Understanding Functional Connectivity in C. elegans

Katherine Lee

May 3, 2016

1 Introduction

We'd like to understand how human neural connections function and interact. However, humans are complicated creatures with many different types of behaviors and billions of neurons in our brain alone. Simple animals like *Caenorhabditis elegans* (C. elegans), which have only 302 neurons, all of which have been anatomically mapped [7], a transparent body, and simple, classifiable movements, are much easier to study. Though we understand the anatomical connectdome of the C. elegans, we don't yet understand how each neuron influences the other, or the functional connectdome[5][1]. Our goal will be to create a functional connectdome with a new data set.

Our collaborators in Andrew Leifers lab at Princeton have successfully imaged up to 120 neurons in the brain of the C. elegans through calcium fluorescence while tracking the animals' behavior. This data-set is complicated by three factors: photobleaching, calcium decay, and motion artifacts. Motion artifacts are difficult to clean as the animal's movement is closely linked to neural activity. We created a model for calcium imaging data that enables us to recover a better estimate of neural activity. Additionally, we have created a model of how neurons interact which enables us to understand how they are connected through Group Automatic Relevance Determination (ARD). In this paper, we will first describe the data and our approach to cleaning motion activity then show results on both simulated and real data. We will introduce our neuron interaction model, the classic ARD, and our modification, Group ARD. We test both algorithms on simulated data and explain how we will apply these algorithms to the cleaned data to develop a functional connectome.

2 Data

Our collaborators in the Leifer lab[5], have been able to record neural activity from up to 120 neurons in the brain of a freely moving C. elegans through calcium imaging. This has finally made it possible to answer questions about how the neurons interact with each other, and how the neuron activity changes with behavior. In this section, we'll discuss the data itself, the artifacts that appear int he data, and how we correct for them.

We utilize two different types of data sets, one with recordings from GCaMP6s (genetically encoded calcium indicator) and RFP (red fluorescent protein) in each available neuron across all time (figure 1 in addition to centerline positioning of the worm on the stage (figure 3), and one with recordings from GFP (green fluorescent protein) and RFP (figure 2) in each available neuron across all time with centerline data.

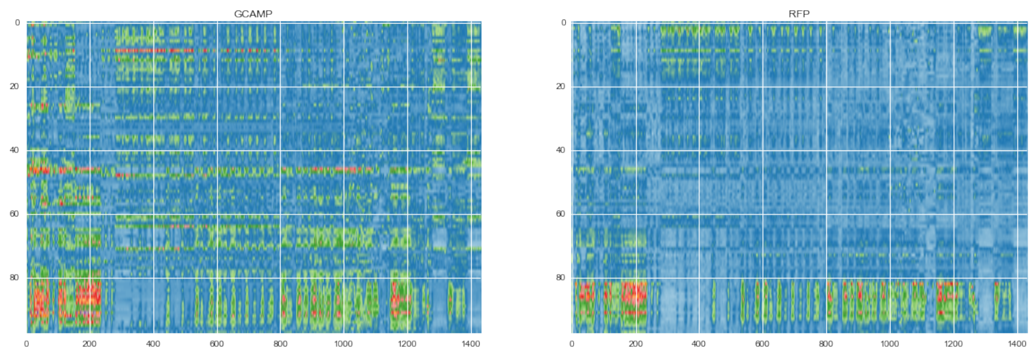


Figure 1: Left: GCaMP6s recordings from 98 neurons from worm 4 of our collaborators' dataset [5]'s. Right: RFP recordings from the same 98 neurons. On the x-axis is time, on the y-axis is the neuron index.

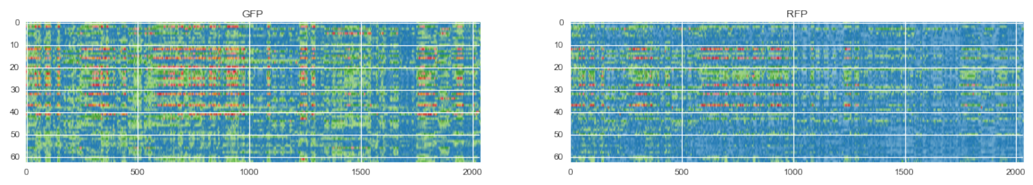


Figure 2: Left: GFP recordings from 63 neurons of a control worm by our collaborators [5]. Right: RFP recordings from the same 63 neurons. On the x-axis is time, on the y-axis is the neuron index.

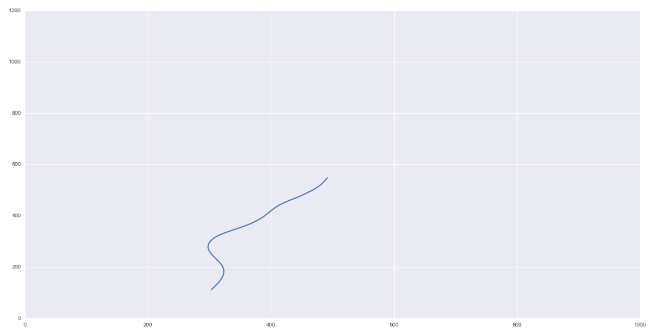


Figure 3: Plot of centerline data for one worm at one instance in time.

2.1 Challenges of Data

Though, ideally, we would have recordings of neural activity directly from each neuron as the worm moves, calcium fluorescence recordings can serve as a non-invasive proxy to estimate neural recordings. Currently, the data is confounded by three sources: photobleaching, calcium decay, and motion artifacts. In the next section, we explore two models for GCaMP and RFP that imply different techniques of motion correction and evaluate how the techniques perform. Then, we explore how well we are able to recover activity on real and simulated data.

3 Cleaning Data

To correct for photobleaching, we simply subtract from the data the exponential decay of the fluorescent proteins[5].

To correct for calcium decay, we can deconvolve the GCaMP recordings with the decay of GCaMP6s to get back the neural signal. Previous work shows that GCaMP6s decays like a exponential distribution [3]. We have not yet done the deconvolution.

The final challenge is to clean motion artifacts. As the worm moves and turns, the neurons in the head of the worm change depth from the camera, and may be deformed. This may result in changes in fluorescence that do not correspond to changes in neural activity. We see this manifested in the RFP recordings. FOr an immobilized worm, we would expect RFP to be constant (with noise) as there is no neural signal. In figure 1, we see that RFP shows similar patterns with GCaMP due to artifacts from motion.

We try to correct for motion in two ways with two different models. In the first, we assume a additive relationship between motion and activity, and in the second we assume a multiplicative relationship.

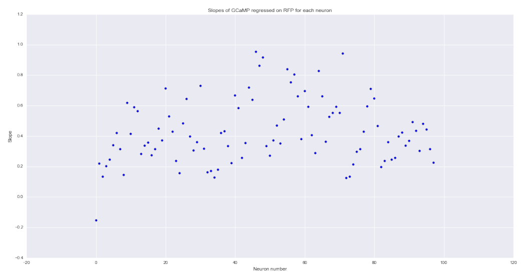
3.1 Additive Model

After cleaning GCaMP and RFP of photobleaching and calcium decay, we propose modeling the recordings as:

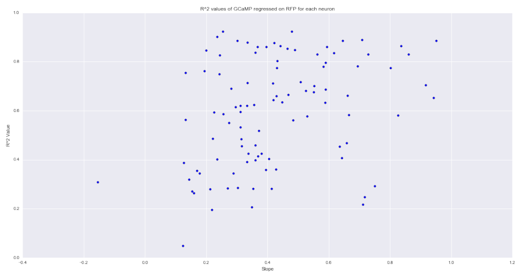
$$\text{GCaMP}: \vec{g}_t = \sum_{s=0}^{t} \alpha_s \vec{a}_{t-s} + \vec{m}_t + \epsilon_g \tag{1}$$

$$\text{RFP}: \vec{r}_t = \vec{m}_t + \epsilon_r \tag{2}$$

where $\vec{g}_t$ is GCaMP, $\vec{a}_{t-s}$ is neural activity and $\vec{m}_t$ is motion artifact. Then we can regress GCaMP by RFP and anything not explained by RFP (i.e. in the residuals) should be our neural activity. This may not be

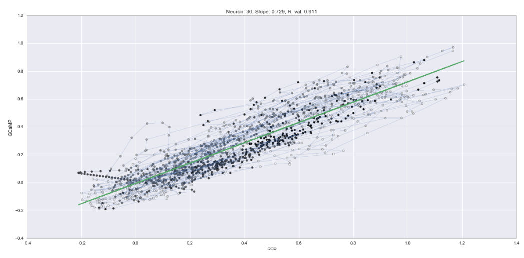


(a) Slopes of GCaMP regressed on RFP

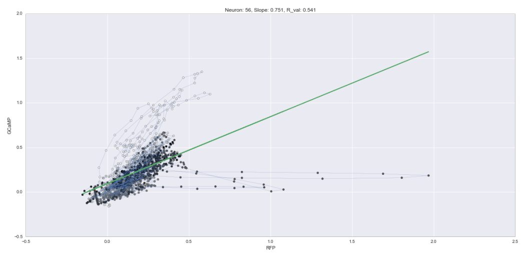


(b) R^2 of GCaMP regressed on RFP

Figure 4: (a): Plot of slopes for RFP regressed on GCaMP6s for each neuron in worm 4 from [5]. Note that there is one negative slope. We discuss this further in the paper. (b): Plot of R^2 values for RFP regressed on GCaMP6s for each neuron in worm 4 from [5].



(a)



(b)

Figure 5: (a): Plot of GCaMP6s versus RFP in neuron 30 of worm 4 from [5] with a regression line superimposed. This is an example of a good correlation between the GCaMP and RFP. The gradient from light to dark in the scatter plot signals time. (b): Plot of GCaMP6s versus RFP in neuron 56 of worm 4 from [5] with a regression line superimposed. Some neurons appear to have multiple regression lines as in this figure. The gradient from light to dark signals time. We can then see that the two separate sections of this plot occur in separate intervals of time.

a valid assumption as motion artifact is linked to neural activity and RFP and GCaMP either may be too well correlated or not at all correlated. To test if the additive model is valid, we regress GFP with RFP and expect a strong correlation between RFP and GFP and that the residuals are uninformative. We, then, add two more features into the regression: eigenworm representations and neural positions to better capture the worm's movements.

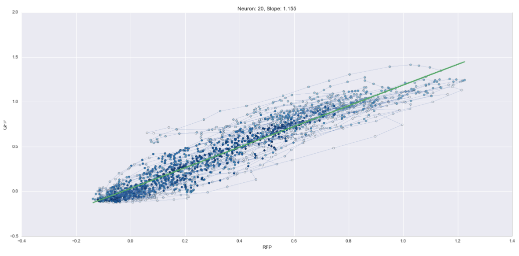
3.1.1 Regress RFP and GCaMP

Figure 4a shows the slopes of the regression of GCaMP on RFP. We expect slopes of the regression to be positive, which would mean that motion affects RFP and GCaMP recordings the same way. However, we note that there is one negative slope in figure 4a which tells us that for that neuron, as RFP increases GCaMP decreases. Looking at regression fits of the neurons, we noted that though many looked like figure 5a, several correlations of GCaMP and RFP looked more like figure 5b. One possible reason may be because motion artifact in the RFP partially explains the neural signal in GCaMP. An alternative reason for a poor correlation between GCaMP and RFP is that what we cannot capture in GCaMP with RFP is due to neural signal.

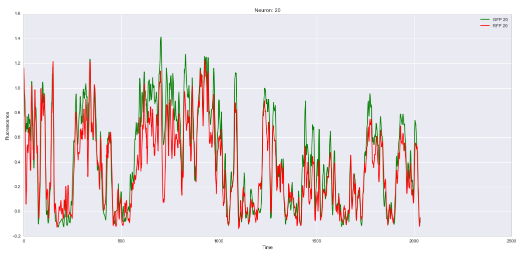
3.1.2 Regress RFP and GFP

To test these hypotheses, we regressed RFP on GFP. Since GFP does not measure neural activity, we expect RFP to explain GFP well. A linear regression between the RFP and GFP shows that for most neurons, GFP and RFP are well correlated, as in figure 6. However, we again see anomalies as in figure 7 where GFP and RFP appear to be correlated at various time intervals and out of correlation in other time intervals. This may be due to a low signal to noise ratio for that neuron and there may be too much error to predict GFP from RFP.

We hypothesized that having RFP values for all other neurons would allow us to better predict GFP values.

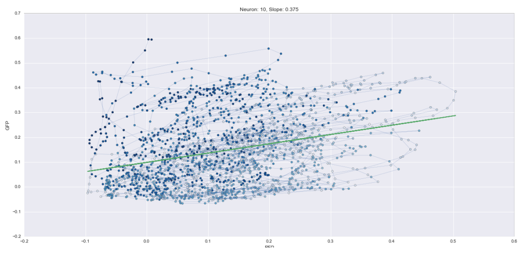


(a) GFP versus RFP

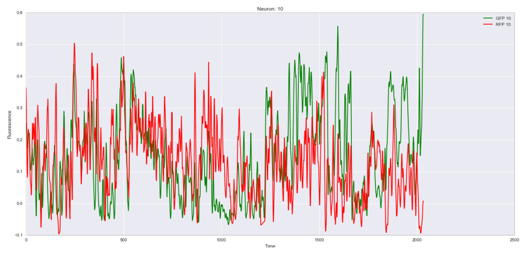


(b) GFP and RFP versus time

Figure 6: This looks like the correlation and regression that we expect to see as both RFP and GFP should only be affected by photobleaching and motion artifacts (unlike GCaMP, which includes neural signal and calcium decay). This data is from neuron 20.



(a) GFP versus RFP



(b) GFP and RFP versus time

Figure 7: Several neurons exhibit the above behavior which appears to be correlated at times, then become uncorrelated leading to the regression and poor correlation we see on the left. This data is from neuron 10 of the same data set as figure 6

Since the motion artifacts affecting one neuron may be indicative of artifacts that also affect another neuron at the same or later time scale, we hypothesized that RFP and GFP can be modeled by autoregressive models, and we can use RFP measurements from previous time-steps and previous neurons to regress GFP.

We incorporated previous time-steps and RFP measurements from all neurons as features in regressing for GFP. Using information from all neurons lowered the Mean Squared Error (MSE). However, figure 9a shows that taking multiple time-steps does not change the MSE significantly for finding GFP. This was surprising, but RFP values from other neurons also depend on the motion of the worm at that time-point and may help infer influence of motion on a neighboring neuron.

3.1.3 Eigenworms and Neural Positions

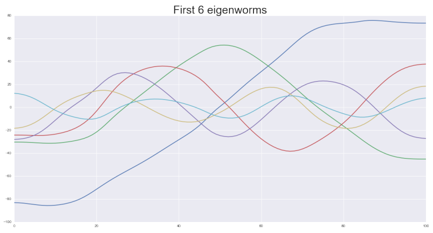
To better understand motion artifacts, we introduce two additional features into the regression: eigenworm representations and pairwise distances between neurons. They should capture the motion of the worm and deformations in the head respectively.

C. elegans have simple movements that can be represented by six principal components of their centerlines (line through the center of their bodies as captured on video. An example centerline is shown in figure 3.) which captures 90% of variance. The principal components, or eigenworms, are shown in figure 8a. This tells us how the worm is moving and incorporates large scale position information of the worm.

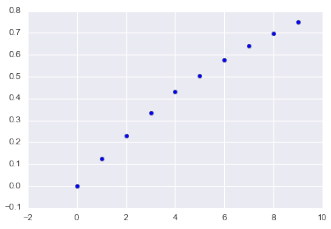
We wanted to further incorporate motion information by understanding how neurons move in the head.

We calculated e^{-d}, where d is the distance between two neurons, for each pair of neurons in the head. To reduce the number of features to the most informative, we took the first 100 principal components of the neural positions. Figure 8b shows how much variance is explainable by neural positions. We can see the neural position data is more variable than centerline data.

We performed ridge regression with neural positions and eigenworm positions as features and found that these additional features did not decrease MSE (figure 9b, 9c). In fact, we saw that ridge regression sends all hyperparameters to ∞, meaning that these additional factors are not useful for our regression. To verify that RFP from all neurons was most informative for the regression, we tried regressing GFP on just neural positions and/or centerline data. Figure 9c shows that not using RFP in the regression results in a much higher MSE.

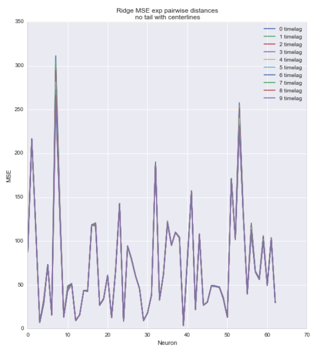


(a)

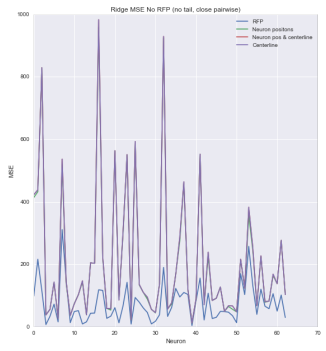


(b)

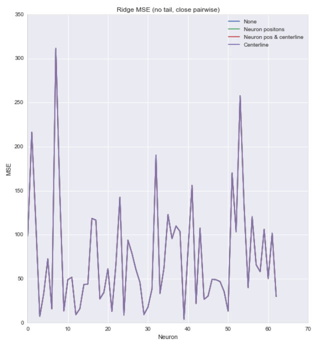
Figure 8: (a) First six Principal Components that capture 90% of variance of centerline data. (b) Amount of variance captured by the first 10 principle components of exponential pairwise neural distance.



(a) Timelags of RFP



(b) Regressing with (1) RFP, (2) Neural Positions, (3) Neural Positions and Centerlines, (4) Centerlines



(c) Regressing with (1) RFP, (2) RFP and Neural Positions, (3) RFP, Neural Positions, and Centerlines, (4) RFP, and Centerlines

Figure 9: MSE of predicting GFP from RFP with various features.

An interesting note is many neurons may not have good signal to noise ratio and the recordings of RFP and GFP are more noisy. That may explain why the MSE of predicting GFP from RFP is much higher for some neurons than for other neurons. This is surprising as we thought changes in neural position from deformations in the head were directly linked to motion artifacts.

3.2 Multiplicative Model

Previously, to recover activity from GCaMP and RFP recordings, researchers would divide GCaMP by RFP (g/r) [5]. This estimate is strongly affected by small values of RFP and can lead to results with high variation from one time-point to the next–inconsistent with how neural activity behaves. We can see these high fluctuations in figure 11. Following the spirit of g/r, which suggests that motion and activity are multiplied to get GCaMP, we create a model and place prior assumptions to derive an estimate of neural activity that more closely follows activity in simulated data. Since we do not have recordings of neural activity to evaluate how close our estimate was, we can check the following three things. First, we must be able to recover simulated neural activity well since we know exactly what it is (figure 11). Then, recovered neural activity from GFP must be constant (as GFP does not respond to neural activity), and GCaMP of an immobilized worm should also be constant. Our model is:

$$\text{RFP:}r\left(t\right)=m\left(t\right)+\epsilon_{r}\left(t\right),n_{r}\sim N\left(0,\sigma_{r}^{2}\right) \tag{3}$$

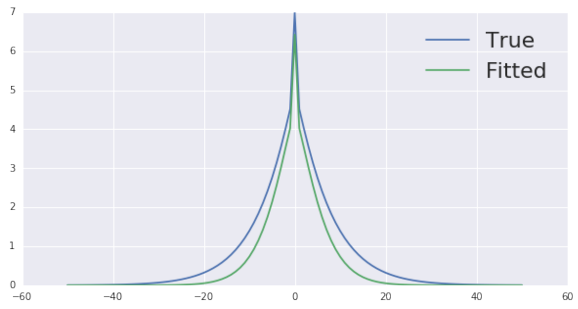
$$\text{GCaMP:}g\left(t\right)=m\left(t\right)a\left(t\right)+\epsilon_{g}\left(t\right),n_{g}\sim N\left(0,\sigma_{g}^{2}\right) \tag{4}$$

where m is the motion artifact, a is measured neural neural activity and ϵ_r, ϵ_g are noise added to RFP and GCaMP respectively.

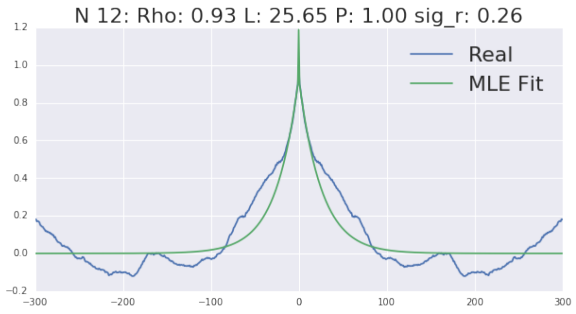
We simplify the problem by adding independent priors to motion artifact and neural activity. This may not be valid as motion artifact and neural activity should be linked, but we can see that even this simplifying assumption recovers activity better than taking the ratio of GCaMP and RFP.

We place a temporal smoothness prior over the motion artifact:

$$m\sim N\left(\mu_{m}1,\Sigma_{m}\right) \tag{5}$$



(a)



(b)

Figure 10: (a) Recovering parameters of $R(t)$ with the maximum likelihood estimator on simulated data. Parameter values from table 1. (b) MLE estimate of parameters for autocorrelation of RFP data for a neuron.

and assume that a is a weighted combination of the true activity y ad the current and past time-steps:

$$a_t = y_t + \alpha y_{t-1} \tag{6}$$

In addition, we choose the following Gaussian process prior for the Σ_m to be:

$$\Sigma_m = \rho e^{-|\tau/a|^p} \tag{7}$$

We can derive the following equations from our above model:

$$\begin{aligned}
\mu_r &= \mu_m, && (8)\\
\mu_g &= R_{a,m}(0) + \mu_a \mu_m, && (9)\\
R_r(\tau) &= R_m(\tau) + \sigma_r^2 \delta(\tau), && (10)\\
R_g(\tau) &= R_a(\tau)R_m(\tau) + R_{a,m}(\tau)R_{m,a}(\tau) + \mu_a^2 R_m(\tau) + \mu_m^2 R_a(\tau) + \sigma_g^2 \delta(\tau), && (11)\\
R_{r,g}(\tau) &= \mu_m R_{a,m}(\tau) + \mu_a R_m(\tau), && (12)
\end{aligned}$$

Where $R_{a,b} = Cov(a(t), b(t'))$. To test our model, we can use these equations to generate simulated neural activity, motion artifacts, RFP and GCaMP then recover the parameters for $\Sigma_m, \mu_m, \sigma_r, \sigma_g, \mu_a$ and then the neural activity.

We know $R \sim N\left(\mu_m, \Sigma_m + \sigma_r^2 I\right)$ and thus we recover parameters for Σ_m and σ_r by maximizing the log-likelihood:

$$-\frac{1}{2}\ln|\Sigma_m + \sigma_r^2 I| - \frac{1}{2}(r-\mu_m)^T\left(\Sigma_m + \sigma_r^2\right)^{-1}(r-\mu_m) \tag{13}$$

Then we can solve equations 10, 11, and 12 to recover neural activity. We generate 100 samples of $R(t)$ then solve the maximization problem each time to get values for the parameters. Then, we calculated the mean and variance of each parameter (given in table 1). Figure 10a shows how well we were able to recover the true covariance with 100 samples of $R(t)$. We can see that we came very close to recovering the true values for ρ, p, and σ_r and further off for l, though the variance here was also high.

Table 1: Estimated Parameters

	ρ	L	p	σ_r
True parameters	5	8	1.1	2
Median estimated	4.402	6.387	1.303	2.018
STD estimated	2.039	3.554	0.508	0.681

In figure 11, we recover activity with our model and with the ratio g/r from simulated data. We can see recovering activity with our model is much less prone to large fluctuations and follows the simulated activity closely.

We have not yet applied the full process to GCaMP and GFP data. However, figure 10b shows the fit of the parameters of Σ_m and σ_r for RFP data for a neuron. We can see from the plot that it achieves a reasonable fit to the data.

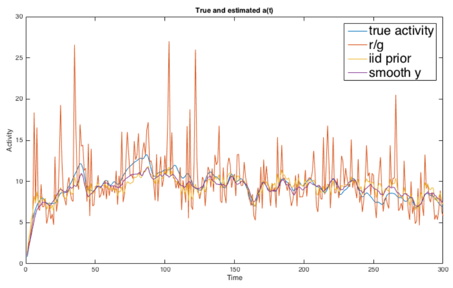


Figure 11: A comparison of recovering activity from g/r and from equations Rr, Rg, Rrg on simulated data.

4 Autoregressive Model

Once we have recovered neural activity from GCaMP and RFP, we can return to the original problem of finding functional connectivity between neurons.

Since the neural activity of one neuron at a time-point may influence itself or another neuron at a later time-point, we create an autoregressive model of our data:

N:	number of neurons.
$w_{j,i,s}$:	weight of neuron j to neuron i at s time-steps back.
$y_{j,t-s}$:	value of the neural activity for neuron j at time $t-s$ where s_b is the furthest time-step back.
ϵ:	Gaussian noise.

Thus:

$$y_{i,t+1} = \sum_{j=1}^{n}\sum_{s=0}^{s_b} w_{j,i,s}\cdot y_{j,t-s} + \epsilon \tag{14}$$

This model takes s_b time-steps back and generates a weighted average of the different contributions of each neuron at each time-step back (until s_b) to get new data points.

We can write this in matrix form. Here is an example weight matrix with only two neurons:

$$\begin{bmatrix} y_{A,t+1} \\ y_{B,t+1} \end{bmatrix} = \begin{bmatrix} W_{A\to A} & W_{B\to A} \\ W_{A\to B} & W_{B\to B} \end{bmatrix} \cdot \begin{bmatrix} Y_{A,t} \\ Y_{A,t} \end{bmatrix} \tag{15}$$

Where $y_{A,t}$ represents a vector of all relevant time-points for neuron A up to time t, $Y_{A,t+1}$ represents the neural values at the next time-point, $t+1$, and $W_{A\to B}$ represents the weights from neuron A to neuron B for all relevant time-points. If we took two time-steps back ($s_b = 2$), this model would look like:

$$\begin{bmatrix} y_{A,2} \\ y_{B,2} \end{bmatrix} = \begin{bmatrix} w_{A,A,0} & w_{A,A,1} & w_{B,A,0} & w_{B,A,1} \\ w_{A,B,0} & w_{A,B,1} & w_{B,B,0} & w_{B,B,1} \end{bmatrix} \cdot \begin{bmatrix} y_{A,0} \\ y_{A,1} \\ y_{B,0} \\ y_{B,1} \\ \\ \end{bmatrix}$$

To determine the strength of the relationship between neurons A and B, we want to find the weight matrix $\boldsymbol{W}$:

$$\begin{bmatrix} W_{A\to A} & W_{B\to A} \\ W_{A\to B} & W_{B\to B} \end{bmatrix}$$

4.1 Automatic Relevance Determination

The model described lends well to Automatic Relevance Determination (ARD). ARD places a constraint function on parameters which tends to send weights to zero–helping us determine which connections exist, in addition to the strength of the connection. Further, unlike methods like SVD, it uses fewer basis functions and has a fixed-point update rule which is computationally faster and less complex. As an added bonus, it provides probabilistic predictions of the weights.

Though ARD generates a sparse weight matrix, it computes each $w_{j,i,s}$ (which represents the weight from neuron j to another neuron, i, at a specific time-step back, s) separately. Instead, we would like to know whether or not there is a connection from neuron j to neuron i, namely if our $W_{j\rightarrow i}$ and our $W_{i\rightarrow j}$ are non-zero. This motivates our modification of ARD to Group ARD.

To understand our modification, I have included part of the derivation of ARD by Tipping [6] and Bishop [2]. For a more detailed derivation, please see [6].

Our data comes in pairs of $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, and x is the predictor variable and y is the target. $\boldsymbol{w}$ is the set of weights we'd like to estimate, and ϵ is a vector of mean-zero, Gaussian noise with variance σ^2 added for each neuron.

We would like to predict y_n given $\boldsymbol{x}_n$. To avoid over-fitting, we add a prior distribution over our parameters, $\boldsymbol{w}$, and choose them to be:

$$p\left(\boldsymbol{w}|\boldsymbol{\alpha}\right) = \prod_{i=0}^{N} N\left(w_i|0, \alpha_i^{-1}\right) \tag{16}$$

Where $\boldsymbol{\alpha}$ are our new hyperparameters. To complete this model, we define our hyperpriors over $\boldsymbol{\alpha}$ and $\beta = \sigma^{-2}$ to be Gamma distributions.

To perform inference for y_{t+1}, we'd like to maximize the marginal distribution

$$p\left(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\alpha}, \sigma^2\right) = \int p\left(\boldsymbol{y}|\boldsymbol{w}, \sigma^2\right) p\left(\boldsymbol{w}|\boldsymbol{\alpha}\right) d\boldsymbol{w} \tag{17}$$

This can be done more easily by taking the log of the marginal distribution (17).

Then, by differentiating the log of the marginal distribution and rearranging the terms as described by Bishop [2] and also MacKay [4], we can get a fixed-point update rule where

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{\mu_i^2}, \gamma_i = 1 - \alpha_i \sigma_{ii}, \left(\sigma^2\right)^{\text{new}} = \frac{||y - \Phi\mu||^2}{N - \Sigma_i \gamma_i} \tag{18}$$

where Φ is the design matrix, Σ is the posterior covariance, and μ is the posterior mean.

4.2 Group ARD

Following the same model as in ARD, we are now interested in whether or not a neuron, A, is ever connected to another neuron, B. We can put a prior over the group of weights:

$$\boldsymbol{w}_i \sim N\left(0, \frac{1}{\alpha_i} I\right) \tag{19}$$

Another paper has also developed Group ARD [8].

Again, we want to maximize the marginal distribution $p\left(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\alpha}, \beta\right)$. We follow the same procedure to take the log of this distribution.

$$\ln p\left(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\alpha}, \beta\right) = -\frac{1}{2}\left(N \ln\left(2\pi\right) + \ln|\boldsymbol{C}| + \boldsymbol{y}^T \boldsymbol{C}^{-1} \boldsymbol{y}\right)$$

$$= -\frac{1}{2}\left(N \ln\left(2\pi\right) + \ln|\Sigma^{-1}| - N\ln\beta - \ln|A| + \beta\|t - \Phi m\|^2 + m^T A m\right)$$

Then, differentiating with respect to α_g, the α of a particular group g, we get:

$$\frac{d}{d\alpha_g} = -\frac{1}{2}\left(\Sigma_{i\in g}\frac{1}{\lambda_i + \alpha_i} - \frac{|g|}{\alpha_i} + m_g^T m_g\right) = 0$$

Again, rearranging following Bishop [2] and MacKay [4], we get:

$$\alpha_i m_g^T m_g = |g| - \alpha_i \Sigma_{i\in g}\frac{1}{\lambda_i + \alpha_i} = \gamma$$

$$\gamma = \Sigma_{i\in g} 1 - \frac{\alpha_i}{\lambda_i + \alpha_i}$$

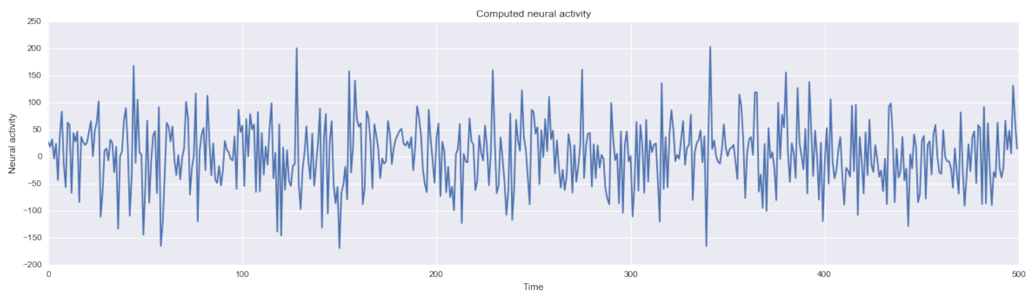


Figure 12: Plot of the time-series y_t across 500 time-steps. The values of the first 50 time-steps are drawn from a normal distribution. The remaining are computed from the weights as described in the paper.

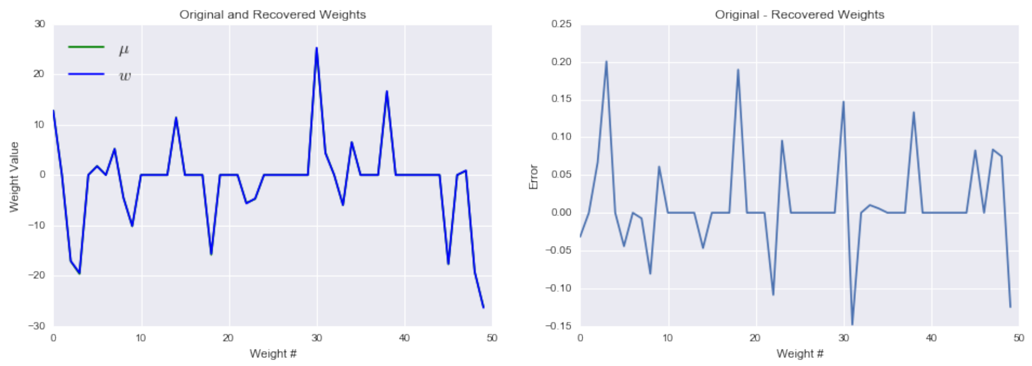


Figure 13: Left: plot of weights recovered from ARD over the original weights. Method for generating the original weights is described in the paper Right: difference between the recovered and original weights. As we can see, the ARD model does a great job of recovering the value of the weight, but not a good job of recovering which weights are zero.

$$\alpha_i = \frac{\Sigma_{i \in g} 1 - \alpha_i \Sigma_{ii}}{m_g^T m_g}$$

Which yields the fixed-point update rules:

$$\gamma = 1 - \alpha_i \Sigma_{ii}, \alpha_i = \frac{\Sigma_{i \in g} \gamma}{m_g^T m_g} \tag{20}$$

4.3 Simulation

To test our implementation of ARD and Group ARD, we created autoregressive models of one neuron and used ARD and Group ARD to recover the weights.

4.3.1 ARD Simulation

To test our ARD model, we randomly drew 50 weights from a Laplace distribution with mean 0 and parameter $\lambda = 10$, then set 60% of the weights to 0 (figure 13). We then created an autoregressive model with 500 time-steps where for $t \leq 50$ we drew from a random normal distribution and for every $t > 50$, the value of the time-series, y, was given by $y_t = \sum_{s=1}^{50} w_s \cdot y_{t-s}$ (figure 12). We then ran the algorithm with the fixed-update rule given in equation (18) and recovered the weights as shown in figure 13. As we can see, this did a good job of recovering the weights. However, we were not able to recover which weights were exactly zero which tells us which neurons are connected. In the future, we would need to modify our algorithm so that it sends values of weights close to zero fully to zero.

4.3.2 Group ARD Simulation

We then checked our Group ARD algorithm by generating a simulation data set with 25 groups of 10 weights for a total of 250 weights. These weights were also randomly drawn from a Laplace distribution of mean 0 and

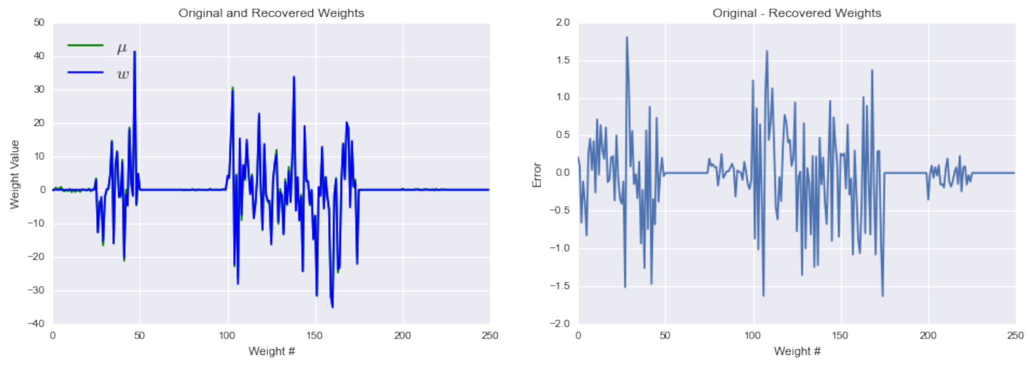


Figure 14: Left: plot of weights recovered from Group ARD over the original weights. Original weights were generated as described in the paper. Right: difference between the recovered and original weights. Again, the ARD model does a great job of recovering the value of the weight, but not a good job of recovering which weights are zero.

parameter $\lambda = 10$, and set 60% of them to zero (figure 14). Then we created our autoregressive model for y_t in the same way as is described in section 4.3.1. Running through the Group ARD algorithm with the fixed points given in (20) we recover the weights as in figure 14. Again, we see that though the algorithm recovered very closely the values of the weights, we still have not set every zero weight to zero. In the future, we will work further on updating our code and the algorithm to determine if a weight is zero or not.

4.3.3 Application to data

We have not yet applied ARD and Group ARD to multineuron examples or the real data. When we do, we expect to have to tune several parameters. For example, we do not currently know how many time-steps back (our value of s_b) will be relevant for predicting the next neural value.

5 Conclusions and Future Work

Thus far, we tackled the challenging of cleaning motion artifact from calcium fluorescence data by creating a model for calcium imaging that may help us recover neural activity with less variance. We will continue by applying this process to real data and evaluating its performance on immobilized worms and GFP data. We have also created a model for neuron interactions. After we recover neural activity, we will apply Group ARD to the cleaned data and tweak the algorithm to find the best number of time-steps back to develop a map of functional connectivity.

6 Acknowledgments

Thank you, Dr. Pillow, for being an incredible adviser this year! I had a great time working with you and in your lab. You do a great job of building a collaborative culture.
Thank you also to the Pillow Lab and the worm team: Nicholas Roy, Mikio Aoi, and Lea Dunker who all mentored and worked with me on this project.
All data were collected in the lab of Dr. Andrew Leifer, by Jeffery Nguyen and Ashley Linder.
Special thanks to Jeremy Cohen, Kiran Vodrahalli, and David Dohan for being a great sounding board!

References

[1] Bargmann. *Beyond the connectome: how neuromodulators shape neural circuits.* BioEssays 34, 458-465. 2014.

[2] Bishop. (2006) *Pattern Recognition and Machine Learning.* Secaucus, NJ: Springer-Verlag New York, Inc.

[3] Chen. et. al.. *Ultrasensitive fluorescent proteins for imaging neuronal activity.* Nature 499, 295300. 2013.

[4] MacKay. *Bayesian interpolation.* Neural Computation, 4(3):415-447, 1992a.

[5] Nguyen, Shipley, Linder, et.al.. *Whole-brain calcium imaging with cellular resolution in freely behaving C. elegans.* Proceedings of the National Academy of Sciences of the United States of America. 10.1073/pnas.1507110112

[6] Tipping. *Sparse Bayesian Learning and the Relevance Vector Machine.* Journal of Machine Learning Research 1: 211-244, 2001.

[7] White, et al. *The structure of the nervous system of the nematode Caenorhabditis elegans.* Philos. Trans. R. Soc. Lond. B Biol. Sci. 314, 1340. 1986.

[8] Yu, et al. *Grouped Automatic Relevance Determination and Its Application in Channel Selection for P300 BCIs.* Neural Systems and Rehabilitation Engineering, IEEE Transactions: 23(6) 1068-1077. 2015.

I pledge my honor I have not violated the honor code on this report.
-Katherine Lee