# From Imaging to Connectivity: A Probabilistic Graphical Model to Learn Neuronal Network Structure

**Alexis Cook**
*alexis_cook@brown.edu*

**Ivana Petrovic**
*ivana_petrovic@brown.edu*

**Guangyao (Stannis) Zhou**
*guangyao_zhou@brown.edu*

## 1 Introduction

The brain is a rich and dense network of neurons, and these networks are believed to be the biological substrate for memory, cognition, and perception. Thus, understanding the brain as a complex network of interacting components can provide insight into learning and intelligence. Mapping its structure could also help generate new treatment modalities for neuropathologies such as Alzheimer's disease and epilepsy.

Neuron-to-neuron communication operates electrochemically, and neuronal signals are well-known to generally be followed by cellular influx of calcium ions into the associated pre-synaptic cell. Calcium-sensitive fluorescent indicators are often used in experimental settings to simultaneously monitor the activity of tens of thousands of neighboring neurons. The purpose of our project is to utilize time series of neural activity visualized with fluorescent indicators in neuronal culture, in order to reverse engineer underlying network connectivity (i.e. the wiring of the neurons).

We downloaded our data from the website `http://www.kaggle.com/c/connectomics`, which includes simulated optical recordings of $N = 100$ neurons, collected over a time period lasting one hour, at a sampling rate of $(\Delta t)^{-1} = 50$ Hz, so with $T = 179498$ equally spaced time points. Based on this data, our goal is to infer directed connections between neurons in the network.

## 2 Our Model

Assume we have $T$ observations of the fluorescence levels of $N$ neurons, which correspond to $T$ time intervals, each of which has length $\Delta t$. The data we have are $T$ vectors $F_1, \ldots, F_T \in \mathbb{R}^N$, where $F_{t,i}$ represents the fluorescence level of neuron $i$ at time $t\Delta t$. The model we are going to use is a nonlinear state space model. At time $t\Delta t$, the observed random variable is the fluorescence levels of the $N$ neurons at time $t\Delta t$, i.e. $F_t \in \mathbb{R}^N$, and the hidden random variable is $(S_t, C_t)$. $S_t \in \mathbb{N}^N$, where $S_{t,i}$ corresponds to the number of spikes in the time interval $[(t-1)\Delta t, t\Delta t]$. $C_t \in \mathbb{R}^N$, where $C_{t,i}$ corresponds to the calcium concentrion level of neuron $i$ at time $t\Delta t$.

To model our system as a state-space model, we needed to define several components:

1. The initial distribution of the hidden variables, i.e. the distribution of $S_1$. Here we define a mean vector $\lambda_1 \in \mathbb{R}^N$, and let $S_{1,i} \sim \text{Poisson}(\lambda_{1,i})$.
2. The transition probability of the hidden variables. For this purpose, we define a connectivity matrix $W \in \mathbb{R}^{N \times N}$ and a vector $b \in \mathbb{R}^N$, and let $S_{t,i} \sim \text{Poisson}\left(e^{b_i + \sum_j W_{ji} S_{t-1,j}}\right)$, $t = 2, \ldots, T$.

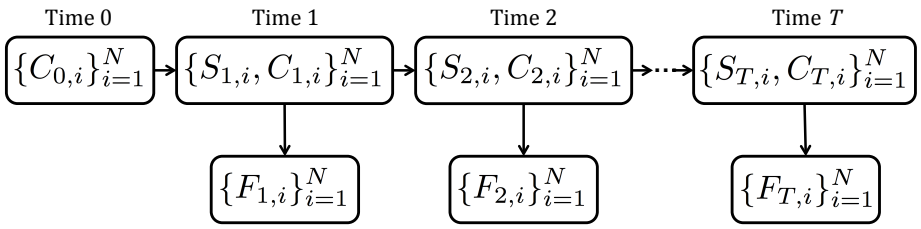


Figure 1: The model.

3. The relationship between $C_t$ and $S_t$. Here $C_{t,i}$ is deterministically determined by $S_{t,i}$ and $C_{t-1,i}$, and the relationship is given by the paper [2].

$$C_{t,i} = \left(1 - \frac{\Delta t}{\tau_{\text{Ca}}}\right) C_{t-1,i} + A_{\text{Ca}} S_{t,i}$$

with $C_{0,i}$'s being some unknown constants and are part of our parameters. Here $\tau_{\text{Ca}}$ and $A_{\text{Ca}}$ are two constants. Using the set-up from the paper [2], we set $\tau_{\text{Ca}} = 1s$, and $A_{\text{Ca}} = 50\mu M$.

4. The conditional distribution of the observed variable $F_t$ conditioned on the hidden variable $(S_t, C_t)$. Using the equations from [2], we have

$$F_{t,i} \sim N\left(\frac{C_{t,i}}{C_{t,i} + K_d}, \sigma_v^2\right)$$

where $K_d$ is a constant. We use $K_d = 300\mu M$ and $\sigma_v^2 = 0.03^2$ following [2].

Using this formulation, we see that the set of parameters in our model is:
$\lambda_1 \in \mathbb{R}^N, C_0 \in \mathbb{R}^N, b \in \mathbb{R}^N, W \in \mathbb{R}^{N \times N}, \tau_{\text{Ca}} \in \mathbb{R}, A_{\text{Ca}} \in \mathbb{R}, K_d \in \mathbb{R}$ and $\sigma_v^2 \in \mathbb{R}$.

If we fix values of the following parameters: $\tau_{\text{Ca}} = 1s, A_{\text{Ca}} = 50\mu M, K_d = 300\mu M$ and $\sigma_v^2 = 0.03^2$ (following [2]), then the unknown parameters in our model are:

$\lambda_1 \in \mathbb{R}^N, C_0 \in \mathbb{R}^N, b \in \mathbb{R}^N, W \in \mathbb{R}^{N \times N}$. We want to recover the connectivity between the neurons by estimating the connectivity matrix $W$. Hence our task is to estimate the parameters of this model.

This is a parameter estimation problem with incomplete observations, and we use the expectation-maximization (EM) algorithm. The joint likelihood of the hidden variables and the observed variables is given by

$$\begin{aligned}
& p(S, C, F | \lambda_1, b, C_{0,i}, W) \\
= & \prod_{i=1}^N \left[ p(S_{1,i}|\lambda_{1,i}) \prod_{t=2}^T p(S_{t,i}|S_{t-1}, b_i, W) \prod_{t=1}^T p(F_{t,i}|C_{t,i}) \right] \\
= & \prod_{i=1}^N \left[ \text{Poisson}(S_{1,i}|\lambda_{1,i}) \prod_{t=2}^T \text{Poisson}\left(S_{t,i} \left| e^{b_i + \sum_j W_{ji} S_{t-1,j}} \right.\right) \prod_{t=1}^T N\left(F_{t,i} \left| \frac{C_{t,i}}{C_{t,i} + K_d}, \sigma_v^2 \right.\right) \right]
\end{aligned}$$

And the log-joint-likelihood is given by

$$
\begin{aligned}
& \log p(S, C, F|\lambda_1, b, C_{0,i}, W) \\
= & \sum_{i=1}^N \left\{ S_{1,i} \log \lambda_{1,i} - \lambda_{1,i} + \log(S_{1,i}!) + \sum_{t=2}^T \left[ S_{t,i} \left( b_i + \sum_j W_{ji} S_{t-1,j} \right) - e^{b_i + \sum_j W_{ji} S_{t-1,j}} + \log(S_{t,i}!) \right] \right\} \\
+ & \sum_{i=1}^N \sum_{t=1}^T \left[ -\frac{1}{2} \log(2\pi\sigma_v^2) - \frac{1}{2\sigma_v^2} \left( F_{t,i} - \frac{C_{t,i}}{C_{t,i}+K_d} \right)^2 \right] \\
= & -\sum_{i=1}^N \lambda_{1,i} \\
+ & \sum_{i=1}^N \left\{ S_{1,i} \log \lambda_{1,i} + b_i \sum_{t=2}^T S_{t,i} + \sum_{t=1}^T \log(S_{t,i}!) + \sum_{t=2}^T \sum_j W_{ji} S_{t-1,j} S_{t,i} - \sum_{t=2}^T e^{b_i + \sum_j W_{ji} S_{t-1,j}} \right\} \\
- & \frac{NT}{2} \log(2\pi\sigma_v^2) - \frac{1}{2\sigma_v^2} \sum_{i=1}^N \sum_{t=1}^T \left( F_{t,i} - \frac{C_{t,i}}{C_{t,i}+K_d} \right)^2
\end{aligned}
$$

### 2.1 E-step

We can carry out the E-step with particle filter. Assuming we use $M$ particles, our goal in the E-step is to get samples of the hidden variables given the observed variables and the parameter values at the current iteration. W.l.o.g., we assume the parameters at the current iteration are $\lambda_1, C_0, b, W$. The particle filter works as follows

1. Sample $S_{1,i}^{(m)}$ i.i.d. from $\text{Poisson}(\lambda_{1,i})$, $m = 1, \ldots, M$. Get $M$ $N$-dimensional vectors $S_1^{(1)}, \ldots, S_1^{(M)}$. Calculate the corresponding $M$ $N$-dimensional vectors $C_1^{(1)}, \ldots, C_1^{(M)}$.
2. Calculate the unnormalized posterior weights of our partices

$$\tilde{w}_1^{(m)} = \prod_{i=1}^N N\left( F_{1,i} \middle| \frac{C_{1,i}^{(m)}}{C_{1,i}^{(m)} + K_d}, \sigma_v^2 \right)$$

   And normalize the weights $w_1^{(m)} = \frac{w_1^{(m)}}{\sum_{\tilde{m}=1}^M w_1^{(\tilde{m})}}$.
3. Resample the particles using the weight, and propagate the samples using the dynamics

$$S_{t,i}^{(m)} \sim \text{Poisson}\left( e^{b_i + \sum_j W_{ji} S_{t-1,j}^{(m)}} \right)$$

   And calculate the weights.
4. Repeat this process.

### 2.2 M-step

To induce the sparsity in the network (which is biologically justified), in the M-step, we also add a $L_1$ regularization term. As a result, we need to maximize

$$
\begin{aligned}
& \mathbb{E} \log p(S, C, F|\lambda_1, b, C_{0,i}, W) \\
= & -\sum_{i=1}^N \lambda_{1,i} + \sum_{i=1}^N \left\{ \log \lambda_{1,i} \mathbb{E} S_{1,i} + b_i \sum_{t=2}^T \mathbb{E} S_{t,i} - \sum_{t=1}^T \mathbb{E} \log(S_{t,i}!) + \sum_{t=2}^T \sum_j W_{ji} \mathbb{E} S_{t-1,j} S_{t,i} - \sum_{t=2}^T \mathbb{E} e^{b_i + \sum_j W_{ji} S_{t-1,j}} \right\} \\
- & \frac{NT}{2} \log(2\pi\sigma_v^2) - \frac{1}{2\sigma_v^2} \sum_{i=1}^N \sum_{t=1}^T \mathbb{E} \left( F_{t,i} - \frac{C_{t,i}}{C_{t,i}+K_d} \right)^2 - \lambda \sum_{i,j=1}^N |W_{ij}|
\end{aligned}
$$

wihch is equivalent to maximizing

$$\begin{aligned}
& \sum_{i=1}^{N}[\log \lambda_{1,i}\mathbb{E}S_{1,i} - \lambda_{1,i}] \\
+ & \sum_{i=1}^{N}\left\{ b_i \sum_{t=2}^{T}\mathbb{E}S_{t,i} + \sum_{t=2}^{T}\sum_{j} W_{ji}\mathbb{E}S_{t-1,j}S_{t,i} - \sum_{t=2}^{T}\mathbb{E}e^{b_i+\sum_j W_{ji}S_{t-1,j}} \right\} \\
- & \frac{1}{2\sigma_v^2}\sum_{i=1}^{N}\sum_{t=1}^{T}\left[ 2F_{t,i}\mathbb{E}\frac{C_{t,i}}{C_{t,i}+K_d} + \mathbb{E}\left(\frac{C_{t,i}}{C_{t,i}+K_d}\right)^2 \right] - \lambda\sum_{i,j=1}^{N}|W_{ij}|
\end{aligned}$$

where the expectation is taken w.r.t. the posterior distribution of $S$ given $F$ and the parameter values at the current iteration, and $C_{t,i} = \sum_{q=0}^{t-1}\left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^q A_{\mathrm{Ca}}S_{t-q,i} + \left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^t C_{0,i}$.

Thus our maximization problem splits into three independent optimization problems over different groups of parametes, that can be solved in parallel.

### 2.2.1 Update for $\lambda_{1,i}, i \in \{1, ..., N\}$

It's not hard to see that we can update $\lambda_{1,i}$ by $\lambda_{1,i}^{(n+1)} = \mathbb{E}S_{1,i}$.

### 2.2.2 Update for W and b

We can use gradient ascent to maximize:

$$f_1(W,b) = \sum_{i=1}^{N}\left\{ b_i \sum_{t=2}^{T}\mathbb{E}S_{t,i} + \sum_{t=2}^{T}\sum_{j} W_{ji}\mathbb{E}S_{t-1,j}S_{t,i} - e^{b_i}\sum_{t=2}^{T}\mathbb{E}e^{\sum_j W_{ji}S_{t-1,j}} \right\}$$

for updating $W$ and $b$, To do this, we need

$$\begin{aligned}
f_1(W,b) &= \sum_{i=1}^{N}\left\{ b_i \sum_{t=2}^{T}\mathbb{E}S_{t,i} + \sum_{t=2}^{T}\sum_{j} W_{ji}\mathbb{E}S_{t-1,j}S_{t,i} - e^{b_i}\sum_{t=2}^{T}\mathbb{E}e^{\sum_j W_{ji}S_{t-1,j}} \right\} \\
\frac{\partial f_1(W,b)}{\partial W_{kl}} &= \sum_{t=2}^{T}\mathbb{E}[S_{t-1,k}S_{t,l}] - e^{b_l}\sum_{t=2}^{T}\mathbb{E}\left[e^{\sum_j W_{jl}S_{t-1,j}}S_{t-1,k}\right] \\
\frac{\partial f_1(W,b)}{\partial b_k} &= \sum_{t=2}^{T}\left[\mathbb{E}S_{t,k} - e^{b_k}\mathbb{E}e^{\sum_j W_{jk}S_{t-1,j}}\right]
\end{aligned}$$

### 2.2.3 Update for $C_{0,i}, i \in \{1, ..., N\}$

Similarly as above, we can use gradient descent to minimize:

$$f_2(C_{0,i}) = \sum_{t=1}^{T}\left[ -2F_{t,i}\mathbb{E}\frac{K_d}{C_{t,i}+K_d} + \mathbb{E}\left(1-\frac{K_d}{C_{t,i}+K_d}\right)^2 \right]$$

for updating $C_{0,i}$, where $C_{t,i} = \sum_{q=0}^{t-1}\left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^q A_{\mathrm{Ca}}S_{t-q,i} + \left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^t C_{0,i}$.

To use gradient descent, we need:

$$\begin{aligned}\frac{\mathrm{d}f_2(C_{0,i})}{\mathrm{d}C_{0,i}} &= \sum_{t=1}^{T} 2\left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^t K_d F_{t,i}\mathbb{E}\frac{1}{(C_{t,i}+K_d)^2} \\ &+ 2\left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^t K_d\mathbb{E}\left(1-\frac{K_d}{C_{t,i}+K_d}\right)\frac{1}{(C_{t,i}+K_d)^2} \\ &= \sum_{t=1}^{T} 2\left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^t K_d F_{t,i}\mathbb{E}\frac{1}{(C_{t,i}+K_d)^2} \\ &+ 2\left(1-\frac{\Delta t}{\tau_{\mathrm{Ca}}}\right)^t K_d\mathbb{E}\frac{C_{t,i}}{(C_{t,i}+K_d)^3}\end{aligned}$$

# 3 Related Work and Novel Contributions

## 3.1 Baseline methods for benchmark, provided by the competition organizers

Two methods were provided for benchmarking the performance of our algortihm.

**Cross-correlation** The simplest approach is to use the so called cross-correlation. This method treats the time-series data at a specific neuron as a giant vector, and calculates the correlation between two vectors as our confidence for a connection between this pair. The resulting weight matrix is symmetric. This method is provided by the competition organizers and is implemented in *challengeFastBaseline.m*.

**Generalized transfer entropy** A more complicated approach is to use generalized transfer entropy, as described in [2]. Transfer entropy is similar to mutual information, but is not symmetric. Given two stochastic processes $X_1, \ldots, X_t, \ldots$ and $Y_1, \ldots, Y_t, \ldots$, the transfer entropy is defined to be:

$$H(Y_t|Y_1, \ldots, Y_{t-1}) - H(Y_t|Y_1, \ldots, Y_{t-1}, X_1, \ldots, X_{t-1}). \tag{1}$$

This idea is very similar to Granger causality. If the knowledge of the history of $X$ $(X_1, \ldots, X_{t-1})$ can help us better predict the future of $Y$ $(Y_t)$ than using the history of $Y$ $(Y_1, \ldots, Y_{t-1})$ alone, then we have evidence for $X$ "causing" $Y$. In the paper [2], they generalize the transfer entropy on two aspects: because of the (relatively) low sampling frequency, they consider the "same bin interactions", i.e. things that happen between two samples, which might include mutiple spikes, and they also considered the underlying dynamic regimes (bursting or non-bursting), and focus their attention on the non-bursting period. This method is provided by the competition organizers and is implemented in the *challengeMain.m* as the GTE score.

## 3.2 Related research work

We surveyed available methods for spike inference and network connectivity inference. The papers cited in the reference cover most dominant methods.

In the paper [3], the authors construct a joint probablistic distribution governing the hidden dynamics and the noisy observations. The hidden variables are spike trains and intracellular calcium concentrations; and the observed states are fluorescence measurements. The goal is to infer spike trains (not the connectivity of the network). In order to do so, the authors design a particle filter smoother.

Similar method, paper [1], is modelling the network of neurons as a collection of coupled hidden Markov chains, where the coupling between the chains reflecting the network's connectivity matrix. Each neuron is modeled as a generalized linear model. The EM algorithm is used to fit the model parameters. Blockwise-Gibbs sampler is used to obtain the sufficient statistics for the E-step.

Another common approach is to first discover spike trains for each neuron, based on the florescence images. This is done by consecutively filtering the florescence recordings with suitable filters. Then use methods

similar to cross-correlation to infer the network structure based on the spike data. An example of this type of work is [4].

Finally, a different type of method is Generalized transfer entropy ( [2], see the outline in the previous section). What differentiate this method from the others is that it is model-independent which has the advantage of not requiring any prior assumption on the neuronal firing and neuronal connections.

### 3.3 Our method and novelties

In our work we combined aspects from each of the above mentioned methods and introduced some of our own ideas.

As in the paper [3], we decided to model our problem as a state-space model with spike trains and intracellular calcium concentrations being the hidden variable and fluorescence measurements being the observed states. But this paper only focused on spike inference. So we pushed the analysis further, we borrow some ideas from the paper [1], by implementing the EM algorithm, and in particular, particle filter for the E-step.

Modeling choices for the dynamics of the system are numerous and vary from paper to paper. The equations are also very complicated with number of parameters. We made some simplyfing choices and came up with our set of equations (equations 1-4, page 2), that fully describes the evolution of the system.

We derived by ourselves all equations in the section 2 of this report. In particular, the equation for the joint likelihood of the hidden and observed variables, the equations for the particle filter in the E-step updates as well as the equations for the updates in the M-step.

In order to initialize parameters in the EM, we got inspired by the paper [4]. In that paper, authors perform a signal processing on the florescence data to deduce spike trains at every time point. We performed signal processing on the fluoresence data only at the first time step to get the estimates for the spike trains at that time point only. To implement this signal processing, we followed the description from the paper. After this we went further, and used a Poisson regression in order to deduce initial values for $b_i$'s (baseline spiking values) and $W$'s (synaptic weights). Indeed: Based on the equation 2 in the section 2, we can approximate: $\log S_{t,i} = b_i + \sum_{j=1}^{N} W_{j,i} S_{t-1,j}$, for all $t = 2, ..., T$ and $i = 1, ..., N$, where $S$'s are considered known since they have been estimated in the signal processing step.
Writing this as a matrix system, $Y = \alpha + \beta X$, and using Matlab's function $fitlm.m$, we get the estimates for $b_i$'s and $W_ij$'s.

Finally, since our implementation of the EM algorithm wasn't giving satisfying performance in terms of estimating $W$'s (many elements of the connectivity matrix were estimated to be negative in the M-step), we had to reconsider whether the simple gradient descent method would is the best for estimating $W$ in the M-steps since it doesn't contrain entries of the matrix to be positive. Instead of gradient descent, we tried several other approaches. In particular, we tried estimating $W$ at every M-step just like we did for its initialization i.e. performing signal processing on the florescence data, then Poisson regression.

## 4 Implementation Details

All code was completed in Matlab. As described above, all data was downloaded online, through an expired Kaggle challenge. To execute the algorithm after linking all handed-in files to the workspace, one need only run the file `finalprojectMain.m`. An outline of the code follows:

1. We make each of the known parameters in the model a global random variable.

2. We use a signal-processing step to give an initial guess as to the unknown parameters $b$, $W$: this step has been pre-executed and stored in the variable `initparameters1`.

3. Next, we execute the EM algorithm through the subroutine `connectomics_EM.m`. After initializing all unknown parameters and setting stopping criteria, we run the E-step and M-step until convergence is achieved.

4. The E-step is executed in another subroutine `particlefilter_connectomics.m`. As described above, the particle filter produces resampled particles.

5. The M-step is completed within `estimate_params.m`. As described above, we can get a closed form expression for the new value of $\lambda_{1,i} = \mathbb{E}S_{1,i}$. We use gradient descent (using `L1General2_TMP.m`) to maximize the remaining unknown parameters.

6. We have left in the code two separate attempts for learning $W, b$ from the output of the M-step. Our first attempt involved directly calculating the log likelihood and its partial derivative in `Wgradient.m` and using the output to drive gradient descent. The other attempt still appears within `estimate_params.m`; it consisted in estimating $W$ at every M-step in the same way as we did it for its initialization i.e. performing first signal processing on the florescence data, then Poisson regression (see the details in the subsection above, 3.3).

   The requirement in the model that all of the $W_{i,j}$ are positive (since they are known to represent excitatory, rather than inhibitory, interactions between neurons) presented a formidable challenge for our algorithm. We have discussed this issue in great detail.

7. *Side Note on the Challenges of Estimating $W$*: Executing gradient descent with no modifications will produce negative values for some of the $W_{ij}$, even if the model is perfect, since $W$ lives in such a highly dimensional space. In addition to the methods that currently appear in the code, we have also attempted executing gradient descent, and then, after each step, sending the negative $W$'s to zero. Since the convergence properties of this modification are not clear, we also tried calculating the gradient, and then only making the step if it produced a positive value for $W$. Although this method is guaranteed to increase the log likelihood at each iteration, the method had the issue that none of the parameters experienced any meaningful change in a reasonable amount of time.

8. In the M-step, the update for $C_0$ was executed through gradient descent, where the log likelihood and its partial derivative (with respect to $C_0$) were calculated in the file `Llik_and_grad_forC0.m`.

9. After each E-step/M-step pair, we produce the total change in the entries of $W$ and the AUC for the current estimate of the unknown parameters.

10. The visualization of our final estimate of the $W$'s is shown in the next section, and generated using `visualizestruct.m`.

# 5 Results

## 5.1 Learning objective

Figure 2 shows the plot of the expected log joint-likelihood as a function of number of iterations. This is an approximation (a lower bound, more precisely) of the marginal log-likelihood of the observed variables. The marginal log-likelihood be monotonically increasing and should serve us as a verification that our algorithm is producing good results.

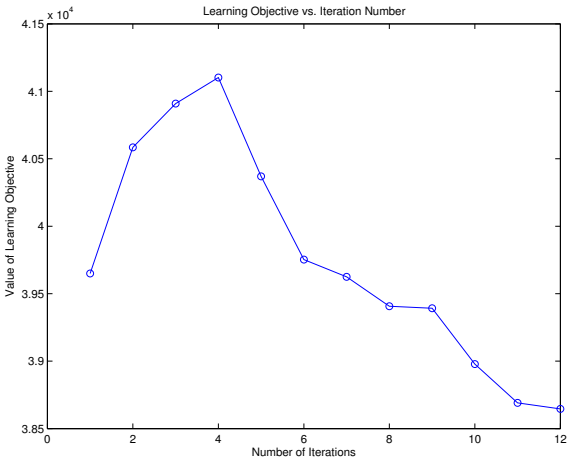


Figure 2: Learning objective as a function of number of iterations

From the figure, we can see that the approximation of the learning objective is not monotonically increasing. The most possible reason for this is that we are estimating the objective using a sampling based method, which might result in some errors.

Another possibility for making sure that our algorithm is learning well would have been to plot a mean square error between learned and true $W$'s, after each iterations. By lack of time, we didn't include this but this is something that can be easily added.

### 5.2 ROC curve

ROC curve detailing the performance of our algorithm is shown in the figure below. Each point of the ROC curve corresponds to a different threshold that we used to transform learned real values of the entries of $W$ into binary values.
Because of the heavy compuations, we run the code on the first 5000 time points only, and got the ROC curve, shown in blue. We also show the ROC curves for the baseline methods of Cross-Correlation and Generalized Transfer Entropy (in red and green, respectively), provided by the organizers of the competition.

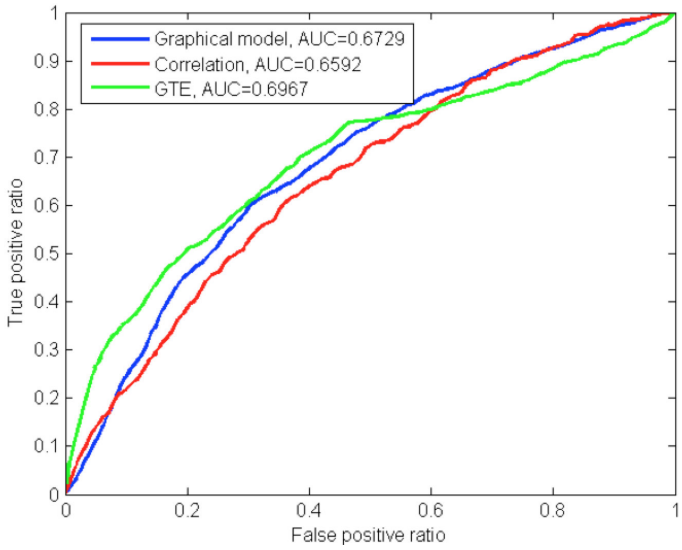


Figure 3: ROC Curve

### 5.3 Learned model structure

By thresholding obtained values for the entries of the matrix $W$ to get binary values, we got the connectivity structure of the network that our algorithm learned. We visualize this learned model structure in the figure below. Each square in the figure corresponds to an entry in the $W$ matrix. The $(i, j)$-th entry is green if the model's prediction of the presence (or absence) of a directed connection from neuron $i$ to neuron $j$ is consistent with the underlying connectivity structure (%TP: 2.17; %TN: 82.13). The $(i, j)$-th entry is blue in the case of a false positive (predicting a connection where there is none), and yellow in the case of a false negative (%FP: 7.83; %FN: 7.87).

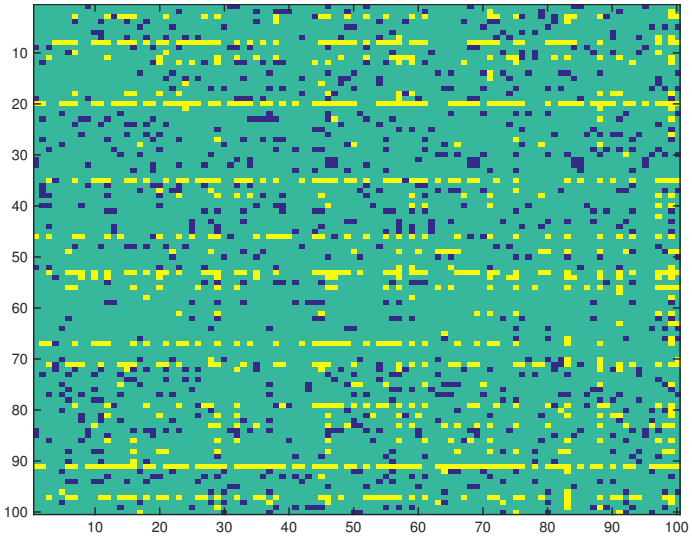


Figure 4: Visualization of Connectivity Parameters

## 6 Contributions of Team Members

All team members read related papers and discussed the associated models. We worked as a team to formulate a new model that used many strengths from the existing literature. Many drafts of the model were produced, and assumptions went through many rounds of edit. We discussed such things as:

- When modeling calcium concentration in the cell, do we need to incorporate a baseline calcium concentration, consistent with the underlying biology, or can we expect that the neurons spike with sufficiently high frequency to permit omitting this variable from the model?
- Should the spiking frequency follow a Normal or Poisson distribution?

We worked very well together, and produced a model that we all deeply understand and believe gives an accurate description of the underlying biology. Stannis was primarily responsible for writing the code for the E-step; Alexis and Ivana were responsible for coding the M-step.

## References

[1] Yuriy Mishchenko, Joshua T. Vogelstein and Liam Paninski. *A Bayesian approach for inferring neuronal connectivity from calcium fluorescent imaging data*. The Annals of Applied Statistics, 5(2B):12291261, 06 2011.

[2] Olav Stetter, Demian Battaglia, Jordi Soriano, and Theo Geisel. *Model-free reconstruction of ex- citatory neuronal connectivity from calcium imaging signals*. PLoS Comput Biol, 8(8):e1002653, 08 2012.

[3] Joshua T. Vogelstein, Brendon O. Watson, Adam M. Packer, Rafael Yuste, Bruno Jedynak, and Liam Paninski. *Spike inference from calcium imaging using sequential monte carlo methods*. Biophysical Journal, 97(2):636 655, 2009.

[4] Antonio Sutera, Arnaud Joly, Vincent Franois-Lavet, Zixiao Aaron Qiu, Gilles Louppe, Damien Ernst, and Pierre Geurts. *Simple connectome inference from partial correlation statistics in calcium imaging*. JMLR: Workshop and Conference Proceedings, 2014.