Active Learning of Functional Networks from Spike Trains

Honglei Liu[*] Bian Wu[†]

Abstract

Learning functional networks from spike trains is a fundamental problem with many critical applications in neuroscience. However, most of existing works focus on inferring the functional network purely from observational data, which could lead to undiscovered or spurious connections. We demonstrate that by adopting experimental data with interventions applied, the accuracy of the inferred network can be significantly improved. Nevertheless, doing interventions in real experiments is often expensive and must be chosen with care. Hence, in this paper, we design an active learning framework to iteratively choose interventions and learn the functional network. In particular, we propose two models, the variance model and the validation model, to effectively select the most informative interventions. The variance model works best to reveal undiscovered connections while the validation model has the advantage of eliminating spurious connections. Experimental results with both synthetic and real datasets show that when these two models are applied, we could achieve substantially better accuracy than using the same amount of observational data or other baseline methods to choose interventions.

1 Introduction

Spike trains are series of neural firing events, which are considered as the language neurons use to encode the external world and communicate with each other. Learning functional networks from spike trains is a fundamental problem with many critical applications in neuroscience. For example, a functional network that describes the temporal dependence relations among neurons is not only the first step to understand the function of neural circuits [9], but also has practical applications such as diagnosing neurodegenerative diseases [10].

Since *Generalized Linear Model* (GLM) is commonly used as a temporal generative model for spike trains [7, 13, 1], the routine [15, 9] of inferring functional networks from spike trains is shown in Figure 1 with an example. A spike train recording of 5 neurons is used to infer the GLM, from which a functional network is derived. The spike train dataset is a set of binary arrays, where "1" represents a firing event (spike) and "0" describes quiet state (no spike). Meanwhile, in the functional network, the edge between node 1 and node 4 with label "+1" represents an excitatory connection with time lag 1 (the firing of neuron 1 at time $t-1$ stimulates the firing of neuron 4 at time t). Similarly, the directed edge from node 4 to node 3 with label "-[1,20]" represents an inhibitory connection with time lags from 1 to 20 (the firings of neuron 4 at time $t-20$ through $t-1$ suppress the firing of neuron 3 at time t).

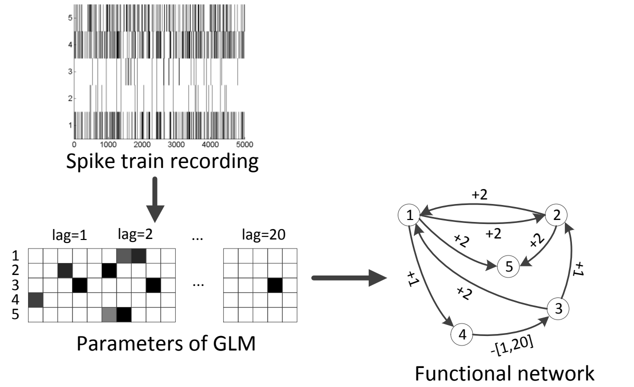
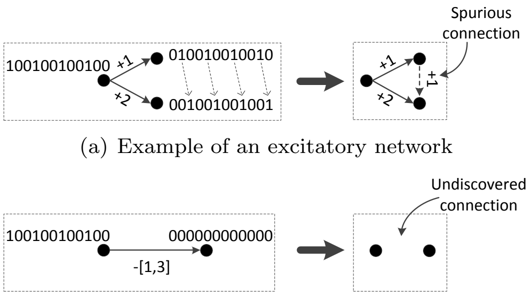


Figure 1: An example of inferring a functional network from spike trains.

Despite the popularity of this approach, we can not rely on it to get accurate functional networks. To illustrate, we give two examples. Figure 2(a) shows spike trains from three neurons where the firings of two of them are being driven by another neuron with different time lags. When a functional network is inferred, the aforementioned algorithm could easily get confused and a spurious excitatory connection will be drawn in the resulted network. In another example shown in Figure 2(b), the activities of a neuron are suppressed by an inhibitory connection and thus, there are not enough evidence to infer the inhibitory connection. Unfortunately, most of existing works [15, 13, 11] suffer from this problem because they are learning functional networks from purely observational data. As we will demonstrate in Section 5, by adopting interventional data, in which we could selectively fix the states of some neurons, the accuracy of the inferred functional network could be significantly improved. However, conducting interventional experiments is often very expensive in terms of time

[*] University of California, Santa Barbara. honglei@cs.ucsb.edu.
[†] Washington State University. bian.wu@wsu.edu

and money, so the interventions must be chosen with care. Hence, in this paper, we focus on the problem of how to design an active learning framework that could utilize as few interventional experiments as possible to get the maximum accuracy gain when inferring a functional network.



(a) Example of an excitatory network

(b) Example of an inhibitory network

Figure 2: Examples of inferred networks with only observational data.

There are previous works [7, 8] that focused on the problem of selecting external stimuli for a better estimation of GLM. But their approach can not be directly used in our problem, because they only consider the case where there is just one neuron while we are interested in inferring functional connectivities among multiple neurons. Meanwhile, learning functional networks should not be confused with learning the structure of causal networks [12] (static or dynamic Bayesian networks). In structure learning of causal networks, possible topological structures are searched and evaluated based on a statistical score function such as Minimum Description Length (MDL) [5] and Bayesian Dirichlet equivalent (BDe) score [4]. In contrast, the structure and parameters of a functional network are jointly learned by inferring a temporal generative model. Several works [17, 3, 16] focused on the problem of active learning for structures of causal networks which is different from our functional network learning problem.

To the best of our knowledge, we are the first to propose active learning models for inferring functional networks from spike trains. Our active learning framework is shown in Figure 3. The functional network is iteratively updated by conducting interventional experiments. In each iteration, the next intervention is chosen based upon the results seen so far towards a full identification of the functional network. In particular, we introduce two models, the variance model and the validation model, to choose interventions that are most beneficial for learning the functional network.

The variance model (Section 3) uses a Gaussian distribution to approximate the posterior distribution of GLM parameters given the data. And then the intervention that can maximally reduce the expected entropy of the posterior distribution is chosen. In addition, we also propose an initialization method that takes higher order interactions into consideration, which could significantly improve the performance of the variance model. Meanwhile, the validation model (Section 4) has the objective to validate the most of our existing connections. It picks interventions by maximizing the expected probability of our current knowledge about the GLM parameters.

These two models represent two different strategies of choosing interventions. The variance model works best to discover hidden inhibitory connection, while the validation model focuses on eliminating spurious excitatory connections. Experimental results with both synthetic and real datasets show that when these two models are applied, we could achieve substantially better accuracy than using the same amount of observational data or other baseline methods to choose interventions.

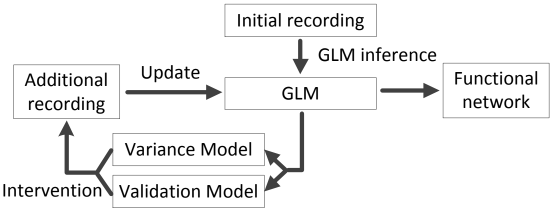


Figure 3: Pipeline of the active learning framework.

2 Preliminaries

In this section, we first briefly introduce the Generalized Linear Model (GLM) and then show the framework to infer GLM when both observational and interventional data are used. Table 1 summarizes some common notations that we are going to use in this paper.

2.1 Generalized Linear Model Let m denote the number of neurons being recorded and $x_{i,t}$ be the number of spikes of neuron i at time t. Usually in spike train data, there are at most one spike at any time point, so $x_{i,t}$ takes the value of 0 or 1. We assume $x_{i,t}$ depends on all the neurons' activities in a history window that spans from time $t - maxlag$ to time $t - minlag$, where $minlag$ and $maxlag$ are the minimum and maximum time lags we consider. Let $\theta_{i,j,t}$ be the parameter that models the effect from neuron j to neuron i at time lag l. For any neuron i, it also has a spontaneous firing rate which is controlled by a bias term b_i. We first model the instantaneous firing rate of of neuron i at time t, $\lambda_{i,t}$, as follows,

$$\lambda_{i,t} = e^{(b_i + \sum_{j=1}^{m} \sum_{l=minlag}^{maxlag} \theta_{i,j,l} x_{j,t-l})}. \tag{2.1}$$

Table 1: Notations

Notations	Description
m	Number of neurons
$minlag$	Minimum time lag to consider
$maxlag$	Maximum time lag to consider
h	$maxlag - minlag + 1$
n	$m \times h + 1$
T	Length of recordings
$x_{i,t}$	The state of neuron i at time point t
$\theta_{i,j,l}$	Parameters for the effect from neuron j to neuron i with time lag l
b_i	The bias term for neuron i which controls the spontaneous firing rate
$\boldsymbol{s_t}$	Input vector at time t with dimensions $n \times 1$
$\boldsymbol{r_t}$	Response vector at time t with dimensions $m \times 1$
$\boldsymbol{s_{1:t}}$	A matrix of input vectors from time point 1 to time point t, with dimensions $n \times t$
$\boldsymbol{r_{1:t}}$	A matrix of response vectors from time point 1 to time point t, with dimensions $m \times t$
$\boldsymbol{W}$	Parameters of GLM as a matrix with dimensions $m \times n$
$\boldsymbol{W}(i, \cdot)$	The i^{th} row of matrix W
$\boldsymbol{w}$	Flattened copy of matrix W

We then assume that $x_{i,t}$ is drawn from a Poisson distribution with mean $\lambda_{i,t}$. In other words, we assume that the firing of neurons follows a Poisson process which is a common assumption [1, 15, 13]. Hence, the log-likelihood for the observation of neuron i at time t, $\log L_{i,t}$, is calculated as

$$\log L_{i,t} = \log p(x_{i,t}|\lambda_{i,t}) = x_{i,t} \log \lambda_{i,t} - \lambda_{i,t}. \tag{2.2}$$

The log-likelihood for all the observations in a recording with length T is

$$\log L = \sum_{i=1}^{m} \sum_{t=maxlag}^{T} \log L_{i,t}. \tag{2.3}$$

To simplify our analysis later, we rewrite the log-likelihood function in matrix format. First, for a recording with T time points, we reconstruct it into an input matrix $\boldsymbol{s_{1:t}}$ and a response matrix $\boldsymbol{r_{1:t}}$, where $t = T - maxlag$. $\boldsymbol{s_{1:t}}$ is a $n \times t$ matrix with each column $\boldsymbol{s_k}$ representing the input vector at time point k, where $n = m \times h + 1$ and $h = maxlag - minlag + 1$. Similarly, $\boldsymbol{r_{1:t}}$ is an $m \times t$ matrix with each column $\boldsymbol{r_k}$ representing the response vector at time point k. $\boldsymbol{s_k}$ and $\boldsymbol{r_k}$ are constructed as follows.

$$\boldsymbol{s_k} = \begin{pmatrix} 1 \\ x_{1,k-minlag} \\ \vdots \\ x_{m,k-minlag} \\ \vdots \\ x_{1,k-maxlag} \\ \vdots \\ x_{m,k-maxlag} \end{pmatrix}, \boldsymbol{r_k} = \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{m,k} \end{pmatrix}$$

We also rewrite the parameters of GLM as a matrix $\boldsymbol{W}$ with dimensions $m \times n$. Each row in W contains the parameters to predict responses of one neuron. For example, $\boldsymbol{W}(i, \cdot)$ contains the parameters responsible for the response of neuron i, where $\boldsymbol{W}(i, \cdot)$ denotes the i^{th} row of $\boldsymbol{W}$. $\boldsymbol{W}$ is constructed as follows.

$$\boldsymbol{W} = \begin{pmatrix} b_1 & \dots & b_m \\ \theta_{1,1,minlag} & & \theta_{m,1,minlag} \\ \vdots & & \vdots \\ \theta_{1,m,minlag} & & \theta_{m,m,minlag} \\ \vdots & & \vdots \\ \theta_{1,1,maxlag} & & \theta_{m,1,maxlag} \\ \vdots & & \vdots \\ \theta_{1,m,maxlag} & \dots & \theta_{m,m,maxlag} \end{pmatrix}^T$$

Following Eq. (2.1), (2.2) and (2.3), the log-likelihood function in matrix format is

$$\log L(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}}) = sum(\boldsymbol{r_{1:t}} \circ (\boldsymbol{W} \cdot \boldsymbol{s_{1:t}}) - e^{\boldsymbol{W} \cdot \boldsymbol{s_{1:t}}}),$$

where sum is a function that sums over all the elements in a matrix and $\circ$ represents Hadamard product which is essentially element-wise multiplication.

2.2 Active Learning of GLM Given a recording with input matrix $\boldsymbol{s_{1:t}}$ and response matrix $\boldsymbol{r_{1:t}}$, to learn the GLM, we use batch gradient ascent to infer the parameters that maximize the log-likelihood function. The gradients with respect to $\boldsymbol{W}$ are calculated as

$$\begin{aligned} D(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}}) &= \frac{\partial \log L(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})}{\partial \boldsymbol{W}} \\ &= \boldsymbol{r_{1:t}} \cdot \boldsymbol{s_{1:t}^T} - e^{\boldsymbol{W} \cdot \boldsymbol{s_{1:t}}} \cdot \boldsymbol{s_{1:t}^T}, \end{aligned} \tag{2.4}$$

where $D(\boldsymbol{W}, \boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})$ is a $m \times n$ matrix.

In the active learning framework, the interventional experiments are conducted iteratively to update the GLM. Let $I = \{\iota_1, \iota_2, ..., \iota_c\}$ be the set of interventions we can choose from. In this paper, we focus on deterministic interventions which means ι_i defines an action of forcing one or several neurons to take a fixed state. For example, ι_i could represent silencing one neuron i. Let Q be the set of neurons that are intervened. Q is empty when only observational data is used. Intuitively, for any neuron q in Q, its state will no longer depend on its parents in the functional network. So when $\boldsymbol{W}$ is being updated, the parameters that are responsible for the response of this neuron will not be changed.

Assuming n recordings has been collected and Q_i is the set of neurons that are intervened in the i^{th} recording. We first calculate the gradients $\boldsymbol{D_i}$ for the i^{th} recording with Eq. (2.4), and then for all the $q \in Q_i$, we set $\boldsymbol{D}_{q,\cdot} = 0$, where $\boldsymbol{D}_{q,\cdot}$ is the q^{th} row in $\boldsymbol{D}$. Eventually, the gradients for all the n recordings are calculated as follows,

$$\boldsymbol{D} = \sum_{i=1}^{n} \boldsymbol{D_i}. \tag{2.5}$$

In summary, the pipeline of the active learning framework is as follows: Given n recordings we have seen so far, infer the GLM using batch gradient ascent (Eq. (2.5)); Then choose an intervention from I and conduct the intervention experiment to collect the $(n+1)^{th}$ recording; Repeat this procedure until the budget for doing experiments has run out. In the following sections, we introduce two models to intelligently choose the next intervention.

3 Variance Model

By conducting interventional experiments, previously undiscovered connections could be revealed. However, how to choose the most informative intervention is still a hard problem to be solved. In this section, we propose the *variance model* to choose interventions based on the following intuitions: (1) Inhibitory connections tend to be undiscovered due to lack of evidence; (2) Lacking of evidence means high uncertainty about our knowledge of the inferred functional network; (3) The uncertainty about our knowledge of the inferred functional network could be quantified as the entropy of the posterior probability distribution of the parameters given the data. Moreover, we also introduce an initialization method that takes higher order interactions into consideration, which proves to be very effective for further improving the performance of the variance model.

3.1 Choose Interventions Assuming we have a recording of t time points which is formalized as an input-output pair $(\boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})$. Let $\boldsymbol{w}$ be the flattened copy of the GLM parameter matrix $\boldsymbol{W}$. Our knowledge about $\boldsymbol{w}$ can be summarized by the posterior probability distribution $p(\boldsymbol{w}|\boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})$ and the entropy of $p(\boldsymbol{w}|\boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}})$, $H(p(\boldsymbol{w}|\boldsymbol{s_{1:t}}, \boldsymbol{r_{1:t}}))$, quantifies the uncertainty of our knowledge. Our goal is to choose the intervention that can maximally reduce the uncertainty.

In this paper, we focus on deterministic interventions which gives us the ability to assume the next input vector with intervention, $\boldsymbol{s_{t+1}}$, is uniquely defined by the intervention type chosen from I. Now the problem can be formalized as choosing $\boldsymbol{s_{t+1}}$ such that the entropy of $p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$ can be maximally reduced. Since the response vector $\boldsymbol{r_{t+1}}$ is unknown, we use the expected entropy instead and the objective of the variance model is

$$\arg\min_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})), \tag{3.6}$$

where $\boldsymbol{r_{t+1}}$ is the response vector for $\boldsymbol{s_{t+1}}$.

However, it's difficult to directly compute and optimize the expected entropy exactly. Since the likelihood function of $\boldsymbol{w}$ also belongs to the exponential family, we approximate
$p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$ as a Gaussian distribution,

$$\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}} \sim \mathcal{N}(\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}}),$$

where $\boldsymbol{u_{t+1}}$ and $\boldsymbol{C_{t+1}}$ denote the mean and covariance of $\boldsymbol{w}$ given $(\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$. Accordingly, we have the following theorem.

Theorem 3.1. *When $p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$ is approximated as a Gaussian distribution, we could solve the objective function (Eq.* (3.6)*) as*

$$\arg\max_{\boldsymbol{s_{t+1}}} (e^{\boldsymbol{W}\cdot\boldsymbol{s_{t+1}}})^T \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}), \tag{3.7}$$

Proof. First, we have

$$H(\mathcal{N}(\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})) = \frac{1}{2}\log|\boldsymbol{C_{t+1}}| + const,$$

where $|\boldsymbol{C_{t+1}}|$ represents the determinant of $\boldsymbol{C_{t+1}}$.

In order to calculate $\boldsymbol{C_{t+1}}$, we have

$$\boldsymbol{C_{t+1}^{-1}} = -\frac{\partial^2 \log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})}{\partial \boldsymbol{w}^2},$$

because the inverse covariance matrix equals to the second partial derivative of the log-Gaussian density function w.r.t. $\boldsymbol{w}$.

By expending $\log p(\boldsymbol{w}|\boldsymbol{u_{t+1}}, \boldsymbol{C_{t+1}})$ (see supplementary materials for more details), we can get

$$\begin{aligned} \boldsymbol{C_{t+1}^{-1}} &= \boldsymbol{C_t^{-1}} + F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}) \\ &= (\boldsymbol{s_{t+1}} \otimes \boldsymbol{I}) \cdot diag(e^{\boldsymbol{W}\cdot\boldsymbol{s_{t+1}}}) \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}), \end{aligned} \tag{3.8}$$

where $\otimes$ represents Kronecker product, *diag* is a function that takes all the elements of a matrix and reconstruct them into a diagonal matrix, $\boldsymbol{I}$ is a $m \times m$ identity matrix, and

$$F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}) = -\frac{\partial^2 \log p(\boldsymbol{r_{t+1}}|\boldsymbol{w}, \boldsymbol{s_{t+1}})}{\partial \boldsymbol{w}^2},$$

which is the Fisher information (the negative of the second derivative of the log likelihood with respect to $\boldsymbol{w}$). It's interesting to see that the Fisher information does not depend on the response vector $\boldsymbol{r_{t+1}}$.

Finally, Eq. (3.6) can be solved as

$$\begin{aligned}
&\operatorname*{arg\,min}_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})) \\
&= \operatorname*{arg\,max}_{\boldsymbol{s_{t+1}}} \log |\boldsymbol{C_t}^{-1} + F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}})| \\
&= \operatorname*{arg\,max}_{\boldsymbol{s_{t+1}}} tr(\log(\boldsymbol{I} + \boldsymbol{C_t} \cdot F(\boldsymbol{w}, \boldsymbol{s_{t+1}}, \boldsymbol{r_{t+1}}))) \\
&= \operatorname*{arg\,max}_{\boldsymbol{s_{t+1}}} (e^{\boldsymbol{W} \cdot \boldsymbol{s_{t+1}}})^T \cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}),
\end{aligned}$$

where *tr* is the function to calculate the trace of a matrix and $\boldsymbol{J}$ is a $m \times 1$ vector with ones in all its entries.

As we can see from Eq. (3.7), the expected entropy relies on the value of $\boldsymbol{W}$. We can use the expectation of $\boldsymbol{W}$ to eliminate this unknown variable. To simplify the calculation, we assume $\boldsymbol{W}(i, \cdot)$, the i^{th} row of $\boldsymbol{W}$, which contains the parameters to predict the responses of neuron i, also follows a Gaussian distribution $\mathcal{N}(\boldsymbol{u_t^i}, \boldsymbol{C_t^i})$. $\boldsymbol{u_t^i}$ and $\boldsymbol{C_t^i}$ are subsets of $\boldsymbol{u_t}$ and $\boldsymbol{C_t}$ that correspond to the parameters in $\boldsymbol{W}(i, \cdot)$.

Now Eq. (3.6) becomes

$$\begin{aligned}
&\operatorname*{arg\,min}_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{W}} E_{\boldsymbol{r_{t+1}}} H(p(\boldsymbol{w}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})) \\
&\approx \operatorname*{arg\,max}_{\boldsymbol{s_{t+1}}} \\
&\begin{pmatrix} E_{\boldsymbol{W}(i,\cdot) \sim \mathcal{N}(\boldsymbol{u_t^1}, \boldsymbol{C_t^1})} e^{\boldsymbol{s_{t+1}}^T \cdot \boldsymbol{W}(i,\cdot)^T} \\ \vdots \\ E_{\boldsymbol{W}(i,\cdot) \sim \mathcal{N}(\boldsymbol{u_t^m}, \boldsymbol{C_t^m})} e^{\boldsymbol{s_{t+1}}^T \cdot \boldsymbol{W}(i,\cdot)^T} \end{pmatrix}^T \\
&\cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I_{m \times m}}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J_{m \times 1}}) \qquad (3.9) \\
&= \operatorname*{arg\,max}_{\boldsymbol{s_{t+1}}} \\
&\begin{pmatrix} e^{\boldsymbol{u_t^1} \cdot \boldsymbol{s_{t+1}} + \frac{1}{2} \boldsymbol{s_{t+1}^T} \cdot \boldsymbol{C_t^1} \cdot \boldsymbol{s_{t+1}}} \\ \vdots \\ e^{\boldsymbol{u_t^m} \cdot \boldsymbol{s_{t+1}} + \frac{1}{2} \boldsymbol{s_{t+1}^T} \cdot \boldsymbol{C_t^m} \cdot \boldsymbol{s_{t+1}}} \end{pmatrix}^T \\
&\cdot (\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J}),
\end{aligned}$$

Eq. (3.9) consists of two terms. The first term is a $1 \times m$ vector and the second term is a $m \times 1$ vector.

From Eq. (3.9), we can get some intuitions about the variance model. The term $e^{\boldsymbol{u_t^i} \cdot \boldsymbol{s_{t+1}}}$ indicates that the model is trying to find the interventions that can increase the activities of the neurons so that previously undiscovered connection would have a higher chance of get revealed. The term $(\boldsymbol{s_{t+1}}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_t} \cdot (\boldsymbol{s_{t+1}} \otimes \boldsymbol{J})$ indicates that the model will give larger weights to the interventions that have influences on the connections with higher variance.

3.2 Update $\boldsymbol{u}$ and $\boldsymbol{C}$ Without losing generality, we assume i recordings has been seen so far and $\boldsymbol{C_i}$ corresponds to the most updated covariance matrix. When the $(i+1)^{th}$ recording comes, we show how to calculate $\boldsymbol{u_{i+1}}$ and $\boldsymbol{C_{i+1}}$. When $i = 0$, we use $\boldsymbol{C_0}$ to denote the initial covariance matrix. In the next section, we will show how to initialize $\boldsymbol{C_0}$ to take higher order interactions into consideration.

Since the log-likelihood function of GLM and the log-Gaussian density function are both concave, every time a new recording comes, we just redo the inference with the method introduced in Section 2.2 and use the inferred $\boldsymbol{w}$ to approximate $\boldsymbol{u_{i+1}}$. Given $\boldsymbol{u_{i+1}}$ and $\boldsymbol{C_i}$, we use Eq. (3.8) to update $\boldsymbol{C_{i+1}}$.

3.3 Initialization When calculating the covariance matrix with the initial recording, we could just set $\boldsymbol{C_0}$ to be an identity matrix. We refer to this method as the *basic variance model*. However, we demonstrate that the performance of the variance model can be further improved by proposing a heuristic initialization method that considers higher order connections.

A deeper analysis about how we update $\boldsymbol{C}$ gives us the following theorem.

THEOREM 3.2. *When $\boldsymbol{C}$ is initialized as an identity matrix and being updated according to equations* (3.8), *$\forall\ i \neq j$, $i \in [1, m]$, $j \in [1, m]$, $k \in [1, n]$ and $c \in [1, n]$, the covariance between $\boldsymbol{W}(i, k)$ and $\boldsymbol{W}(j, c)$ will always equal to 0, where $\boldsymbol{W}(i, k)$ is the GLM parameter in i^{th} row and k^{th} column of $\boldsymbol{W}$ (similarly for $\boldsymbol{W}(j, c)$).*

Proof. According to Eq. (3.8), we have

$$\boldsymbol{C} = (\boldsymbol{C_0}^{-1} + (\boldsymbol{s} \otimes \boldsymbol{I}) \cdot diag(e^{\boldsymbol{W} \cdot \boldsymbol{s}}) \cdot (\boldsymbol{s}^T \otimes \boldsymbol{I}))^{-1}$$

By applying the Sherman-Morrison-Woodbury formula, we get

$$\begin{aligned}
&\boldsymbol{C} = \boldsymbol{C_0} - \boldsymbol{C_0} \cdot (\boldsymbol{s} \otimes \boldsymbol{I}) \cdot \\
&(diag(e^{\boldsymbol{W}\boldsymbol{s}})^{-1} + (\boldsymbol{s}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_0} \cdot (\boldsymbol{s} \otimes \boldsymbol{I}))^{-1} \cdot (\boldsymbol{s}^T \otimes \boldsymbol{I}) \cdot \boldsymbol{C_0}
\end{aligned}$$

When $\boldsymbol{C_0}$ is initialized as an identity matrix, we can prove Theorem 3.2 by carrying out matrix operations.

The intuition behind Theorem 3.2 is that the parameters responsible for different neurons (different rows in $\boldsymbol{W}$) are independent with each other. As an example shown in Figure 4, two connections form a chain and the covariance between their corresponding parameters will not be updated. However, this chain represents higher order interactions in the functional network. Taking them into consideration is beneficial when choosing interventions. Accordingly, we propose a heuristic initialization method that proves to be working very well.

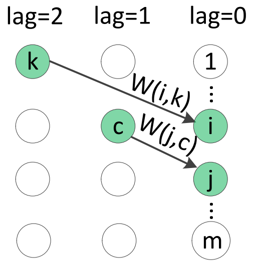


Figure 4: An example of higher order interactions.

We first calculate the average firing rates, $(a_1, a_2 \ldots a_m)$, for all the neurons using the initial recording. Then an input vector $\boldsymbol{s}$ is constructed by using the average firing rates as the values for each neuron in all time lags. For any two parameters $\boldsymbol{W}(i,k)$ and $\boldsymbol{W}(j,c)$ where $i \neq j$, let $C_{\boldsymbol{W}(i,k)-\boldsymbol{W}(j,c)}$ denote their covariance. We initialize this value as follows,

$$C_{\boldsymbol{W}(i,k)-\boldsymbol{W}(j,c)} = \frac{1}{a_k a_c e^{\boldsymbol{W}(\boldsymbol{i},\cdot)\cdot \boldsymbol{s}} e^{\boldsymbol{W}(\boldsymbol{j},\cdot)\cdot \boldsymbol{s}}}$$

where we use the most updated $\boldsymbol{u}$ to approximate $\boldsymbol{W}$. This initialization method is designed to follow the intuition that more information indicates smaller (co)variance. Here, the amount of information is quantified by the average or predicted firing rate.

4 Validation Model

The variance model works pretty well for many cases. However, it still has some weaknesses. For example, in Figure 5, we have three neurons connected in a chain with spontaneous firing rates $(0.05, 0.0001, 0.0001)$ and the firings of neuron 2 and 3 are mainly driven by neuron 1. When a functional network is inferred, a spurious connection is likely to appear. Assuming we have the ability to silence one of the neurons, and our goal is to use the interventional data to maximally decrease the strength of the spurious connection. Using the variance model, neuron 3 will be picked to be silenced. Clearly, it's not the best option as when the state of neuron 3 is fixed, all the parameters for the incoming connections will not be updated. The variance model picks neuron 3 because it's trying to reduce the uncertainty about the parameters by increasing the neuronal activities of the whole network. Picking other neurons would reduce more activities than neuron 3. However, if we are able to pick neuron 2 as the target, the spurious connection will be filtered because the incoming connection that is driving the firing of neuron 3 is blocked and we will know whether there is a connection from neuron 1 to neuron 3 or not. So in some cases, the variance model is not picking the best interventions.

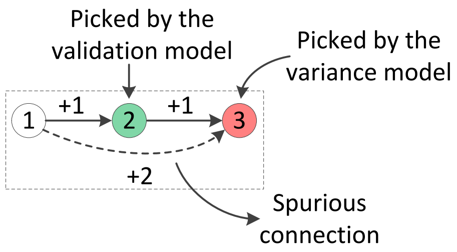


Figure 5: A chain network where the variance model picks neuron 3 and the validation model picks neuron 2.

Hence, we propose another model, called validation model, in accompany with the variance model. Instead of trying to increase activities of the system so that we can discover previously missed connections, the goal of the validation model is to maximally validate our existing knowledge about the functional network. Our current knowledge can be represented as the most updated GLM parameters, $\boldsymbol{u_t}$. For a new inverventional input vector $\boldsymbol{s_{t+1}}$, the objective is to maximally increase our confidence about $\boldsymbol{u_t}$, which is measured by $p(\boldsymbol{u_t}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}})$. The objective function is formalized as

$$\underset{\boldsymbol{s_{t+1}}}{\arg\max} \log p(\boldsymbol{u_t}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}}). \tag{4.10}$$

To have more intuitions about the validation model, consider a procedure of making decisions about the connections in a functional network given the GLM parameters. The significance of the parameters is measured by their posterior probabilities. Any parameter that has a posterior probability higher than a threshold will result in a connection in the functional network. By pursuing the objective function, we can increase our confidence about connections in the functional network or filter out spurious connections.

Since $\boldsymbol{r_{t+1}}$ is unknown, we use its expectation and rewrite the objective functions as follows,

$$
\begin{aligned}
&\arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} \log p(\boldsymbol{u_t}|\boldsymbol{s_{1:t+1}}, \boldsymbol{r_{1:t+1}}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} (\log \boldsymbol{u_t} + \log p(\boldsymbol{r_{1:t}}|\boldsymbol{s_{1:t}}, \boldsymbol{u_t}) + \\
&\log p(\boldsymbol{r_{t+1}}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}) + const) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} E_{\boldsymbol{r_{t+1}}} \log p(\boldsymbol{r_{t+1}}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} E_{\boldsymbol{r_{t+1}(i)}} \log p(\boldsymbol{r_{t+1}(i)}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}),
\end{aligned}
\tag{4.11}
$$

where $\boldsymbol{r_{t+1}(i)}$ represents the i^{th} element in the response vector $\boldsymbol{r_{t+1}}$. When spike train data is considered, $\boldsymbol{r_{t+1}(i)}$ can only take the value of 0 or 1. So, we have

$$
\begin{aligned}
&\arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} \sum_{\boldsymbol{r_{t+1}(i)}=0,1} \log p(\boldsymbol{r_{t+1}(i)}|\boldsymbol{s_{t+1}}, \boldsymbol{u_t}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} (-e^{\boldsymbol{u_t^i s_{t+1}}} \cdot e^{-e^{\boldsymbol{u_t^i s_{t+1}}}} + \\
&(\boldsymbol{u_t^i s_{t+1}} - e^{\boldsymbol{u_t^i s_{t+1}}}) \cdot e^{\boldsymbol{u_t^i s_{t+1}}} \cdot e^{-e^{\boldsymbol{u_t^i s_{t+1}}}}) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} (e^{\boldsymbol{u_t^i s_{t+1}}} \cdot e^{-e^{\boldsymbol{u_t^i s_{t+1}}}} (\boldsymbol{u_t^i s_{t+1}} - e^{\boldsymbol{u_t^i s_{t+1}}} - 1)) \\
&= \arg\max_{\boldsymbol{s_{t+1}}} \sum_{i=1}^{m} \lambda_i \cdot e^{-\lambda_i} \cdot (\log \lambda_i - \lambda_i - 1),
\end{aligned}
\tag{4.12}
$$

where $\lambda_i = e^{\boldsymbol{u_t^i s_{t+1}}}$ and $\boldsymbol{u_t^i}$ represents the parameters in $\boldsymbol{u_t}$ that are responsible for the response of neuron i in a row vector.

5 Experiments

In this experimental study, we use both synthetic and real spike train data sets to test the effectiveness of our active learning models. All the computations are conducted on a server with 2.67GHz Intel Xeon CPU (32 cores) and 1TB RAM.

5.1 Data Sets

Interventions. A very recent equipment called Neuronal Circuit Probe (NCP) was developed to do interventional experiments when recording spike trains from neurons. NCP could locate a single neuron and deliver drugs locally to this neuron. In our experiments, a drug that could silence neurons is used. In other words, assuming we have m neurons being recorded, there are m types of interventions we can do with each one corresponding to fixing the sate of a neuron to 0.

Synthetic data. We use three steps to generate simulated spike train data. First, the structure of the functional network is proposed. Then, a GLM parameter matrix is created according to the functional network. Finally, simulated spike train data is generated by running the GLM. If a neuron is intervened in the simulated experiment, its value will be always set to 0. We use 1 millisecond as the time bin in the recordings and each recording has a length of 20 seconds which are 20,000 data points. All the parameters in the simulation process are chosen to mimic real neurons. Due to space constraints, more details about the synthetic data could be found in the supplementary materials.

Real data. We use a Multielectrode Array (MEA) with 120 channels to record signals from neurons on a culture. Each channel corresponds to a node in the functional network we want to learn. We use 1 millisecond as the time bin to discretize neuronal signals to ensure there is at most 1 spike at each time bin.

The spike train recordings can be divided into two categories: observational recording and interventional recording. For the observational recording, the neurons are recorded without any drug deliveries. For the interventional recording, the neurons are recorded while drugs that can silence neurons are delivered at channels selected by different methods. Each interventional experiment is conducted after the neurons have fully recovered from the previous experiment. We use an observational recording with 60 seconds as the initial recording and each additional recording has a length of 20 seconds. Finally, another 60 seconds observational recording is reserved as the test set.

5.2 Evaluation

Methods. To illustrate the effectiveness of our active learning models, we compare our approaches with several baselines. The models we have proposed could be organized as four approaches: (1) ***Basic variance model***. The variance model using identity matrix as initialization; (2) ***Variance model***. The variance model using our initialization method; (3) ***Validation model***; (4) ***Mixture***. Alternately using *variance model* and *validation model* to choose interventions. We use two baselines to compare with: (1) ***Extend***. Simply adding more observational recordings without any interventions. (2) ***Firing rate***. Choosing the neuron that has the highest firing rate as the intervention target.

Metrics. For the synthetic datasets, since we have the ground truth which is the GLM parameter matrix $\boldsymbol{W}$, the inferred $\bar{\boldsymbol{W}}$ is directly compared with $\boldsymbol{W}$. The Frobenius norm of their difference is used to characterize the error of the inferred model,

$$e = \|\bar{\boldsymbol{W}} - \boldsymbol{W}\|_F.$$

Since we want to repeat our tests with different experimental settings (structure of the functional networks and parameters of the GLM) and report the average, we need to normalize the errors. Let $E = \{e_1, e_2, ..., e_c\}$ denote the set of errors when different number of recordings and different models are used. We normalize e_i as

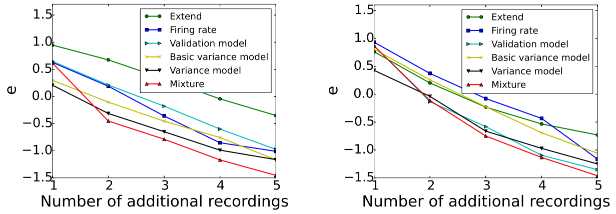
$$\frac{e_i - u}{\sigma},$$

where u is the mean of E and σ is the standard deviation of E.

For real datasets, since we don't have the ground truth, we reserve some observational recordings as the test set, and use the predictive ability of the inferred model to measure its accuracy. So, the negative log-likelihood on the test dataset is used as the evaluation metric. A lower negative log-likelihood means the inferred model is more accurate.

5.3 Random Networks To test the effectiveness of our models, we conduct simulated experiments with random networks of different sizes. Given the size of the network, we randomly generate 10 networks and report the average of the normalized errors. All the simulated experiments are done interactively which means every time a new additional recording is added, the intervention models are re-calculated to pick the next intervention.

We first test the case when the functional network contains 10 nodes. As shown in Figure 6(a), for all the methods, when more additional recordings are added, the inferred model is getting more accurate. However, when *Mixture* is used to guide the intervention experiments, we can achieve the most accuracy gains. Another observation is that the variance model works better than the basic variance model because of our initialization method. It's worth mentioning that the validation model is not working very well because the size of the network is too small such that there are not a lot of spurious connections when the network is inferred.



(a) Random networks with 10 nodes.

(b) Random networks with 20 nodes.

Figure 6: Averages of normalized errors with random networks.

We then increase the size of the random networks to 20 nodes and redo the experiments. As shown in Figure 6(b), the variance model, the validation model and the mixture method achieves the best results. The variance model shows consistent advantages over other models. The validation model shows a huge performance improvement compared to the previous experiment for the reason that the size of the random networks is larger and more spurious connections will be eliminated by the validation model. Interestingly, when the interventions are chosen by firing rate, it performs even worse than simply adding observational recordings.

5.4 Real Data For biological reasons the neurons can not be recorded for too long. So, in real experiments, instead of choosing interventions interactively, we use batch experimental design. An initial recording of 60 seconds is collected to train the GLM and intervention models. Then a ranking of interventions is generated by each intervention model. We use this ranking without updating it to guide following experiments. We also use the negative log-likelihood on a test set with a recording of 60 seconds to measure the accuracy of the inferred model.

As shown in Figure 7, the variance model, the validation model and the mixture method could achieve lower negative log-likelihood (higher accuracy) than simply extending observational recording or picking interventions according to firing rate. The performance gain of our models over the *Firing rate* method is not as obvious in the second intervention experiment as in the first one. The reason may be because we are not able to update our intervention models by using the new recording. When the experiments can be done interactively, more accuracy gain will be achieved.

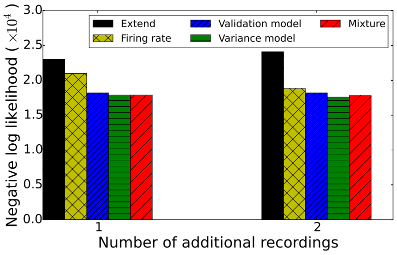


Figure 7: Evaluation by real data.

6 Related Work

The problem of learning functional networks should not be confused with the problem of learning the structure of a causal network [12] (static or dynamic Bayesian networks). In structure learning of static or dynamic causal Bayesian networks, numerous works [2, 19, 18, 4, 5] have been proposed. However, these works focus on how to efficiently search the structure space or how to evaluate the the proposed structure.

When learning a static causal Bayesian network, it can be proved that given only observational data, we can not differentiate networks in a Markov equivalence class, in which the networks have the same skeleton but may have different directions for some edges [14]. So some researchers [3, 16] try to tackle this problem with an active learning framework. In these methods, they will choose interventions that can orientate most edges. Another work [17] based on active learning framework keeps a distribution of possible structures and choose interventions that can maximally reduce the entropy of this distribution, but the ordering of nodes needs to be given.

We study the problem of learning functional networks. The structure and parameters are jointly learned by inferring a Generalized Linear Model. GLM is widely used in spike train analysis, but most works [15, 13] focus on learning GLM from observational data. J. Lewi *et al* proposes methods [8, 6, 7] to select external stimuli for a better estimation of GLM when there is only one neuron. However, we are interested in modeling interactions among multiple neurons, which is a different problem.

7 Conclusions

In this work, we study the problem of learning functional networks from spike trains in an active learning setting. In particular, we propose two models, the variance model and the validation model, to choose the most informative intervention so that we can get the maximum accuracy gain for the inferred network. Our experimental results with both synthetic and real data show that by applying our approaches, we could achieve substantially better accuracy than using the same amount of observational data or other baseline methods to choose interventions.

8 Acknowledgement

We would like to thank Prof. Paul K. Hansma, Connor Randall and Daniel Bridges for collecting the data. We also want to thank Prof. Xifeng Yan and Prof. Kenneth S. Kosik for the useful discussions. This research was sponsored in part by NSF IIS 0954125. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein.

References

[1] Z. Chen, *An overview of bayesian methods for neural spike train analysis*, Computational intelligence and neuroscience, 2013 (2013), p. 1.

[2] N. Dojer, *Learning bayesian networks does not have to be np-hard*, in Mathematical Foundations of Computer Science 2006, Springer, 2006, pp. 305–314.

[3] Y.-B. He and Z. Geng, *Active learning of causal networks with intervention experiments and optimal designs*, Journal of Machine Learning Research, 9 (2008).

[4] D. Heckerman, D. Geiger, and D. M. Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, Machine learning, 20 (1995), pp. 197–243.

[5] W. Lam and F. Bacchus, *Learning bayesian belief networks: An approach based on the mdl principle*, Computational intelligence, 10 (1994), pp. 269–293.

[6] J. Lewi, R. Butera, and L. Paninski, *Sequential optimal design of neurophysiology experiments*, Neural Computation, 21 (2009), pp. 619–687.

[7] J. Lewi, R. J. Butera, and L. Paninski, *Efficient active learning with generalized linear models*, in International Conference on Artificial Intelligence and Statistics, 2007, pp. 267–274.

[8] J. Lewi, D. M. Schneider, S. M. Woolley, and L. Paninski, *Automating the design of informative sequences of sensory stimuli*, Journal of computational neuroscience, 30 (2011), pp. 181–200.

[9] S. Linderman, C. H. Stock, and R. P. Adams, *A framework for studying synaptic plasticity with neural spike train data*, in Advances in Neural Information Processing Systems, 2014, pp. 2330–2338.

[10] P. D. Maia and J. N. Kutz, *Compromised axonal functionality after neurodegeneration, concussion and/or traumatic brain injury*, Journal of computational neuroscience, 37 (2014), pp. 317–332.

[11] D. Patnaik, S. Laxman, and N. Ramakrishnan, *Discovering excitatory networks from discrete event streams with applications to neuronal spike train analysis*, in Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, IEEE, 2009, pp. 407–416.

[12] J. Pearl, *Causality: models, reasoning and inference*, Economet. Theor, 19 (2003), pp. 675–685.

[13] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. Chichilnisky, and E. P. Simoncelli, *Spatio-temporal correlations and visual signalling in a complete neuronal population*, Nature, 454 (2008), pp. 995–999.

[14] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search*, MIT press, 2000.

[15] I. H. Stevenson, J. M. Rebesco, N. G. Hatsopoulos, Z. Haga, L. E. Miller, and K. P. Körding, *Bayesian inference of functional connectivity and network structure from spikes*, Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 17 (2009), pp. 203–213.

[16] M. Steyvers, J. B. Tenenbaum, E.-J. Wagenmakers, and B. Blum, *Inferring causal networks from observations and interventions*, Cognitive science, 27 (2003), pp. 453–489.

[17] S. Tong and D. Koller, *Active learning for structure in bayesian networks*, in International joint conference on artificial intelligence, vol. 17, LAWRENCE ERLBAUM ASSOCIATES LTD, 2001, pp. 863–869.

[18] N. X. Vinh, M. Chetty, R. Coppel, and P. P. Wangikar, *Globalmit: learning globally optimal dynamic bayesian network with the mutual information test criterion*, Bioinformatics, 27 (2011), pp. 2765–2766.

[19] Z. Wang and L. Chan, *Using bayesian network learning algorithm to discover causal relations in multivariate time series*, in Data Mining (ICDM), 2011 IEEE 11th International Conference on, IEEE, 2011, pp. 814–823.