Contents

1 Motivation

1.1 Finding highly correlated networks

We wanted to find highly correlated communities among larger ensembles of neurons. In order to do this, we induced an undirected weighted network in the neural ensemble by measuring the spike count correlations between every pair of neurons. We then used a cutting edge community detection method [1] on this network to find any highly correlated communities present in the ensemble.

1.2 Within regions or across regions

Information in the brain is carried in correlated network activity. Correlations can carry sensory information [2]. Recent findings how that spontaneous behaviours explain correlations in parts of the brain not associated with motor control [3]. We wanted to know if the highly correlated communities that we found existed within anatomical regions, or between anatomical regions. That is, did all the members of the community come from one anatomical region, or from many.

1.3 Changing timescales

The time bin width used when binning spikes into spike counts affects spike count correlation measures [4]. We found that spike count correlations increased as the width of the bin width used increased. However, we also found that the difference between *within-region* and *between-region* correlations decreased as the width of the bin width used increased.

2 Data

2.1 Brain regions

Neuropixels probes were used to collect extracellular recordings [5] from three different mice.

1. male, wild type, P73.
2. female, TetO-GCaMP6s, Camk2a-tTa, P113
3. male, Ai32, Pvalb-Cre, P99

Eight probes were used to collect readings from 2296, 2668, and 1462 cells respectively. Data were collected from nine brain regions in each mouse:

- Caudate Putamen (CP)
- Frontal Motor Cortex (Frmoctx)
- Hippocampal formation (Hpf)
- Lateral Septum (Ls)

- Midbrain (Mb)
- Superior Colliculus (Sc)
- Somatomotor cortex (Sommoctx)
- Thalamus (Th)
- Primary visual cortex (V1)

Readings were continuous and lasted for about 1 hour [3].

3 Methods

3.1 Binning data

The data were divided into time bins of various widths ranging from 0.01s to 4s. If the total length of the recording period was not an integer multiple of the time bin width, we cut off the remaining time at the end of the recording period. This period was at most 3.99s. This is far less than the total recording time of around 1 hour.

3.2 Correlation coefficients

We calculated Pearson's correlation coefficient for pairs of neurons. For jointly distributed random variables X and Y, Pearson's correlation coefficient is defined as:

$$\begin{aligned}\rho_{XY} =& \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} && (1)\\ =& \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} && (2)\end{aligned}$$

where E denotes the expected value, μ denotes the mean, and σ denotes the standard deviation. The correlation coefficient is a normalised measure of the covariance. It can take values between 1 (completely correlated) and -1 (completely anti-correlated). Two independent variables will have a correlation coefficient of 0. But, having 0 correlation does not imply independence.

If we do not know the means and standard deviations required for equation 1, but we have samples from X and Y, Pearson's sample correlation coefficient is defined as:

$$r_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3}$$

where $\{(x_i, y_i)\}$ for $i \in \{1, \ldots, n\}$ are the paired samples from X and Y, and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ are the sample means.

In practice we used the python function `scipy.stats.pearsonr` to calculate the correlation coefficients.

3.2.1 Spike Count Correlation, r_{SC}

The spike count correlation (r_{SC}) of two cells is the correlation between the spike counts of those cells in response to a given stimulus condition. In this study, there was only one stimulus condition, that of no stimulus. The subjects engaged in spontaneous behaviour during recording.

3.2.2 Separating Correlations & Anti-correlations

In order to compare the effect of bin width on measures of negative r_{SC} (anti-correlation) and positive r_{SC} separately, we had to separate correlated and anti-correlated pairs. To do this, we simply measured the mean r_{SC}, taking the mean across all the bin widths. If this quantity was positive or zero we regarded the pair as positively correlated. If this quantity was negative we regarded the pair as anti-correlated.

3.3 Network analysis

3.3.1 Communities

Given some network represented by an adjacency matrix $\mathbf{A}$, a community within that network is defined as a collection of nodes where the number of connections within these nodes is higher than the expected number of connections between these nodes. In order to quantify the 'expected' number of connections, we need a model of a community-less network.

3.3.2 Correlation networks

In order to analyse functional networks created by the neurons in our ensemble, we measured the spike count correlation between each pair of neurons. These measurements induced an undirected weighted network between the neurons. The weight of each connection was equal to the spike count correlation between each pair of neurons.

3.3.3 Rectified correlations

At the time of writing, the community detection method outlined in [1] could only be applied to networks with positively weighted connections. But many neuron pairs were negatively correlated. To apply the community detection method, we *rectified* the network, by setting all the negative weights to zero.

(IDEA: We should also do negatively correlated network analysis. Reverse the signs of the correlations, then set the negative correlations to zero, then do the analysis.)

3.3.4 Weighted configuration model

The *weighted configuration model* is a canonical null network model for weighted networks. Given some data network, the weighted configuration model null network will preserve the degree sequence and weight sequence of each node in the data network. But the edges will be distributed randomly [6]. Any structure in the data network beyond its degree sequence and weight sequence will be removed in the weighted configuration model. So, this model can be used in testing the hypothesis that this extra structure exists.

3.3.5 Sparse weighted configuration model

The *sparse weighted configuration model* is another null network model. Similar in nature to the weighted configuration model (see section 3.3.4), but the sparsity of the data network is preserved in the null network. This is achieved by sampling from a probability distributon for the creation or non-creation of each possible connection, then distributing the weight of the data network randomly in this sparse network [1].

3.3.6 Spectral rejection

We made use of the spectral rejection algorithm as outlined in [1]. The spectral rejection algorithm is a method for finding structure in a network not captured by a supposed null model, if such structure exists.

To describe the method, we denote our data network matrix $\mathbf{W}$, we denote the expected network of our null network model as $\langle\mathbf{P}\rangle$. Then the departure of our data network from the null network can be described by the matrix

$$\mathbf{B} = \mathbf{W} - \langle\mathbf{P}\rangle \tag{4}$$

a common choice for $\langle\mathbf{P}\rangle$ in community detection is the ‘configuration model’ [6, 7]. The matrix $\mathbf{B}$ is often called the configuration matrix, in this context we will use the term ‘deviation matrix’ as it captures the deviation of $\mathbf{W}$ from the null model.

To test for structure in the network represented by $\mathbf{W}$, we examine the eigenspectrum of $\mathbf{B}$ and compare it the eigenspectrum of our null model. Firstly, note that since our data model doesn’t allow self loops, and is not directed, the matrix representing the network will be symmetric and positive semi-definite, and will therefore be invertible with real eigenvalues. We selected a null model with the same characteristics.

To find the eigenspectrum of the null model, we generated N samples from our null model $P_1, \ldots, P_N$, and we measured their deviation matrices $B_1, \ldots, B_N$. We then calculated the eigenspectrum of each of those samples. We calculated the upper bound of the null model eigenspectrum by taking the mean of the largest eigenvalues of $B_1, \ldots, B_N$. We calculated a lower bound on the null model eigenspectrum by taking the mean of the smallest eigenvalues of $B_1, \ldots, B_N$.

We then calculated the eigenspectrum of $\mathbf{B}$, our data network deviation matrix. If any of those eigenvalues lay outside of the upper or lower bounds of the null model eigenspectrum, this is evidence of additional structure not captured by the null model. If we chose the sparse weighted configuration model (seec section 3.3.5) as our null network model, then eigenvalues lying below the lower bound indicate n-partite structure in the network. For example, if one eigenvalue lay below the lower bound, this would indicate some bipartite structure in the data network. If any eigenvalues lay above the upper bound of the null model eigenspectrum, this is evidence of community structure in the data network. For example, one eigenvalue of $\mathbf{B}$ lying above the upper bound of the null model eigenspectrum indicates the presence of two communities in the network [7].

3.3.7 Node rejection

If there are d data eigenvalues lying outside of the null network eigenspectrum, the d eigenvectors corresponding to these eigenvalues will form a vector space. If we project the nodes of our network into this vector space, by projecting either rows or colmns of the data matrix, we can see how strongly each node contributes to the vector space. Nodes that contribute strongly to the additional structure will project far away from the origin, nodes that do not contribute to the additional structure will project close to the origin. We want to use this information to discard those nodes that do not contribute.

We can test whether a node projects *far* away from the origin or *close* to the origin using the eigenvalues and eigenvectors of $B_1, \ldots, B_N$. The jth eigenvector and eigenvalue of B_i gives a value for a null network's projection into the jth dimension of the additional structure vector space. The matrices $B_1, \ldots, B_N$ give N projections into that dimension. These projections are a distribution of null networks' projections. If the data nodes projection exceeds that of the null network projections this node is judged to project *far* from the origin, and therefore contribute to the additional structure. Otherwise, the node is judged to project *close* to the origin, and is therefore rejected [1].

3.3.8 Community detection

Another application for this d dimensional space is community detection.

References

[1] Mark D. Humphries, Javier A. Caballero, Mat Evans, Silvia Maggi, Abhinav Singh, *Spectral rejection for testing hypotheses of structure in networks*. arXiv:1901.04747, (2019)

[2] Marlene R Cohen, John H R Maunsell, *Attention improves performance primarily by reducing interneuronal correlations*. Nature Neuroscience 12, 1594–1600, (2009)

[3] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, Kenneth D. Harris, *Spontaneous behaviors drive multidimensional, brain-wide activity*. Science 364, (2019)

[4] Marlene R Cohen, Adam Kohn, *Measuring and interpreting neuronal correlations*. Nature Neuroscience 14, 811-819, (2011)

[5] J. J. Jun et al., *Fully integrated silicon probes for high-density recording of neural activity*. Nature 551, 232–236, (2017)

[6] Bailey K. Fosdick, Daniel B. Larremore, Joel Nishimura, Johan Ugander, *Configuring random graph models with fixed degree sequences*. arXiv:1608.00607 (2017)

[7] Mark D. Humphries, *Spike-train communities: finding groups of similar spike trains*. Journal of Neuroscience 31, 2321–2336, (2011)