

Approximate, Computationally Efficient Online Learning in Bayesian Spiking Neurons

Levin Kuhlmann

levink@unimelb.edu.au

NeuroEngineering Laboratory, Department of Electrical & Electronic Engineering, The University of Melbourne, and the Centre for Neural Engineering, The University of Melbourne, Victoria 3010, Australia.

Michael Hauser-Raspe

michael.hauser-raspe@cantab.net

The MARCS Institute, University of Western Sydney, Penrith, Australia.

Jonathan H. Manton

jonathan.manton@gmail.com

Control & Signal Processing Laboratory, Department of Electrical & Electronic Engineering, The University of Melbourne, Victoria 3010, Australia.

David B. Grayden

grayden@unimelb.edu.au

NeuroEngineering Laboratory, Department of Electrical & Electronic Engineering, The University of Melbourne, and the Centre for Neural Engineering, The University of Melbourne, Victoria 3010, Australia.

Jonathan Tapson

j.tapson@uws.edu.au

The MARCS Institute, University of Western Sydney, Penrith, Australia.

André van Schaik

a.vanschaik@uws.edu.au

The MARCS Institute, University of Western Sydney, Penrith, Australia.

Running title: Approximate, Efficient Learning in Bayesian Neurons

Bayesian Spiking Neurons (BSNs) provide a probabilistic interpretation of how neurons perform inference and learning. Online learning in BSNs typically involves maximum-likelihood expectation-maximisation (ML-EM) based parameter estimation, which is computationally slow and limits the potential of

studying networks of BSNs. An online learning algorithm, Fast Learning (FL), is presented that is more computationally efficient than the benchmark ML-EM for a fixed number of time steps as the number of inputs to a BSN increases (e.g. 16.5 times faster run times for 20 inputs). Although ML-EM appears to converge 2-3.6 times faster than FL, the computational cost of ML-EM means that ML-EM takes longer to simulate to convergence than FL. FL also provides reasonable convergence performance that is robust to initialization of parameter estimates that are far from the true parameter values. However, parameter estimation depends on the range of true parameter values. Nevertheless, for a physiologically meaningful range of parameter values, FL gives very good average estimation accuracy, despite its approximate nature. The FL algorithm, therefore, provides an efficient tool, complementary to ML-EM, for exploring BSN networks in more detail in order to better understand their biological relevance. Moreover, the simplicity of the FL algorithm means it can be easily implemented in neuromorphic VLSI such that one can take advantage of the energy efficient spike coding of BSNs.

1 Introduction

Bayesian inference provides an intuitive understanding of sensory, cognitive and motor processing by describing these processes in probabilistic terms (Knill & Richards, 1996; Kording & Wolpert, 2004). In addition to the application of Bayesian inference to neural population models (Zemel et al., 1998; Barber et al., 2003), more

recently Bayesian inference has been applied at the level of single neurons in the form of Bayesian Spiking Neurons (BSNs) (Denève 2008a; 2008b; Lochmann & Denève, 2008). BSNs provide a probabilistic interpretation of how neurons can perform inference and learning, and BSN theory has been extended to provide explanations of sensory processing (Lochmann et al., 2012) and working memory (Boerlin & Denève, 2011). Learning in BSNs is local to a single neuron, meaning that BSNs can self-organise using only their inputs, although their organisation is still constrained by underlying assumptions about how the inputs were generated. As a result of the self-organisation of single BSNs, hierarchical feed-forward networks of BSNs can generally be thought of as clustering systems. Moreover, BSN networks can be used to perform perceptual inference and learning in sensory networks, such as feature and object recognition, and this can be done using the efficient spike coding properties of BSNs. Specifically, ‘object’ neurons at the top of a feed-forward hierarchical system which receive no supervisory signals can learn to connect to lower level ‘feature’ neurons that are simultaneously active when the object stimulus is present. In addition, lateral interactions can be used to activate different ‘object’ neuron clusters when different object classes are present in order to allow ‘clustered’ learning of object classes, where groups of few to many neurons could represent the presence or absence of different object classes. Although some BSN networks have been studied (Boerlin & Denève, 2011; Lochmann et al., 2012), learning in large feed-forward, and potentially feedback, BSN networks has not yet been studied in detail.

Given that learning in BSNs involves Maximum-Likelihood Expectation-Maximisation (ML-EM) (Denève, 2008b; Mongillo and Denève, 2008), learning is computationally intensive and, therefore, it can be time consuming to study networks of interesting complexity. Here, we present the Fast Learning (FL) algorithm, which increases the speed of learning in BSNs and should therefore make it easier to study the properties of networks of BSNs. The FL algorithm presented here is more robust and stable with respect to initialisation of parameter estimates and a large set of true parameter combinations than an earlier version (Kuhlmann et al, 2012).

This paper is organised as follows. The methods are first presented, outlining the definition of a BSN, the new FL algorithm and a benchmark ML-EM algorithm. In the results, the performance (parameter estimation accuracy and run time) of the FL algorithm and its dependence on various factors is explored. The FL and ML-EM algorithms are compared for a large number of true parameter combinations and for varied degrees in the uncertainty of the initial parameter estimates relative to the true parameter values. This is done in order to assess robustness of the FL algorithm and highlight its strengths and weaknesses compared to ML-EM. We also consider the performance of FL in three-layer networks. It is worth noting that the FL and ML-EM algorithms are both online and adaptive, meaning that they can be applied to data in real-time as it enters the system and that their parameter estimates will converge when the input statistics are stationary, but can adapt to estimate new parameter values if the input statistics change to a different stationary distribution.

2 Methods

2.1 Bayesian Spiking Neuron (BSN) Equations

A BSN (Denève, 2008a; 2008b) is defined to have an explicit space and an implicit space (see Figures 1A and 1B). The explicit space represents the neuron, with N synaptic inputs indexed by i , input spike sequences, s_t^i , ($s_t^i = 1$ corresponds to occurrence of a spike at time t on the i th synapse, otherwise $s_t^i = 0$), synaptic weights, w_i , and output spike sequence, O_t . The implicit space is defined by a 2-state Hidden Markov Model (HMM), where the hidden state takes on values of $x_t = \{0,1\}$ that, for example, model the presence or absence of a stimulus. When the hidden state is $x_t = 1$ it is considered “on” or “present”, whereas when the hidden state is $x_t = 0$ it is considered “off” or “absent”. The hidden states transition with the probabilities,

$$P(x_t = d \mid x_{t-\Delta t} = c) = a_{cd} = \begin{bmatrix} (1-r_{on}\Delta t) & r_{on}\Delta t \\ r_{off}\Delta t & (1-r_{off}\Delta t) \end{bmatrix}, \quad (1)$$

where r_{on} and r_{off} are the rates of $off \rightarrow on$ and $on \rightarrow off$ transitions, t is the current time, and Δt is the time step.

The HMM produces N independent observable outputs that can produce one of two observations within a time step, $s_t^i = \{0,1\}$, where i indexes the N outputs. These N independent output observations of the HMM correspond to the inputs to the BSN and are modelled as inhomogeneous Poisson processes, where the process occurs as a homogeneous Poisson process either with rate q_{on}^i when $x_t = 1$ or with rate q_{off}^i when

$x_t = 0$. For small time steps, the Poisson process can be described by a binomial process and the observation probabilities for each synapse become

$$P(s_t^i = e | x_t = d) = b_{de}^i = \begin{bmatrix} (1 - q_{off}^i \Delta t) & q_{off}^i \Delta t \\ (1 - q_{on}^i \Delta t) & q_{on}^i \Delta t \end{bmatrix}. \quad (2)$$

For a BSN when the stimulus is either present or absent, the distribution of the number of output spikes per a given time interval matches closely to a Poisson distribution (Denève, 2008a) and, therefore, it is possible to build hierarchies of BSNs. The aim of a single BSN is to code the log-odds ratio of the hidden state, x_t , in its output spikes. This is achieved by numerically integrating the differential equation of the log-odds ratio, L_t , along with a differential equation for the prediction of the log-odds ratio, G_t , which depends on the output spikes of the BSN. When the log-odds ratio goes above its prediction by $g_o/2$, an output spike is generated (represented by $O_t = 1$). This can be seen in the following equations (Denève, 2008a):

$$\Delta L_t = \left[r_{on} (1 + e^{-L_t}) - r_{off} (1 + e^{L_t}) - \theta_{th} \right] \Delta t + \sum_i w_i s_t^i \quad (3)$$

$$\Delta G_t = \left[r_{on} (1 + e^{-G_t}) - r_{off} (1 + e^{G_t}) \right] \Delta t + g_o O_t \quad (4)$$

$$O_t = \begin{cases} 1 & \text{iff } L_t > G_t + \frac{g_o}{2} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where

$$w_i = \log \left(\frac{q_{on}^i}{q_{off}^i} \right), \quad (6)$$

$$\theta_{th} = \sum_i (q_{on}^i - q_{off}^i) \quad (7)$$

and g_o is a free parameter that controls coding efficiency. Unless otherwise stated, for all simulations $g_o = 1.45$. Learning in a BSN involves estimation of the unknown parameters r_{on} , r_{off} , q_{on}^i , and q_{off}^i . Here, we compare learning with ML-EM and the FL algorithms. In what follows the parameter estimates are denoted as \hat{r}_{on} , \hat{r}_{off} , \hat{q}_{on}^i , and \hat{q}_{off}^i , and θ and $\hat{\theta}$ denote the vectors of the parameters and the parameter estimates, respectively.

2.2 Benchmark ML-EM algorithm

The benchmark online ML-EM algorithm is based on computing the sufficient statistics of the maximum likelihood function (Denève 2008b; Mongillo & Denève 2008). The algorithm converges to a local maximum if run for an adequate amount of time. Mongillo & Denève (2008) do not provide a proof of convergence for their online algorithm, although they explain that their online method converges to the same estimate as batch EM (Rabiner, 1989) for an appropriate discount factor and scheduling of parameter updates. For each time step, at time t , the following equations (8)-(11) are computed in the order presented. Equation (8) defines $\gamma_{cd}^i(s_t^i; \hat{\theta}(t - \Delta t))$,

which is a weighting term that depends on the previous parameter estimates and determines the trajectories of the variables in equations (9) and (10),

$$\gamma_{cd}^i(s_t^i; \hat{\theta}(t-\Delta t)) = \frac{\hat{a}_{cd}(t-\Delta t) \hat{b}_{d,s_t^i}^i(t-\Delta t)}{\sum_{m,n} \hat{a}_{mn}(t-\Delta t) \hat{b}_{n,s_t^i}^i(t-\Delta t) Q_m^i(t-\Delta t)}, \quad (8)$$

where $\hat{\theta}(t-\Delta t)$ is the set of previous parameter estimates which determine the previous estimates of the transition, $\hat{a}_{cd}(t-\Delta t)$, and observation, $\hat{b}_{de}^i(t-\Delta t)$, probabilities as defined in equations (1) and (2), respectively. The variable, $Q_l^i(t-\Delta t)$, represents the probability of being in state l at time t and is updated according to

$$Q_l^i(t) = \sum_m \gamma_{ml}^i(s_t^i; \hat{\theta}(t-\Delta t)) Q_m^i(t-\Delta t). \quad (9)$$

A statistic is then computed from which the current transition and observation probabilities can be estimated:

$$\begin{aligned} \phi_{cde}^{hi}(t) = & \sum_l \gamma_{lh}^i(s_t^i; \hat{\theta}(t-\Delta t)) \\ & \times \left\{ \phi_{cde}^{li}(t-\Delta t) + \eta \left[\delta(s_t^i - e) \delta(c-l) \delta(d-h) Q_l^i(t-\Delta t) - \phi_{cde}^{li}(t-\Delta t) \right] \right\} \end{aligned} \quad (10)$$

where η is a positive-valued temporal forgetting factor. The current transition and observation probabilities are given by:

$$\hat{a}_{cd}(t) = \frac{\sum_{e,h,i} \phi_{cde}^{hi}(t)}{\sum_{d,e,h,i} \phi_{cde}^{hi}(t)}; \quad \hat{b}_{de}^i(t) = \frac{\sum_{c,h} \phi_{cde}^{hi}(t)}{\sum_{c,e,h} \phi_{cde}^{hi}(t)}. \quad (11)$$

The BSN parameter estimates, \hat{r}_{on} , \hat{r}_{off} , \hat{q}_{on}^i , and \hat{q}_{off}^i , can then be inferred from equations (1) and (2). The initialisation and settings of the algorithm that were applied are: $\phi_{cde}^{hi}(0) = 0$, $Q_1^i(0) = 0.5$, $\eta = 10^{-5}$. The statistic, $\phi_{cde}^{hi}(t)$, is updated for the first 10^2 time steps without applying equation (11) so that it can stabilise. It should also be noted that this form of learning of the BSN only directly depends on the input spikes to the BSN and its parameters and not on any other variables of the BSN.

2.3 The Fast Learning (FL) algorithm

As noted by Denève (2008), the parameters of the BSN (i.e. the transition rates and observation rates of the HMM) depend on the following intermediate statistics: the proportion of time spent in the *on* state, $\tau_{on}(t)$, the number of *on* \rightarrow *off* transitions, $N_{on \rightarrow off}(t)$, the number of *off* \rightarrow *on* transitions, $N_{off \rightarrow on}(t)$, the number of spikes occurring at the *i*th synapse during the *on* state, $N_{on}^i(t)$, and the number of spikes occurring at the *i*th synapse, $N^i(t)$. For each time step, the FL algorithm first estimates the hidden state from the log-odds ratio of the hidden state, then estimates these statistics in order to determine the parameters.

First we note that the definition of the log-odds ratio of the hidden state, x_t , is

$$L_t = \log \left(\frac{P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})}{P(x_t = 0 | \mathbf{s}_{0 \rightarrow t})} \right) \quad (12)$$

implying that the probability that the hidden state, x_t , is equal to 1 given the past synaptic inputs from all synapses can be computed using

$$P(x_t = 1 | \mathbf{s}_{0 \rightarrow t}) = \frac{e^{L_t}}{1 + e^{L_t}}. \quad (13)$$

A robust way to determine if a hidden state transition has occurred is to set an auxiliary variable, \tilde{x}_t , (i.e. a hidden state estimate) such that

$$\tilde{x}_t = \begin{cases} 1 & \text{if } P(x_t = 1 | \mathbf{s}_{0 \rightarrow t}) > U_t \\ 0 & \text{if } P(x_t = 1 | \mathbf{s}_{0 \rightarrow t}) < D_t \\ \tilde{x}_{t-\Delta t} & \text{otherwise,} \end{cases} \quad (14)$$

where U_t and D_t are dynamic thresholds that depend on the maximum and minimum values of $P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})$, respectively, over a window of length T :

$$U_t = \theta_U (M_t - m_t) + m_t \quad (15)$$

$$D_t = \theta_D (M_t - m_t) + m_t \quad (16)$$

with

$$M_t = \max\{P(x_u = 1 | \mathbf{s}_{0 \rightarrow u}) : u \in [t - T, t]\} \quad (17)$$

$$m_t = \min\{P(x_u = 1 | \mathbf{s}_{0 \rightarrow u}) : u \in [t - T, t]\} \quad (18)$$

and θ_U and θ_D are factors defining the position of the thresholds within the range of $P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})$ over the last T seconds. These factors should be set such that $\theta_U > 0.5$ and $\theta_D < 0.5$, simply because $P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})$ values above the mid-range of $P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})$ over the last T seconds best indicate x_t is likely to be 1, whereas

$P(x_t = 1 | s_{0 \rightarrow t})$ values below this mid-range best indicate x_t is likely to be 0. In all simulations, $T=0.5$ s. The dynamic thresholding is a heuristic method that provides stability of estimation that cannot be achieved with static thresholds alone (such as by Kuhlmann et al., 2012). In particular, it ensures that state transitions can still be detected when transition rates are low, the number of input spikes is low, and/or the parameter estimates are far from the true parameter values.

An *on* \rightarrow *off* transition variable is defined as

$$T_{off}(t) = \begin{cases} 1 & \text{if } \tilde{x}_t = 0 \text{ and } \tilde{x}_{t-\Delta t} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

and an *off* \rightarrow *on* transition variable is defined as

$$T_{on}(t) = \begin{cases} 1 & \text{if } \tilde{x}_t = 1 \text{ and } \tilde{x}_{t-\Delta t} = 0 \\ 0 & \text{otherwise.} \end{cases} \quad (20)$$

It is then possible to write update rules for the intermediate statistics respectively as follows:

$$\tau_{on}(t) = \eta \tilde{x}_t + (1 - \eta) \tau_{on}(t - \Delta t) \quad (21)$$

$$N_{on \rightarrow off}(t) = \eta T_{off}(t) + (1 - \eta) N_{on \rightarrow off}(t - \Delta t) \quad (22)$$

$$N_{off \rightarrow on}(t) = \eta T_{on}(t) + (1 - \eta) N_{off \rightarrow on}(t - \Delta t) \quad (23)$$

$$N_{on}^i(t) = \eta s_t^i \tilde{x}_t + (1 - \eta) N_{on}^i(t - \Delta t) \quad (24)$$

$$N^i(t) = \eta s_t^i + (1 - \eta) N^i(t - \Delta t), \quad (25)$$

where each equation represents a moving average with exponential decay with a time constant τ and $\eta = \Delta t / (\Delta t + \tau)$ is a positive valued forgetting factor. The forgetting factor η is present in both the FL and ML-EM algorithms. A small η means that learning/parameter estimation is smoother but also takes longer to converge. A large η means the algorithm forgets quickly but can also converge quicker provided τ is of adequate duration to accurately estimate the intermediate statistics required to estimate the parameters.

The parameter estimates can then be calculated from equations (21)-(25) as follows:

$$\hat{r}_{on}(t) = \frac{N_{off \rightarrow on}(t)}{\Delta t(1 - \tau_{on}(t))} \quad (26)$$

$$\hat{r}_{off}(t) = \frac{N_{on \rightarrow off}(t)}{\Delta t(\tau_{on}(t))} \quad (27)$$

$$\hat{q}_{on}^i(t) = \frac{N_{on}^i(t)}{\Delta t(\tau_{on}(t))} \quad (28)$$

$$\hat{q}_{off}^i(t) = \frac{N^i(t) - N_{on}^i(t)}{\Delta t(1 - \tau_{on}(t))}. \quad (29)$$

In equations (26) and (29), it is important to note that the denominator includes 1 rather than the time constant of the online forgetting window, τ , and $\tau_{on}(t)$ represents a proportion of time rather than an actual time, due to our normalised low-pass filter (see equations (21)-(25)) and the use of a discrete time step, Δt .

The FL algorithm runs online by updating L_t using equation (3) and the previous parameter estimates. Then L_t is used in equation (13) to determine the auxiliary state estimate, \tilde{x}_t , in equation (14) which is needed to estimate the intermediate statistics in equations (21)-(25) via equations (15)-(20). From these statistics the current parameter estimates can be determined using equations (26)-(29). The current parameter estimates are then used to determine L_t in the next time step.

The FL algorithm is initialized by setting the intermediate statistics in equations (21)-(25) to zero and, as with the ML-EM algorithm, $\eta = 10^{-5}$. Moreover, the intermediate statistics were updated for the first 10^5 time steps without applying equations (26)-(29). In all simulations, the time step was set to $\Delta t = 0.1$ ms. To ensure numerical stability of the FL algorithm, it is important to prevent the observation rate estimates \hat{q}_{on}^i and \hat{q}_{off}^i from being very close to zero because this can cause the synaptic weights w_i to be undefined, or to approach very large positive or negative numbers (as can be seen by inspecting equation (6)). Moreover, during simulation, it is necessary to provide a small lower bound on the transition rate estimates \hat{r}_{on} and \hat{r}_{off} in order to prevent either parameter estimate becoming too small and causing the log odds ratio to grow very large in either the positive or negative direction, as the resting potential of the BSN corresponds to $\log(\hat{r}_{on} / \hat{r}_{off})$. This is especially important when a BSN receives only a small number of input spikes, which makes it harder to detect

off \rightarrow *on* and *on* \rightarrow *off* transitions. In the simulations presented the estimated transition and observation rates were forced to be above 0.1 s^{-1} and 10^{-3} s^{-1} , respectively. For similar reasons, the intermediate statistic, $\tau_{on}(t)$, was prevented from being zero by adding a small number, 10^{-15} to this statistic.

The FL algorithm emulates the ML-EM algorithm in that it performs iterative online estimation of the state and the parameters, although ML-EM only provides a partial state estimate. In addition, there is no proven guarantee that the FL algorithm will converge on a fixed set of parameters. In contrast and as mentioned above, the ML-EM algorithm presented here converges to the same estimates as batch EM under certain conditions, although a general proof of convergence has not been published. Nevertheless, the computational simplicity of the FL algorithm means that it will be more computationally efficient than the ML-EM algorithm.

2.4 Computational comparison of FL and ML-EM algorithms

Here we explore online learning in a simple network with two BSNs where Neuron 1 receives $N=20$ Poisson process inputs and Neuron 2 receives the output of Neuron 1 as its only input (see Figure 1C). Many simulations of the FL algorithm were performed over a large set of true parameter values to give evidence that the FL algorithm converges reliably in the absence of convergence proofs. The performances of the FL and ML-EM algorithms are compared.

The online learning problem that is analysed first involves simulation of the implicit space HMM of Neuron 1. The simulated hidden state time series (which depends on r_{on} and r_{off}) reflects the presence or absence of a ‘feature’ out in the world that the BSN response encodes. From the simulated hidden states, the simulated observations are produced and fed as input to Neuron 1. When the hidden state is 1, the observations are simulated as a homogenous Poisson process with rate q_{on}^i ; when the hidden state is 0, the observations are simulated as a homogenous Poisson process with rate q_{off}^i . Learning in Neuron 1 then involves estimation of the parameters r_{on} , r_{off} , q_{on}^i , and q_{off}^i .

The output of a BSN is approximately Poisson (Denève, 2008a) and therefore the output of Neuron 1 can be used as the only input to Neuron 2. Note that the hidden state, x_t , is the same for Neuron 1 and Neuron 2 because Neuron 2 only receives input from Neuron 1 and therefore will only learn the same state transition rates coded in the output of Neuron 1. This is synonymous with a hierarchical causal generative HMM model with two states where the top state causes the bottom state (i.e. it follows the top state) and the bottom state produces the observations (Denève, 2008a). Neuron 2 is considered here to highlight some features of the FL algorithm in simple chain networks.

2.5 The FL algorithm applied to three-layer networks

The FL algorithm is also applied to three-layer networks to explore parameter estimation and hidden state decoding accuracy of the FL algorithm across layers in a network. Two three-layer networks are considered, one small, one large. The small three-layer network has 4, 2, and 1 neurons in the 1st, 2nd, and 3rd layers, respectively (see Figure 1D), and each neuron in layer 1 receives 20 inputs. The large three-layer network is similar to the small network; however, it involves a doubling of the synaptic inputs to the higher layers. This second network has 16, 4, and 1 neurons in the 1st, 2nd, and 3rd layers, respectively. Learning is considered for a hierarchical causal generative HMM model where the top state corresponding to layer 3 causes the states corresponding to layer 2, which in turn cause the states corresponding to layer 1, and these bottom states produce the observations. BSN theory has been derived within the context of hierarchical causal models (Denève, 2008a) therefore we restrict our study to these models.

3 Results

3.1 An example simulation using FL

For the two neuron network, Figure 2 shows an example simulation of Neuron 1 and Neuron 2 while running the FL algorithm over 10^6 time steps. The time series that are shown correspond to the last 10^5 time steps of the simulation when the parameter estimates have stabilized. For both neurons, it can be seen that the log odds ratio of the hidden state, L_t , and the output spike sequence, O_t , show strong

correlation with the hidden state, x_t . Moreover, there is less fluctuation in the log odds ratio of Neuron 2 than there is for Neuron 1 because Neuron 1 receives many more input spikes. Given this greater number of inputs, the output spikes of Neuron 1 also provide a better representation of the hidden state.

3.2 FL estimation accuracy depends on various factors

In addition to the BSN's free parameters, g_o and N (free in the sense that they are selected prior to running any algorithm), the FL algorithm introduces three free parameters to select before running the algorithm; the factors θ_U and θ_D , and the window length, T , used to define the dynamic thresholds. Based on simulations, a value of $T = 0.5$ s was empirically found to adequately capture the range of state transition rates considered here, and therefore we focus our analysis on threshold factors θ_U and θ_D , the spiking threshold, g_o , and the range of true parameter values instead.

Based on equations (1) and (2) and given that a time step of 0.1 ms is used, the transition and observation rates should be limited to the range of $0-10^4$ s⁻¹. For the two neuron network, in simulations with FL it was found that if the transition and observation rates covered the full range of $1-10^4$ s⁻¹ and $\theta_U = 0.75$ and $\theta_D = 0.25$, then the transition rates are significantly underestimated while observation estimates are reasonably accurate (Neuron 1 median estimation error values- r_{on} : -97.6%, r_{off} : -97.7%, q_{on}^i : 0.01%, q_{off}^i : -0.04%). Nevertheless, if we restrict ourselves to a

physiologically plausible range of transition rates ($1-115\text{s}^{-1}$, these rates are adequate for capturing ecologically relevant changes of stimuli in the outside world) and observation rates ($1-1000\text{ s}^{-1}$, these rates span the allowed rates of neuronal spiking assuming a 1ms refractory period) we can achieve excellent average estimation performance as the remaining figures will show.

For the two neuron network, Figures 3A-3I show the sensitivity analysis of the FL algorithm for the threshold factors θ_U and θ_D for 100 simulations of 10^6 time steps with different true parameter combinations for each factor pair (i.e. (θ_U, θ_D)), with the true parameters lying in the ‘physiologically plausible’ range. For each simulation the initial parameter estimates were set to five times the values of the true parameters. It can be seen that the median estimation errors in r_{on} and r_{off} for Neuron 1 (top row of Figures 3A and 3B) are quite varied, showing best performance away from low θ_U and high θ_D values, whereas the median error for q_{on}^i and q_{off}^i for Neuron 1 (top row of Figures 3C and 3D) is minimally affected by the threshold factors with errors under about 1% for all factor pairs. The median errors in r_{on} and r_{off} for Neuron 2 (top row of Figures 3E and 3F) show a similar sensitivity pattern to that seen for Neuron 1 although errors are larger for Neuron 2. These greater errors result from Neuron 2 receiving less input spikes and, therefore, less information about state transitions. Note that sensitivity/error maps are not provided for q_{on}^i and q_{off}^i for Neuron 2 because there are no ‘true’ parameter values for this neuron to determine

the error. The sign of the error/bias as a function of threshold factors θ_U and θ_D is captured in the bottom row of Figures 3A-3F. Generally it can be seen that when the errors are large for r_{on} and r_{off} for both Neuron 1 and Neuron 2 the bias tends to be positive (i.e. overestimation) and θ_U and θ_D are both closer to 0.5. For q_{on}^i and q_{off}^i of Neuron 1 it can be seen that the bias is generally negative (i.e. underestimation) and positive, respectively.

For the same case as Figures 3A-3F, Figure 3G demonstrates the median RMS error in the probability that the hidden state is 1, P_{rms} , as a function of the threshold factors for Neuron 1. For a given 10^6 time step simulation, P_{rms} was computed over the last 10^5 time steps as the difference between the simulated $P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})$ time series using the estimated parameters and the true parameters

$$P_{rms} = 100 \sqrt{\frac{1}{N_{rms}} \sum_{j=0}^{N_{rms}-1} \left(P(x_{t_j} = 1 | \mathbf{s}_{0 \rightarrow t_j}; \hat{\theta}(t_j)) - P(x_{t_j} = 1 | \mathbf{s}_{0 \rightarrow t_j}; \theta) \right)^2}, \quad (30)$$

where t_f is the end time of the simulation, $N_{rms} = 10^5$ is the number of time steps over which the RMS error is calculated, and $t_j = t_f - j\Delta t$. Moreover, Figures 3H and 3I show the percent Hamming error between the true and estimated hidden states as a function of the threshold parameters for Neurons 1 and 2, respectively. Percent Hamming error was computed in a similar way to P_{rms} and was defined as follows

$$H = 100 \sqrt{\frac{1}{N_{rms}} \sum_{j=0}^{N_{rms}-1} |\tilde{x}_{t_j} - x_{t_j}|}, \quad (31)$$

where $t_j = t_f - j\Delta t$ also.

In Figures 3G-3I it can be seen that both P_{rms} and the percent Hamming error are lowest for factors θ_U and θ_D close to 0.75 and 0.25, respectively.

To assess how the true parameter value ranges effect parameter estimation, the maximum values for the range of the transition rates, r_{max} , and the observation rates, q_{max} , are varied within a physiologically plausible range in Figures 3J and 3K. In Figure 3J it can be seen that if $q_{max} = 1000 \text{ s}^{-1}$ and r_{max} is varied then the transition rates are overestimated or underestimated if r_{max} is below or above 115 s^{-1} . In Figure 3K it can be seen that if $r_{max} = 115 \text{ s}^{-1}$ and q_{max} is varied then the transition rates are underestimated or overestimated if q_{max} is below or above 1000 s^{-1} . Generally it was found that to minimise transition rate estimation error bias one needs to have transition rates that are sufficiently lower than the observation rates, but not too low. Figures 3J and 3K also demonstrate that the median estimation error of the observation rates is always quite small and is independent of the true parameter range.

To assess the influence of the spiking threshold, g_o , in one layer on the estimation performance in the next layer, the spiking threshold for Neuron 1 is varied for true parameter values corresponding to our ‘physiologically plausible’ range and the estimation performance is plotted in Figure 3L. It can be seen that small and large

values of Neuron 1 g_o lead to overestimation and underestimation of the transition rates for Neuron 2, respectively, with an optimal value of around $g_o = 1.45$. Note that observation rate errors are not plotted because no true observation rates are defined for Neuron 2. A low spiking threshold value means more input spikes to Neuron 2 and leads to too many false state transition detections, while a high spiking threshold value leads to less input spikes and fewer true state transition detections.

In Figures 3J-3K it can generally be seen that the on and off observation rate estimation errors closely overlap each other, as do the on and off transition rate estimation errors in Figures 3J-3L.

3.3 A comparison between FL and ML-EM for the two neuron network

For the two neuron network, Figure 4 shows box whisker plots of the parameter estimation errors for the FL and ML-EM algorithms in cases where the initial parameter estimates were set to the true parameter values (0% initial parameter perturbation case) or the initial parameter estimates were set to five times the values of the true parameters (400% initial parameter perturbation case). Moreover, $\theta_U = 0.75$ and $\theta_D = 0.25$ and the true parameter values lie in our ‘physiologically plausible’ range. Each case was simulated 1000 times for 10^6 time steps with different true parameter combinations each time. It can be seen that ML-EM performs the best for the 0% perturbation case, whereas for the 400% perturbation case, FL performs best for r_{on} and r_{off} , and is comparable to ML-EM for q_{on}^i and q_{off}^i . The FL estimation errors do not change much between the 0% and 400% perturbation cases.

We focus here on a fixed 400% initial parameter estimate perturbation because we are mostly interested in robustness and stability when the true parameters are varied over a large range. The behaviour of FL for 0% and 400% initial parameter estimate perturbations is similar to that shown in Figure 3.

Figures 5A and 5B show that, in terms of the percent Hamming error and P_{rms} , respectively, FL performs better than ML-EM for the 400% perturbation case, and increasing the number of time steps by a factor of 10 further improves the convergence of FL for the 400% perturbation case. ML-EM was only simulated for 10^6 time steps because the run time was too long. Moreover, for ML-EM the P_{rms} is most meaningful, since the percent Hamming error values for ML-EM should only be considered to be approximate. This is because ML-EM only gives a partial state estimate, i.e. the probability $P(x_t = 1 | \mathbf{s}_{0 \rightarrow t})$, which needs to be thresholded (threshold set to 0.5) in order to obtain a non-optimal state estimate used to compute the percent Hamming error.

Figure 5C captures the median run times for the FL and ML-EM algorithms applied to Neurons 1 and 2 simulated for 10^6 time steps over 1000 different true parameter value combinations in the 0% initial parameter estimate perturbation case. For Neuron 1, it can be seen that FL is approximately 16.5 times more computationally efficient than ML-EM for a 20 synapse BSN. For Neuron 2, it can be seen that FL is approximately 6.1 times more computationally efficient than ML-EM for a 1 synapse BSN. These results indicate that FL becomes more computationally

efficient than ML-EM as the number of synapses, and also the number of input spikes to a BSN increases. Note also that for Neuron 1, simulating FL for 10^7 time steps still results in run times nearly less than half the time required for ML-EM to run 10^6 time steps.

The ‘real’ time (seconds of synaptic input) convergence properties of FL and ML-EM are summarised in Figures 5D and 5E. Figure 5D shows an example of parameter estimation for Neuron 1 when the initial parameter estimates are 400% above the true values. As mentioned in the methods the parameter estimates are held constant for the first 10^5 and 10^2 time steps for FL and ML-EM, respectively, in order to let the intermediate statistics stabilise. It can be seen that FL estimates converge close to the true values for all parameters (note that only r_{on} , r_{off} , q_{on}^1 and q_{off}^1 are shown), while the ML-EM estimates converge close to the true values for all the observation rates (note only q_{on}^1 and q_{off}^1 are shown), but not for the transition rates, r_{on} and r_{off} . These observations for FL and ML-EM were generally true as can be observed in Figure 4. Figure 5E shows the ‘real’ time convergence distributions for Neuron 1 for 100% and 400% initial parameter estimate perturbation cases under the same simulation conditions as Figure 4. For the FL and ML-EM algorithms, convergence was deemed to have occurred if over 7.5 seconds of ‘real’ time all the parameter estimates remained within a deviation of 3%. It can be seen that the ‘real’ times to convergence for ML-EM are approximately 2-3.6 times shorter than for FL. Note, the benefits of the faster convergence of ML-EM than FL for the 400% initial

perturbation case are negated by the poorer performance of ML-EM as seen in Figure 5B.

3.4 Three-layer network simulations

To look at FL operating in larger networks and to demonstrate the effects of FL estimation accuracy in one network layer on subsequent network layers, Figure 6 shows the estimation accuracy across the layers of the small and large three-layer networks described in the methods. Similar simulation settings as in Figures 4 and 5 are applied, with the exception that the initial transition and observation rate estimates are selected uniformly within the physiologically plausible parameter range. Figures 6A and 6B illustrate the distributions of the percent estimation error for r_{on} and r_{off} across the layers of the small three-layer network, respectively. By looking at the median values, it can be seen there is a slight increase in underestimation of the transition rates. Figure 6C shows the distributions of the percent estimation error for the observation rates for the neurons in layer 1 of the small network. As in the other figures, the median percent estimation error for the observation rates is small. Figure 6D shows the distributions of the percent Hamming error across the layers of the small network. It can be seen there is a slight increase in the median percent Hamming error; however, this decoding error remains small at the final layer of the network.

Similar results are obtained when each unit in layers 2 and 3 pools over twice as many synaptic inputs; however, the increase in pooling leads to better decoding

performance in layers 2 and 3. This can be seen by comparing the Hamming error in Figures 6D (small network) and 6E (large network with twice as many inputs in higher layers). The increased accuracy in the higher layers in the large network occurs because increasing the input to an adequate level provides a more informative level of evidence about the state changes.

An important property in layered networks is to recode relevant information in the inputs in a sparse manner with fewer output spikes at the top layers. This can be achieved with stronger synaptic weights at the higher layers. Here the informativeness of a given network layer is quantified by the distribution and median value of the final absolute value of the synaptic weights, $|\log(\hat{q}_{on}^i(t_f)/\hat{q}_{off}^i(t_f))|$, in the corresponding layer. Figures 6F (small network) and 6G (large network with twice as many inputs in higher layers) demonstrate that the informativeness of each network layer increases across the layers. Moreover, comparing Figures 6F and 6G indicates that doubling the number of inputs to the higher layers leads to more informative higher layers. This reflects the improved decoding performance of the large network with twice as many inputs in the higher layers.

To quantify the sparsity of spike coding, output spiking rates are estimated over the last 10^5 time steps of each simulation when learning has effectively stabilised. For the small three-layer network although the median output spiking rate per neuron in each layer increases from layer 1 to layers 2 and 3 by 27.6% and 55.8%, respectively, the median total output spiking rate of each layer decreases from layer 1

to layers 2 and 3 by 38.5% and 64.3%, respectively. This indicates that fewer spikes are needed at the higher layers to encode the presence or absence of the input stimuli. Qualitatively similar changes were observed for the large three-layer network. The cost of spiking in higher layers can be reduced by increasing the spiking threshold g_o at each layer, however, for the case of FL-based learning this needs to be traded off against parameter estimation accuracy as indicated in Figure 3L.

4 Discussion

We have demonstrated for a physiologically plausible range of parameters that, FL is more computationally efficient than the benchmark ML-EM algorithm presented, and that FL is also more robust when there is high uncertainty about the initial parameter estimates. The latter occurs because the ML-EM algorithm seeks out local maxima of the ML function and therefore typically will find local maxima rather than the global maximum when there is a large mismatch between the initial parameter estimates and the true parameter values. FL on the other hand, is a heuristic approach that seeks to count the number of state transitions and input spikes, determine the time spent in each state and then estimate the BSN parameters. The FL estimation is robust to high uncertainty about the initial parameter estimates because the dynamic threshold enables estimation of the state even when the transition rate estimates are inaccurate (i.e. the resting potential of the log-odds ratio of the hidden state is offset from the true value) or when the synaptic rate estimates are inaccurate (i.e. synaptic weight magnitudes are not the appropriate size or sign). This is because the dynamic

threshold fits to the short-term range of the log-odds ratio of the hidden state and, for the most part, can capture important log-odds ratio changes induced by input spike events regardless of their size or sign, or the offset of the resting membrane potential/log-odds ratio.

It was observed that the computational run times for a fixed number of time steps are much longer for ML-EM than for FL, while the ‘real’ (i.e. biological) time to convergence for the algorithms was shorter for ML-EM. This faster ‘real’ time convergence for ML-EM, which can also be thought of as a more efficient use of the available data, may simply be due to the smoother convergence trajectory of ML-EM and the greater sensitivity of FL to fluctuations in the input statistics. It is also worth noting that these ‘real’ convergence times are only approximate and primarily capture the times over which major estimate changes are taking place. Both FL and ML-EM can potentially give more accurate estimates if the simulations are run for longer. Figures 5A and 5B show FL estimates become more accurate for longer simulations. We did not consider longer simulations for ML-EM given that it becomes less tractable to complete a large number of simulations for different true parameter value combinations. The reason for the shorter computational run times for a fixed number of time steps of FL compared to ML-EM stems largely from the number of equations that need to be calculated by each algorithm at each time step. As the number of input synapses, N , increases, the number of equations grows more quickly for ML-EM than for FL. For FL and ML-EM, this number is $4N + 13$ and $28N + 6$, respectively. Although, ML-EM appears to converge faster than FL (i.e. requires less time steps),

the computational cost of ML-EM means that ML-EM still takes a much longer time to simulate to convergence than FL. Both the FL and ML-EM algorithms were implemented as sequential programs. For a single neuron, it would be possible to improve the computational efficiency of ML-EM compared to FL if multi-core or GPU implementations were used. In particular, computations for the N synapses can be parallelised for both algorithms. Assuming full parallelisation of the synapse computations/equations, this would effectively reduce the number of equations updated each time step to $4+13=17$ and $28+6=34$ for FL and ML-EM, respectively. Although this would improve performance for a single neuron, parallelisation still consumes computational resources and, therefore, will provide a limitation of the size of the network that can be studied. Where only small networks are considered and computational resources are not an issue, FL would still be useful given its robustness to uncertainty in the initial parameter estimates.

The computational speed and robustness of FL indicates that it will be useful in studying hierarchical feedforward networks of BSNs within tractable simulation times. The study of larger networks is important for more complex inference problems such as feature and object recognition. Moreover, the simplicity of the FL algorithm means it is easier to take advantage of the spike coding efficiency of Bayesian spiking neurons in energy efficient neuromorphic VLSI circuits, compared to if online EM were to be implemented. The FL algorithm that we present is not necessarily intended to be biologically motivated, although its simplicity may make it computationally and mechanistically possible for real neurons to implement. Rather

the FL algorithm, which specifically applies to BSNs, represents a balance between simplicity and efficiency, and provides us with a tool to study BSNs in greater detail in order to further understand their biological relevance and plausibility.

The study of larger networks is beyond the scope of this letter but here we have considered the first step in this direction by simulating learning in a two neuron network and in three-layer networks. Considering the two neuron network, given the one input and a lower number of input spikes to Neuron 2 compared with Neuron 1, it was difficult for Neuron 2 to accurately capture state transitions without the incorporation of the dynamic thresholds to estimate the auxiliary state. The dynamic thresholds thus enhance the stability and accuracy of the FL algorithm. Given that learning in BSNs is local to the neuron and here the FL algorithm appears stable over a large number of true parameter combinations, it is expected that the FL algorithm will be stable in larger networks and this is supported by the simulations with the three-layer networks.

It is important to note that due to symmetry of the HMM, although the FL algorithm can estimate the *on* and *off* parameters reasonably accurately, it cannot determine whether *on* is *on* or *on* is *off*. For example, the estimate \hat{r}_{on} may be close to the true value for r_{off} , and vice versa. Thus Figures 3-6 have been produced by checking if such ‘parameter flips’ have occurred for both the FL and ML-EM algorithms. Although this may seem undesirable, the labelling of “on” and “off” is arbitrary and can be reversed without effecting estimation in the network.

It is also worth noting that, given that the BSN only spikes when the hidden state is “on” (or “off” depending on how the parameters are defined or evolve), the learned observation/synaptic rates in the receiving neuron evolve such that the observation rate for the “on” state (i.e. q_{on}^i) takes a high value while the observation rate for the “off” state (i.e. q_{off}^i) takes a value close to zero. This applies to both the FL and ML-EM algorithms, and was observed for BSNs in feed-forward networks that are not in the first layer of the network (not shown). To ensure this property did not affect numerical stability of FL, very small observation rates were prevented (see methods).

The ML-EM algorithm analysed here (Denève, 2008b; Mongillo and Denève, 2008) is just one possibility and it may be possible to find faster online EM algorithms to estimate HMM parameters (Cappé, 2011) and many variants of online EM algorithms for HMMs exist (Krishnamurthy & Moore, 1993; Elliot et al., 1995; Lindgren & Hoist, 1995; Le Gland & Mevel, 1997; Rydén, 1997; Stiller & Radons, 1999; Andrieu & Doucet, 2003; Tadić, 2010; Cappé, 2011; LeCorff and Fort, 2012). However, it is expected that FL will still be more computationally efficient because many of these methods aim for exactness of estimation whereas FL focuses on simplicity of the algorithm.

One possible shortcoming of the application of FL in hierarchical networks could be an accumulation of parameter estimation errors across layers of neurons. However, as is shown in Figure 6, the hidden state decoding performance can be

reasonably stable, with median percent Hamming errors below 7% across the layers in the three-layer networks. Based on the results in Figure 3L, it may be possible in future versions of FL to control any under- or over-estimation of parameters in subsequent layers by adapting the spiking threshold in each layer.

Perhaps the main shortcoming of FL is the dependence of estimation accuracy/bias on the range of the true parameters of the generative model. This is a reflection of the approximate nature of the FL algorithm. ML-EM on the other hand works for the full theoretically allowed true parameter range. Future work will be needed to find an FL variant that can improve this feature. Nevertheless, for a given problem, provided one knows what parameter range one is interested in, one can test FL for its estimation accuracy/bias on this parameter range. As is shown for a physiologically plausible parameter range, average FL-based estimation accuracy can be very good for an appropriately defined parameter range. Moreover, if FL is not very accurate over a certain parameter range, this may not be a significant problem because in many cases approximate solutions may be adequate and there will be high uncertainty about the true values of parameters, therefore limiting the accuracy of EM approaches, which can become trapped in local minima. On the other hand, FL gives much more consistent parameter estimation regardless of the degree in uncertainty about the true parameter values. Moreover, the problems to which BSN networks can be applied are stochastic in nature and thus small errors can be better tolerated.

An additional possibility, that reflects the complementarity of the FL and ML-EM algorithms, will be to first run the FL algorithm to get reasonably small

parameter estimation error and position the parameter estimates close to the optimum, and then apply ML-EM, which is more exact and has a better chance of finding the optimum if the initial parameter estimates are close to the true parameter values. These ideas should be the subject of future study, with a major aim of seeking to understand how neurons can perform probabilistic inference in sensory (Lochmann et al., 2012) and cognitive networks (Boerlin & Denève, 2011), and how such networks can efficiently learn to extract and/or integrate complex information about the external and internal environment.

Acknowledgements

This work was supported by the Australian Research Council Discovery Project grant DP1096699 and the University of Western Sydney.

References

- Andrieu, C., & Doucet, A. (2003) Online expectation-maximization type algorithms for parameter estimation in general state space models. In Int. Conf. Acoustics, Speech and Signal Processing (Vol. 6, pp. 96–72). Piscataway, NJ: IEEE Press.
- Barber, M., Clark, J., & Anderson, C. (2003) Neural representation of probabilistic information. *Neural Comput.*, 15(8), 1844–1853.
- Boerlin, M., & Denève, S. (2011) Spike-based population coding and working memory. *PLoS Comput Biol* 7(2): e1001080. doi: 10.1371/journal.pcbi.1001080.
- Cappé, O. (2011) Online EM algorithm for Hidden Markov Models. *J. Comput. Graph. Statist.*, 20(3):728–749.
- Denève, S. (2008a) Bayesian spiking neurons i: inference. *Neural Comput.* 20: 91–117.
- Denève, S. (2008b) Bayesian spiking neurons ii: learning. *Neural Comput.* 20: 118–145.
- Elliot, R.L., Aggoun, L., & Moore, J.B. (1995) *Hidden Markov Models: Estimation and Control*. Springer, New York, NY.
- Knill, D., & Richards, W. (1996) *Perception as Bayesian Inference*. Cambridge: Cambridge University Press.
- Kording, K., & Wolpert, D. (2004) Bayesian integration in sensorimotor learning, *Nature*, 427, 244–247.

- Krishnamurthy, V. & Moore, J. B. (1993) On-line estimation of hidden Markov model parameters based on the Kullback-Leibler information measure. *IEEE Trans. Sig. Process.* SP-41,2557-2573.
- Kuhlmann, L., Hauser-Raspe, M., Manton, J., Grayden, D.B., Tapson, J., & Van Schaik, A. (2012). Online learning in Bayesian Spiking Neurons. *Proceedings of IJCNN12*, In press.
- Le Gland, F. & Mevel, L.. (1997) Recursive estimation in HMMs. In *Proc. IEEE Conf. Decis. Control*, pages 3468–3473.
- Lindgren, G. & Hoist, H. (1995) Recursive estimation of parameters in Markov-modulated Poisson processes. *IEEE Transactions on Communications*, vol. 43, no. 11, pp. 2812–2820.
- Le Corff, S. & Fort, G. (2012) Online Expectation Maximization based algorithms for inference in hidden Markov models. *arXiv.org*, vol. math.ST. arXiv:1108.3968v2.
- Lochmann, T., & Denève, S. (2008) Information transmission with spiking bayesian neurons. *New J Phys* 10: 055019 (19pp).
- Lochmann, T., Ernst, U.A., & Denève, S. (2012) Perceptual inference predicts contextual modulations of sensory responses. *J Neurosci.* 32(12):4179–4195.
- Mongillo, M., & Denève, S. (2008) Online expectation-maximization in hidden Markov models. *Neural Comput.* 20: 1706-1716.
- Rabiner, L. R. (1989) A tutorial on hidden Markov models and selected applications

- in speech recognition. *Proceedings of the IEEE*, 77, 257–286.
- Rydén, T. (1997) On Recursive Estimation for Hidden Markov Models. *Stochastic Processes and Their Applications*, 66 (1): 79–96.
- Stiller, J.C. & Radons, G. (1999) Online Estimation of Hidden Markov Models. *IEEE Signal Processing Letters*, 6(8): 213-215.
- Tadić, V.B. (2010) Analyticity, convergence, and convergence rate of recursive maximum-likelihood estimation in hidden Markov models. *IEEE Trans. Inf. Theor.*, 56:6406–6432.
- Zemel, R., Dayan, P., & Pouget, A. (1998) Probabilistic interpolation of population code. *Neural Comput.*, 10(2), 403–430.

Figure Captions:

Figure 1: (A) The implicit space of a BSN involves a 2-state HMM (i.e. $x_t = \{0,1\}$) that produces N independent outputs with 2 possible observations and the outputs are indexed by i (i.e. $s_t^i = \{0,1\}$). The state transition rates are represented by r_{on} and r_{off} , and the observation rates are represented by q_{on}^i and q_{off}^i . (B) The explicit space of the BSN involves N synapses with weights w_i , the input spike sequences are assumed to be the outputs of the HMM (i.e. $s_t^i = \{0,1\}$), and the output spike sequence is given by O_t . (C) The simple two neuron network studied in this paper. Neuron 1 receives $N = 20$ inputs derived from independent Poisson processes and provides the only input to Neuron 2. (D) The small three-layer network studied in this paper. Each layer 1 neuron receives $N = 20$ inputs.

Figure 2: Example BSN simulations for (A) Neuron 1 and (B) Neuron 2 in the two neuron network while running the FL algorithm. Each sub-figure shows the last 10^5 time steps of a 10^6 time step simulation with $r_{on} = 8 \text{ s}^{-1}$, $r_{off} = 10.5 \text{ s}^{-1}$, and q_{on}^i and q_{off}^i uniformly selected in the range of 10-80 spikes/s for Neuron 1. Initial parameter estimates were set to the true parameter values for Neuron 1, while for Neuron 2 initial transition rates were set to the same values as Neuron 1. The initial observation rates for the single input synapse to Neuron 2 were set equal to the values of the first input synapse to Neuron 1. In each neuron-specific sub-figure, the top plot shows the

log odds ratio L_t , the prediction G_t , and the state transition thresholds, U_t and D_t , mapped onto the log odds ratio axis. The middle plot shows the output spike sequence, O_t , and the lower plot shows the HMM state, x_t , unknown to the BSN. $g_o = 2$.

Figure 3: Estimation accuracy as a function of the dynamic threshold and the true parameter range for the FL algorithm for the two neuron network. The median percent parameter estimation error (top row) and the corresponding sign of the error/bias (bottom row) plotted as functions of the dynamic threshold factors θ_U and θ_D for Neuron 1 parameters (A) r_{on} , (B) r_{off} , (C) q_{on}^i , (D) q_{off}^i , and Neuron 2 parameters (E) r_{on} and (F) r_{off} . The Neuron 1 median RMS error of (G) $P(x_t = 1 | s_{0 \rightarrow t})$, P_{rms} , and (H) median percent Hamming error, H , and the (I) Neuron 2 median percent Hamming error, H , as functions of θ_U and θ_D . For each (θ_D, θ_U) pair, the median error was calculated over 100 simulations lasting 10^6 time steps for the 400% initial parameter perturbation case and, for Neurons 1 and 2, r_{on} and r_{off} are uniformly selected in the range of 1-115 s^{-1} , and the q_{on}^i and q_{off}^i are uniformly selected in the range of 1-1000 spikes/s. To improve image clarity, median state transition rate error was clipped at 150 s^{-1} . In the bias map legend white indicates positive bias/overestimation and black indicates negative bias/underestimation for a given θ_U and θ_D pair. Median parameter estimation error of Neuron 1 as a function

of the true parameter range when (J) r_{\max} is varied and the q_{on}^i and q_{off}^i are uniformly selected in the range of 1-1000 spikes/s and (K) q_{\max} is varied and r_{on} and r_{off} are uniformly selected in the range of 1-115 s^{-1} . (L) Parameter estimation error of Neuron 2 as a function of Neuron 1 threshold, g_o , for the same true parameter range as in (A)-(I). The legend in (J) applies to (K) and (L). For (J)-(L) $\theta_U = 0.75$ and $\theta_D = 0.25$, and 1000 simulations were performed to obtain each median value.

Figure 4: Comparison of parameter estimation performance between FL and ML-EM for the 0% and 400% initial parameter perturbation cases for the two neuron network. The Neuron 1 and Neuron 2 distributions of percent parameter estimation error for (A) r_{on} and (B) r_{off} . The Neuron 1 distributions of percent parameter estimation error for (C) q_{on}^i and (D) q_{off}^i . For each FL and EM case, data were collected over 1000 simulations lasting 10^6 time steps where for Neurons 1 and 2, r_{on} and r_{off} are uniformly selected in the range of 1-115 s^{-1} , and the q_{on}^i and q_{off}^i are uniformly selected in the range of 1-1000 spikes/s. To gauge the HMM simulation errors resulting from finite simulation times, the SIM distributions in (A)-(D) show the percent error of the parameters calculated from the hidden state, x_t , known to us but not the BSN, as well as the observations, s_t^i . In each box-whisker the gray box bounds the 25th and 75th percentiles of the data, the black horizontal line represents

the median and the black dashed whiskers span the extreme points within 1.5 times the interquartile range.

Figure 5: Estimation accuracy, computation time and time to convergence in the two neuron network. Comparison between FL and ML-EM for (A) % Hamming error distributions for Neuron 1 and Neuron 2, (B) P_{rms} distributions for Neuron 1 and (C) run-time distributions for Neuron 1 and Neuron 2 for the 0% and 400% initial parameter perturbation cases. FL6 and FL7 refer to FL simulations involving 10^6 and 10^7 time steps, respectively. EM refers to EM simulations involving 10^6 time steps. (D) Example of parameter estimation evolution for Neuron 1 for both FL (solid black) and ML-EM (solid gray) when the estimates begin 400% above the true parameter values. The dashed gray line indicates the true parameter values. (E) Distributions of the ‘real’ time to convergence for the FL and ML-EM algorithms for Neuron 1 for 100% and 400% initial parameter perturbation cases. For (E) all simulations lasted 10^6 time steps (i.e. 100s). For (A)-(E) simulation settings and box whisker details are the same as for Figure 4.

Figure 6: Three-layer feedforward network simulations using FL. The network layer 1, 2 and 3 distributions of percent parameter estimation error for (A) r_{on} and (B) r_{off} for the small network. (C) The layer 1 distributions of percent parameter estimation error for q_{on}^i and q_{off}^i for the small network. The layer 1, 2 and 3 distributions of percent Hamming error of the hidden state estimate for (D) the small network and (E) the large network with twice as many inputs in the higher layers. The layer 1, 2 and 3

distributions of informativeness for (F) the small network and (G) the large network. Data were collected over 1000 simulations lasting 10^6 time steps where r_{on} and r_{off} are uniformly selected in the range of 1-115 s^{-1} , and the q_{on}^i and q_{off}^i are uniformly selected in the range of 1-1000 spikes/s. Initial parameter estimates were also selected uniformly in the same ranges.

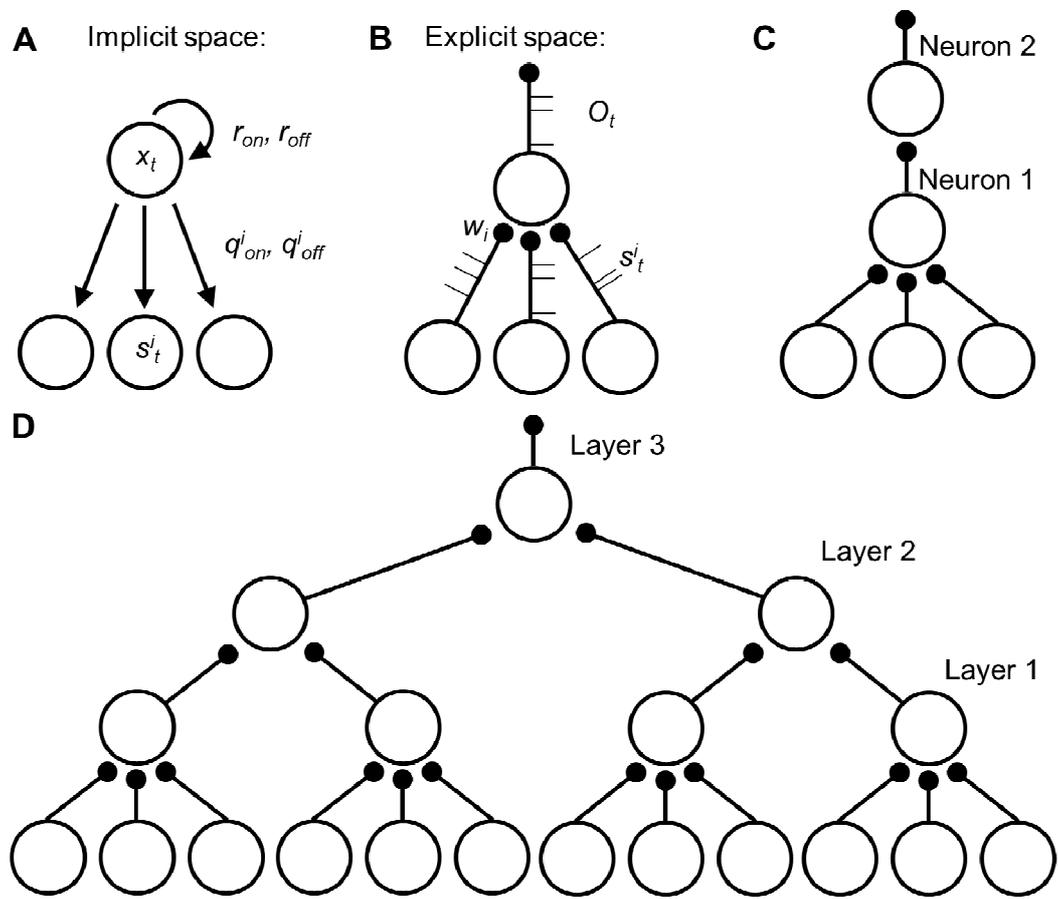


Figure 1

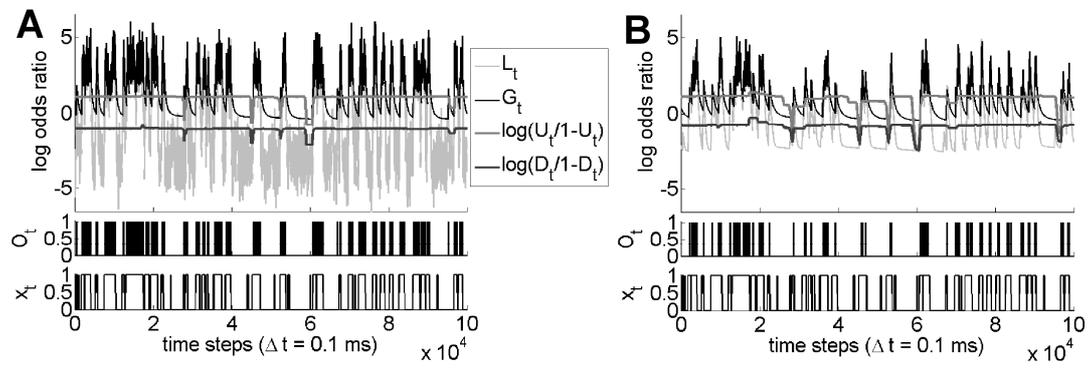


Figure 2

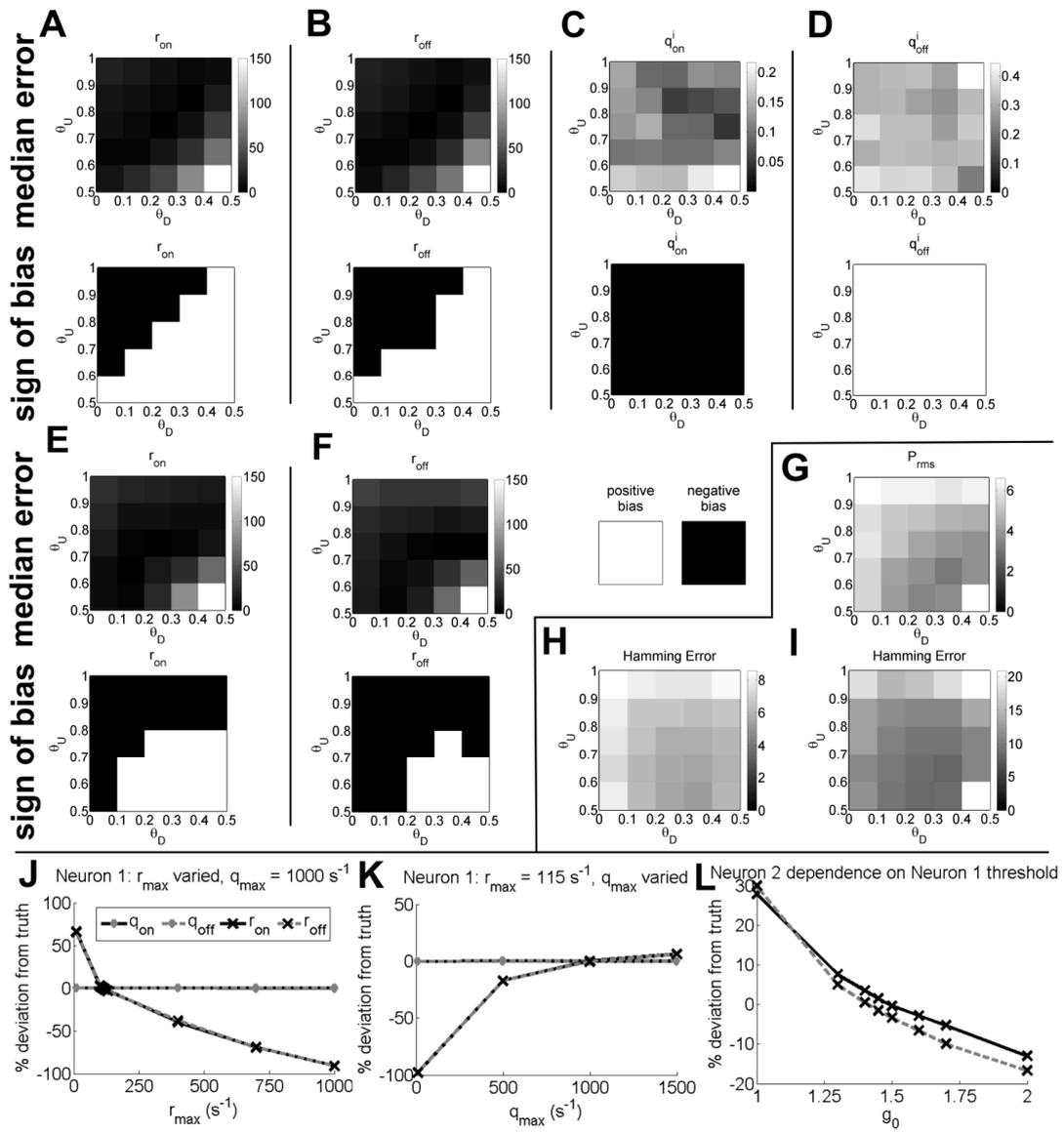


Figure 3

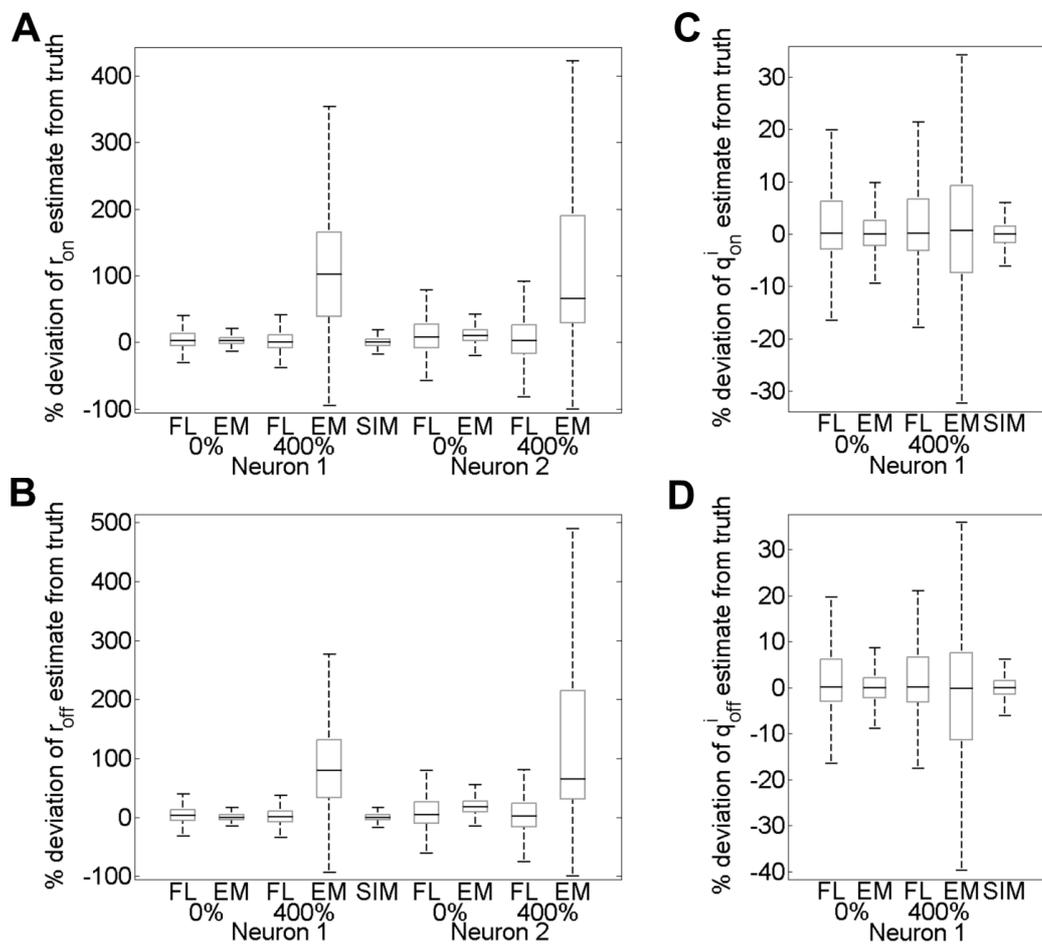


Figure 4

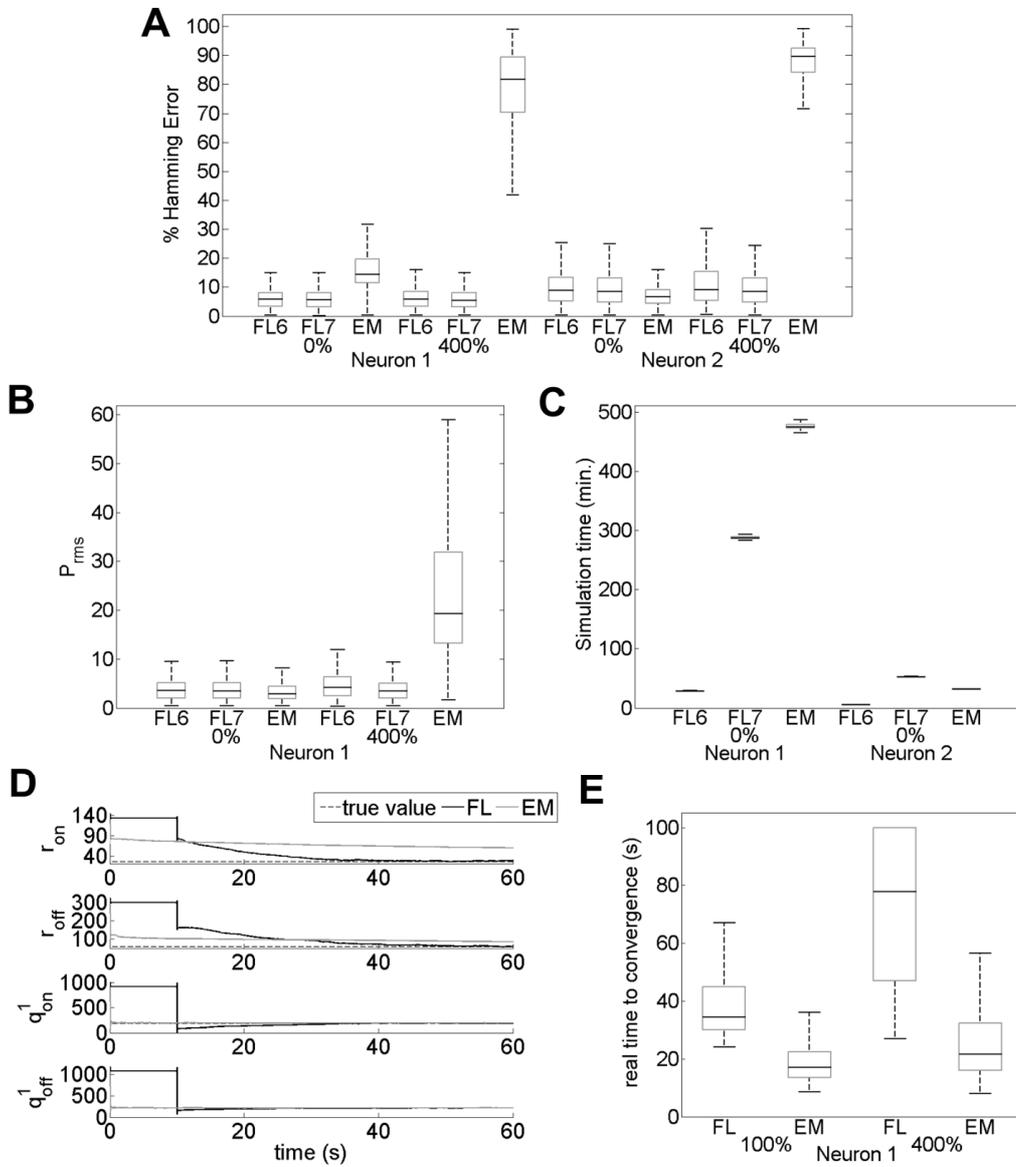


Figure 5

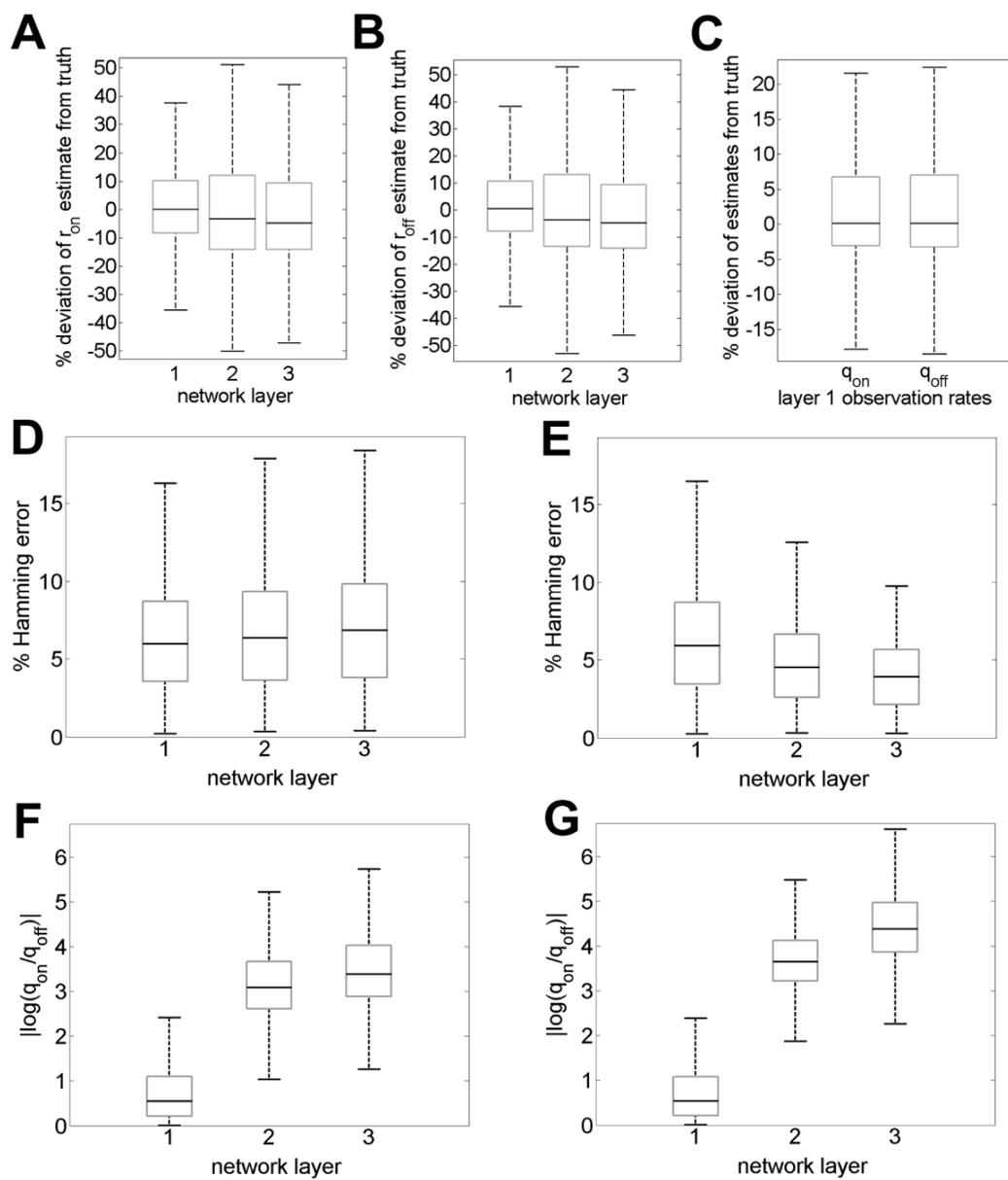


Figure 6