag{15}$$

for some matrix $\mathbf{A}$, and related to the decoding matrix $\boldsymbol{\Theta}_Y$ of the prior population by

$$\boldsymbol{\Theta}_Y = \boldsymbol{\Theta}_Z \cdot \mathbf{B}, \tag{16}$$

for some matrix $\mathbf{B}$.

Given these definitions, if we consider the LPPC posterior (8) given the prior rate vector $\mathbf{y}$ and the response $\mathbf{n}$, we find that

$$p_{X|N}(\mathbf{x} \mid \mathbf{n}) \propto e^{\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_N\cdot\mathbf{n}+\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_Y\cdot\mathbf{y}} = e^{\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_Z\cdot(\mathbf{A}\cdot\mathbf{n}+\mathbf{B}\cdot\mathbf{y})},$$

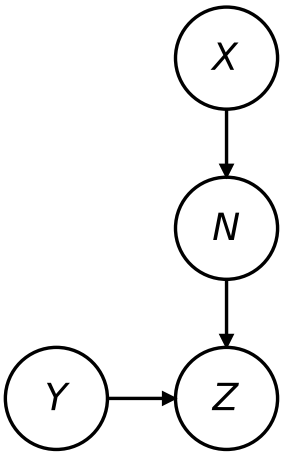

Figure 3: **Neural Circuit for Bayesian Inference**: Here we depict a random response N from a population of Poisson neurons to a random stimulus X. The random rates Y and Z represent the rates of the prior and posterior populations, respectively. The random response N is combined with the random prior rate Y to compute the random posterior rate Z in accordance with neural Bayes' rule.

which implies that

$$p_{X|N}(\mathbf{x} \mid \mathbf{n}) = q_{X|Z}(\mathbf{x} \mid \mathbf{A} \cdot \mathbf{n} + \mathbf{B} \cdot \mathbf{y}). \tag{17}$$

We refer to this equation as neural Bayes' rule. In words, as long as the prior population encodes the true prior, and the observation, prior, and posterior populations satisfy equations 15 and 16, then Bayes' rule may be implemented as a linear combination of the rates of the prior population $\mathbf{y}$ and the response of the observation population $\mathbf{n}$ (Ma et al., 2006).

In accordance with neural Bayes' rule, if we define the rates of the posterior population by

$$Z = \mathbf{A} \cdot N + \mathbf{B} \cdot Y, \tag{18}$$

where Y is drawn from a distribution over the set of $\mathbf{y}$ that satisfy $\boldsymbol{\theta}_X = \boldsymbol{\Theta}_Y \cdot \mathbf{y}$, we may ensure that the posterior population encodes the posterior density for any realization of the circuit. We depict the neural circuit composed of N, X, Y, and Z in figure 3, and we demonstrate an application of this circuit in the bottom right panel of figure 1.

3 Dynamics

Bayesian inference formalizes statistical inference as the answer to the question, "What is the conditional probability distribution over the unknown variable given the available observations?" If we are given a sequence of population

responses to a dynamic stimulus, we may compute the corresponding conditional probability over the stimulus with a recursive, two-step algorithm known as a Bayes filter. A Bayes filter recursively computes beliefs about the stimulus at time $k+1$ by computing predictions of the stimulus at time $k+1$ as a function of the beliefs at time k, and combining these predictions with the response at time $k+1$ using Bayes' rule (Thrun et al., 2005; Särkkä, 2013). This has important implications for the Bayesian brain hypothesis, because if neural populations are performing Bayesian inference, then not only must populations of neurons implement Bayes' rule, but they must also implement prediction.

For some neuroscientists this is not surprising, as many argue that prediction is essential to neural computation and constitutes a guiding principle for the architecture of the brain (Bubic et al., 2010; Friston, 2010; Clark, 2013). For example, forward models are ubiquitous in computational models of motor control (Miall and Wolpert, 1996; Todorov and Jordan, 2002; Franklin and Wolpert, 2011). Forward models in the brain predict the result of a given motor command, and this information is combined with the responses of proprioceptors in order to estimate body position. In addition to the observation, prior, and posterior populations introduced in section 2, this entails the existence of an additional neural network which transforms efference copies of the motor command into predictions of the consequences of that command.

We begin this section by formalizing dynamic stimuli and dynamic Poisson populations, and continue by introducing the prediction and update equations which define the Bayes filter. We then introduce closed-form solutions to the Bayes filter on population responses for the case of finite-state systems and linear dynamical systems, generalizing an approach described in Makin et al. (2015). We conclude by demonstrating how these solutions can be implemented in theoretical neural circuits based on the work of Beck and Pouget (2007) and Beck et al. (2011), and thereby lay the groundwork for how approximate solutions can be similarly defined.

3.1 Dynamic Poisson Populations

We define a dynamic Poisson population as a sequence of pairs of random variables $(X_k, N_k)_{k\in\mathbb{N}}$, where X_k is the random stimulus at time k, and N_k is the random response at time k. We assume that the dynamic stimulus is Markov, such that each stimulus X_{k+1} is conditionally independent of the past random variables given the previous stimulus X_k, where X_0 is drawn from some initial density p_{X_0}. We also assume that the densities $p_{X_{k+1}|X_k}$ are invariant with respect to k, such that we may recursively generate the sample sequences of $(X_k)_{k\in\mathbb{N}}$ with a single, time-invariant conditional density, which we refer to as the transition density. Because the transition density is time-invariant, we denote it by

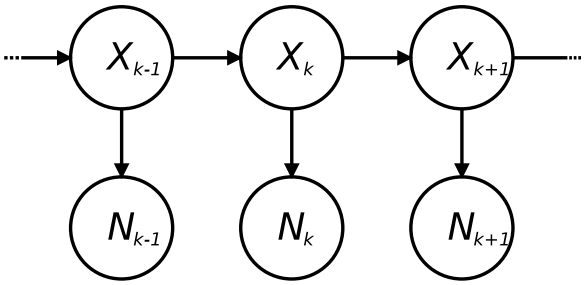

Figure 4: **Dynamic Poisson Population**: The graphical representation of a dynamic Poisson population. The arrow from the current stimulus X_k to the subsequent stimulus X_{k+1} represents the dependence of the subsequent stimulus on the current stimulus, and is described by transition density $p_{X'|X}$. The arrow from the current stimulus X_k to the current response N_k represents the dependence of the response on the stimulus, and is described by emission density $p_{N|X}$.

$p_{X'|X} = p_{X_{k+1}|X_k}$ for any k.

We assume that each response N_k is conditionally independent of all the other random variables given the simultaneous response X_k, and that the conditional density $p_{N_k|X_k}$ is given by the likelihood of a population of Poisson neurons, as defined in equation 3 and relation 7. The tuning curves and gain of the dynamic Poisson population often depend on time, but for the purposes of developing the theory in this section we assume that they are constant. As such, we denote the time-invariant conditional density by $p_{N|X} = p_{N_k|X_k}$ for any k, and refer to it as the emission density. We depict the graphical representation of the stimulus and Poisson population in figure 4.

3.2 Bayesian Filtering

Given the sequence of responses $\mathbf{n}_0, \ldots, \mathbf{n}_k$ from a dynamic Poisson population, a Bayes filter is an algorithm for computing the beliefs $p_{X_k|N_0=\mathbf{n}_0,\ldots,N_k=\mathbf{n}_k}$. A Bayes filter is defined by two equations. The first is a form of Bayes' rule (2), and is given by

$$p_{X_{k+1}|N_0,\ldots,N_{k+1}}(\mathbf{x}_{k+1} \mid \mathbf{n}_0, \ldots, \mathbf{n}_{k+1}) \propto p_{N|X}(\mathbf{n}_{k+1} \mid \mathbf{x}_{k+1}) p_{X_{k+1}|N_0,\ldots,N_k}(\mathbf{x}_{k+1} \mid \mathbf{n}_0, \ldots, \mathbf{n}_k). \tag{19}$$

When normalized, this relation is known as the update equation, and computes the beliefs at time $k+1$ by applying Bayes' rule to the emission density $p_{N|X}$ and the prior $p_{X_{k+1}|N_0=\mathbf{n}_0,\ldots,N_k=\mathbf{n}_k}$.

This prior constitutes beliefs about the stimulus at time $k+1$ given only the sequence of responses up to time k. Because these prior beliefs transform

available information into information about the future, these prior beliefs are known as predictions. The prediction density may be expressed as

$$p_{X_{k+1}|N_0,\ldots,N_k}(\mathbf{x}_{k+1} \mid \mathbf{n}_0, \ldots, \mathbf{n}_k) = \int_{\mathscr{X}} p_{X'|X}(\mathbf{x}_{k+1} \mid \mathbf{x}_k) p_{X_k|N_0,\ldots,N_k}(\mathbf{x}_k \mid \mathbf{n}_0, \ldots, \mathbf{n}_k) d\mathbf{x}_k, \tag{20}$$

where $\mathscr{X}$ is the state space of the stimulus. If the state space is countable, then the integral in this equation becomes a sum.

The predictions are a function of the transition density $p_{X'|X}$ and the beliefs $p_{X_k|N_0=\mathbf{n}_0,\ldots,N_k=\mathbf{n}_k}$ at time k. Therefore, given the transition and emission densities, we may recursively compute the beliefs at any time k given the sequence of responses $\mathbf{n}_0, \ldots, \mathbf{n}_k$ by using the update equation to calculate the beliefs as a function of the predictions, and by using the prediction equation to compute the predictions as a function of the previous beliefs. This recursion ultimately completes at the prior over the initial stimulus p_{X_0}.

3.3 Closed-Form Solutions

For arbitrary dynamic stimuli and dynamic Poisson populations, the prediction and update equations cannot be evaluated in closed-form. Nevertheless, dynamic Poisson populations do have a particular flexibility when it comes to Bayesian filtering. Before we explain this, first note that the prediction equation (20) depends on the transition density $p_{X'|X}$, but not the emission density $p_{N|X}$. This implies that if we know how to solve the prediction equation for a particular dynamic stimulus, then we may apply this solution towards evaluating the Bayes filter for any form of emission density.

Now recall from section 2.3 that the posterior (8) computed from the response of a Poisson population (3) and an appropriate prior (12) is a member of an exponential family determined by the form of the tuning curves of the Poisson population (7). Moreover, as described in section 2.4, if the sum of the tuning curves is constant (11), then the prior is a member of the same exponential family as the posterior. As such, if the sum of the tuning curves of the emission density is constant (11), and if the predictions at time k (20) are elements of the same exponential family as the beliefs, then we can compute the exponential family beliefs at time k (19) by computing the posterior (8) as a function of the emission density and the exponential family predictions.

More formally, suppose that $\mathscr{M}_X$ is the exponential family which matches the tuning curves of the emission density $p_{N|X}$, that the sum of the tuning curves of $p_{N|X}$ is constant (11), and that the parameters of the initial prior $p_{X_0} \in \mathscr{M}_X$ are $\boldsymbol{\theta}^*$. Then given the initial response $\mathbf{n}_0$, the natural parameters of the initial

belief density $p_{X_0|N_0=\mathbf{n}_0} \in \mathscr{M}_X$ are $\boldsymbol{\theta}_0 = \boldsymbol{\Theta}_N \cdot \mathbf{n}_0 + \boldsymbol{\theta}^*$ in accordance with the posterior of linear probabilistic population codes (8). Now suppose that the belief density $p_{X_k|N_0,\ldots,N_k}$ at time k is in $\mathscr{M}_X$ with natural parameters $\boldsymbol{\theta}_k$, that the prediction density $p_{X_{k+1}|N_0,\ldots,N_k}$ at time $k+1$ is also in $\mathscr{M}_X$, and that there exists a function $\mathbf{h}$ which computes the natural parameters $\mathbf{h}(\boldsymbol{\theta}_k)$ of $p_{X_{k+1}|N_0,\ldots,N_k}$. Then given the sequence of population responses $\mathbf{n}_0, \ldots, \mathbf{n}_{k+1}$, we may compute natural parameters of the belief density at time $k+1$ by computing

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\Theta}_N \cdot \mathbf{n}_{k+1} + \mathbf{h}(\boldsymbol{\theta}_k), \tag{21}$$

and therefore by induction, we may compute the parameters of the belief density at any k.

The question then becomes whether we can find such a function $\mathbf{h}$. Outside the context of neuroscience, there are two well-known cases for which Bayes filters may be evaluated in closed-form. The first is when the latent variable (i.e. the stimulus) and the observation (i.e. the response) only take on a finite number of values. In this case both the normalization in the update equation and the sum in the prediction equation can be computed brute-force. The second case is when the transition and emission densities are given by linear transformations with additive Gaussian noise. In this case the solution is known as a Kalman filter, and solving the corresponding prediction and update equations may be reduced to straightforward linear algebra (Thrun et al., 2005; Särkkä, 2013).

Now suppose we wish to apply the solutions to the prediction equation provided by these two Bayes filters to computing beliefs given sequences of population responses. The prediction and belief densities are given by categorical densities in the finite-state case, and (multivariate) normal densities in the Kalman filter case, both of which are exponential family densities. This implies that if $\boldsymbol{\theta}_k$ are the natural parameters of the belief density at any time k, then there exists a function $\mathbf{h}$ such that $\mathbf{h}(\boldsymbol{\theta}_k)$ are the natural parameters of the prediction density at time $k+1$, and we may therefore use equation 21 to recursively compute the beliefs at all times.

Moreover, even when the prediction densities are not elements of an exponential family, it often remains a good strategy to approximate them with exponential family densities. For example, the most well-known extension of Kalman filtering to nonlinear dynamical systems is the extended Kalman filter (Thrun et al., 2005; Särkkä, 2013). Although the extended Kalman filter does not compute the true predictions, it does compute good exponential family approximations to the true predictions, and we may use this approximation to define the function $\mathbf{h}$, and thereby approximate a Bayes filter. We apply these techniques later in this paper when we validate our method for learning to compute approximate predictions.

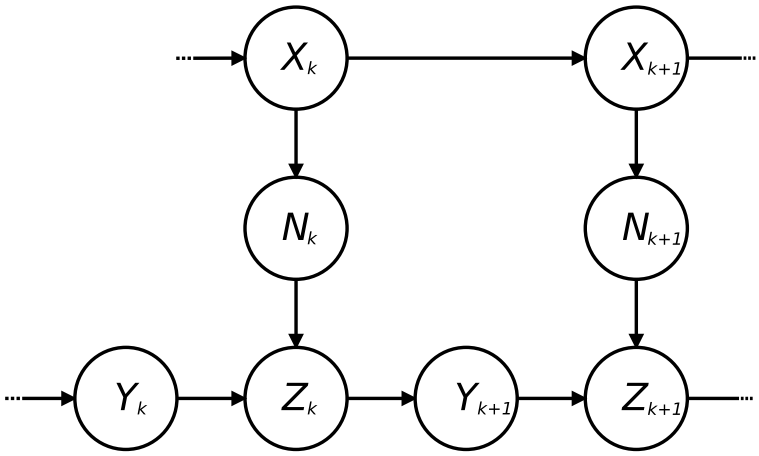

Figure 5: **Generic Neural Circuit**: Here we depict the graphical representation of the generic neural circuit proposed in this paper. At time k, the circuit is composed of a random dynamic stimulus X_k, a random dynamic response N_k, and the random rates of the prediction and filtering populations, Y_k and Z_k. The arrow from Z_k to Y_{k+1} represents the output of the prediction network $\mathbf{g}$, such that $Y_{k+1} = \mathbf{g}(Z_k)$. The arrows from Y_{k+1} and N_{k+1} to Z_{k+1} represent the linear combination of Y_{k+1} and N_{k+1}, such that $Z_{k+1} = \mathbf{A} \cdot N_{k+1} + \mathbf{B} \cdot Y_{k+1}$.

3.4 Optimal Filtering in Neural Circuits

In the previous section we demonstrated how to solve the prediction and update equations based on dynamic Poisson populations, and in this section we wish to encode these solutions in a dynamic neural circuit. In section 2.5 we introduced the concept of prior and posterior populations. In the context of Bayesian filtering, we refer to these populations as the prediction population and the filtering population.

We begin by defining the components of the dynamic neural circuit. Let $(Y_k)_{k\in\mathbb{N}}$ and $(Z_k)_{k\in\mathbb{N}}$ be the sequence of random firing rates of the prediction and filtering populations. Given the emission density $p_{N|X}$ with parameters $\mathbf{\Theta}_N$, let $\mathbf{\Theta}_Y$, $\mathbf{\Theta}_Z$, $\mathbf{A}$, and $\mathbf{B}$ be matrices which satisfy equations 15 and 16, and let us define the densities encoded by the prediction and filtering populations Y_k and Z_k at any time k as the decoding densities $q_{X|Y}$ and $q_{X|Z}$ in relations 13 and 14, respectively. Let the rates of the filtering population be $Z_k = \mathbf{A} \cdot N_k + \mathbf{B} \cdot Y_k$, let the initial rates of the prediction population Y_0 be given by some density p_{Y_0}, and finally, for $k > 0$, let the rates of the prediction population be $Y_k = \mathbf{g}(Z_{k-1})$, where $\mathbf{g}$ is a neural network which we refer to as the prediction network. We depict the graphical representation of this dynamic neural circuit in figure 5.

Let us now consider initial rates of the prediction population $\mathbf{y}_0$ which satisfy $q_{X|Y=\mathbf{y}_0} = p_{X_0}$, the initial population response $\mathbf{n}_0$ and the initial rates of the

filtering population $\mathbf{z}_0 = \mathbf{A} \cdot \mathbf{n}_0 + \mathbf{B} \cdot \mathbf{y}_0$. Since the prior p_{X_0} is equal to $q_{X|Y=\mathbf{y}_0}$ by assumption, the encoded beliefs $q_{X|Z=\mathbf{z}_0}$ are equal to the beliefs $p_{X_0|N_0=\mathbf{n}_0}$ in accordance with neural Bayes' rule (17). Let us now suppose that at an arbitrary time k, the rates of the filtering population are $\mathbf{z}_k$, the subsequent rates of the prediction population are $\mathbf{y}_{k+1} = \mathbf{g}(\mathbf{z}_k)$, and the subsequent response is $\mathbf{n}_{k+1}$. Let us also suppose that the prediction network $\mathbf{g}$ has the property that if $q_{X|Z=\mathbf{z}_k}$ is equal to the belief density $p_{X_k|N_0=\mathbf{n}_0,\ldots,N_k=\mathbf{n}_k}$ at time k, then $q_{X|Y=\mathbf{y}_{k+1}}$ is equal to the prediction density $p_{X_{k+1}|N_0=\mathbf{n}_0,\ldots,N_k=\mathbf{n}_k}$ at $k+1$. Since the update equation (19) is simply an application of Bayes' rule to the emission and prediction densities, neural Bayes' rule (17) implies that $q_{X|Z=\mathbf{z}_{k+1}}$ is equal to $p_{X_{k+1}|N_0=\mathbf{n}_0,\ldots,N_{k+1}=\mathbf{n}_{k+1}}$ where $\mathbf{z}_{k+1} = \mathbf{A} \cdot \mathbf{n}_{k+1} + \mathbf{B} \cdot \mathbf{y}_{k+1}$. Therefore, by induction, $\mathbf{y}_k$ and $\mathbf{z}_k$ encode the predictions and beliefs at any k, and the dynamic neural circuit depicted in figure 5 exactly implements a Bayes filter for any realization of the system.

The question, again, is whether there in fact exists a neural network $\mathbf{g}$ which computes encodings of the true predictions. The work of Beck and Pouget (2007) and Beck et al. (2011) answers this question in the affirmative, where the authors derived stochastic differential equations which describe how to encode beliefs in a neural population when the stimulus has either finite-states or is driven by linear dynamics, respectively. Critically, these solutions involve linearly combining the responses of the observation population with encoded predictions at every time step, in accordance with equation 18.

In the linear dynamical system case, under the assumption that the dynamic population code satisfies equation 11, the prediction network $\mathbf{g}$ may be expressed as

$$\mathbf{g}(\mathbf{z}) = (\mathbf{G}^{(2)} \cdot \mathbf{z} + \mathbf{z} \cdot \mathbf{G}^{(3)} \cdot \mathbf{z} + \mathbf{1}(z^0 - \frac{\mathbf{1} \cdot \mathbf{z}}{m}))dt + \mathbf{z}, \tag{22}$$

where dt is the time-step in the time-discretized system. Intuitively, $\mathbf{G}^{(2)}$ drives the rate of the population in proportion to the linear dynamics, $\mathbf{G}^{(3)}$ quadratically drives the rate of the population in proportion to the noise in the dynamics, and z^0 is a parameter which encourages the component-wise average of the rate process to remain near z^0 (Beck et al., 2011). Although this is the optimal solution, computing this $\mathbf{g}$ depends on knowing the parameters of linear the stimulus dynamics, which may not always be available. In the next section we describe how to maximize the likelihood of of a parameterized $\mathbf{g}$ based the responses of the observation population.

4 Learning

In the previous section we described how to compute beliefs about an unknown, dynamic stimulus with a Bayes filter. Evaluating the Bayes filter requires solving the prediction and update equations, which in turn depend on access to the transition and emission densities, respectively. Although assuming that we can access these densities allows us to define optimal neural circuits directly, this assumption is not ecologically valid. Animals do not, in general, have direct access to emission and transition densities, and must rather learn to implement Bayes filters in their neural circuitry.

In this paper we reduce learning to the optimization of the parameters of a theoretical neural circuit. Different tasks and experimental designs call for training some of these parameters, and leaving others fixed. In our case, we focus on optimizing the parameters of a neural network for computing approximate predictions, and assume that the rest of the parameters of the neural circuit have already been optimized. This scenario applies when dealing with an adult subject with well-developed neural populations for sensation, but no familiarity with the task or dynamic stimulus.

A concrete example of such a task is sequence learning in a psychophysics experiment (Clegg et al., 1998). Many colour sensitive neurons are found in the visual area V4, which connect further down the ventral stream with neurons in the inferior temporal cortex (ITC) (Roe et al., 2012), which in turn has been found to play a role in sequence learning (Meyer and Olson, 2011). We may thus interpret the neural populations in V4 as the observation population, and populations in the ITC as the prediction and filtering populations. In a given sequence learning task, the prediction network is then trained to ensure that the rates of the filtering population in the ITC encode accurate beliefs about the stimulus.

In this section we begin by defining the architecture of our theoretical neural circuit, and introducing the general negative log-likelihood gradient on the parameters of the neural network. We continue by deriving an expression for this gradient, and then introduce contrastive divergence minimization and a novel exponential family approximation in order to descend it.

4.1 Approximate Filtering in Neural Circuits

Suppose we know the emission density $p_{N|X}$ of a dynamic Poisson population $(X_k, N_k)_{k\in\mathbb{N}}$, and that we wish to train a neural circuit to approximately implement a Bayes filter on responses of the dynamic Poisson population. We take a parametric approach in this paper, which means we must choose a form for the approximate beliefs. We therefore assume that the approximate belief densi-

ties are elements of some exponential family $\mathscr{M}_X$ with sufficient statistic $\mathbf{s}$ and a constant base measure.

As discussed in sections 2.3 and 2.4, in order apply the update equation (19) to compute such exponential family beliefs given a response from the observation population, the approximate predictions must be proportional to $e^{\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\theta}_X+\gamma\sum_{i=1}^{d_N} f_i(\mathbf{x})}$ for some parameters $\boldsymbol{\theta}_X$. Moreover, as described in section 3.3, since many algorithms for Bayesian filtering yield predictions and beliefs which are in a single exponential family, and since we typically aim to approximate these algorithms as well as possible, we typically assume that the sum of the tuning curves of the observation population is constant such that the approximate prediction densities are also elements of $\mathscr{M}_X$.

Nevertheless, the brain cannot work with such abstract prediction and belief densities directly. The theoretical neural circuit described in section 3.4 and depicted in figure 5 is designed to encode predictions and beliefs with such exponential family forms by way of the prediction and filtering population decoders $q_{X|Y}$ (13) and $q_{X|Z}$ (14). We therefore assume that the neural circuit considered in this section has the same structure as in section 3.4, except now the prediction network $\mathbf{g}$ is parameterized by $\boldsymbol{\phi}$. In our simulations, $\mathbf{g}$ will be a three layer perceptron with d_Y input neurons, d_Z output neurons, d_H hidden neurons, and $\boldsymbol{\phi}$ will be a pair of matrices and biases.

The prediction network $\mathbf{g}$ computes encodings of the parameters of prediction densities over the stimuli, and so if we could access sample sequences of the stimuli $\mathbf{x}_0, \ldots, \mathbf{x}_k$, then optimizing $\mathbf{g}$ would reduce to a regression problem. However, since the purpose of a Bayes filter, approximate or otherwise, is to compute beliefs about unknown stimuli, using these stimuli for training purposes would violate the spirit of the problem. Therefore, we aim instead to optimize $\mathbf{g}$ based on sequences of population responses $\mathbf{n}_0, \ldots, \mathbf{n}_k$.

In order to optimize $\mathbf{g}$ based on sequences of responses, we first define the approximate generative model

$$q_{XN|Y}(\mathbf{x}, \mathbf{n} \mid \mathbf{y}) \propto \frac{e^{\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_N\cdot\mathbf{n}+\mathbf{s}(\mathbf{x})\cdot\boldsymbol{\Theta}_Y\cdot\mathbf{y}+\mathbf{n}\cdot\boldsymbol{\theta}_N}}{n_1!\cdots n_{d_N}!}, \tag{23}$$

which is equal to the harmonium defined in relation 6 where $\boldsymbol{\Theta}_Y \cdot \mathbf{y} = \boldsymbol{\theta}_X$. As discussed in sections 2.3 and 2.4, this density is the only approximate generative model over stimuli and responses which is consistent with exponential family beliefs, a likelihood given by a population of Poisson neurons, and predictions encoded by $\mathbf{y}$. Where $q_{N|Y}$ is the marginal density of $q_{XN|Y}$, and given the sequence of responses $\mathbf{n}_0, \ldots, \mathbf{n}_k$, we may maximize the likelihood of the parameters $\boldsymbol{\phi}$ by following the stochastic negative log-likelihood gradient

$$-\nabla_{\boldsymbol{\phi}} \log q_{N|Y}(\mathbf{n}_k \mid \mathbf{y}_k), \tag{24}$$

where $\mathbf{y}_k$ is computed as a function of the neural network and the sequence of responses $\mathbf{n}_0, \ldots, \mathbf{n}_{k-1}$ (see Welling et al., 2004; Bengio, 2009).

4.2 Computing the Gradient

As shown in Welling et al. (2004), the component partial derivatives of the negative log-likelihood gradient of $\boldsymbol{\theta}_X$ for the harmonium in relation 6 are

$$-\partial_{\boldsymbol{\theta}_X} \log p_N(\mathbf{n}) = \mathbb{E}_p[\mathbf{s}(X) \mid N = \mathbf{n}] - \mathbb{E}_p[\mathbf{s}(X)], \tag{25}$$

where the random variables are distributed according to p_{XN}. Given a sequence of responses $\mathbf{n}_0, \ldots, \mathbf{n}_k$ and the corresponding rates of the prediction population $\mathbf{y}_0, \ldots, \mathbf{y}_k$, if we consider the derivative of the negative log-likelihood with respect to the component parameters ϕ_i of $\boldsymbol{\phi}$, and define the biases as $\boldsymbol{\theta}_X = \boldsymbol{\Theta}_Y \cdot \mathbf{y}_k$, we may apply the chain rule to combine derivative 25 with the partial derivative $\partial_{\phi_i} \mathbf{y}_k$. By taking the expectations with respect to $q_{XN|Y}$ and adding the dependencies on Y_k, we may then write the partial derivatives as

$$\begin{aligned} &-\partial_{\phi_i} \log q_{N|Y}(\mathbf{n}_k \mid \mathbf{y}_k) = \\ &\quad (\mathbb{E}_q[\mathbf{s}(X) \mid N = \mathbf{n}_k, Y = \mathbf{y}_k] - \mathbb{E}_q[\mathbf{s}(X) \mid Y_k = \mathbf{y}_k]) \cdot \boldsymbol{\Theta}_Y \cdot \partial_{\phi_i} \mathbf{y}_k. \end{aligned} \tag{26}$$

Observe that this gradient is zero when the predictions of the network match the posterior, and is thus a form of prediction error.

Because $\mathbf{y}_k$ depends recurrently on $\mathbf{y}_0, \ldots, \mathbf{y}_{k-1}$, we must recursively apply the chain rule to the gradient $\partial_{\phi_i} \mathbf{y}_k$ until $\mathbf{y}_0$ is reached. This leads to the algorithm known as backpropagation-through-time (Werbos, 1990; Sutskever et al., 2009; Makin et al., 2016) for computing the recursive gradient of $\mathbf{y}_k$. Unfortunately, backpropagation-through-time is known to be problematic (Bengio et al., 1994; Pascanu et al., 2013), and introduces complexities into the gradient calculation that we wish to avoid in this paper.

In section 3.4 we showed how to define optimal and near-optimal filtering populations such that $(Z_k)_{k \in \mathbb{N}}$ loses (nearly) no information about the stimulus over time. This implies that $(Z_k)_{k \in \mathbb{N}}$ is (approximately) Markov, and so at the optimal parameters $\boldsymbol{\phi}$ of $\mathbf{g}$, Y_k is (nearly) independent of Z_{k-j} for $j > 1$. This suggests that we may ignore the long-range dependencies in the gradient and still hope to find good parameters. We therefore consider a one-step approximation to the true derivative of $\mathbf{y}_k$, and assume that the rates of the filtering population at the previous time $\mathbf{z}_{k-1}$ are independent of the parameters $\boldsymbol{\phi}$. This allows us to express the components of gradient 24 as

$$\begin{aligned} &-\partial_{\phi_i} \log q_{N|Y}(\mathbf{n}_k \mid \mathbf{y}_k) = \\ &\quad (\mathbb{E}_q[\mathbf{s}(X) \mid N = \mathbf{n}_k, Y = \mathbf{y}_k] - \mathbb{E}_q[\mathbf{s}(X) \mid Y_k = \mathbf{y}_k]) \cdot \boldsymbol{\Theta}_Y \cdot \partial_{\phi_i} \mathbf{g}(\mathbf{z}_{k-1}), \end{aligned} \tag{27}$$

and thereby reduce the problem of maximizing the likelihood of the parameters $\boldsymbol{\phi}$ to computing equation 25 and the partial derivatives $\partial_{\phi_i}\mathbf{g}$ at $\mathbf{z}_{k-1}$. When $\mathbf{g}$ is a multilayer perceptron, we may apply standard backpropagation to compute these derivatives (Rumelhart et al., 1986). The last remaining problem is therefore to compute the expectations $\mathbb{E}_q[\mathbf{s}(X) \mid N = \mathbf{n}_k, Y = \mathbf{y}_k]$ and $\mathbb{E}_q[\mathbf{s}(X) \mid Y = \mathbf{y}_k]$.

4.3 Approximating the Harmonium Expectations

Computing the first conditional expectation $\mathbb{E}_q[\mathbf{s}(X) \mid N = \mathbf{n}_k, Y = \mathbf{y}_k]$ is not challenging, as $q_{X|N=\mathbf{n}_k,Y=\mathbf{y}_k}$ is given by the decoder of the filtering population $q_{X|Z=\mathbf{A}\cdot\mathbf{n}_k+\mathbf{B}\cdot\mathbf{y}_k}$ (14) in accordance with neural Bayes' rule (17). By design $q_{X|Z}$ is always an element of some exponential family, and for many exponential families the expected value of the sufficient statistics can be computed exactly, or in the very least can be approximated by sampling. On the other hand, $\mathbb{E}_q[\mathbf{s}(X) \mid Y = \mathbf{y}_k]$ is determined by the encoded prediction density $q_{X|Y}$ (13), which cannot, in general, be trivially evaluated or sampled. In this paper we consider two strategies for approximating this expectation.

Firstly, the expectations of the marginal densities of an exponential family harmonium may be approximated by Gibbs sampling (Geman and Geman, 1984; Roberts and Polson, 1994). In this context, gibbs sampling involves constructing a Markov chain through recursive sampling of the densities $q_{N|X,Y}$ and $q_{X|N,Y}$ based on an arbitrary initial response $\mathbf{n}^0$. It can be shown that the sample stimuli and responses generated after many such iterations are distributed approximately according to the density $q_{XN|Y}$. Since it often takes a long time for this Markov chain to converge, the contrastive divergence gradient was developed as an alternative to the negative log-likelihood gradient (Hinton, 2002; Bengio and Delalleau, 2009). Approximating the contrastive divergence gradient leads to a similar approximation scheme as Gibbs sampling, however, instead of an arbitrary starting condition which we wish the sampler to forget, we let $\mathbf{n}^0 = \mathbf{n}_k$, which allows a useful gradient to be calculated after a handful of iterations.

Secondly, if the sum of the tuning curves of the emission density $p_{N|X}$ are constant (11), then the decoder of the prediction population $q_{X|Y}$ is in the same exponential family as the decoder of the filtering population $q_{X|Z}$, and therefore $\mathbb{E}_q[\mathbf{s}(X) \mid Y = \mathbf{y}_k]$ can be evaluated as easily as $\mathbb{E}_q[\mathbf{s}(X) \mid N = \mathbf{n}_k, Y = \mathbf{y}_k]$. In particular, where $\boldsymbol{\tau}$ is the coordinate transform from the natural parameters to the expectation parameters of the exponential family of $\mathbf{s}$ (Amari and Nagaoka, 2007; Wainwright and Jordan, 2008; Nielsen and Garcia, 2009), equation 25

is then given by

$$\mathbb{E}_p[\mathbf{s}(X) \mid N = \mathbf{n}] - \mathbb{E}_p[\mathbf{s}(X)] = \boldsymbol{\tau}(\boldsymbol{\Theta}_N \cdot \mathbf{n}_k + \boldsymbol{\theta}_X) - \boldsymbol{\tau}(\boldsymbol{\theta}_X). \tag{28}$$

In the case of the 1-dimensional Gaussian tuning curve, $\boldsymbol{\tau}$ may be computed in closed-form, and is given by

$$\boldsymbol{\tau}(\theta_{X,1}, \theta_{X,2}) = \left(-\frac{\theta_{X,1}}{2\theta_{X,2}}, \frac{\theta_{X,1}}{4\theta_{X,2}} - \frac{1}{2\theta_{X,1}} \right). \tag{29}$$

Nevertheless, equation 11 is often only approximately satisfied, and so this strategy may not always be optimal. In the subsequent section, we compare the performance of this exponential family approximation with contrastive divergence minimization.

5 Simulations

In this section we describe in detail how to apply the methods we have developed in three simulated experiments. In each experiment we aim to understand how a subject learns to compute accurate beliefs about an unknown dynamic stimulus with a neural circuit composed of an observation population which generates responses to stimuli, a prediction population which encodes approximate predictions, a filtering population which encodes approximate beliefs, and a prediction network which computes rates of the prediction population as a function of the rates of the filtering population. In the first half of this section we describe the details of the neural circuits, the training procedures, and the validation procedures, which are more or less the same across all experiments. In the second half of this section we present the results of the three simulated experiments.

5.1 Methods

The theoretical neural circuits we consider are composed of the observation, prediction, and filtering populations, and the prediction network. The parameters of the observation population are the parameters of the emission density $\boldsymbol{\Theta}_N$ and $\boldsymbol{\theta}_N$, and the observation population recoder $\mathbf{A}$; the parameters of the prediction population are the decoding matrix $\boldsymbol{\Theta}_Y$, the prediction population recoder $\mathbf{B}$, and the initial rates $\mathbf{y}_0$; and the parameters of the filtering population are the decoding matrix $\boldsymbol{\Theta}_Z$. We set these parameters of the three neural populations by hand. The remaining parameters are the parameters $\boldsymbol{\phi}$ of the

prediction network $\mathbf{g}$, which we optimize by stochastic gradient descent on the negative log-likelihood of $\boldsymbol{\phi}$ given sequences of population responses.

In order to test what kind of population codes are likely used by the brain, we propose two candidate neural circuits which differ in how the prediction and filtering populations encode probabilities, as defined by $\boldsymbol{\Theta}_Y$ and $\boldsymbol{\Theta}_Z$. Moreover, when training the prediction network in each circuit, we apply and compare contrastive divergence minimization (Hinton, 2002) and the exponential family approximation to the negative log-likelihood gradient (28), leading to a total of four sub-experiments in each experiment. Because we keep the stimuli mathematically simple, we may then validate the learned filter against the corresponding optimal, or mostly optimal, filter.

All simulations presented in this paper were developed in Haskell, and are available at the repository of the author at *hub.darcs.net/alex404/goal*.

5.1.1 Population Parameters

In each simulation we define the parameters of the emission density $\boldsymbol{\Theta}_N$ and $\boldsymbol{\theta}_N$ by first defining the gain γ and tuning curves $(f_i)_{i=1}^{d_N}$ such that the sum of the tuning curves is (approximately) constant. We then equate the likelihood of the Poisson population (3) with its exponential family form (7) and solve for $\boldsymbol{\Theta}_N$ and $\boldsymbol{\theta}_N$. We define the decoding matrix $\boldsymbol{\Theta}_Y$ as equal to the decoding matrix $\boldsymbol{\Theta}_Z$. This implies that the prediction population recoder $\mathbf{B} = \mathbf{I}$, since the population codes of the prediction and filtering populations are the same. In this case we may intuitively think of the prediction and filtering populations as a single population for encoding approximate beliefs. Finally, since the sum of the tuning curves of the emission density is constant (11), we set the initial rates of the prediction population to $\mathbf{y}_0 = \mathbf{0}$, such that the prediction population initially encodes a flat prior over the stimulus.

We define the parameters of the filtering population $\boldsymbol{\Theta}_Z$ in one of two ways. The first is by setting $\boldsymbol{\Theta}_Z = \boldsymbol{\Theta}_N$, which we refer to as the *naive* code. In the case, the observation population recoder is given by $\mathbf{A} = \mathbf{I}$. We refer to the second code as the *orthogonal* code, based on the code presented in the supplementary material of Beck et al. (2011). In this case we construct $\boldsymbol{\Theta}_Z$ from a set of mutually orthogonal rows, which are also orthogonal to the vector of ones, such that $\boldsymbol{\Theta}_{Z,(i)} \cdot \boldsymbol{\Theta}_{Z,(j)} = 0$ for $i \neq j$, and $\boldsymbol{\Theta}_{Z,(i)} \cdot \mathbf{1} = 0$. In this case $\boldsymbol{\Theta}_Z \cdot \mathbf{1} = \mathbf{0}$, which implies that for any rates of the filtering population $\mathbf{z}$, the rates $\mathbf{z} + c\mathbf{1}$ encode the same beliefs for any scalar c. The details of constructing $\boldsymbol{\Theta}_Z$, as well as a corresponding observation population recoder $\mathbf{A}$ which satisfies equation 15, can be found the supplementary material of Beck et al. (2011).

For the naive code, because $\boldsymbol{\Theta}_N = \boldsymbol{\Theta}_Y = \boldsymbol{\Theta}_Z$, all three neural populations in the circuit have the same number of neurons. In the case of the orthogonal

Parameter	Experiment 1	Experiment 2	Experiment 3
$d_N = d_Y = d_Z$	10	10	20
d_H	100	200	500
n_t	10,000	10,000	20,000

Table 1: **Summary of Circuit and Training Parameters**: In this table we show the simulation parameters which change in each experiment. These are the sizes of the observation, prediction, and filtering populations d_N, d_Y, d_Z, respectively; the number of hidden neurons in the prediction network d_H; and the number of steps in the training simulation n_t. In all experiments, $\mathbf{\Theta}_Z = \mathbf{\Theta}_Y$ and $\mathbf{y}_0 = \mathbf{0}$; in the naive circuit $\mathbf{\Theta}_Z = \mathbf{\Theta}_N$, and in the orthogonal circuit $\mathbf{\Theta}_Z$ is constructed from orthogonal rows which are also orthogonal to $\mathbf{1}$. Where i_e is the epoch, the parameters of the Adam algorithm are $\alpha = 0.00005 \cdot 1.25^{-(i_e-1)}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, and contrastive divergence is run with i_e contrastive divergence steps.

code, because $\mathbf{\Theta}_Y = \mathbf{\Theta}_Z$, the prediction and filtering populations continue to have the same number of neurons. Although we could set the number of neurons in these populations to be different from the number in the observation population, in order to ensure that differences in circuit performance are not simply due to differences in the number of parameters, we continue to define $\mathbf{\Theta}_Z$ to have the same number of columns as $\mathbf{\Theta}_N$. This implies that in all the cases we consider, $d_N = d_Y = d_Z$, which is to say that the observation, prediction, and filtering populations always have the same number of neurons. For the number of neurons in each experiment, and a summary of all simulation parameters, see table 1.

5.1.2 Prediction Network and Training Procedure

In all experiments we define the prediction network $\mathbf{g}$ as a 3-layer perceptron with d_H hidden neurons, where the number of input and output units is determined by the number of neurons in the filtering population. We define the activation functions of the neural network to be a sigmoid activation function in the hidden layer, and an exponential activation function in the output layer. We use the exponential function on the output layer in order to ensure that the rates computed by the prediction network are always positive. This is especially important in the case of the naive code, as negative rates cannot be reliably decoded by $\mathbf{\Theta}_N$.

We denote the parameters of $\mathbf{g}$ by $\boldsymbol{\phi}$, such that $\boldsymbol{\phi}$ is composed of a pair of matrices and biases. We thus compute the gradient in equation 27 by applying backpropagation to compute $\partial_{\phi_i} \mathbf{g}(\mathbf{z}_{k-1})$ (Rumelhart et al., 1986), and

one of the two methods proposed in section 4.3 to compute the expectations in equation 25. Given the proposed neural circuits and gradient approximation schemes, we run four parallel simulations in every experiment, by applying either contrastive divergence minimization (CD) or the exponential family approximation (EF) to computing the stochastic gradient (24), in order to train the neural circuits based on either the naive (NV) or orthogonal (OT) population codes. We denote these four simulations and corresponding circuits by NV-EF, NV-CD, OT-EF, and OT-CD.

In each experiment we train each neural circuit over the course of twenty epochs, where each epoch is composed of a training simulation of n_t steps. During the training simulation, where i_e is the number of the current epoch, we reset the rates of the prediction population Y_k to $\mathbf{0}$ every $(i_e - 1)^2$ number of steps. We do this because newly initialized prediction networks $\mathbf{g}$ tend to be unstable, in that recursively evaluating Z_k for large k tends to result in rates which diverge and fail to encode accurate beliefs. By first training $\mathbf{g}$ on shorter, stable paths, we may avoid this problem and better maximize the likelihood of the parameters $\boldsymbol{\phi}$.

When updating $\boldsymbol{\phi}$, we apply the Adam algorithm in order to dynamically adapt the step size of the gradient descent (Kingma and Ba, 2014). In every epoch we define the initial learning rate to be $\alpha = \frac{0.00005}{1.25^{i_e - 1}}$, we set the momentum parameters of the Adam algorithm to $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the regularizer to $= 10^{-8}$. Finally, when applying contrastive divergence minimization, we also set the number of contrastive divergence steps equal to the epoch number i_e.

5.1.3 Validation

After each training epoch we validate the trained neural circuits on a simulation of $n_v = 200,000$ steps. We compute the sequence of rates $\mathbf{z}_0, \ldots, \mathbf{z}_{n_v}$ as a function of the sequence of validation responses $\mathbf{n}_0, \ldots, \mathbf{n}_{n_v}$ without resetting the rates of the prediction population. Where ε is a function which measures error given a stimulus and the natural parameters of a belief density, we then compute the average of the error measure $E_Z = \sum_{i=0}^{n_v} \frac{\varepsilon(\mathbf{x}_i, \boldsymbol{\Theta}_Z \cdot \mathbf{z}_i)}{n_v}$.

By computing the average error $E_{Opt} = \sum_{i=0}^{n_v} \frac{\varepsilon(\mathbf{x}_i, \boldsymbol{\theta}_k)}{n_v}$ on the belief parameters $\boldsymbol{\theta}_k$ of the closed-form filters described with equation 21, we compute a lower-bound on the error of the trained circuits. Conversely, since any useful filter must provide more information about the stimulus then the instantaneous responses, we compute $E_N = \sum_{i=0}^{n_v} \frac{\varepsilon(\mathbf{x}_i, \boldsymbol{\Theta}_N \cdot \mathbf{n}_i)}{n_v}$ to provide a performance upper-bound. Finally, by computing the ratio $r = \frac{E_Z - E_N}{E_{Opt} - E_N}$, we may express the performance of the neural circuit in question as a percentage of the distance

achieved from the upper- to the lower-bound.

In the first two experiments we validate our model by computing the average negative log-likelihoods of the approximate beliefs given the true stimuli; that is, $\varepsilon(\mathbf{x}, \boldsymbol{\theta}) = -\log q(\mathbf{x})$, where q is the exponential family density described in relation 5 with parameters $\boldsymbol{\theta}$. In these experiments we can compute the true beliefs of the Bayes filter based on knowledge of the stimulus dynamics. The true beliefs have minimal negative log-likelihood, and the smaller the difference between the average negative log-likelihood of the approximate beliefs and true beliefs, the better our neural circuit has implemented a Bayes' filter.

In the third experiment we model a two-dimensional stimulus with a stochastic pendulum, where the first dimension in the angle and the second is the angular velocity. We define the tuning curves of the observation population as the concatenation of a set of von Mises and normal tuning curves, and the exponential family density which corresponds to these tuning curves is the product density of a von Mises density and a normal density. Because von Mises densities cannot be computed in closed-form, we cannot evaluate the corresponding negative log-likelihoods. Therefore, we instead take our error measure to be the average squared error of the mean of the belief density from the true stimulus, such that $\varepsilon(q, \dot{q}, \boldsymbol{\theta}_{vM}, \boldsymbol{\theta}_n) = \frac{(q-\mu_1(\boldsymbol{\theta}_{vM}))^2+(\dot{q}-\mu_2(\boldsymbol{\theta}_n))^2}{2}$, where $(\mu_{vM}(\boldsymbol{\theta}_{vM}), \mu_n(\boldsymbol{\theta}_n))$ is the mean of of the bivariate von Mises-normal density with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_{vM}, \boldsymbol{\theta}_n)$.

Since pendula are nonlinear dynamical systems and the tuning curves are von Mises, we cannot exactly compute the true Bayes filter for the third experiment. An extended Kalman filter (EKF) is an algorithm for approximate filtering of nonlinear dynamical systems, however the predictions of an EKF can only be computed as a function multivariate normal beliefs, not von Mises-normal beliefs. Nevertheless, when the concentration parameter of the von Mises density is high, it is approximately equal to the inverse variance of the density, and the von Mises density is approximately normal. We therefore compute approximate EKF predictions in the following manner.

First note that if the von Mises-normal beliefs at some time are given by some natural parameters $\boldsymbol{\theta}$, then we may equivalently express this density with the parameters $(\mu_{vM}, \kappa, \mu_n, \sigma^2)$, where κ is the concentration of the von Mises density, and σ^2 is the variance of the normal density. We convert this density into a multivariate normal density by setting the mean μ_{EKF} of the multivariate normal to $\mu_{EKF} = (\mu_{vM}, \mu_n)$, and the covariance matrix Σ_{EKF} to a diagonal matrix with first component $\Sigma_{EKF,(1,1)} = 1/\kappa$ and second component $\Sigma_{EKF,(2,2)} = \sigma^2$. After computing the multivariate normal predictions with parameters μ^*_{EKF} and Σ^*_{EKF} as a function of the parameters μ_{EKF} and Σ_{EKF} with the EKF, we convert the multivariate normal predictions back into von

Mises-normal predictions by setting $(\mu^*_{vM}, \mu^*_n) = \mu^*_{EKF}$, $\kappa^* = \Sigma^*_{EKF,(1,1)}$, and $(\sigma^2)^* = \Sigma^*_{EKF,(2,2)}$. After computing the approximate predictions, we apply Bayes' rule by evaluating equation 21 in the usual manner. As we later show, when the concentration of the von Mises density is high, this provides an effective filter for this nonlinear, von Mises-normal system.

Finally, in the last two experiments we also estimate the tuning curves of the hidden layer of the prediction network with respect to the stimuli. We estimate these tuning curves by simulating the trained neural circuits for n_v steps, and then sorting these steps into bins, where each bin contains the activity of the hidden layer of the prediction network when the stimulus is near a particular stimulus value. We then average the rate of each hidden neuron in each bin, in order to estimate the mean activity of the hidden neuron given the stimulus which corresponds to the bin.

5.2 Results

In this section we simulate three experiments which model how a neural circuit learns to implement a Bayes filter. The three stimuli are colour sequences which we model as a finite-state Markov chain; the position of a mouse on a track which we model as a 1-dimensional linear dynamical system; and the angle and angular velocity of a human arm, which we model as a pendulum. In the first experiment the tuning curves of the observation population are designed to ensure that the sum of the tuning curves is exactly constant. In the second experiment the tuning curves are Gaussian, and in the third experiment the tuning curves are a product of a von Mises and a Gaussian tuning curve. In both the second and third experiments the tuning curves tile the space of the stimulus so that the sum of the tuning curves is approximately constant.

5.2.1 Colour Sequence Learning

In this simulated experiment we imagine that subjects are shown sequences of colours drawn from red, green, and blue. The colours are described by a Markov chain, such that each colour has a certain probability of appearing based on the previously seen colour. Subjects must learn to predict the sequence as well as possible. We assume that the stimuli change quickly, so that subjects do not always perceive the stimulus before it transitions to the next stimulus value. We consider how this problem might be solved in the ventral stream, and model colour-sensitive neurons in the visual area V4 with the observation population, and sequence-learning neurons in the inferior temporal cortex with the prediction and filtering populations (Roe et al., 2012).

For simplicity, let us denote the three colour values by r, g, and b. The transition probabilities of the Markov chain are

$$p_{X'|X}(r \mid r) = p_{X'|X}(b \mid b) = 0.8, \qquad p_{X'|X}(g \mid g) = 0.5,$$
$$p_{X'|X}(r \mid g) = p_{X'|X}(b \mid g) = 0.25, \qquad p_{X'|X}(g \mid r) = p_{X'|X}(g \mid b) = 0.15,$$
$$p_{X'|X}(b \mid r) = p_{X'|X}(r \mid b) = 0.05.$$

Intuitively, blue tends to stay blue and red tends to stay red, whereas green is a relatively transitory state. Moreover, red and blue tend to first transition through green before reaching blue and red, respectively.

We assume that the observation population has $d_N = 10$ neurons, and that the gain $\gamma = 1$. We define the tuning curve of neuron i given blue as $f_i(b) = e^{0.4(i-1)-5}$, given red as $f_i(r) = f_{10-i}(b)$ and given green as $f_i(g) = \frac{1}{n}\sum_{i=1}^{10} f_i(b)$. This construction ensures that equation 11 is satisfied exactly. Intuitively, the low-index neurons of the observation population respond to red, the high-index neurons respond to blue, and the observation population responds with a uniform pattern of activity to green, which provides little information about the true colour. Finally, we set the number of hidden neurons in the prediction network to $d_H = 100$.

We depict the results of the simulations of the NV-EF, NV-CD, OT-EF, and OT-CD circuits in figure 6. As displayed in the left panel, the circuit which best approximates the true beliefs of the Bayes filter is the orthogonal circuit trained with the exponential family approximation of the stochastic cross-entropy gradient (solid red), which achieves $r = 95.4\%$ of the performance of the Bayes filter. In this experiment, because equation 11 is exactly satisfied, it is unsurprising that the EF gradient produces the best results, as it is in fact equal to the true stochastic gradient. It is surprising, however, that the choice of population code has such a dramatic effect on the learning. Where the orthogonal circuits more or less completely recover the true beliefs, the naive circuits cannot even surpass the baseline provided by the responses.

In the right panels of figure 6 we display 30 steps of a simulation from this system. In the top right panel we use the opacities of coloured squares to show the dynamic beliefs of both the optimal and OT-EF filter, and one can see how the beliefs of the two filters are nearly identical. In the bottom right panel we show the corresponding population responses. In total, the observation population spikes 25 times over the course of the simulation. Both filters initially recognize that the stimulus is blue. However, in the middle of the simulation when the stimulus changes to red and back to blue again, the filters cannot recognize this, as no spikes reveal this transition.

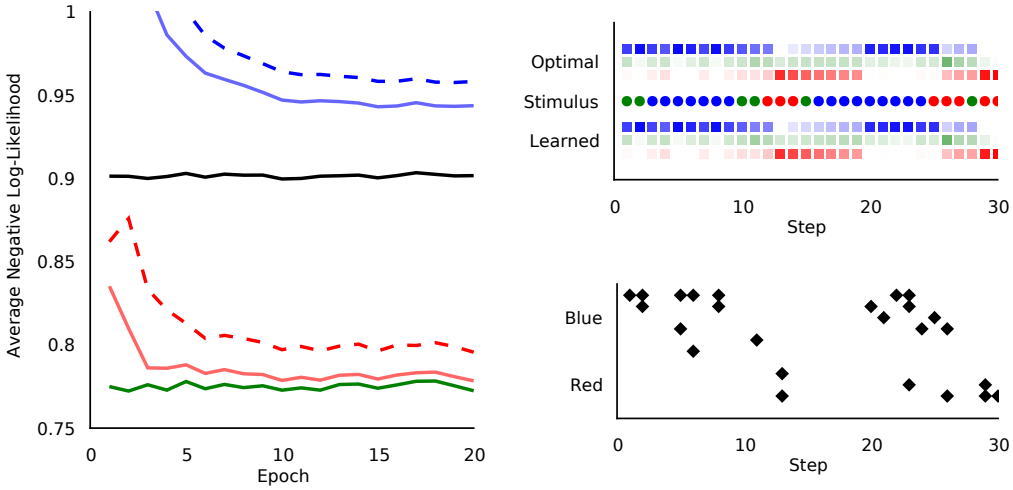

Figure 6: **Colour Sequence Training and Simulation**: Here we depict the training of the proposed neural circuits, and a simulation with the OT-EF circuit. *Left*: The average negative log-likelihood of the approximate beliefs given the stimuli over each epoch. We display the descent of the NV-EF circuit (blue), the NV-CD circuit (dashed blue), the OT-EF circuit (red) and the OT-CD circuit (dashed red). We also depict the baseline (black) provided by the population responses and the optimum (green) computed by the discrete Bayes filter. *Top Right*: A simulation from the Markov chain (coloured circles), as well as the learned an optimal filters. The beliefs of the optimal filter (top) and the learned filter (bottom) are indicated by the opacity of a colour, which corresponds to the inferred probability of the stimulus value. *Bottom Right*: The responses of the observation population over the 30 steps of the simulation. Spikes (black diamonds) from a particular neuron are arranged along the x-axis in accordance with the neuron index.

5.2.2 Self-Localization

In this simulated experiment we imagine that a mouse is confined to a one-dimensional track, and explores the local track while avoiding straying too far from its home position. We model the dynamics of the position of the mouse with a stochastic, one-dimensional, linear dynamical system. We wish to understand how the mouse learns to track its position in a novel environment with place cells in the hippocampus, given noisy position estimates provided by visual cues (McNaughton et al., 2006). We model place cells with the filtering population, and cue-sensitive cells with the observation population.

Since the position of the mouse is a continuous-time variable, we describe it with the linear stochastic differential equation

$$dX_t = aX_t dt + bdW_t,$$

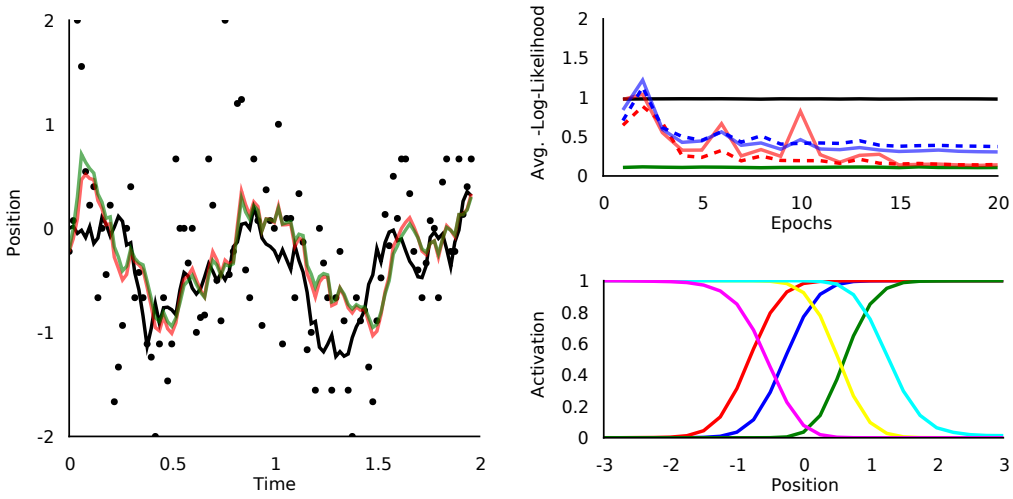

Figure 7: **Self-Localization Training and Simulation**: Here we depict the training of the proposed neural circuits, and simulations with the OT-EF circuit. *Top Right*: The descent of the negative log-likelihood using the same colour scheme as in figure 6. *Left*: A simulation from the dynamical system, where we depict the stimulus (black line), and the dynamic mean of the response posteriors (black dots), the optimal belief density (green line), and the learned belief density (red line). *Bottom Right*: Six tuning curves from the hidden layer of the prediction network.

where W_t is a Wiener process. Where h is the time-step, this implies that the transition density $p_{X'|X}$ of the time-discretized system at x is a normal density with mean $x + hax$ and variance hb^2. In our case we let $a = -1$, $b = 1$, and $h = 0.02$. We then define the observation population to have $d_N = 10$ neurons with the 1-d Gaussian tuning curves defined in equation 4, with gain $\gamma = h \cdot 100 = 2$, preferred stimuli x_i^0 distributed evenly over the interval $[-7, 7]$, and variance $\sigma^2 = 2$. Finally, we set the number of hidden neurons of the prediction network to $d_H = 200$.

We depict the results of the four simulations in figure 7. As depicted in the top right panel, the orthogonal circuit trained with the exponential family approximation (red) best approximates the optimal filter, achieving $r = 96.0\%$ of the performance of the optimal filter, which is slightly better than the OT-CD circuit. In the left panel we display a 2 second simulation from the system. The black dots indicate the mean of the posterior $p_{X|N=\mathbf{n}_k}$ of each response $\mathbf{n}_k$ from equation 8 under the assumption of a flat prior. The mean of the optimal beliefs (green line) given these responses is very close to the true stimulus, and the mean of the learned beliefs (red line) is nearly identical to the optimum. In the bottom right panel we display six approximate tuning curves from the

hidden layer of the trained multilayer perceptron $\mathbf{g}$, which we find to have learned sigmoid tuning curves over the stimuli.

5.2.3 Proprioception

In this final simulated experiment we imagine that a human is trying to optimize its forward model of the swing of its arm. We focus on the role of the cerebellum in proprioception, and assume that Purkinje cells in the cerebellum receive information about the angle and angular velocity of the shoulder from proprioceptors, and use this information to drive a forward model of arm position (Kawato et al., 2003; Franklin and Wolpert, 2011). We model the neural populations in the cerebellum with the prediction and filtering populations, and the proprioceptors with the observation population.

For simplicity, we assume that the arm may be described by a single rigid body at a joint, and that the subjects use random motions of the arm in order to explore its dynamics. We therefore model the arm as a stochastic pendulum, which is a two-dimensional stochastic process over the angular position q, and the angular velocity $\dot{q}$. We define the discrete-time transition dynamics $p_{X'|X}$ of the stochastic pendulum at $\mathbf{x} = (q, \dot{q})$ as a multivariate normal density with mean $\mathbf{x} + h\mathbf{a}(\mathbf{x})$ and covariance matrix $h^2\mathbf{\Sigma}$, where h is the time-step. The function $\mathbf{a}$ is known as the drift, and is given by

$$\begin{aligned} a_1(q, \dot{q}) &= \dot{q}, \\ a_2(q, \dot{q}) &= -g \sin(q) - c\dot{q}, \end{aligned}$$

where $g = 9.81$ is the gravitational constant and $c = 0.1$ is the coefficient of friction. We define the covariance matrix of the process by

$$\begin{aligned} \mathbf{\Sigma}_{(1,1)}(q, \dot{q}) = \mathbf{\Sigma}_{(1,2)}(q, \dot{q}) = \mathbf{\Sigma}_{(2,1)}(q, \dot{q}) = 0, \\ \mathbf{\Sigma}_{(2,2)}(q, \dot{q}) = \sigma_{\dot{q}}^2, \end{aligned}$$

where $\sigma_{\dot{q}}^2 = 1$ is the variance of the noise process. By restricting the noise to the velocity, we may interpret the noise to be the result of the subject applying random forces to its arm. Finally, we define $h = 0.02$.

We define the gain of the emission density $p_{N|X}$ by $\gamma = h \cdot 100 = 2$, and we define the tuning curves of $p_{N|X}$ with two independent sets of tuning curves over the angle and angular velocity, such that half the neurons in the observation population respond to angle, and the other half to angular velocity. Since the angle is periodic, we define the tuning curves over the angle as a set of von Mises tuning curves

$$f_i(q) = e^{\kappa \cos(q - q_i^0)},$$

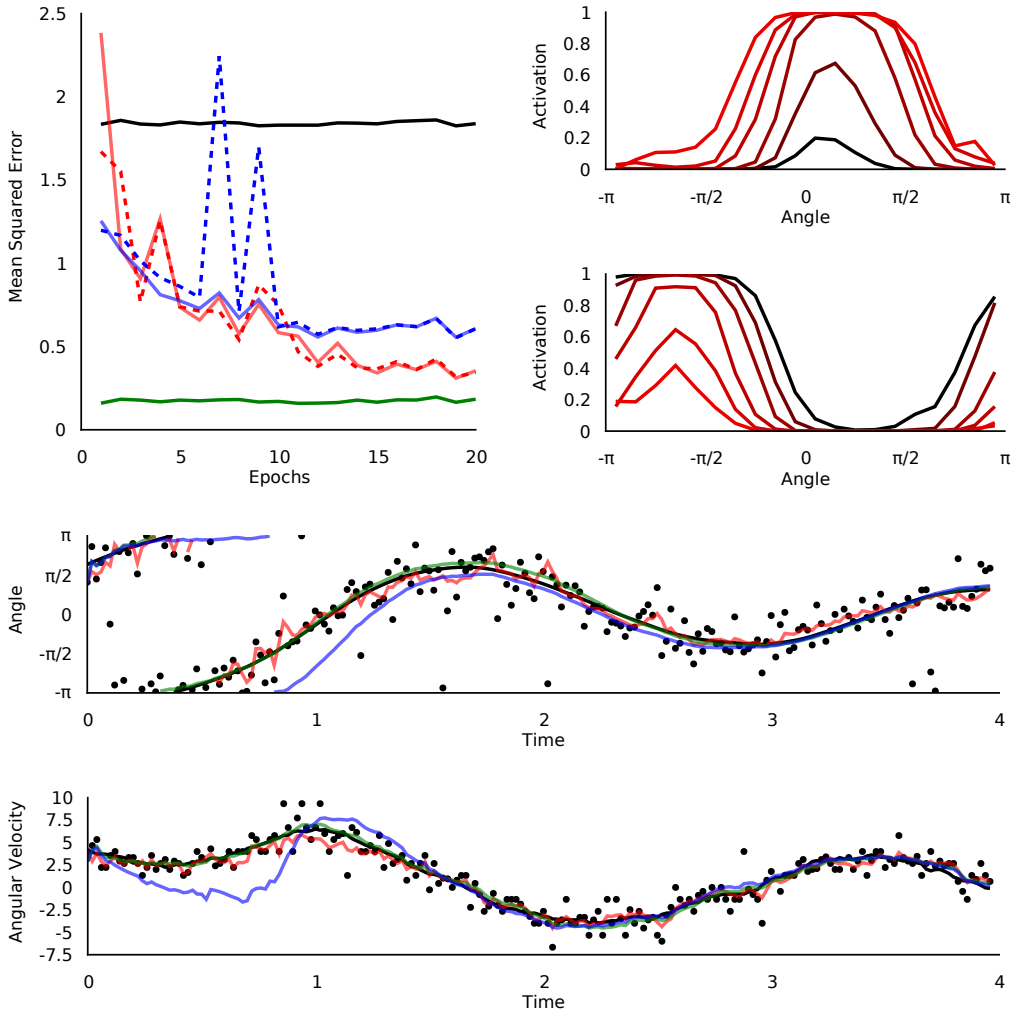

Figure 8: **Proprioception Training and Simulation**: Here we depict the training of the proposed neural circuits, and simulations with the OT-CD circuit. *Top Right*: The descent of the mean squared error of the approximate beliefs using the same colour scheme in figure 6, where green indicates the approximate EKF beliefs. *Top Right*: Two tuning curves from the hidden layer of the multilayer perceptron. The stimulus angle is plotted on the x-axis, and the stimulus angular velocity is indicated by the colours from red to black, where black indicates an angular velocity of -6, and red indicates a velocity of 6. *Bottom*: Simulation of the angle and angular velocity from the dynamical system. We depict the stimulus (black line), and the dynamic mean of the response posteriors (black dots), the approximate EKF belief density (green line), the approximate KF belief density (blue line), and the learned belief density (red line).

with 10 preferred stimuli q_i^0 distributed evenly over the period $[-\pi, \pi]$, and concentration $\kappa = 1/2$. The tuning curves over the angular velocity are again 1-dimensional Gaussian tuning curves as defined in equation 4 with 10 preferred stimuli distributed evenly over the interval $[-12, 12]$, and covariance $\sigma^2 = 4$. The sufficient statistic of the exponential family determined by these tuning curves is $\mathbf{s}(q, \dot{q}) = (\cos q, \sin q, \dot{q}, \dot{q}^2)$. The matrix $\mathbf{\Theta}_N$ is a diagonal block matrix in four quadrants, with the parameters in the upper-left and lower-right quadrants defined by the parameters of the two sets of tuning curves, and with the parameters in the upper-right and lower-left quadrants equal to zero. In total, the neural populations have $d_N = d_Y = d_Z = 20$ neurons, and we set the number of neurons in the hidden layer of the prediction network $\mathbf{g}$ to $d_H = 500$.

We depict the results of the four simulations in figure 8. As displayed in the top left panel, the circuit which best approximates the optimal beliefs, by an extremely slim margin over the OT-EF circuit, is the orthogonal circuit trained with the contrastive divergence minimization (dashed red), which achieves $r = 89.7\%$ of the performance of the approximate EKF. In the lower two panels we depict a 4 second simulation from the system. The black dots, green line, and red line indicate the mean of the response posteriors, the approximate EKF, and the learned beliefs, as in the previous section. The blue line indicates the mean of an approximate Kalman filter with linear dynamics given by the small-angle approximation, and which updates its beliefs with the same strategy as the approximate EKF. As can be seen, a straightforward linear model is not sufficient for tracking the nonlinear stimulus.

In the upper right two panels we depict two tuning curves from the hidden layer of the trained multilayer perceptron $\mathbf{g}$. We plot the two-dimensional tuning curves by plotting the stimulus angle on the x-axis, and indicating the angular velocity with the colour of the line, where black corresponds to -6, and red to 6. As can be seen in these plots, the tuning curve over the angle is a von Mises tuning curve, and the angular velocity is a monotonic function, and the two components interact multiplicatively. Such multiplicative interactions in neural populations are known as gain-fields, and have been widely applied in theory (Zipser et al., 1988; Sejnowski, 1995; Pouget and Sejnowski, 1997) and reported in experiment (Salinas and Thier, 2000; Hwang et al., 2003; Paninski et al., 2004; Herzfeld et al., 2015).

6 Discussion

In this paper we demonstrated how to define and train a theoretical neural circuit to approximately implement a Bayes filter, and thereby encode accurate beliefs about an unknown, dynamic stimulus. As depicted in figure 5,

this neural circuit is composed of three neural populations called the observation population, the prediction population, and the filtering population, as well as a neural network called the prediction network. The observation population generates responses to the stimulus, the prediction population encodes predictions of the stimulus, the filtering population encodes beliefs about the stimulus, and the prediction network computes the rates of the prediction population as a function of the rates of the filtering population.

In our work we assume that the parameters of the three neural populations are fixed, and that our goal is exclusively to optimize the parameters of the prediction network. Towards this end, we derived the negative log-likelihood gradient of the parameters of the prediction network given the responses of the observation population. In addition, we developed a novel approach to approximating this gradient based on the theory of exponential families.

We demonstrated our methods in three simulated experiments. In the first experiment we modelled a sequence learning task and neural populations along the ventral stream, in the second we modelled self-localization and neural populations in the visual cortex and hippocampus, and in the third we modelled proprioception optimization, and shoulder joint receptors and neural populations in the cerebellum. In each experiment we demonstrated how our circuit recovered most of the performance of an optimal, or near-optimal filter. Moreover, we found that the hidden layer of the prediction network developed tuning curves which reproduce well-known experimental findings in the literature.

In concluding this paper we discuss three topics. Firstly, we discuss how our work relates to other algorithms for filtering in machine learning and computational neuroscience. Secondly, we provide a deeper analysis of the models developed and trained in the previous section, and the broader relevance of this work for experimental neuroscience. Thirdly and finally, we discuss how the theory we have developed in this paper could be extended to continuous-time spiking networks, and how it could be applied to modelling noise correlations in the brain.

6.1 Related Computational Work

To begin, let us consider the relationship between the exponential family gradient (EF) we have introduced in this paper and the contrastive divergence gradient (CD) for approximating the negative log-likelihood gradient of the exponential family harmonium (EFH). As can be seen in the gradient descent panels of figures 6, 7, and 8, the EF gradient in general perform as well or better than the CD gradient. Although the advantage is relatively slight, the EF gradient is also much easier to compute – it requires no additional sampling,

and no tuning of the number of contrastive divergence steps. As such, when the EFH in question approximately satisfies equation 11, it is arguable that the EF gradient should be applied.

Our method for approximate Bayesian filtering based on an EFH is related to previous work on approximate filtering with restricted Boltzmann machines (Sutskever et al., 2009; Boulanger-Lewandowski et al., 2012). Although there are many differences in the details, our model can essentially be viewed as a special case of the RNN-RBM model presented in Boulanger-Lewandowski et al. (2012). However, the predictions and beliefs of Bayesian filtering are not clearly separated in the RNN-RBM, and the updating of the dynamic parameters of the RNN-RBM is rather justified as a "mean-field" approximation. In our case, by extending the work on optimal Bayesian inference with probabilistic population codes (Ma et al., 2006), we present conditions under which these updates are exact, leading to equations 15, 16, and 18 for optimally implementing Bayes' rule, and allowing us to describe optimal filters (Beck and Pouget, 2007; Beck et al., 2011) as special cases of our general model. In short, the RNN-RBM is arguably overparameterized, as many of the neural circuits it describes suboptimally implement Bayes' rule.

Another importance difference between our model and the RNN-RBM is that we do not apply backpropagation-through-time (BPTT). It has been argued that BPTT is necessary for these models to learn implicit higher-order temporal structure (Sutskever, 2013; Makin et al., 2016). In particular, in the experiment of section 5.2.3, if the observation population does not respond to the angular velocity of the arm, then BPTT should be required to infer it. Indeed, in our simulations we have found that training a neural circuit to filter responses to a pendulum fails when the observation population does not respond to the angular velocity. Nevertheless, there are proprioceptors for both position and motion (McCloskey, 1978), so this fact does not limit our ability to model proprioception with our circuit, and if it were required, we could in principle apply BPTT to infer the missing state variables.

Another model related to our own is the rEFH presented in Makin et al. (2015), which also depends on an EFH trained with contrastive divergence minimization to approximate a Bayes filter based on the responses of a dynamic Poisson population. Despite these similarities, however, there are important differences between the rEFH and our approach. At every time k, the rEFH optimizes the joint density of the rates of the filtering population Z_k and the concatenated vector (N_k, Z_{k-1}) of the response of the observation population and the previous rates of the filtering population, whereas in our circuit and the RNN-RBM it is the conditional probability of N_k given Z_{k-1} that is optimized. Moreover, in the rEFH architecture it is this joint density over Z_k and (N_k, Z_{k-1}) that is modelled as an EFH (Makin et al., 2013), whereas in our circuit the EFH

is over the stimuli X_k and responses N_k.

Because the optimization problem is based on the joint density of Z_k and (N_k, Z_{k-1}), the structure of the rEFH circuit is more strict, and so cannot, for example, incorporate a multilayer perceptron for computing predictions. Moreover, it is not clear if the rEFH can exactly implement Bayes' rule, or the optimal solutions discussed in sections 3.3 and 3.4. Nevertheless, as reported in Makin et al. (2016), the rEFH can learn to infer velocities without direct observation, and without using BPTT. Moreover, in our circuit the gradient descent procedure involves computing expectations in the space of the stimulus directly, which is difficult to justify biologically, whereas the rEFH applies Gibbs sampling between two neural populations, which is less problematic. As such, the training procedure of the rEFH is more biologically realistic Makin et al. (2015), and choosing either our circuit or the rEFH comes down to a trade-off between a flexible prediction network and theoretical exactness on one hand, and a flexible and biologically realistic learning rule on the other.

6.2 Neuroscience Applications

The three simulated experiments we presented in this paper were kept theoretically simple so that they could be validated against optimal models, yet they can easily be extended to more complex and realistic experimental designs as required. We consider simulation 5.2.1 in its current form to constitute a sound experiment, as such a sequence learning experiment could easily be performed with real subjects, allowing hypothetical networks and circuits to be compared against subject performance and recorded neural activity. The self-localization experiment of section 5.2.2 could be expanded to 2-dimensional place cells by applying 2-d Gaussian tuning curves, and could incorporate models of spatially periodic grid cells in the entorhinal cortex (Moser et al., 2008; Giocomo et al., 2011) through the application of von Mises tuning curves. Finally, our work can trivially be extended to include control variables, which would allow the proprioception task in section 5.2.3 to depend on motor commands and efference copies (Thrun et al., 2005; Särkkä, 2013; Makin et al., 2015).

One surprising result of our simulations was the dramatic effect that the choice of population code can have on learning. We considered the naive circuit which uses the same population code across the observation, prediction, and filtering populations, and the orthogonal circuit, which uses the code presented in the supplementary material of Beck et al. (2011) for the prediction and filtering populations. In the self-localization and proprioception experiments, the naive circuit performs reasonably well, though not nearly as well as the orthogonal circuit. In the sequence learning experiment, however, the naive circuit fails to even achieve the upper-bound on the error provided by the

instantaneous information in the responses. This finding is in line with computational (Boulanger-Lewandowski et al., 2012) and experimental (Chang and Snyder, 2010) evidence which suggests that diverse population codes improves performance in neural circuits.

Moreover, an important feature which distinguishes these two codes is the importance of the sum of the rates of the population. When the sum of the rates of the observation population is constant, then the sum of the rates is also proportional to the precision of the encoded density (Ma et al., 2006; Beck et al., 2007). In the naive circuit, this implies that the sums of the rates of the prediction and filtering populations are also proportional to the precision of the encoded densities, and since the rates of the three neural populations are always positive, this enforces a trade-off between encoding accurate beliefs and adding too much precision when adjusting the rates of the prediction and filtering populations. In the orthogonal circuit, however, because the rows of the decoding matrix are mutually orthogonal and orthogonal to the vector of ones, , the parameters of the encoded density may be adjusted independently, and the magnitude of the sum of the rates of the filtering population does not influence the encoded beliefs. Evidently, this provides a much better code for learning to implement a Bayes filter.

We have also shown that the hidden layer of the prediction network learns tuning curves over stimuli. In the self-localization experiment (5.2.2), training the network resulted in sigmoid tuning curves over the unobserved stimuli. Although sigmoid tuning curves are often found in the brain (Pouget and Sejnowski, 1997; Pouget et al., 2000), to the best our knowledge, sigmoid tuning curves for self-location have not been found in the limbic system. Because it is a linear circuit with no hidden activity, the optimal circuit described in equation 22 could avoid this discrepancy. In our experiments, however, we found that a nonlinear network is required for learning stable neural circuits, and that we could not successfully train a neural circuit based on a linear prediction network. Moreover, self-localization in general is a nonlinear problem, for which a linear prediction network would in any case not suffice.

Although it could be the case that there exists an as of yet undiscovered neural population in the limbic system with tuning curves which match those of our learned hidden layer, we rather suspect that the model neural circuit which we tested fails to capture essential features of the self-localization circuitry, and that the sigmoid tuning curves are a result of this. The exact structure and connectivity of recurrent connections in the hippocampus and entorhinal cortex remains a highly active area of research, and we believe that our work can contribute to this research by providing a general framework for exploratory modelling. By matching the observation, prediction, and filtering populations, as well as the prediction network, to hypotheses about neural circuitry, the re-

sulting performance and internal structure of the circuit can serve to validate the hypothesis in question. In our simple case, our experiment emphasizes that a local neural network is insufficient for explaining the activity of hippocampal place cells, and that a more realistic neural circuit must incorporate additional neural circuitry, for example from the entorhinal cortex.

In simulation 5.2.3, training the neural circuit resulted in von Mises tuning curves over the angle, and sigmoid tuning curves over the angular velocity, which interact via multiplication. When the tuning curve over one stimulus interacts multiplicatively with the tuning curve over another stimulus, it is known as a gain-field or gain modulation (Salinas and Thier, 2000), and gain-fields have been found in many areas of the brain (Salinas and Thier, 2000; Hwang et al., 2003; Paninski et al., 2004). In particular, Herzfeld et al. (2015) demonstrated that eye position and velocity is encoded by Purkinje cells in the cerebellum with the same gain-field structure as in our arm-localization circuit. Although the stimuli in this experiment and our simulated experiment are different, both respective neural circuits must ultimately predict the motion of parts of the body, and they do so in similar manners.

It is well-known that given data which match population activity for encoding stimuli, gain-fields can arise spontaneously in the hidden layer of multilayer perceptrons (Zipser et al., 1988). At the same time, Gaussian/sigmoid gain-fields have been used to model the neural computation of coordinate transformations in the posterior parietal cortex (Sejnowski, 1995; Pouget and Sejnowski, 1997), and were found to be especially apt for computing addition over the encoded variables. In our proprioception experiment, although our neural circuit is not performing a coordinate transformation per se, it is learning to add velocity to position at every time, and therefore the emergence of this particular gain-field fits well into the existing theory. What is novel in our work however, is that our neural network is not trained solve a standard regression problem as in Zipser et al. (1988), but is trained rather as part of a more complex neural circuit for implementing a Bayes filter. As we have demonstrated, gain-fields continue to emerge in this context.

In our simulated experiments we implemented the prediction networks with multilayer perceptrons for performance reasons. Nevertheless, the maximum likelihood approach we have presented in this paper can be applied to any parameterized network, and it is entirely possible to apply our method to optimize the parameters of more biologically plausible prediction networks and thereby validate more realistic neural circuit models. To reiterate, our theory is not dependent on multilayer perceptrons, but rather only populations of Poisson-spiking neurons and probabilistic population codes, which are well-established for explaining the activity of populations of neurons (Dayan and Abbott, 2005; Pouget et al., 2013).

6.3 Future Directions

In concluding our paper we discuss two ways in which we hope to extend and apply our work in the future. In particular, we discuss how to model a continuous-time, spiking neural circuit with our methods, and how our model might be applied to understanding noise correlations in the brain.

Although we described the neural circuit for linear Bayesian filtering presented in Beck et al. (2011) as a special case of our model, the circuit in Beck et al. (2011) is both a spiking and continuous-time circuit, which ours is not. Nevertheless, extending our neural circuit to be spiking and continuous-time is relatively straightforward. On one hand, as shown in Beck et al. (2011), generating spikes from the rates of the prediction and filtering populations and using them as the exclusive basis for neural communication ultimately results in little loss of information. On the other hand, a parameterized network with the form of 22 can in principle be optimized by our method, and the linear transformations could be made nonlinear for stability and more expressive power. In unpublished work we have done exactly this, and the initial results are promising. Nevertheless, there are details and pitfalls specific to the training of such a continuous-time circuit, which are beyond the scope of this paper.

Understanding noise correlations in neural populations is a major research area in neuroscience, as they represent a breakdown of the simple understanding of neurons as independent Poisson processes, and affect the efficacy of neural coding (Averbeck et al., 2006). The only source of noise in our neural circuit is in the observation population, which is indeed a population of independent Poisson neurons, and the rates of the prediction and filtering populations are only random by virtue of being functions of the observation population. This might suggest that our theoretical neural circuit cannot model noise correlations, however recent research has shown that many patterns of noise correlation in the brain can be explained by correlations resulting from downstream responses to sensory noise (Kanitscheider et al., 2015). In initial simulations we have indeed found that the prediction and filtering populations in our neural circuits exhibit significant noise correlations, and in the future we hope to use our neural circuit model to explore the extent to which noise correlations in dynamic neural populations can be explained as the result of sensory noise.

Acknowledgements

This work was partially funded by the DFG Priority Program 1527, Autonomous Learning. The author would like to thank Nihat Ay, Guido Montufar, Keyan Zahedi, and Anna Erzberger, for their comments, advice, and support.

References

Amari, S.-i. and Nagaoka, H. (2007). *Methods of information geometry*, volume 191. American Mathematical Soc.

Arnold, B. C., Castillo, E., Sarabia, J. M., and others (2001). Conditionally specified distributions: an introduction (with comments and a rejoinder by the authors). *Statistical Science*, 16(3):249–274.

Arnold, B. C. and Press, S. J. (1989). Compatible conditional distributions. *Journal of the American Statistical Association*, 84(405):152–156.

Averbeck, B. B., Latham, P. E., and Pouget, A. (2006). Neural correlations, population coding and computation. *Nature Reviews Neuroscience*, 7(5):358–366.

Beck, J., Ma, W. J., Latham, P. E., and Pouget, A. (2007). Probabilistic population codes and the exponential family of distributions. *Progress in Brain Research*, 165:509–519.

Beck, J. M., Latham, P. E., and Pouget, A. (2011). Marginalization in Neural Circuits with Divisive Normalization. *The Journal of Neuroscience*, 31(43):15310–15319.

Beck, J. M. and Pouget, A. (2007). Exact inferences in a neural implementation of a hidden Markov model. *Neural computation*, 19(5):1344–1361.

Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127.

Bengio, Y. and Delalleau, O. (2009). Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236.

Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*.

Bowers, J. S. and Davis, C. J. (2012). Bayesian just-so stories in psychology and neuroscience. *Psychological bulletin*, 138(3):389.

Bubic, A., Von Cramon, D. Y., Schubotz, R. I., Bubic, A., Cramon, D. Y. v., and Schubotz, R. I. (2010). Prediction, cognition and the brain. *Frontiers in Human Neuroscience*, 4:25.

Chang, S. W. C. and Snyder, L. H. (2010). Idiosyncratic and systematic aspects of spatial representations in the macaque parietal cortex. *Proceedings of the National Academy of Sciences*, 107(17):7951–7956.

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(03):181–204.

Clegg, B. A., DiGirolamo, G. J., and Keele, S. W. (1998). Sequence learning. *Trends in cognitive sciences*, 2(8):275–281.

Coen-Cagli, R., Kohn, A., and Schwartz, O. (2015). Flexible gating of contextual influences in natural vision. *Nature Neuroscience*, 18(11):1648–1655.

Dayan, P. and Abbott, L. F. (2005). Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.

Doya, K. (2007). *Bayesian brain: Probabilistic approaches to neural coding*. MIT press.

Ernst, M. O. and Banks, M. S. (2002). Humans integrate visual and haptic information in a statistically optimal fashion. *Nature*, 415(6870):429–433.

Fetsch, C. R., Pouget, A., DeAngelis, G. C., and Angelaki, D. E. (2011). Neural correlates of reliability-based cue weighting during multisensory integration. *Nature Neuroscience*, 15(1):146–154.

Fischer, B. J. and Peña, J. L. (2011). Owl's behavior and neural representation predicted by Bayesian inference. *Nature Neuroscience*, 14(8):1061–1066.

Franklin, D. W. and Wolpert, D. M. (2011). Computational mechanisms of sensorimotor control. *Neuron*, 72(3):425–442.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741.

Giocomo, L., Moser, M.-B., and Moser, E. (2011). Computational Models of Grid Cells. *Neuron*, 71(4):589–603.

Herzfeld, D. J., Kojima, Y., Soetedjo, R., and Shadmehr, R. (2015). Encoding of action by the Purkinje cells of the cerebellum. *Nature*, 526(7573):439–442.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Hwang, E. J., Donchin, O., Smith, M. A., and Shadmehr, R. (2003). A Gain-Field Encoding of Limb Position and Velocity in the Internal Model of Arm Dynamics. *PLOS Biol*, 1(2):e25.

Jaynes, E. T. (2003). *Probability theory: the logic of science*. Cambridge university press.

Kanitscheider, I., Coen-Cagli, R., and Pouget, A. (2015). Origin of information-limiting noise correlations. *Proceedings of the National Academy of Sciences*, 112(50):E6973–E6982.

Kawato, M., Kuroda, T., Imamizu, H., Nakano, E., Miyauchi, S., and Yoshioka, T. (2003). Internal forward models in the cerebellum: fMRI study on grip force and load force coupling. *Progress in brain research*, 142:171–188.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knill, D. C. and Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719.

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

Makin, J. G., Dichter, B. K., and Sabes, P. N. (2015). Learning to Estimate Dynamical State with Probabilistic Population Codes. *PLoS Comput Biol*, 11(11):e1004554.

Makin, J. G., Dichter, B. K., and Sabes, P. N. (2016). Recurrent Exponential-Family Harmoniums without Backprop-Through-Time. *arXiv:1605.05799 [cs, stat]*. arXiv: 1605.05799.

Makin, J. G., Fellows, M. R., and Sabes, P. N. (2013). Learning Multisensory Integration and Coordinate Transformation via Density Estimation. *PLoS Comput Biol*, 9(4):e1003035.

McCloskey, D. I. (1978). Kinesthetic sensibility. *Physiological Reviews*, 58(4):763–820.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., and Moser, M.-B. (2006). Path integration and the neural basis of the 'cognitive map'. *Nature Reviews Neuroscience*, 7(8):663–678.

Meyer, T. and Olson, C. R. (2011). Statistical learning of visual transitions in monkey inferotemporal cortex. *Proceedings of the National Academy of Sciences*, 108(48):19401–19406.

Miall, R. C. and Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural networks*, 9(8):1265–1279.

Moser, E. I., Kropff, E., and Moser, M.-B. (2008). Place Cells, Grid Cells, and the Brain's Spatial Representation System. *Annual Review of Neuroscience*, 31(1):69–89.

Nielsen, F. and Garcia, V. (2009). Statistical exponential families: A digest with flash cards. *arXiv:0911.4863 [cs]*.

Paninski, L., Shoham, S., Fellows, M. R., Hatsopoulos, N. G., and Donoghue, J. P. (2004). Superlinear population encoding of dynamic hand trajectory in primary motor cortex. *The Journal of neuroscience*, 24(39):8551–8561.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Pouget, A., Beck, J. M., Ma, W. J., and Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. *Nature Neuroscience*, 16(9):1170–1178.

Pouget, A., Dayan, P., and Zemel, R. (2000). Information processing with population codes. *Nature Reviews Neuroscience*, 1(2):125–132.

Pouget, A., Dayan, P., and Zemel, R. S. (2003). Inference and computation with population codes. *Annual review of neuroscience*, 26(1):381–410.

Pouget, A. and Sejnowski, T. J. (1997). Spatial transformations in the parietal cortex using basis functions. *Journal of cognitive neuroscience*, 9(2):222–237.

Roberts, G. O. and Polson, N. G. (1994). On the geometric convergence of the Gibbs sampler. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 377–384.

Roe, A., Chelazzi, L., Connor, C., Conway, B., Fujita, I., Gallant, J., Lu, H., and Vanduffel, W. (2012). Toward a Unified Theory of Visual Area V4. *Neuron*, 74(1):12–29.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Salinas, E. and Thier, P. (2000). Gain modulation: a major computational principle of the central nervous system. *Neuron*, 27(1):15–21.

Sejnowski, A. P. T. J. (1995). Spatial representations in the parietal cortex may use basis functions. *Advances in Neural Information Processing Systems 7*, 7:157.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory.

Sutskever, I. (2013). *Training recurrent neural networks*. PhD thesis, University of Toronto.

Sutskever, I., Hinton, G. E., and Taylor, G. W. (2009). The Recurrent Temporal Restricted Boltzmann Machine. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1601–1608. Curran Associates, Inc.

Särkkä, S. (2013). *Bayesian filtering and smoothing*. Number 3. Cambridge University Press.

Talbott, W. (2015). Bayesian Epistemology. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Summer 2015 edition.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Todorov, E. and Jordan, M. I. (2002). Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2004). Exponential family harmoniums with an application to information retrieval. In *Advances in neural information processing systems*, pages 1481–1488.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Zemel, R. S., Dayan, P., and Pouget, A. (1998). Probabilistic interpretation of population codes. *Neural computation*, 10(2):403–430.

Zipser, D., Andersen, R. A., et al. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, 331(6158):679–684.