Implementing a Bayes Filter in a Neural Circuit: The Case of Unknown, Nonlinear 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 unknown stimuli which caused them. The Bayesian solution to this problem is known as a Bayes filter, and previous work has shown how to exactly implement a Bayes filter in a theoretical neural circuit when the stimulus dynamics are known and linear. In this paper we develop a method for approximating a Bayes filter when the stimulus dynamics are unknown and nonlinear, by training a recurrent neural network to approximate the predictions of the Bayes filter. To train the network, we use a combination of contrastive divergence minimization and backpropagation, in order to maximize the likelihood of the parameters of the network given the population responses. We demonstrate this method on a problem where the stimulus is a stochastic pendulum, and show how the learned network displays many of the characteristic properties found in research on populations of neurons.

1 Introduction

Although our conscious world is filled with vivid colours, clear lines, and distinct objects, this apparent precision and clarity is merely inferred from the stochastic responses of neural populations. Moreover, this inference problem may not be reduced to inferring the stimuli which triggered a population response at a particular time, because sometimes what we are attempting to infer is hardly triggering our senses at all. For example, we know from visual neuroscience that our perception of a scene is constructed dynamically from the point-like samples provided by our fovea, and yet we do not lose track of
objects upon which we are no longer foveating. Similarly, a rustle in the grass is sufficient to inform a particular prey to the presence of a predator, and allows the prey to estimate how near the predator might be at a later time, even when no trace of the predator remains.

This inference problem is formally known as the filtering problem, and the Bayesian solution to the filtering problem is known as a Bayes filter (Särkkä, 2013). In the case of neural systems, a Bayes filter is a recursive process which computes beliefs about the stimulus over time based on the stochastic responses of a neural population. At every time, a Bayes filter proceeds by transforming beliefs about the stimulus into predictions about the subsequent stimulus based on knowledge of the stimulus dynamics, and applying Bayes rule to combine these predictions with the subsequent population response (Beck and Pouget, 2007; Bobrowski et al., 2009).

Probabilistic population codes are stochastic models of how populations of neurons respond to stimuli (Zemel et al., 1998; Doya, 2007; Pouget et al., 2013), and population codes may be generalized to dynamic population codes by applying them to a dynamic stimulus. If the stimulus dynamics of a dynamic population code are known and linear, then it is possible to define a second neural population such that the firing rates of the second population form dynamic parameters of the belief distributions computed by a Bayes filter (Deneve et al., 2007; Beck et al., 2011). Although these techniques have been applied to modelling some computations performed by the brain, animals do not generally know the dynamics of the stimuli they are attempting to estimate, nor can they assume that these dynamics are linear. As such, this work is not sufficient for explaining how animals solve the filtering problem.

The goal of this paper is to generalize the application of population code filters to systems where the stimulus dynamics are nonlinear and unknown, towards a more complete model of filtering in neural systems. In particular, we develop a coupled Markov process composed of a dynamic population code and a population rate process, where the dynamics of the rate process are determined by a recurrent neural network which computes approximations of the predictions of a Bayes filter. By taking advantage of the exponential family structure of population codes (Welling et al., 2004; Beck et al., 2007), we then show how to combine contrastive divergence minimization (Hinton, 2002) and backpropagation (Rumelhart et al., 1986) to maximize the likelihood of the parameters of the network given the responses of the dynamic population code.

In order to demonstrate our method, we train a neural network to predict a dynamic population code which responds to a stochastic pendulum, and

---

1In Bayesian inference, probabilities are interpreted as degrees of belief.
show how the learned rate process successfully approximates a Bayes filter. Moreover, we demonstrate that the gain-invariance properties of population codes (Salinas and Thier, 2000; Beck et al., 2011) allow the rate process to successfully estimate the stimulus even when the amount of information in the population responses is varied over time. Finally, we show that the rate process is composed of a stable hill of activity in a topographic representation of the rates, as is often found in population code research (Pouget et al., 2000; Deneve et al., 2007).

2 Coding Theory

In this section we introduce probabilistic population codes, as well as the key mathematical properties of population codes which form the foundation of the work in this paper. In particular, we demonstrate the relation between population codes and the family of machine learning models known as exponential family harmoniums (Smolensky, 1986; Welling et al., 2004), and how Bayes rule may be computed as a linear combination of a population response and a prior parameterized by a rate vector.

The linear solution to Bayes rule was originally derived in Ma et al. (2006), however the relationship between population codes and exponential family harmoniums has not, to our knowledge, been previously explored. The mathematical relationship itself is more or less trivial, and has been strongly implied by previous work (Beck et al., 2007). However, as we later demonstrate, the consequence of this relationship is that we may apply machine learning algorithms to gather learning statistics from the population code, allowing us to develop a novel approach to modelling how neural systems learn to solve the filtering problem.

2.1 Population Codes

A population code is a pair of random variables \((X, N)\), where \(X\) is a random stimulus and \(N\) is a random response. Conditioned on a particular stimulus \(x\), each component response \(N_i\) of the population code is Poisson distributed with mean \(\gamma f_i(x)\), where \(f_i(x)\) is the tuning curve of that component, and \(\gamma\) is the population gain. The component responses are conditionally independent of each other given the stimulus, such that the conditional density of the complete response given the stimulus may be written

\[
p_{N|X}(n | x) = \prod_{i=1}^{d_N} p_{N_i|X}(n_i | x) = \prod_{i=1}^{d_N} e^{-\gamma f_i(x)} \frac{(\gamma f_i(x))^{n_i}}{n_i!}, \tag{1}
\]

\(\text{(1)}\)
Figure 1: Population Codes: These plots demonstrate stochastic encoding and decoding with population codes. **Top Left:** The rates of ten Gaussian tuning curves with uniformly distributed preferred stimuli, and gain $\gamma = 2$. **Top Right:** The components of the response $n_i$ (black dots) to the stimulus 0 (black line) generated with $\gamma = 2$, and the resulting decoding density $p_{X|N=n_i}$ (red line). **Bottom Left:** Response $n_2$, generated with $\gamma = 4$. **Bottom Right:** The summed response $n_3 = n_1 + n_2$, which, as predicted by Bayes rule, is a more accurate encoding of the stimulus than either $n_1$ or $n_2$.

where $d_N$ is the number of neurons in the population. We refer to this conditional density as the encoding density.

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

$$f_i(x) = e^{-\frac{(x-x_0^i)^2}{2\sigma^2}},$$

with preferred stimuli $x_0^i$ (figure 1: Top Left). The theory we develop may nevertheless be generalized to multivariate Gaussian and von Mises tuning curves by following derivations analogous to the 1-dimensional Gaussian case.

For many populations of neurons, the sum of the component responses $\sum_{i=1}^{d_N} N_i$ is approximately independent of the stimulus $X$ itself. This leads naturally to defining population codes with a stimulus-independent gain parameter, which scales the rates of the component neurons (Salinas and Thier, 2000). Now, for the total rate $\sum_{i=1}^{d_N} \gamma f_i(x)$ to be independent of the stimulus, the tuning curves must satisfy

$$\sum_{i=1}^{d_N} f_i(x) = \alpha$$

4
for some constant $\alpha$. It turns out that for many families of tuning curves, if one distributes the preferred stimuli uniformly over the space of the stimulus, then $\sum_{i=1}^{d_N} f_i(x)$ converges to a constant as the number of the preferred stimuli is increased (Ma et al., 2006). This allows us to define population codes which approximately satisfy equation 3 to arbitrary degrees of precision.

### 2.2 Exponential Family Harmoniums

Although the population code parameters are sufficient for determining the encoding density $p_{N|X}$, they are not sufficient for determining the decoding density $p_{X|N}$. One way to specify $p_{X|N}$ is to compute Bayes rule

$$p_{X|N}(x | n) \propto p_{N|X}(n | x)p_X(x)$$ (4)

with respect to a given prior density $p_X$. Now, for arbitrary priors $p_X$, the posterior $p_{X|N}$ will not have a closed-form expression. However, if we re-express the encoding density $p_{N|X}$, restrict the form of $p_X$, and assume that the tuning curves of the population code $(X, N)$ satisfy equation 3, then we may arrive at a closed-form expression for $p_{X|N}$ of sufficient generality.

For many families of tuning curves, the encoding density (1) may be expressed in the log-linear form

$$p_{N|X}(n | x) \propto e^{s(x) \cdot \Theta_N \cdot n + n \cdot \theta_N}$$ (5)

where $s$ is the sufficient statistic of an appropriate family of distributions. 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 $\sigma$, then we may satisfy relation 5 by setting the elements of the matrix $\Theta_N$ equal to

$$\theta_{N,1i} = \frac{x_i^0}{\sigma^2}, \quad \theta_{N,2i} = -\frac{1}{2\sigma^2},$$ (6)

and the elements of the vector $\theta_N$ equal to

$$\theta_{Ni} = \log \gamma - \frac{(x_i^0)^2}{2\sigma^2},$$ (7)

and by letting $s(x) = (x, x^2)$, which is the sufficient statistic of a normal distribution.

If we restrict ourselves to prior densities of the form $p_X(x) \propto e^{s(x) \cdot \theta_X}$ for an arbitrary vector $\theta_X$, then by applying Bayes rule (4) to this prior and the log-linear form of the encoding density (5), we may express the decoding density at $n$ as

$$p_{X|N}(x | n) \propto e^{s(x) \cdot \Theta_N \cdot n + s(x) \cdot \theta_X - \gamma \sum_{i=1}^{d_N} f_i(x)}.$$
Figure 2: Harmoniums: Here we depict the graphical model of an exponential family harmonium. From this graph we may infer that the component responses $N_1$, $N_2$, and $N_3$ are mutually independent given the component stimuli $X_1$ and $X_2$, and that $X_1$ is independent of $X_2$ given $N_1$, $N_2$, and $N_3$.

Therefore, if the tuning curves of the population code satisfy equation 3, then the decoding density is given by

$$p_{X|N}(x \mid n) \propto e^{s(x) \cdot \theta_N \cdot n + s(x) \cdot \theta_X},$$

such that $p_{X|N}$ also has a log-linear form. The family of densities to which $p_{X|N}$ belongs is determined by the sufficient statistic $s$ in relation 5, and in the case of the 1-dimensional Gaussian tuning curve (2), the decoding density is a conditional normal density (figure 1: Top Right, Bottom Left).

Under fairly general conditions, two conditional densities $p_{X|Y}$ and $p_{Y|X}$ derived from a joint density $p_{XY}$ are sufficient for determining the joint density itself (Arnold and Press, 1989; Arnold et al., 2001). In our case, given the parameters $\theta_Y$ and $\Theta_N$ of the population code $(X, N)$, and the parameters $\theta_X$ of the prior $p_X(x) \propto e^{s(x) \cdot \theta_X}$, if $p_{X|N}$ satisfies relation 8, then the joint density of the population code is given by

$$p_{XN}(x, n) \propto e^{s(x) \cdot \theta_N \cdot n + s(x) \cdot \theta_X + n \cdot \theta_N} \frac{1}{n_1! \cdots n_d!},$$

because the conditional densities of $p_{XN}$ are equal to the encoding and decoding densities defined in relations 5 and 8, respectively (Welling et al., 2004). In the machine learning literature, models with the quadratic form of $p_{XN}$ are known as exponential family harmoniums (figure 2).

An exponential family harmonium is a generalization of the well-known restricted Boltzmann machine (see Bengio, 2009), and affords maximizing the likelihood of the model parameters by contrastive divergence minimization (Hinton, 2002). In our case, if we maximize the likelihood of $\theta_X$ given a set of population responses, then we may indirectly maximize the likelihood of the learned prior density given the unobserved stimuli. We later apply this technique for approximating the priors of a Bayes filter.
2.3 Linear Bayes Rule

The principles of Bayesian inference may be applied to maintaining beliefs about the stimulus given a sequence of population responses. In this paper, the likelihood $p_{N|X}$ in Bayes rule (4) is given by the encoding density of a population code. In order to implement Bayes rule in a theoretical neural circuit, we describe how the prior $p_X$ is encoded by a neural population, and how $p_X$ and the population response are combined to compute the posterior $p_{X|N}$.

Let us imagine a population of neurons with firing rates represented by the random rate vector $Z$ of dimension $d_Z$, where the component rates $Z_i$ are positive real values. We define this population with tuning curves given by the matrix parameters $\Theta_Z$, and we define the decoding density of the rate vector $z$ by

$$p_{X|Z}(x | z) \propto e^{s(x) \cdot \Theta_Z \cdot z}, \quad (10)$$

which is equivalent to the decoding density $p_{X|N}$ of a population code where the prior density is given by $\theta_X = 0$.

In order to define a neural circuit for computing the posterior in relation 4, we assume that the matrix parameters of the population code $(X, N)$ and the random rates $Z$ are related by

$$\Theta_N = \Theta_Z \cdot A \quad (11)$$

for some matrix $A$. Different values for $A$ allow the neural populations of $(X, N)$ and $Z$ to differ in size and tuning curve structure, while maintaining a linear relationship. At the same time, it easier to visualize the relationship between the two neural populations when $A$ is the identity matrix, and so we assume that $A$ is the identity matrix for the simulations we present in this paper.

Suppose we are given a rate vector $z$ and a response $n$. If the tuning curves of the population code satisfy equation 3, and the prior is given by the parameters $\theta_X = \Theta_Z \cdot z$, then

$$p_{X|N}(x \mid n) \propto e^{s(x) \cdot \Theta_N - n + s(x) \cdot \Theta_Z \cdot z} = e^{s(x) \cdot \Theta_Z \cdot (z + A \cdot n)},$$

which implies that

$$p_{X|N}(x \mid n) = p_{X|Z}(x \mid z + A \cdot n). \quad (12)$$

Therefore, in the context of population codes, the application of Bayes rule may be approximately reduced to the linear combination of a rate vector and a response (Ma et al., 2006) (figure 1: Bottom Right).
A prior is known as a conjugate prior if the combination of the likelihood and prior in Bayes rule allows the posterior to be expressed in the same parametric form as the prior (Wainwright and Jordan, 2008). Conjugate priors allow Bayes rule to be iteratively applied while maintaining a posterior with a specified form. In our case, the conjugate priors encoded by rate vectors allow us to construct a rate process for a fixed number of neurons which encodes a dynamic posterior over the stimuli.

3 Dynamics

A Bayes filter processes observations of a given dynamical system, and so before we may develop a Bayes filter, we must formalize how to generate these observations. A hidden Markov model is defined by a transition density over hidden states – in our case, the stimulus – and an emission density which describes how to generate observations of these hidden states. In order to define a generic hidden Markov model for observations in the form of the responses of a neural population, we let the emission density be the encoding density given by relation 5, and let the transition density be a normal density with a dynamic mean and variance.

Given a history of population responses, a Bayes filter recursively computes the posterior beliefs about the stimulus at time \( t + h \) by computing prior beliefs about the stimulus at time \( t + h \) as a function of the posterior beliefs at time \( t \), and combining these prior beliefs with the response at time \( t + h \) using Bayes rule (Särkkä, 2013). These prior beliefs can be thought of as predictions, in the sense that they estimate the stimulus at time \( t + h \) as a function of the posterior beliefs at time \( t \), but independently of the response at time \( t + h \).

In order to construct a dynamic neural circuit which approximates a Bayes filter, we define a recurrent neural network which computes approximations of the priors, and a dynamic neural population which encodes approximations of the posteriors. We then define the dynamics of this second neural population to be the result of evaluating the neural network, and adding the result to the response at time \( t + h \) in accordance with equation 12. We conclude this section by describing a network which exactly implements a Bayes filter for the case of linear stimulus dynamics, as derived in (Beck et al., 2011, supplementary material). From this section onwards, we abandon the usage of subscripts to indicate elements of a random vector, and instead use subscripts to indicate time.
3.1 Dynamic Population Codes

Given a time step $h$, a dynamic population code is a sequence of population codes $(X_{hk}, N_{hk})_{k \in \mathbb{N}}$, where the stimuli $(X_{hk})_{k \in \mathbb{N}}$ are generated with the time-invariant transition density $q_{X' | X} := p_{X_{t+h} | X_t}$, $\forall t$. We define $q_{X' | x}$ at every $x$ as a normal density with mean $x + ha(x)$ and covariance $hB(x) \cdot B^T(x)$, where $a$ and $B$ are known as the drift and diffusion, respectively.

The parameters of the tuning curves of the dynamic population code are typically assumed to be constant, but in practice it is often useful to vary the gain of the dynamic population code over time. Nevertheless, for the purposes of developing the theory in this section, we assume that the gain is constant, which allows us to denote the time-invariant encoding density by $q_{N|X} := p_{N_t | X_t}$, $\forall t$. We depict this dynamic population code in figure 3.

3.2 Rate Processes

We aim to construct a rate process $(Z_{hk})_{k \in \mathbb{N}}$ based on a dynamic population code $(X_{hk}, N_{hk})_{k \in \mathbb{N}}$ such that the decoding density of $Z_t$ approximates the true dynamic belief density $p_{X_t | N_0, \ldots, N_t}$ as well as possible. Let us assume that the decoding densities $p_{X_t | Z_t}$ are given by the matrix $\Theta_Z$ as defined in relation 10, and denote the time-invariant decoding density by $q_{X_t | z} := p_{X_t | Z_t}$, $\forall t$. Because the initial posterior of the Bayes filter $p_{X_t | N_0}$ is given by the decoding density of the responses (8), and the decoding densities of the responses and rates are equal up to a linear transformation, we may encode $p_{X_t | N_0}$ in the initial state of the rate process by letting $Z_0 = A \cdot N_0$.

Suppose that $z$ are the rates of the rate process at an arbitrary time $t$, and that $n'$ is the response of the dynamic population code at time $t + h$. If we interpret $q_{N_t | x}$ as the likelihood in Bayes rule (4), then we may express the
posterior beliefs about the subsequent stimulus $x'$ at time $t + h$ as

$$q_{X' | Z, N'}(x' | z, n') \propto q_{N' | x'}(n')q_{X' | Z}(x' | z),$$

(13)

where $q_{X' | Z}(x' | z)$ is a prior which updates the previous beliefs $q_{X | Z}(x | z)$ based on the stimulus dynamics $q_{X' | X}$ (Särkkä, 2013). In order to approximate these priors, let us define $g$ as a form of recurrent neural network which maps rate vectors to rate vectors, and let us aim to minimize, for any $z$, the KL-divergence

$$D(q_{X' | Z}(x' | z) \parallel q_{X' | Z}(x' | g(z))).$$

(14)

This divergence measures the number of bits of information lost by approximating the true prior with the neural network (Kullback and Leibler, 1951).

Suppose that we use the neural network $g$ to approximate the prior in relation 13, and that the tuning curves of the dynamic population code satisfy equation 3. Then the posterior in relation 13 is given by

$$q_{X' | Z, N'}(x' | z, n') = q_{X | Z}(x' | g(z) + A \cdot n'),$$

(15)

which is the result of applying equation 12 to the subsequent response $n'$ and the prior encoded by $g(z)$. Therefore, by recursively defining the rates $Z_{t+h} = g(Z_t) + A \cdot N_{t+h}$ given the base case $A \cdot N_0$, we may construct a coupled Markov process $(X_{hk}, N_{hk}, Z_{hk})_{k \in \mathbb{N}}$, where $(Z_{hk})_{k \in \mathbb{N}}$ form the parameters of the time-dependent posterior density over $(X_{hk})_{k \in \mathbb{N}}$ (figure 4). Moreover, as divergence 14 goes to 0 for all $z$, the rate process converges to an exact encoding of the posteriors $p_{X | N_0, \ldots, N_t}$ computed by a Bayes filter.

Since both the posterior $q_{X' | Z, N'}$ and the likelihood $q_{N' | X}$ are log-linear densities, we may determine the joint density $q_{X', N' | Z}$ at $z$ in the manner of section 2.2 to be the density of the exponential family harmonium defined by relation 9, where the prior is given by $\Theta_X = \Theta_Z \cdot g(z)$. The harmonium structure of $q_{X', N' | Z}$ will later allow us to minimize 14 based on collections of population responses.

### 3.3 Continuous-Time Limits

In this section we explain the relationship between the approach developed in this paper and previous work on population codes filters for known, linear dynamical systems (Deneve et al., 2007; Beck et al., 2011). Although this section is not strictly necessary for developing the methods of this paper, it is worth demonstrating that our method can be seen as an extension of previous work to the case of nonlinear, unknown dynamics.
Figure 4: **Coupled Markov Process**: Here we depict a coupled Markov process composed of a dynamic stimulus, a dynamic response, and a rate process. The arrow from $Z_t$ to $Z_{t+h}$ represents the contribution of the neural network $g$, the result of which is added to $N_{t+h}$ in order to fully calculate $Z_{t+h}$. In the context of this process, the goal of approximate filtering is to ensure that $p_{X_t|Z}$ approximates $p_{X_t|N_0, ..., N_t}$ as well as possible.

By taking continuous-time limits of the stimulus dynamics $q_{X'\mid X}$ and the responses generated by $q_{N'\mid X}$, we may form the Ito diffusion $(X_t)_{t \in \mathbb{R}^+}$ defined as a solution of the stochastic differential equation

$$dX_t = a(X_t)dt + B(X_t) \cdot dW_t,$$

where $(W_t)_{t \in \mathbb{R}^+}$ is a Brownian motion, as well as the inhomogenous Poisson process $(N_t)_{t \in \mathbb{R}^+}$ with stochastic rates $\gamma_f(X_t)$ (Cox and Isham, 1980; Figueroa-López, 2012; Ceci and Colaneri, 2012). Moreover, if we suppose for small $h$ that $g(z) \approx z + h\dot{g}(z)$, then we may construct the continuous-time rate process $(Z_t)_{t \in \mathbb{R}^+}$ which solves the stochastic differential equation

$$dZ_t = \dot{g}(Z_t)dt + A \cdot dN_t,$$

where $(Z_t)_{t \in \mathbb{R}^+}$ depends implicitly on $(X_t)_{t \in \mathbb{R}^+}$ through $(N_t)_{t \in \mathbb{R}^+}$.

If we assume that that the drift is a linear function and that the diffusion $B$ is independent of the stimulus, then a $\dot{g}$ which sets divergence 14 to zero exists, and has the form

$$\dot{g}(z) \mapsto G^{(2)} \cdot z - z \cdot G^{(3)} \cdot z + 1(z^0 - \frac{1}{m} \cdot z).$$

Intuitively, $G^{(2)}$ drives the rate of the population in proportion to the linear dynamics, $G^{(3)}$ suppresses the rate of the population in proportion to the diffusion, 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).
4 Maximum Likelihood

The goal of this paper is to develop a neural network for the case where the drift and diffusion of the stimulus dynamics are unknown and nonlinear. In doing so, we abandon closed-form expressions for the optimal neural network, and instead define a parameterized family of neural networks \( g \), and apply the method of maximum likelihood to minimizing KL-divergence \( 14 \).

For the purposes of optimizing the neural network, we assume that a hypothetical neural system may access the responses \( n_t \) and rates \( z_t \) at any time \( t \) over the course of a simulation. If we would have access to conditional samples \( (z, x') \) from \( q_{X'}|Z \) in relation \( 13 \), then minimizing divergence \( 14 \) with \( g \) would reduce to a regression problem. However, since we cannot access the stimuli \( x_t \), we instead maximize the likelihood of the parameters of the network based on conditional samples \( (z, n') \) from \( q_{N'}|Z \). Due to the joint distribution of \( X_t \) and \( N_t \) defined by the parameters of the dynamic population code, maximizing this likelihood indirectly minimizes divergence \( 14 \).

Since we define the initial rates as \( Z_0 = A \cdot N_0 \), maximizing the likelihood given samples of the first transition of the coupled Markov process is simple, because the initial training pairs \( (z_0, n'_0) \) may be generated independently of the neural network. Using rates \( z_t \) after the initial time, however, must be carefully motivated, because if the neural network is not yet well-trained, then the rates that it computes will not be good targets for conditional likelihood maximization. As we argue, we may circumvent this problem with a form of bootstrapping, by training the neural network on initial transitions, and then on transitions from sample paths of gradually increasing length.

4.1 Contrastive Divergence

Suppose the neural network \( g \) is parameterized by \( \theta \), and that we use it to define the harmonium density \( q_{X',N'|Z} \) as described in section 3.2. We aim to maximize the likelihood of \( \theta \) given samples \( (z, n') \) by gradient descent, and in order to compute the negative log-likelihood gradient \( -\partial_{\theta} \log q_{N'|Z}(n' | z) \), we must be able to compute expectations with respect to the \( q_{X',N'|Z}(x', n' | z) \) (see Bengio, 2009). In general, the expectations of an exponential family harmonium may be approximated by Gibbs sampling (Geman and Geman, 1984; Roberts and Polson, 1994). Given the rates \( z \) and an arbitrary vector \( n^0 \), Gibbs sampling for \( q_{X',N'|Z} \) proceeds by iteratively generating a sample from the random variables

\[
X^0 \sim q_{X | Z}(x^0 | g(z) + n^0), \\
N^k \sim q_{N | X}(n^k | X^{k-1}),
\]
\[ X^k \sim q_{X|Z}(x^k | g(z) + N^k), \] (19)

and based on the fact that \((X^\infty, N^\infty) \sim q_{X', N'|Z}(x', n' | z)\), using samples generated by large numbers of iterations to approximate the expectations.

Because computing a large number of such iterations is expensive, the contrastive divergence gradient was developed as an approximation of the log-likelihood gradient (Hinton, 2002; Bengio and Delalleau, 2009). Given a sample \((z, n')\), computing the contrastive divergence gradient proceeds in the same manner as computing the log-likelihood gradient with Gibbs sampling, except instead of an arbitrary starting condition which we wish the sampler to forget, we let \(n^0 = n'\), which allows a useful gradient to be calculated after a handful of iterations.

In our case, the gradient of the negative log-likelihood of \(q_{N'|Z}(n' | z)\) given the observation \((z, n')\) is approximated by

\[ -\partial_i \log q_{N'|Z}(n' | z) \approx \mathbb{E}[s(X^0) - s(X^k)] \cdot \Theta_Z \cdot \partial_i g(z), \] (20)

where \(k\) is the number of contrastive divergence steps, and \(X^0\) and \(X^k\) depend implicitly on \(n'\) through definition 19. As \(k \to \infty\), the contrastive divergence gradient converges to the negative log-likelihood gradient, allowing us to define a range of algorithms based on the same general procedure.

Intuitively, the gradient in this equation minimizes the difference between an estimate of the stimulus based on the response \(n'\), and an estimate of the stimulus based on the predictions of the harmonium. This quantity tends to zero as the divergence in expression 14 is minimized, and so in this way we may see \(\mathbb{E}[s(X^0) - s(X^k)] \cdot \Theta_Z\) as a form of error function. By applying the back-propagation algorithm to this error function, we may thus fully compute the contrastive divergence gradient with respect to the neural network (Rumelhart et al., 1986). Although other function approximation architectures for \(g\) are possible, in this paper we assume that \(g\) is a form of multilayer perceptron.

### 4.2 Generating Sample Paths

Now that we can approximately maximize the likelihood of the parameters of the neural network \(g\) given conditional samples \((z, n')\), let us address the issue of how to generate appropriate training samples. Given an initial stimulus \(x_0\), we may generate an initial response \(n_0\) with the encoding density and define the initial rate \(z_0 = A \cdot n_0\). We may then generate the subsequent stimulus \(x_h\) with the transition density of the dynamic population code, and then the subsequent response \(n_h\) again with the encoding density. These samples may
be generated independently of the neural network, and may be used to successfully train $g$ to minimize divergence 14 with respect to the first transition of the coupled Markov process.

In order to generate training samples of $z_t$ for $t$ greater than 0, we must make use of approximations based on the neural network $g$. However, training on such rate samples $z_t$ based on a poorly trained $g$ leads to maximizing the conditional likelihood $q_{n|h}(n_{t+h} | z_t)$ of rates $z_t$ which are both uninformative about the stimulus, and unpredictable in distribution. As such, following the gradient based on these poor quality samples leads to high variance and instability in the gradient calculations, which worsens as the sample paths are increased in length.

In this paper we therefore begin optimizing $g$ on correct initial transitions, and then on transitions from sample paths of gradually increasing length. This allows us to follow a more stable gradient through the parameter space without changing the objective defined by divergence 14. We validate this method in the next section with simulations based on a stochastic pendulum.

5 Simulations

In this section we describe in detail how to train and apply a neural network to approximately filter a dynamic population code which responds to a damped stochastic pendulum. An advantage of relying on computer simulations of for demonstrating our method, is that we may compute the likelihoods of the rate process given the generated stimuli for the purposes of validation, while training the rate process itself under the assumptions of the filtering problem. We make use of this feature throughout this section.

After training the neural network and validating it with the relevant likelihoods, we test the learned rate process for some of the canonical properties found in population code research. In particular, we show that the learned rate process is able to successfully filter the population responses as the gain of the population code is varied over time, even though the rate process itself was only trained on fixed gain simulations. We also analyze a topographic representation of the rates in a single step of the rate process, and find that the rate process forms a moving hill of activity in the topographic rate space.

5.1 Model Construction

A stochastic pendulum is a two-dimensional process where the first dimension is the angular position, and the second dimension is the angular velocity. The
drift of the stochastic pendulum featured in this section is given by

\[
\begin{align*}
a_1(q, \dot{q}) &= \dot{q}, \\
a_2(q, \dot{q}) &= -g \sin(q) - c\dot{q},
\end{align*}
\]

where \( g = 9.81 \) is the gravitational constant and \( c = 0.1 \) is the coefficient of friction, and the diffusion is given by

\[
\begin{align*}
b_{11}(q, \dot{q}) &= b_{12}(q, \dot{q}) = b_{21}(q, \dot{q}) = 0, \\
b_{22}(q, \dot{q}) &= \sigma \dot{q},
\end{align*}
\]

where \( \sigma^2 = 0.25 \) is the variance of the noise process. By restricting the noise to the velocity, we may imagine that the diffusion models unpredictable perturbations in the environment like strong winds and other annoyances.

We define the gain of the dynamic population code by \( \gamma = 20 \), and we define the tuning curves as the product of two 1-dimensional Gaussian tuning curves with identical tuning widths given by \( \sigma^2 = 2 \), and preferred stimuli distributed uniformly over a \( 12 \times 12 \) grid on the rectangle \( [-\pi - 2, \pi + 2] \times [-7, 7] \subset \mathbb{R}^2 \). In this case, the sufficient statistic in the definition of the harmonium density is \( s(q, \dot{q}) = (q, q^2, \dot{q}, \dot{q}^2) \), which is the sufficient statistic of a 2-dimensional normal distribution with a diagonal covariance matrix.

We define the neural network as a three-layer perceptron. We let the activation function of the hidden layer be the classic sigmoid function, and set the dimension of the hidden layer to 50. We set the activation function of the 144-dimensional output layer to be the exponential function, to force the components of the output of the network to be positive. Although the usefulness of negative rates may be worth exploring, prior belief encodings are formally defined as having positive rates, and so for the sake of mathematical consistency we adopt the exponential activation function. Finally, as discussed in section 2.3, we let \( A \) be the identity matrix.

5.2 Training

We set the time step in our simulations to \( h = 0.1 \). We generate samples from the jump-diffusion by first generating an initial stimulus \( x_0 \) from a uniform distribution over \( [-\pi, \pi] \times [-5, 5] \), then generating a response \( n_0 \) of this stimulus from the encoding density \( q_{N,h}^X \) and setting this response to the initial rate \( z_0 = n_0 \). Given these initial states, we then generate a sample path of length \( k + 1 \) from the coupled Markov process, thus producing \( k \) sample transitions \( \{(z_{ih}, n_{(i+1)h})\}_{i=0}^{k-1} \). In order to prevent the neural network from taking advantage of the order in which the training samples are presented (Taylor, 2009), we shuffle the order of the samples before training the network.
We begin training the multilayer perceptron by using a learning rate of $\epsilon = 0.01$ and sample paths of length given by $k = 1$. We set the number of contrastive divergence steps to $n_{CD} = 1$, and when running contrastive divergence we generate 10 samples in parallel to approximate the expectation. We train the model on a total of 2,000,000 sample transitions, over the course of which we gradually change the training parameters to $\epsilon = 0.001$, $n_{CD} = 5$, and $k = 10$. Note that based on this scheme, gradient steps take progressively longer to compute over the course of training, with the final iterations taking an order of magnitude longer than the initial iterations.

In order to monitor the training progress, and in particular the descent of the divergence in expression 14, we occasionally pause the training process and generate 2500 sample paths in the manner previously described with $k = 200$, which is equivalent to 20 seconds of simulation time. Since the duration of such paths grant the stochastic pendulum enough time to exceed the bounds of the period $[-\pi, \pi]$, and thus eventually the bounds of the preferred stimuli, we discard those paths for which the pendulum flips. At each transition pair $(z_t, x_{t+h})$ of each accepted simulation, we then estimate the the time-average of divergence 14 up to an additive constant, by calculating the negative logarithm of $q_{X|Z}(x_{t+h} | z_t)$, and averaging the result across all generated transitions. Since recently initialized neural networks compute priors which often result in numerically infinite log-likelihoods, we plot the descent only after the first 200,000 training iterations, as depicted in the left panel of Figure 5: **Negative Log-Likelihood Statistics.** Here we plot two kinds of likelihood statistics for the purposes of monitoring the progress of training, and validating the model, respectively. Left: A plot of the approximate descent of the divergence in expression 14 over the course of training. We begin the plot only after 200,000 training iterations, as recently initialized neural networks tend to result in numerically infinite log-likelihood estimates. Right: A histogram of the negative log-likelihood of the rate (red) and the response (black) given the stimulus. Over 99.9% of the generated rate-based negative-log likelihoods were within the bounds of the histogram, and the maximum value in the tail was 31.7. Rate-based estimates provide an average of 1.01 bits more information about the stimulus than the response alone.
Figure 6: **Coupled Markov Process Simulation**: In this simulation of 20 seconds (x-axes) of a coupled Markov process based on a well trained neural network, the neural network adds an average of 1.1 bits of information about the stimulus to each response. **Top**: We compare the angular position of the stimulus (black line) with the mean of the response decoding density (black dots) and the mean of the rate decoding density (red line). **Middle**: Here, the y-axis represents the angular velocity. **Bottom**: The component-wise sum of the response over time (black dots) compared with the component-wise sum of the rates over time (red line). The time averages of these quantities are 21.2 and 52, respectively.

5.3 Validation

Although we defined the minimization of expression 14 as the training objective of our optimization problem, the ultimate goal of filtering is to maximize the likelihood of the rate process given the stimuli. As such, in order to validate the trained model, we calculate the negative logarithms of $q_{X|Z}(x_t | n_t)$ and $q_{X|Z}(x_t | z_t)$ based on 2500 sample paths given by $k = 200$ as described in the previous section, and plot the histogram of the resulting negative log-densities (figure 5: Right). This histogram visualizes how much more information the rate process contains about the stimulus than the population responses alone. The distribution of the belief negative log-likelihoods has a long tail, and so we leave a small number of values off of the histogram for the purposes of
Based on the results presented in figure 5, we conclude that the trained neural network has found a locally optimal representation of the true prior with respect to the KL-divergence in expression 14, and that the rate process based on this neural network reliably contains more information about the stimulus than the population responses alone. The population responses serve as a good baseline for performance in the filtering problem, because if the population rates do not reliably provide more information about the stimulus than the response alone, then one should discard the rates and rely exclusively on the responses.

In order to better understand the result of a single simulation of the coupled Markov process, we generate a sample path with an initial stimulus of $x_0 = (1, 0)$, and visualize the results (figure 6). In particular, we visualize the angular position and velocity of the pendulum over time, and compare them with the mean of the decoding density based on the responses, on one hand, and the mean of the decoding density based on the rate vectors, on the other. In addition, since the total rate of a population encoding is approximately proportional to the precision of the decoding density (Zemel et al., 1998), we visualize the precision of the decoding densities by comparing the total spike count of each response with the total rate of the rate process.

### 5.4 Computational Issues

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 under version 0.2. The simulations were computed on a 4-core 2.7 gigahertz processor, and ran primarily on a single core. With this system, training the neural network takes a few hours, computing the validation histogram takes about five minutes, and generating a single sample path for visualization takes less than a second. Although training a neural network is slow, the approximate filtering itself may be computed in real time.

An alternative to rejecting simulations which run out of bounds would be to make use of von Mises distributions for describing the pendulum angle, rather than relying purely on normal distributions. For the purposes of training, we only require the ability to generate samples from the belief distribution, and generating samples from the von Mises distribution is indeed possible (Best and Fisher, 1979). However, since there is no closed-form expression for the von Mises density, we cannot directly analyze the log-likelihood statistics of the results, and so we rely on normal distributions for the purposes of validation.
Figure 7: **Variable Gain Simulation**: This simulation was generated with the same parameters as the simulation in figure 6, except instead of a constant gain of $\gamma = 20$, we sinusoidally vary the gain between 5 and 45. The learned rate process maintains an accurate encoding of the stimulus regardless of the gain.

### 5.5 Properties of the Rate Process

In order train our model, we relied on the hypothesis that a parametric neural network trained on short simulations can generalize to filtering problems of a longer duration. This hypothesis is based on the intuition that since the parameters computed by a Bayes filter are Markov (Ceci and Colaneri, 2012), a good approximate model of a single transition of the rate process should be as Markov as possible. If we therefore assume that the transitions of the rate process which minimize divergence 14 can be described by their recent history, then we would expect that short training simulations would be sufficient for approximating them. The fact that our simulations demonstrate that short paths are indeed sufficient for training a good model of the optimal jump-diffusion appears to support our hypothesis.

A key property of any good filtering algorithm is the ability to weigh the importance of the observations against the predictions of the model when the amount of information available in the observations is dynamic. In order to demonstrate the ability of the learned rate process to adapt to conditions of variable information, we test it on a filtering problem where the gain of the
Figure 8: Rate Process Step: A complete step from an update of the trained rate process at 4 seconds into a simulation. Each image is a pixel map, where the opacity of each pixel equals the rate of a neuron normalized to within 0 and 3.85, and the (row, column) location of the pixel is ordered according to the (velocity, position) of the preferred stimulus of the neuron, where the preferred stimuli are distributed over the rectangle $[-\pi - 2, \pi + 2] \times [-7, 7]$. Counting from left to right, the first pixel map is of the current rate (red), the second is of the result of the prior belief function (blue), the third is of the subsequent response (black), and the fourth is of the subsequent rate (red), which is the sum of the prior and response. At this point in the simulation the stochastic pendulum is moving rightwards across the top of the phase space, which is matched by the evolution of the rate process.

(population response is varied sinusoidally over time, as depicted figure 7. This simulation demonstrates that a neural network trained with our method may automatically solve filtering problems under conditions of dynamic gain, and thus, dynamic information content, even though the neural network itself was only trained on fixed gain simulations.

It is not surprising that our neural network learns to solve the filtering problem in a gain-independent way, as gain-invariance properties of population codes have been widely explored in the literature (Zipser et al., 1988; Salinas and Thier, 2000; Beck et al., 2011). In our case, since the decoding density developed in section 2 is independent of the gain of the population code, changing the gain should not change the optimality of the neural network with respect to divergence. As such, the rate process implementations of Bayes filters described in this paper are naturally adaptive to changes in the population gain, and correspondingly, to changes in the amount of information available about the stimulus over time.

Topographic representations of the rate space allow us to qualitatively understand how rates and responses encode stimuli. In figure 8, we illustrate a complete step of the trained process by organizing the component rates of the various encodings into a matrix, such that the row indicates the preferred velocity of the corresponding component, and the column indicates the preferred direction, and by visualizing the matrix by indicating the value of the elements with the transparency of a particular colour. This allows us to see,
for example, that the rate process, indicated with red, is constituted by a hill of activity in the rate space, as is typically the case in population code models (Zemel et al., 1998; Deneve et al., 2007; Beck et al., 2011).

Decoding in the context of population codes approximately reduces to a sum of the preferred stimuli weighted by the component rates to compute the mean, and a sum of the component rates to compute the precision (Zemel et al., 1998). By comparing the prior encoding computed by the neural network, indicated with blue, and the response indicated with black, we see that the prior encoding and the population response encode similar means. Even though it does so with a more diffuse hill of activity, the prior encoding in figure 8 provides a more precise estimate of the stimulus than the response, because the sum of the components of the prior encoding is greater than the sum of the components of the response.

The neural network makes use of a diffuse encoding scheme because it must not only account for noise in the response process, but also for noise in the evolution of the stochastic pendulum. Consequently, the diffuse hill of activity computed by the neural network ensures that the response will always overlap somewhere with the prior encoding. Due to the product of experts structure of the population code (Hinton, 1999; Ma et al., 2006), such overlapping regions result in highly precise posteriors over the stimulus.

6 Discussion

We have shown how to construct a neural circuit composed of a dynamic population code, a population rate process, and a recurrent neural network, and how to train the neural network with a combination of backpropagation and contrastive divergence so that the neural circuit approximately implement a Bayes filter. The rate process based on this neural network reliably contains more information about the stimulus than the population responses alone, which demonstrates both the validity of the gradient descent method developed in section 4, and the practical value of the learned neural network.

We have also shown that the learned rate process successfully filters the population responses as the gain of the population code is varied over time, even though the rate process itself was only trained on fixed gain simulations. This demonstrates that our method is an efficient solution to the more general filtering problem where the average amount of information in the population response is dynamic. Moreover, this work provides another example of the practical value of population codes with stimulus-independent total rates (Salinas and Thier, 2000; Beck et al., 2011).

In order to understand how the rate process encodes the stimulus, we an-
alyzed a topographic representation of the rates from a single step of the rate process. This allowed us to compare the behaviour of the learned rate process with previous research, where population code filters encode stimuli with moving hills of activity in the topographic rate space (Deneve et al., 2007; Beck et al., 2011). We found that the neural network learns a diffuse representation of the underlying stimulus, which accounts for the noise in both the population responses and the dynamics of the stimulus. This diffuse representation nevertheless provides accurate prior beliefs about the stimulus, and allows for accurate posterior beliefs when added with the population response.

We developed our method with the stated aim of modelling how animals solve the filtering problem. If our goal is to offer a phenomenological account of the firing activity of populations of neurons, then we may apply our method by identifying a population of neurons which fires according to a dynamic population code, training the recurrent neural network on responses generated by the population, and then finding a second population of neurons with firing rates given by the learned population rate process. Such an experiment, if successful, would demonstrate that the neural circuit composed of the two populations is approximating a Bayes filter.

Nevertheless, if our goal is to model how the neural system itself learns to solve the filtering problem, then we must also explain how the various algorithms we apply could be feasibly implemented by animal brains as we know them. Justifying the use of contrastive divergence is relatively straightforward, as contrastive divergence grew out of a long tradition on Hebbian learning (Hopfield, 1982; Hinton, 1989, 2002; Makin et al., 2013)\(^2\). Our use of backpropagation is more problematic, and we use backpropagation in this paper entirely for pragmatic reasons. Nevertheless, the problems of backpropagation have been known for some time (Hinton, 1989), and we point to recent work by Bengio et al. (2015) as an attempt to address this issue.

Of deeper importance is that the rate process we have developed is not a continuous-time, spiking neural network, which arguably prevents it from being a realistic model of neural activity in the sense of Deneve et al. (2007); Beck et al. (2011). Beck et al. (2011) explained how to extend the continuous-time rate process in equations 17 to a spiking network, by evaluating the time-derivative \(\dot{g}\) of the neural network on stochastic responses from the rate process, generating another set of responses based on the evaluated time-derivative, and then adding these responses to the rate process. Intuitively, this ensures that the neurons in the network only communicate through spikes,

\(^2\)Makin et al. (2015) have also used exponential families and contastive divergence to model how neural systems learn to solve the filtering problem for linear dynamical systems, although this approach differs substantially from our own.
such that no neuron knows the exact rate of any other neuron. As Beck et al. (2011) also show, this introduction of noise into the rate process has little impact on its ability to represent the stimulus, allowing the authors to unproblematically extend their work to spiking networks.

In our case, it is not difficult to sample the rates of a multilayer perceptron in the manner just described, as there exists a natural correspondence between multilayer perceptrons and stochastic neural networks, as exploited in Hinton et al. (2006). We have also developed a neural network implementation and algorithm for training a continuous-time $\dot{g}$, and have found that it performs comparably to the time-discretized version. Nevertheless, the aim of this paper is to provide the essential theory for our approach to approximate filtering with unknown stimulus dynamics, and so we leave the complete development of the continuous-time extension to future work.

**Acknowledgments**

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**

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.

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., Lee, D.-H., Bornschein, J., and Lin, Z. (2015). Towards Biologically Plausible Deep Learning. *arXiv preprint arXiv:1502.04156*.

Best, D. J. and Fisher, N. I. (1979). Efficient Simulation of the von Mises Distribution. *Applied Statistics*, 28(2):152.

Bobrowski, O., Meir, R., and Eldar, Y. C. (2009). Bayesian filtering in spiking neural networks: Noise, adaptation, and multisensory integration. *Neural computation*, 21(5):1277–1320.

Ceci, C. and Colaneri, K. (2012). The Zakai equation of nonlinear filtering for jump-diffusion observation: existence and uniqueness. *arXiv:1210.4279 [math]*. arXiv: 1210.4279.

Cox, D. R. and Isham, V. (1980). *Point Processes*. CRC Press.

Deneve, S., Duhamel, J.-R., and Pouget, A. (2007). Optimal Sensorimotor Integration in Recurrent Cortical Networks: A Neural Implementation of Kalman Filters. *The Journal of Neuroscience*, 27(21):5744–5756.

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

Figueroa-López, J. E. (2012). Jump-diffusion models driven by Lévy processes. In *Handbook of Computational Finance*, pages 61–88. Springer.

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.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial intelligence*, 40(1):185–234.

Hinton, G. E. (1999). Products of experts. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 1–6. IET.

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

24
Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

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

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.

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.

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.

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

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

Taylor, G. W. (2009). Composable, Distributed-state Models for High-dimensional Time Series. PhD.
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.

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.