AMP 205 Final Project Report

Xincheng Tan, Yaxin Lei
Harvard University
Cambridge, United States

For all code, refer to:
`https://github.com/XinchengTan/AM205_Final_Project`

December 18, 2020

Abstract

Graphical models have a wide range of applications due to their interpretable mathematical properties and the computational efficiency compared with certain neural-network-based models. In neuroscience, graphical models enable researchers to estimate a sparse functional connectivity graph among neurons given simultaneous neuron signal recordings. However, with the advances of the neuron imaging technology, the size of simultaneous neuron signal recordings grows significantly and thus poses computational challenges to graphical models. In this study, we aim to apply randomized linear algebra techniques to reduce the computational complexity of graphical models and examine whether it preserves the final estimation. Specifically we attempt to investigate how randomized matrix multiplication and randomized regression may improve existing graphical models.

Keywords: functional connectivity, covariance, graphical model, randomized linear algebra, optimization

1 Introduction

The connectivity pattern among neuron populations has been a fundamental topic in understanding the brain. There are two major types of connectivity: structural and functional connectivity. Structural connectivity is defined as the existence of the white matter tracts physically interconnecting the brain regions, while functional connectivity is the statistical correlation between neural signals, such as fMRI signals [9]. Although structural connectivity provides anatomical basis for how the neurons may interact with each other, it does not necessarily contain functional connectivity – researchers have discovered that certain brain regions without physical connections can exhibit strong functional connectivity [9]. Moreover, functional connectivity provides extra insight into how neurons function together. In particular, studies have found a strong correlation between static functional connectivity and certain neurological diseases such as depression, schizophrenia, and Alzheimer's disease [5].

Recently, with the development in functional neuroimaging, researchers are able to quantify and record the activities of a large neuron population in various animals. In particular, two-photon calcium imaging technology enables monitoring thousands of neurons simultaneously across weeks. To derive meaningful insights from these video datasets, a powerful calcium imaging analysis pipeline, which includes neuron detection, motion correction, spike inference etc., has been developed to extract more computationally tractable signals [7].

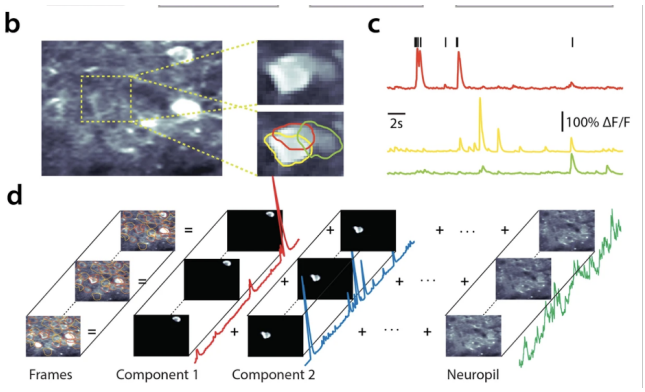

Figure 1: Two-photon calcium imaging captures neuron activity

In this study, we aim to apply appropriate statistical models on the preprocessed neuron signal datasets to estimate the functional connectivity graph. Specifically, we model the pairwise functional connectivity as the conditional dependence of a pair of neurons, given the activity of all other neurons in the population. By definition, it is the Pearson partial correlation coefficients between two neuron signal series. Therefore, sparse graphical models, which estimate the partial correlation matrix, can be used to estimate the functional connectivity between every neuron pair. However, as the size of the neuron population grows larger, applying graphical model on the entire dataset can be computationally inefficient. So we also aim to utilize matrix sketching techniques from randomized linear algebra (RLA) to reduce the overall computational complexity. However, there has not been any prior literature on such a combination, so our study is mainly exploratory and focuses on whether applying various sketching schemes could estimate a similar connectivity graph.

In Section 2, we give an overview of the graphical model and how it applies to functional connectivity estimation. Next, we discuss two application scenarios of RLA that can be applied in the graphical model estimation pipeline. In Section 3, we explain the basic randomized matrix sketching techniques and how they may speed up the connectivity estimation. Moreover, in section 4, we also investigate a few different randomized Lasso regression methods that may be applied in a crutial step of the graphical model. In Section 5, we will describe our experiment setup and demonstrate our results. Finally, we will conclude our finding and discuss potential directions for improvement in Section 6 and Section 7.

2 Graphical Models

Graphical models combine graph theory and probability theory to infer complex network structure within large systems. They aim to uncover an underlying graph $G(V, E)$, where each node represents a feature of interest and each edge reflects the conditional dependence between the two features. To estimate the functional connectivity, we focus on the existence of the connection instead of the direction of such connection. Thus, we only discuss undirected graphical models for the rest of this paper.

Formally, let $X = (X_v : v \in V = \{1, 2, ..., p\})$ be a p-dimensional random vector indexed by the vertices of an undirected graph $G = (V, E)$ where $E \subseteq V \times V$. Moreover, we say X satisfies the *pairwise Markov property* with respect to G if:

$$X_i \perp X_j | X_{V \setminus \{i,j\}} \iff (i, j) \notin E \qquad \forall i, j \in V, i \neq j \tag{1}$$

The pairwise Markov property translates each absence of an edge into 'full' conditional indepen-

dence. In other words, two random variables are conditionally independent if and only if the nodes representing them are not connected under the graphical model representation.

2.1 Gaussian Graphical Model

A graphical model is Gaussian when $X = (X_1, ..., X_p)$ is from a Gaussian distribution $\mathcal{N} = (\mu, \Sigma)$. A Gaussian conditional dependence graph can be estimated by determining the zero entries of the inverse covariance matrix $\Theta = \Sigma^{-1}$, also referred to as *precision* matrix. That is, if $\Theta_{ij} = 0$, $(i, j) \notin E$. Therefore, to estimate the underlying conditional dependence graph, we need to estimate the inverse covariance matrix of the underlying data distribution [2].

Gaussian graphical model (GGM) typically uses the maximum likelihood estimator (MLE) to estimate Θ. Let S be the sample covariance matrix for an i.i.d. Gaussian sample of size n. The MLE maximizes the following log-likelihood function

$$L(\Theta) = \log\det(\Theta) - \mathrm{tr}(S\Theta) \tag{2}$$

subject to Θ is positive definite. Note that if $n > p$, L admits a unique maximizer with probability one because S is almost surely positive definite [2]. On the other hand, if $n \leq p$, S is singular and L is unbounded. However, if the graph G is sparse in nature, the MLE of Θ may exist uniquely with probability one even if the number of observations are insufficient.

2.2 Graphical Lasso

A popular variant of general undirected Gaussian graphical model is the *graphical lasso* (Glasso) proposed by Yuan & Lin in [10] and Banerjee *et al.* in [1]. Based on the general Gaussian graphical model, it adds an ℓ_1 regularization term in the MLE objective function:

$$L_{gl}(\Theta) = \log\det(\Theta) - \mathrm{tr}(S\Theta) - \lambda\|\Theta\|_1 \tag{3}$$

where λ is a regularization parameter that controls the sparsity of the graph estimation.

The formulation of Glasso has a few advantages. First, after adding the regularization term, its objective function is still convex and the convex optimization can be solved in polynomial time. Second, when the number of observations is insufficient ($n < p$), Glasso has strong statistical guarantees which include convergence in Frobenius and spectral norm [4]. Third, Glasso provides a hyperparameter λ to control the sparsity in the graph while preserving the statistical significance of the selected connections, which is very convenient if we are only interested in certain strong connections and wish to ignore the weaker ones.

2.3 Optimization in Glasso

Given that the ℓ_1 regularization term in the objective function Equation (3) is non-differentiable, we cannot apply the traditional gradient descent methods to find its optimum. However, Banerjee *et al.* prove in [1] that instead of estimating Θ, consider estimate $\Sigma = \Theta^{-1}$ as follows. Let W be an estimate of Σ. They show that the problem can be solved by optimizing over each row and column of W in a block coordinate descent fashion.

Partition W and S as follows:

$$W = \begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix}, \quad S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12}^T & s_{22} \end{pmatrix} \tag{4}$$

They show that the solution w_{12}^* in W^* satisfies

$$w_{12}^* = argmin_w\{w^T W_{11}^{-1} w : \|w - s_{12}\|_\infty \leq \lambda\}. \tag{5}$$

Using convex duality, they further show that the dual program of Equation (5) is

$$\min_{\beta} \left\| W_{11}^{1/2}\beta - b \right\|^2 + \lambda \|\beta\|_1 \text{ where } b = \frac{1}{2} W_{11}^{1/2} s_{12}. \tag{6}$$

Based on the above derivation, Friedman *et al.* claim in [3] that Equation (6) looks like a lasso least square regression problem if W_{11} is considered as known. With this observation, they propose that at each step of solving the vector w_{12} during block coordinate descent in Equation (3), w_{12} can be solved by the lasso problem in Equation (6) treating W_{11} as known. The lasso least square problem, in turn, can be solved by regular coordinate descent method that descends along the objective function one direction at a time until convergence.

The full algorithm is presented below. At the end of each iteration of block descents in all p directions, the precision matrix Θ needs to be updated as in line 10.

Algorithm 1 Glasso($S \in R^{p\times p}, \lambda \in R$)

1: Initialize $W = S + \lambda I$ where $I \in R^{p\times p}$ is an identity matrix;
2: $\hat{B} \leftarrow 0_p \in R^{p\times p}$ is a zero matrix;
3: Before convergence **do**
4: **for** $j = 1$ to p **do**
5: Swap j^{th} column and row of W to the last column w_{12} and row w_{12}^T;
6: $\beta^* \leftarrow$ LASSO(W_{11}, s_{12});
7: Fill in w_{12} with $2W_{11}\beta^*$;
8: Store β^* in j^{th} row and column of $\hat{B}$;
9: **end for**
10: Update $\hat{\Theta}$ by $\hat{\Theta}_{pp} \leftarrow \frac{1}{W_{pp} - 2\sum k\neq p \hat{B}_{kp} W_{kp}}$ and $\hat{\Theta}_{kp} \leftarrow -2\hat{\Theta}_{pp}\hat{B}_{kp}$;
11: **return** $W, \hat{\Theta}$;

Note that LASSO in line 6 is a separate optimization routine which uses coordinate descent along each direction β_i for $i = 1, 2, ..., p-1$ to solve the objective function Equation (6).

2.4 Estimate Functional Connectivity

Suppose $X \in R^{p\times n}$ is the neuron signal data matrix where each entry X_{ij} is the activity intensity of neuron i at time j. To estimate the pairwise functional connectivity $\Theta \in R^{p\times p}$ among all p neurons, we use the following procedure:

Algorithm 2 EstimateFC($X \in R^{p\times p}, \lambda \in R$)

1: Standardize X by $X \leftarrow X - \mu_X$;
2: Compute sample covariance $S = XX^T$;
3: $W, \hat{\Theta} \leftarrow$ Glasso(S, λ);
4: $A \leftarrow 0_{p\times p}$;
5: Set $A_{ij} \leftarrow 1$ if $\Theta_{ij} \neq 0$;
6: **return** $A, W, \hat{\Theta}$;

2.5 RLA's Application in Graphical Lasso

Using the EstimateFC routine defined above, we observe that randomized linear algebra techniques (RLA) may help reduce the overall computation complexity in the following two ways:

1. **Covariance matrix multiplication approximation (Algorithm 2 line 2), which we will explain in section 3 Randomized Matrix Multiplication**;
2. **Solving Lasso regression inspired by randomized regression methods (Algorithm 1 line 6), which we will explain in section 4 Randomized Lasso Regression**

In the remaining parts of this report, we first implement various randomized algorithms on matrix multiplication and Lasso regression with our own extensions, and then evaluate these methods based on their computation efficiency and numerical performance in an isolated as well as integrated manner within the overall graphical model context.

3 Randomized Matrix Multiplication

As data size gets increasingly large in recent years, common computations like matrix multiplication, decomposition can be really slow and researchers start to explore randomized techniques in linear algebra to efficiently find solutions. In our functional connectivity estimation algorithm, one of the most important matrix multiplications lies in the calculation of sample covariance matrix that takes the form of XX^T. Neuron signal data in our project contains a large amount of observations that can potentially be omitted in sample covariance estimation. Randomized linear algebra in matrix multiplication, in this scenario, performs a sampling over the time axis. However, before we apply the technique, we first separately examine its mathematical foundation, actual implementation performance.

For traditional multiplication of matrices AB where $A \in R^{m\times n}, B \in R^{n\times p}$, the $(i,j)^{th}$ element is calculated element-wise by:

$$(AB)_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj}$$

Since there are $O(mp)$ elements in AB, the time complexity of this traditional matrix multiplication is quadratic $O(mnp)$.

However, in randomized matrix multiplication, we can rewrite the product AB as a summation of outer products of columns of A and B $AB = \sum_{i=1}^{n} A_{:,i}B_{i,:}$, where $A_{:,i}B_{i,:}$ is an $m \times p$ rank-one matrix. The main idea of randomized matrix multiplication is to sample among the n outer products according to a probability distribution to approximate for AB.

Depending on the selected sample size, randomized matrix multiplication may improve asymptotic runtime significantly. In this section, we implement and evaluate the performance of matrix multiplication under two different sampling distributions.

3.1 Basic Uniform Distribution Randomized Matrix Multiplication

The most straightforward way is to sample these rank-one matrices $A_{:,i}B_{i,:}$ from a uniform distribution $i \in 1, 2, ..., n$ and take the average of all samples (Algorithm 3 line 4). With limited sample size, although it might fail to include certain important samples compared to the below Min-Variance method, it has little overhead and is very efficient when sampling in a uniform distribution.

3.2 Min-variance Randomized Matrix Multiplication

The second method we consider is drawing outer products $A_{:,i}B_{i,:}$ from a sampling distribution that minimizes the sample variance. The key idea to choose columns of A and rows of B according to the probability distribution that is proportional to $\left\|A_{:,i}B_{i,:}\right\|_2$. In other words, we want to bias towards the rank-one matrices that have higher spectral norms. Since each outer-product matrix is rank-one, we can simplify its spectral norm as $\left\|A_{:,i}\right\|_2\left\|B_{i,:}\right\|_2$. Based on the pseudo code presented

Algorithm 3 Uniform Randomized Matrix Multiplication)

Input: $A \in R^{m\times n}, B \in R^{n\times p}$, a positive integer c representing the number of samples, $p_i = \frac{1}{n}$.
Output: Approximation $M \approx AB$

1: for t=1 to c;
2: Pick $i_t \in 1, 2, ..., n$ with probability p_i in i.i.d. trials with replacement;
3: $M = M + \frac{1}{p_i} A_{:,i_t} B_{i_t,:}$;
4: **return** $\frac{1}{c}M$

in Algorithm 3 , we set $p_i = \frac{\left\|A_{:,i}\right\|_2 \left\|B_{i,:}\right\|_2}{\sum_{i'=1}^{n} \left\|A_{:,i'}\right\|_2 \left\|B_{i',:}\right\|_2}$ in line 2-3.

Obviously, when matrices are large, with any sample size $c << n$, randomized methods should be faster than traditional matrix multiplication. However, this is only beneficial if we guarantee that the approximation is within tolerable error from the true result. The accuracy of the randomized matrix multiplication is crucial especially in our context where each step of the coordinate descent algorithm depends on the accuracy of sample covariance matrix calculated through randomized matrix multiplication. We attempt to evaluate its performance both through theory and simulation before adding it into our graphical model.

As shown below, the probability distribution $p_i = \frac{\left\|A_{:,i}\right\|_2 \left\|B_{i,:}\right\|_2}{\sum_{i'=1}^{n} \left\|A_{:,i'}\right\|_2 \left\|B_{i',:}\right\|_2}$ gives unbiased approximation, existing literature has also proved that it minimizes sample variance [6].

Proof: Fix i, j. For $t = 1, ..., c$ define $X_t = (\frac{A_{:,i}B_{i,:}}{cP_{i_t}})_{ij} = \frac{A_{ii_t}B_{i_tj}}{cp_{i_t}}$.

$$E[X_t] = \sum_{k=1}^{n} p_k \frac{A_{ik}B_{kj}}{cp_k} = \frac{1}{c}(AB)_{ij}$$

Therefore since $M_{ij}E = \sum_{t=1}^{c} X_t$, we have

$$E[M_{ij}] = \sum_{t=1}^{c} E[X_t] = (AB)_{ij}$$

3.3 Randomized Matrix Multiplication Performance Evaluations

Would the overhead of random sampling be too large to apply to our covariance matrix calculation? How accurate would the approximation be for different types of matrices? How does the approximation accuracy change as sample size varies? We aim to answer these questions before adding randomized matrix multiplication into our graphical model.

Fig.2 and Fig.3 show the relative error of randomized matrix multiplication with varying matrix size. In both simulations, sampling ratio is 0.5. Fig.1 shows that matrices with elements i.i.d drawn from uniform distribution, and Fig.2 uses sparse matrices where each element is drawn i.i.d from $Bernoulli(0.2, 0.8)$. We see that as matrix size increases, the relative error significantly decreases especially in the case where matrix elements are uniformly generated. We also observe that Min-variance randomized matrix multiplication performs slightly better than the uniform randomized matrix multiplication in the sparse case. Both trends show that the randomized techniques may faithfully restore the result of the traditional matrix multiplication.

We continue to test how random sampling size influences the similarity between traditional matrix multiplication and randomized approximations. We randomly generated two matrices $A \in R^{400\times 600}$, $B \in R^{600\times 380}$ ($n = 600$). In Fig.4, we observe that indeed uniform randomized method has higher variance. As sample size approaches 200 with a sampling ratio of $\frac{1}{3}$, the Frobenius norm

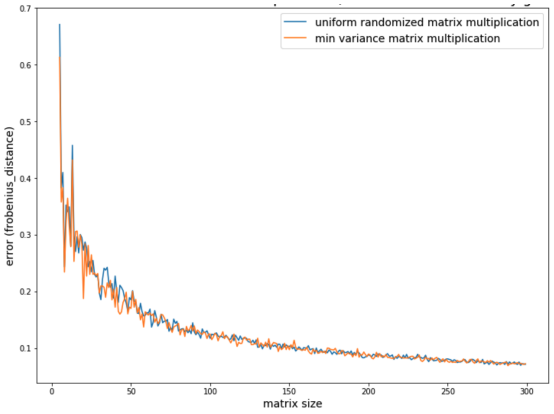

Figure 2: Relative error of randomized matrix multiplication (matrix element uniform randomly generated)

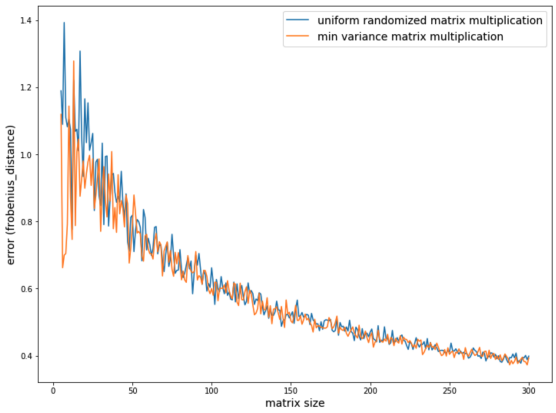

Figure 3: Relative error of randomized matrix multiplication (matrix element sparsely generated)

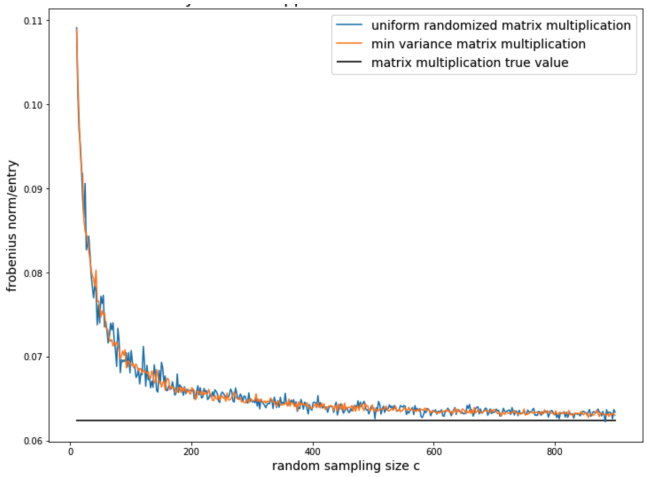

Figure 4: Frobenius norm per entry for different types of matrix multiplication

per entry approaches the straight black line representing the true Frobenius norm value. Therefore, from above three simulation results, we can see that the approximation accuracy of randomized matrix multiplication yields reasonable accuracy even when sample ratio is less than half.

In terms of speed, as expected, we also see how despite the large overhead for randomized method to compute the random distributions, as matrix size increase, traditional multiplications soon exceed that of the randomized methods as shown in Fig.5.

(We are aware that evaluating speed using computer timing functions may sometimes produce controversial results due to different ways of timing runtime across multiple CPU cores. Rigorous timing often needs to take into account parallelism, code type, computer memory etc. In our case, the purpose of this timing check is to roughly confirm our guess that the overhead in randomized algorithms can be ignored as matrix size increases. During the implementation of these simulation comparisons, we reduced the use of numpy as much as possible since it is backed by highly optimized C code which is inherently faster. Our comparison is between a naive implementation of traditional matrix multiplication using for loops.)

Based on the above exploratory results on randomized matrix multiplication, we have confirmed that its accuracy and speed improvement is satisfactory to be included in our graphical model.

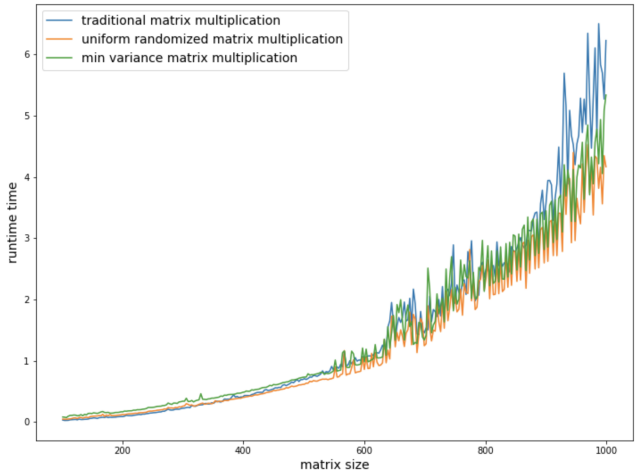

Figure 5: Matrix multiplication speed

4 Randomized Lasso Regression

A thorough examination of Graphical model inspires us on a second place where we may be able to optimize through randomized linear algebra. Recall that in graphical model, every block coordinate descent step is actually a Lasso optimization problem, which is a regularized linear regression problem. Therefore, we also did an exploratory study on randomized linear regression. Specifically, we implemented, extended and tested the below three methods:

- Random sampling method for linear regression;
- Random projection method for linear regression;
- Accelerated sparse linear regression via random projection.

Since most existing literature focuses on randomization techniques in general linear regression, we extended them to include the $l1$ regularizing factor, and adapted them to solve Lasso Regression in our context. Before including our randomized regression algorithms into our graphical model, we aim to do isolated evaluations first on each individual method just like in matrix multiplication.

4.1 Random sampling method for Lasso regression

In traditional regression settings, we usually attempt to solve for x in the system $Ax = b$ to minimize least square error, sometimes with the addition of a penalizing factor. The essence of random sampling based methods is to find a matrix P such that we can transform solving $Ax = b$ into a smaller problem of $PAx = Pb$ by randomly sampling information from data matrix A. This method was introduced by Drineas in his work titled *Faster Least Squares Approximation*.

In Algorithm 4, the algorithm attempts to find an optimal sample size r (line 1) to minimize error as well as construct a transformation matrix $P = S^T HD$ to randomly sample from matrix A. Different from the original algorithm, in line 9-10, we extend the algorithm to solve the new system $PAx = Pb$ using Lasso regression with a penalizing factor of λ. Therefore, our new objective function is $||S^T HD\ Ax' - S^T HDb||_2^2 + \lambda||b||_1$.

Our initial testings with randomized lasso regression proves successful. Our simulation data is randomly generated from a linear function where $y = 2 + 0.6X + stochasticity$. In Fig.4, the orange line depicts a traditional deterministic fitting of Lasso regression, and the blue line depicts the lasso

Algorithm 4 Random sampling method for Lasso regression

Input: $A \in R^{n \times d}, b \in R^n$, error tolerance ϵ, penalizing factor λ
Output: x'

1: Compute $r = max\{48^2 d \ln(40nd) \ln(100^2 d \ln(40nd)),\ 40d \ln(40nd)/\epsilon\}$
2: Set $S = 0, S \in R^{n \times r}$;
3: for $l = 1, ..., r$ do
4: Choose $k_l \in 1, ..., n$ with replacement with probability $P(k_l = k) = \frac{1}{n}$
5: Let $S_{k_l l} = \frac{1}{\sqrt{r p_k}} = \sqrt{n/r}$
6: end for loop
7: Let $H \in R^{n \times n}$ be the normalized Hadamard transform matrix;
8: Let $D \in R^{n \times n}$ be a random diagonal matrix with entries 1, -1 with equal probability;
9: Solve for x' in reduced problem Lasso $S^T HD\ Ax = S^T HDb$
10: **return** x' such that it minimizes $||S^T HD\ Ax' - S^T HDb||_2^2 + \lambda||b||_1$

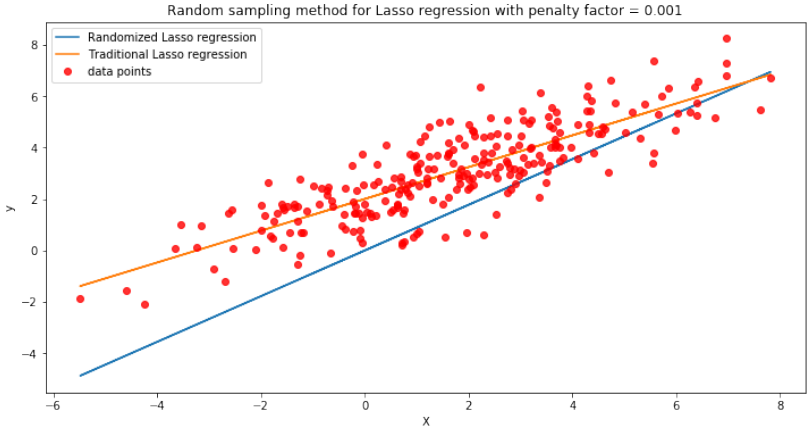

Figure 6

regression generated by our randomized sampling method. Although the blue line is not perfectly optimal, we can still see that it yields reasonable result in a single lasso regression.

However, a closer look into the code generates concerns for integration into our graphical models, a concern that is confirmed through testing within graphical models. In line 1 of Algorithm 4, we need to first compute the sample size r which takes the max of two different values determined by the dimensions of A and error tolerance. In most circumstances, especially when A is large and a square matrix, our sample size r blows up to a huge number that violates our original purpose of achieving efficiency while compromising minimal accuracy. Since graphical models need to solve a Lasso optimization problem in every coordinate descent step, this algorithm needs to be highly efficient to cope with the repetitiveness.

Another phenomenon we've observed is that, while error is tolerable and reasonable in a single iteration of Lasso test, the error accumulates as we iterate through the coordinate descent step. Common outcomes we see include inability to converge, certain matrices loosing full rank, and emergence of complex matrices after raising to the half power.

4.2 Random projection method for Lasso regression

Next, we move on to evaluate random projection method, also introduced by Drineas. The essence of random projection method is to project matrix A in $Ax = b$ onto a smaller subspace to decrease the size of the problem and solve the new problem $PAx = b$.

Algorithm 5 Random projection method for Lasso regression

Input: $A \in R^{n\times d}, b \in R^n$, error tolerance ϵ, penalizing factor λ
, C_q, C_k Output: x'

1: Compute q s.t. $q \geq \frac{C_q d \ln(40nd)}{n}(2\ln n + 16d + 16)$, and r s.t. $r \geq max\{C_k(118^2 d + 98^2), \frac{60d}{\epsilon}\}$
2: Let $T \in R^{r\times n}$ be a matrix with i.i.d random entries, $T_ij = \{\sqrt{\frac{1}{rq}}, -\sqrt{\frac{1}{rq}}, 0\}$ with probabilities $\frac{q}{2}, \frac{q}{2}, 1-q$, respectively;
3: Let $H \in R^{n\times n}$ be the normalized Hadamard transform matrix;
4: Let $D \in R^{n\times n}$ be a random diagonal matrix with entries 1, -1 with equal probability;
5: Solve for x' in reduced problem Lasso $THD\ Ax = THDb$
6: **return** x' such that it minimizes $||THD\ Ax' - THDb||_2^2 + \lambda||b||_1$

In Algorithm 5, we first calculate probability q and sample size r in line 1, and then construct our transformation matrix $P = THD$ to facilitate the mapping of A and b on to a smaller subspace PA and Pb.

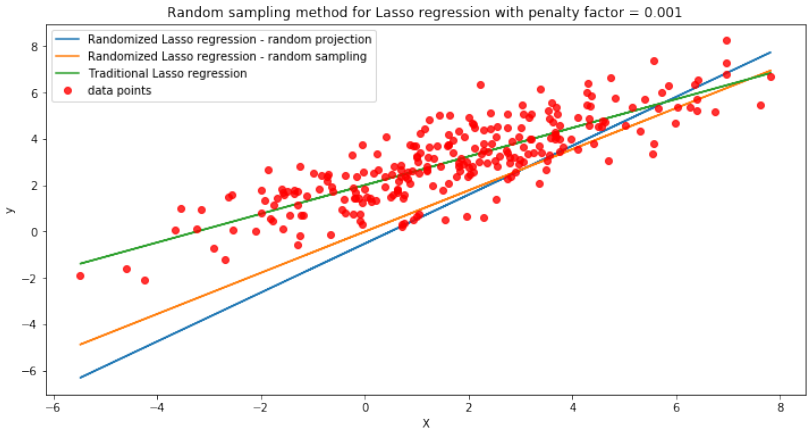

Figure 7

Fig.7 shows three Lasso regression in a single graph. Orange line represents Lasso regression using random sampling, and Blue line represents Lasso regression using random projection. We draw the same conclusion as that of random sampling. The approximation accuracy is acceptable for a single Lasso, but error accumulates once the random projection method is integrated into our graphical model. Apart from these, the selection of C_k, C_q and error tolerance is also extremely important, as they determine probability q (line 1-2) as well as sample size r (line 1). When testing such algorithm, careless selections of C_q and C_k, as well as invalid q greater than 1 or smaller than 0, easily lead to huge sample size r unrealistic for computation. The algorithm only works under careful hyper parameter tuning. Such attribute makes this algorithm costly in our case as we require repetitive Lasso problems solving procedures for each step of coordinate descent.

4.3 Accelerated sparse Lasso regression via random projection

The next algorithm we attempt to test is another variation of random projection specifically for data points fewer than the number of parameters, $n < d$. It was originally published by Zhang in [11].

It first computes the low-rank approximation $\hat{A}$ for the data matrix A efficiently. And then, uses the homotopy strategy to iteratively recover the optimal pattern x_T by solving a sequence of subproblems in line 6 of Algorithm 6. As x approaches the optimal x_T, the penalizing factor λ decreases by a factor of η.

Algorithm 6 Accelerated Sparse Linear Regression via Random Projection

Input: the data matrix $A, y, \lambda_0, \lambda_{min}, \gamma, k, \eta$
Output: x_T

1: Sample a $d \times k$ random matrix Z with $Z_i j\ N(0,1)$
2: Compute the QR decomposition of $A^T Z$, i.e., $A^T Z = QR$
3: Approximate A by $\hat{A} = (AQ)Q^T = W^T Q^T$
4: Initialize $X_0 = 0$
5: for $t = 0$ to $T - 1$ do:
6: Update $X_{T+1} = \min_{X \in R^d}\{-\frac{1}{n}(X - x_t)^T \hat{A}(y - \hat{A}^T X_t) + \lambda_t |X|_1 + \frac{\gamma}{2}|x - x_t|_2^2\}$, where $\lambda_t = max(\lambda_{min}, \lambda_0 \eta^t)$.
7: **return** x_T

Like the previous algorithm, the success of this algorithm and accurate results also relies largely on the smart selection of hyper parameters, except with even more hyper parameters.

4.4 Randomized Regression Takeaways

By testing these three methods, we find that in most cases where we only concern a single Lasso problem, randomized methods produce reasonable results, often facilitated with a smart selection of hyper parameters. But in our context, a Lasso optimization is solved in every coordinate descent step. Through isolated and integrated exploratory testing, we conclude that the below three reasons significantly hinder the use of RLA in our regression:

1. Large sample size: Our sample size r is often dependent on the dimension of the input matrix A, error tolerance, and hyper parameters. We observe that the sample size r often blows up, which defeats the purpose of using randomized regression for efficiency in the first place.
2. Error accumulation: As coordinate descent iterates, the error generated by randomized approximation accumulates, the the loss of information also increases.
3. Difficulty in hyper parameter tuning: This phenomenon is especially evident for the later two algorithms proposed above. The difficulty of selecting hyper parameters is only exacerbated when we need to solve Lasso optimization problems within every iteration.

5 Experiments

The fundamental question of this study is to compare whether applying RLA methods at any step would influence the estimated functional connectivity. We further ask these questions: Can RLA faithfully restore the connectivity estimated by graphical lasso without RLA? If there is a difference, how can we quantify it and what is the qualitative relation of the difference with respect to the hyper parameter in RLA? Would different RLA methods incur the same difference?

5.1 Dataset

In order to answer these questions, we apply graphical model on a real calcium imaging dataset published by Stringer in [8]. It contains nine simultaneous recordings as preprocessed neuron signals in the primary visual cortex of a mouse while it is performing spontaneous activities on a treadmill. Each recording spans a consecutive 90 to 140 minutes, with 3 or 4 datapoints per second.

Each dataset records around 10,000 neurons spanning over 9 or 11 depth levels 30 or 35 μm apart. Fig.8 shows the location of these neurons from one of the recordings. On the right are three randomly-selected neuron signals at different depths recorded over a total of 120 minutes.

In our experiments, we fit a graphical model on a subset of 200 neurons randomly selected at each depth for all nine datasets, and we take minute 10 to minute 30 as full observations, which already provides sufficient observations for $p = 200$.

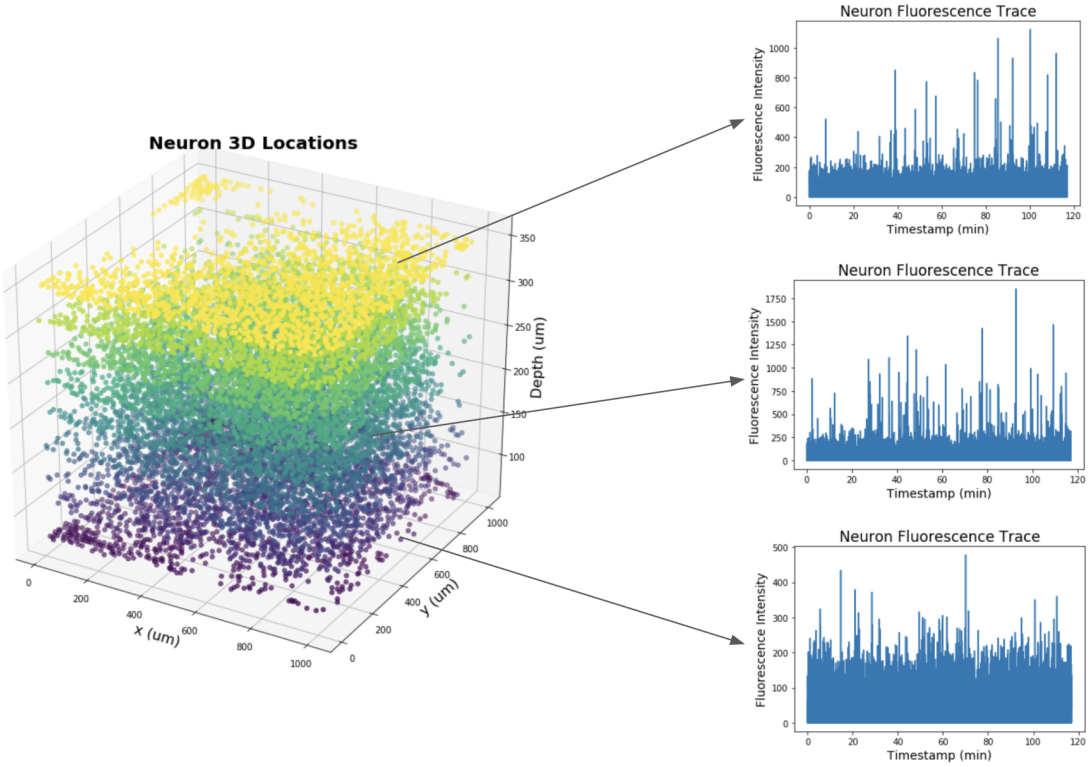

Figure 8: Relative location of nine layers of neurons (left) and example neuron signals from three different layers (right)

5.2 Preprocessing

We preprocess each signal data matrix $X \in R^{p \times n}$ in three main steps:

(1) we remove all neurons whose signal variance is zero because these are likely recording error. By our examination, each recording only contains a very small portion of such erratic neuron signals.

(2) we center X by subtracting the mean $X \leftarrow X - \hat{\mu}$. This step is, by definition, required to compute the sample covariance. Moreover, since the $\hat{\mu}$ is also an estimation of the true mean. We use the unbiased covariance $\frac{1}{n-1} XX^T$ as the sample covariance.

(3) we standardize the sample covariance S to a correlation matrix S' where $S'_{ij} = \frac{S_{ij}}{\sqrt{S_{ii}S_{jj}}}$. This step significantly enhances the numerical stability of graphical model since pairwise covariance only measures direction of pairwise linear relationship while correlation measures both direction and strength.

5.3 Results

In our experiments, the column sampling percentage c is the only hyper parameter used by the RLA matrix multiplication. Therefore, we vary this percentage and compare the final connectivity estimations by the resulting sample correlation matrices. In Section 5.3.1, we visualize the functional connectivity graphs under $c = 0.35$ and $c = 0.85$ respectively. In Section 5.3.2, we propose five quantitative comparison metrics and discuss how their measure changes with respect to c.

5.3.1 Functional connectivity estimation

Regularization strength $\lambda = 0.1$

c = 0.35

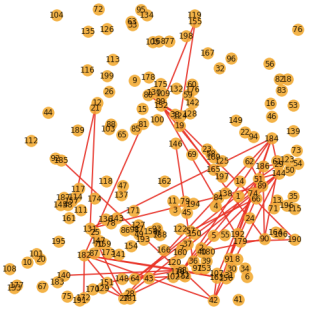

Figure 9: Full sample

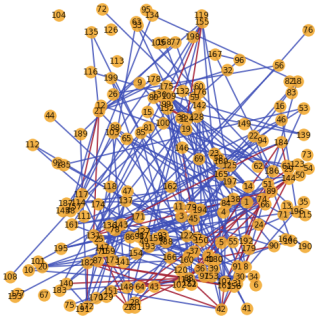

Figure 10: RLA uniform

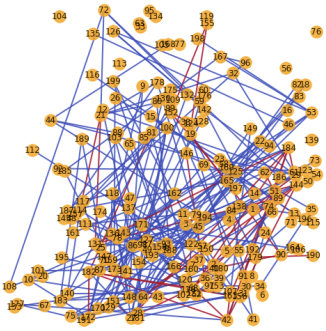

Figure 11: RLA min-variance

c = 0.85

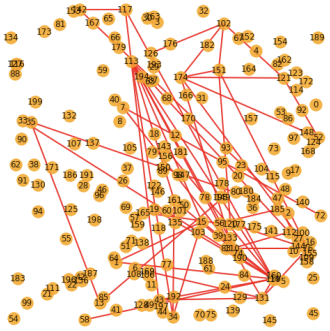

Figure 12: Full sample

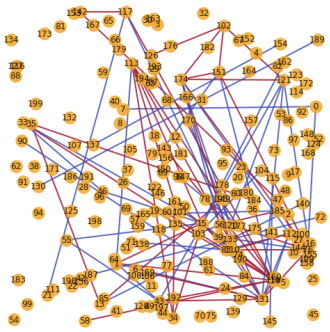

Figure 13: RLA uniform

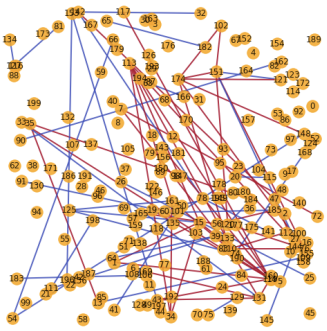

Figure 14: RLA min-variance

The yellow dots in Fig.9 and 14 represent 200 neurons at the same depth and their relative positions are determined by their actual physiological location in the mouse brain. The red edges

are estimated by using the full sample while the blue ones are estimated by the RLA sampled matrices.

Clearly, both RLA techniques recovers the edges estimated by the graphical lasso on the full data. However, when c is small, the connectivity graph is significantly less sparse than the one estimated by the full sample.

Moreover, comparing Fig.10 to Fig.11 as well as Fig.13 to Fig.14, we observe that the minimum-variance RLA technique seems to better recover the sparsity of the graph without losing the important edge connections.

5.3.2 Comparison Metrics

To quantify the estimation difference, we propose the following five metrics where the first two analyze the numerical difference of the estimated precision matrix $\hat{\Theta}$ and the last three quantifiy the difference in the connectivity graph structure.

- **Entry-wise Frobenius norm**

The first metric is the entry-wise Frobenius norm of $\hat{\Theta}_{full}$ with $\hat{\Theta}_{uni}$ and $\hat{\Theta}_{min-var}$, respectively. From Fig.16 and 17, on all nine recording datasets, as c increases, the recovered precision matrix has smaller Frobenius norm compared to the precision matrix of the full sample. Fig.15 is the overall average of Frobenius norm across all sessions under the two different RLA techniques. Obviously, minimum-variance method has a significantly smaller norm deviation from the baseline estimation.

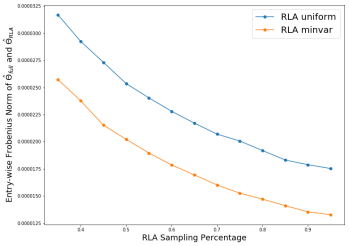

Figure 15

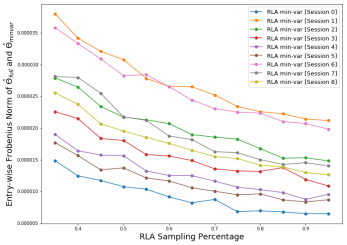

Figure 16

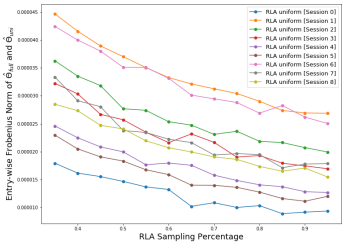

Figure 17

- **Matrix correlation**

Although the numerical difference in matrix norm is indicative of the difference, it is not clear whether either of the individual estimation produces a similar matrix as the baseline. Therefore, we further examine the correlation over the non-diagonal entries. That is, we flatten the non-diagonal entries of each $\hat{\Theta}$ to a vector and compute the correlation coefficient of $\hat{\Theta}_{uni}$ and $\hat{\Theta}_{min-var}$ with the baseline. From Fig.18, we can see that both as a relatively high correlation and minimum-variance RLA estimation is more similar to the baseline.

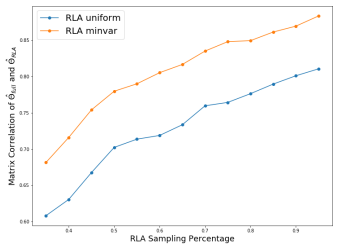

Figure 18

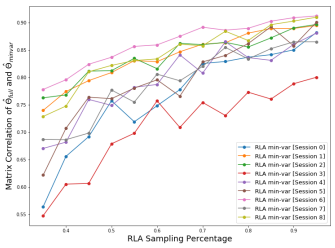

Figure 19

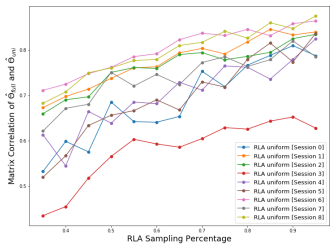

Figure 20

- **Graph structure accuracy**

Aside from the numerical difference of the estimated precision matrix, we also look into the difference in graph structure, i.e. the adjacency matrix. We first compute the estimation accuracy: the number of adjacency entry matches with respect to A_{full}. From Fig.21 – 23, we can see that increasing c enhances the accuracy, but the two overlapping curves in Fig.21 shows that there is no significant difference between the two RLA techniques. The main reason is that the ultimate graphs estimated by all three methods are fairly sparse, so the matches in the lack of edges largely boost the overall accuracy such that the two curves coincide.

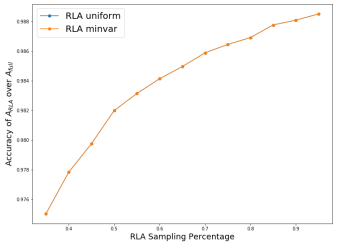

Figure 21

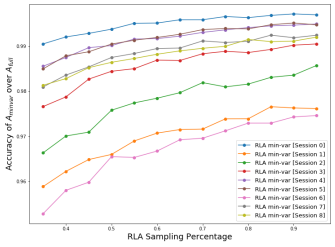

Figure 22

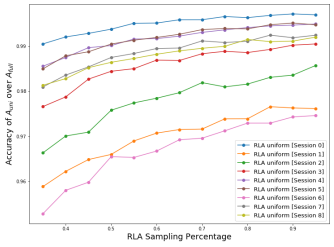

Figure 23

- **True positive rate**

Next, we consider the true positive rate (TPR), that is, out of all the edges in A_{full}, what percentage of them are recovered by A_{uni} and A_{RLA} respectively? From Fig.24, we see that both RLA techniques has a fairly high TPR, but minimum-variance RLA has a significantly higher TPR than uniform RLA. In Fig.25 and 26, we see that unlike other metrics, increasing c does not always increase TPR, although the overall trend is positive.

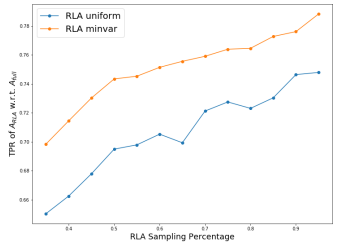

Figure 24

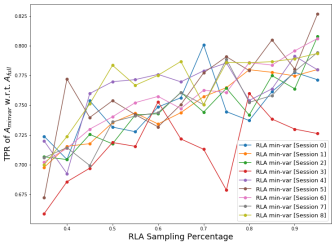

Figure 25

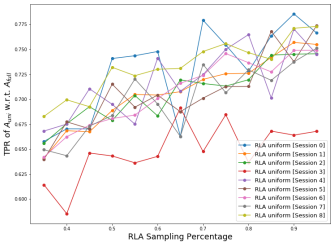

Figure 26

- **False discovery rate**

Last, we also consider the false discovery rate (FDR), which is the percentage of estimated edges that does not belong to A_{full}. From Fig.27 – 29, as c increases, false discovery rate of both methods drop, but minimum-variance RLA has a lower FDR.

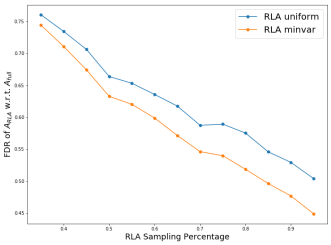

Figure 27

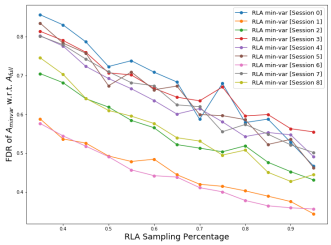

Figure 28

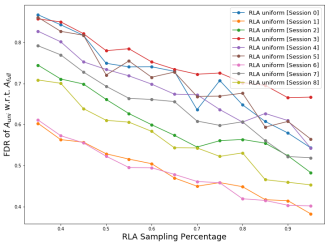

Figure 29

6 Conclusion

Based on our experiments in all neuron planes in the nine different recording datasets, we conclude our findings:

(1) As the column sampling percentage increases, graphical lasso estimation becomes closer to the baseline model (without RLA);

(2) Minimum-variance RLA matrix multiplication generates estimations significantly more similar to the baseline model than uniform RLA in almost every comparison metric that we propose;

(3) Although randomized methods produce reasonable results in single Lasso regression problem, graphical lasso isn't suitable for randomized regression due to large sample size, accumulation in loss of information and error, and difficulty in hyperparameter tuning.

7 Future Work

The major goal of this project is to conduct exploratory studies on how RLA can be used in graphical models for neuron functional connectivity estimation with implementations and extensions on current RLA techniques. We focus on the overall precision matrix and graph structure estimation. For future works, the below related extensions would be interesting to explore.

- **Neighborhood connectivity:** In the future, it would be interesting to examine neighborhood connectivity patterns and the interpretation of specific sub-graph with or without connections.
- **Data-oriented sampling method:** Current random sampling method used in randomized regression is reliant upon the calculation of $P = S^T HD$. The selection of samples largely depends on the Hadamard transformation matrix H and the randomly generated diagonal matrix D. In the future, we would like to explore data-driven sampling methods to improve

estimation accuracy, e.g., how Min-variance Matrix Multiplication samples outer products from a data-dependent probability distribution.

- **General Modeling Improvement - Generalization to Non-Guassian Data:** One of our model assumptions in the current project is Guassian data. It would be interesting to generalize current model and evaluate performances on non-Guassian data.
- **Speed Optimization:** In addition to the isolated speed comparisons between naive implementations of traditional matrix multiplication and randomized matrix multiplication, we would also like to conduct a speed test on overall models with further acceleration techniques such as extensive hyper parameter tuning and parallel computing implementation. More rigorous testings and improvement in speed would add further evidence in evaluating the appropriateness of RLA applications in graphical models.

References

[1] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9(15):485–516, 2008.

[2] M. Drton and M. H. Maathuis. Structure learning in graphical modeling, 2016.

[3] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the lasso, 2007.

[4] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Quic: Quadratic approximation for sparse inverse covariance estimation. *J. Mach. Learn. Res.*, 15(1):2911–2947, Jan. 2014.

[5] M. M. Jones DT, Vemuri P. Non-stationarity in the resting brain's modular architecture. *PLos One*, 7, 2012.

[6] P.-G. Martinsson and J. Tropp. Randomized numerical linear algebra: Foundations algorithms, 2020.

[7] E. A. Pnevmatikakis. Analysis pipelines for calcium imaging data. *Current Opinion in Neurobiology*, 55:15 – 21, 2019. Machine Learning, Big Data, and Neuroscience.

[8] C. Stringer, M. Pachitariu, N. Steinmetz, C. B. Reddy, M. Carandini, and K. D. Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437), 2019.

[9] L. Q. Uddin. Complex relationships between structural and functional brain connectivity. *Trends in Cognitive Sciences*, 17:600 – 602, 2013.

[10] M. Yuan and Y. Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

[11] W. Zhang, L. Zhang, R. Jin, D. Cai, and X. He. Accelerated sparse linear regression via random projection. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2337–2343. AAAI Press, 2016.