A deep learning approach to recover conditional independence graphs

Harsh Shrivastava[1] **Urszula Chajewska**[1] **Robin Abraham**[1] **Xinshi Chen**[2]

[1]Microsoft Research, Redmond, USA, [2]School of Mathematics, Georgia Institute of Technology

Abstract

Probabilistic Graphical Models are generative models of complex systems. They rely on conditional independence assumptions between variables to learn sparse representations which can be visualized in a form of a graph. Such models are used for domain exploration and structure discovery in poorly understood domains. This work introduces a novel technique to perform sparse graph recovery by optimizing deep unrolled networks. Assuming that the input data $X \in \mathbb{R}^{M \times D}$ comes from an underlying multivariate Gaussian distribution, we apply a deep model on X that outputs the precision matrix Θ. Then, the partial correlation matrix P is calculated which can also be representated as the conditional independence graph. Our model, `uGLAD`[1] , builds upon and extends the state-of-the-art model `GLAD` [43] to the unsupervised setting. The key benefits of our model are (1) `uGLAD` automatically optimizes sparsity-related regularization parameters leading to better performance than existing algorithms. (2) We introduce multi-task learning based 'consensus' strategy for robust handling of missing data in an unsupervised setting. We evaluate performance on synthetic Gaussian data, non-Gaussian data generated from Gene Regulatory Networks, and present a case study in anaerobic digestion.

Keywords: Graphical Lasso, Deep Learning, Unrolled Algorithms, Deep unfolding, Conditional Independence graphs, Sparse graphs

1 Introduction

Probabilistic graphical models (PGMs) [30, 25] are generative models of complex systems, used to describe dependencies within a set of random variables and visualize the structure of a domain. The models rely on conditional independence assumptions between variables, which result in sparse representation and enable efficient inference. In the graphical representation of the models, conditional independence is indicated by an absence of an edge between two variables. Such models can be learned from observational data [17, 12]. Structure discovery enabled by PGMs is important for new and poorly understood domains where relationships between variables are not known. PGMs have been used in various domains, including medical diagnosis [19, 18], fault diagnosis, genomic data analysis via gene regulatory networks [27, 33, 1, 45], speech recognition, and in finance [16].

The problem of recovering the structure from observational data is particularly difficult in high-dimensional settings, where the number of features may be larger than the number of observations. We focus specifically on learning undirected models where the data is assumed to have been generated from a multivariate Gaussian distribution [12, 3, 52, 51, 43]. In such cases, the goal is to estimate a sparse inverse covariance matrix. Sparsity is typically enforced by the use of ℓ_1 (lasso) regularization.

Assume we have a d-dimensional multivariate Gaussian random variable $X = [X_1, \ldots, X_d]^\top$ with m observations. The goal is to estimate its covariance matrix Σ^* and precision matrix $\Theta^* = (\Sigma^*)^{-1}$.

[1]`uGLAD` code: `https://github.com/Harshs27/uGLAD`

NeurIPS 2022 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2022).

Θ^* encodes conditional independence assumptions between variables: the ij-th component is zero if and only if X_i and X_j are conditionally independent given the other variables $\{X_k\}_{k\neq i,j}$. This problem is known as the sparse graph recovery problem and usually formulated (following [12]), as the ℓ_1-regularized maximum likelihood estimation

$$\hat{\Theta} = \arg\min_{\Theta \in \mathcal{S}^d_{++}} -\log(\det \Theta) + \text{tr}(\hat{\Sigma}\Theta) + \rho \left\|\Theta\right\|_{1,\text{off}}, \tag{1}$$

where $\hat{\Sigma}$ is the empirical covariance matrix based on m samples, $\mathcal{S}^d_{++}$ is the space of $d \times d$ symmetric positive definite matrices (SPD), and $\|\Theta\|_{1,\text{off}} = \sum_{i\neq j} |\Theta_{ij}|$ is the off-diagonal ℓ_1 regularizer with regularization parameter ρ. The use of this estimator is justified even for non-Gaussian X, since it is minimizing an ℓ_1-penalized log-determinant Bregman divergence [36]. The problem in Eq. 1 is a convex optimization problem. It can be solved by many algorithms, see Section 2 for examples.

However, classic approaches have their limitations in both the statistical aspect and computational aspect. Statistically, the classic formulation uses a single regularization parameter ρ for all entries in the precision matrix Θ, which may not be optimal. A recent theoretical work [47] validates the use of adaptive parameters. [43] has proposed a model with multiple regularization parameters called `GLAD` and has shown emprical evidence that such a model pushes the sample complexity limits. Based on that evidence, we hypothesise that one may obtain better recovery results by allowing the regularization parameters to vary across the entries in the precision matrix. However, it is hard for traditional approaches to search over a large number of hyperparameters. Computationally, the complexity of solving the optimization depends on the convexity of the objective, the step sizes of the algorithm, the initialization, the design of the update steps, etc. Different problems may require different designs of the algorithm to achieve a better efficiency. A unified design for all problems may not be optimal.

In this work, we propose `uGLAD` (unsupervised-`GLAD`), which is a deep model that can recover sparse graphs in an unsupervised manner. As the name suggests, it builds upon and extends the `GLAD` model, which recovers sparse graphs under supervision. `uGLAD` uses the same objective function as `uGLAD`. Using an additional variable Z, the ℓ_1-regularized maximum likelihood from Eq. 1 can be written as

$$\hat{\Theta} = \arg\min_{\Theta \in \mathcal{S}^d_{++}} -\log(\det \Theta) + \text{tr}(\hat{\Sigma}\Theta) + \rho \left\|Z\right\|_1, s.t. \quad Z = \Theta$$

Now, including the constraint as squared penalty term λ we obtain the reformulated objective as

$$\hat{\Theta}_\lambda, \hat{Z}_\lambda = \arg\min_{\Theta, Z \in \mathcal{S}^d_{++}} -\log(\det \Theta) + \text{tr}(\hat{\Sigma}\Theta) + \rho \left\|Z\right\|_1 + \tfrac{1}{2}\lambda \left\|Z - \Theta\right\|_F^2 \tag{2}$$

Note that introducing the variable Z helps in splitting the objective into 2 parts and those can be optimized alternately using the Alternating Minimization algorithm. A Conditional Independence (CI) graph can be obtained from this precision matrix by calculating the partial correlation matrix. The CI graphs are very informative as they describe how the features in a domain interact with each other. Furthermore, they can additionally capture negative partial correlations between features which are usually not modeled by traditional graphs.

Key contributions of this work:

- *Extending `GLAD` to unsupervised setting*: The `uGLAD` doesn't rely on availability of ground truth to do graph recovery.
- *Adaptive hyperparameters*: The `uGLAD` architecture design enables the hyperparameters to optimally adapt at each step of the unrolled Alternating Minimization (AM) algorithm [43] (also known as 'deep unfolding') that leads to its superior performance.
- *Automatically decide optimum sparsity parameters*: The sparsity of the recovered graph is highly sensitive to the choice of the regularization hyperparameters. Instead, `uGLAD` models hyperparameters within the neural network framework and they are directly optimized for the `uGLAD` objective defined above. So, there is no need to separately optimize the sparsity hyperparameters which is otherwise a computationally expensive process.
- *Runtime efficiency*: The `uGLAD` software can run on GPUs for higher time efficiency and scalability.
- *Missing data handling*: The `uGLAD` framework can also be used for multi-task learning. We leverage this property further to develop a novel 'consensus' strategy to robustly handle missing data.

2 Related work

Traditional algorithms for graphical lasso: These are primarily iterative methods for optimizing the graphical lasso objective. Main methods developed for this problem in the last two decades are

detailed in the survey paper by Witten et al. [49]. The variant of the Block Coordinate Descent (BCD) algorithm by Friedman et al. [12] is widely used to recover graphs using the graphical lasso method. This method is also implemented in the popular python `sklearn' package. The G-ISTA algorithm by Rolfs et al. [44] is based on the iterative shrinkage thresholding procedure; it is one of the prominent methods based on using the proximal gradient descent approach. The Alternating Direction Method of Multipliers (ADMM) [9] has also been successfully used in various graphical lasso based applications.

Deep Learning approaches for graph recovery: DeepGraph (DG) [31], is a supervised deep learning method that takes in the input samples and outputs the corresponding adjacency matrix which shows the connections between input features. DeepGraph architecture consists of many convolutional layers followed by multi-layer perceptrons that finally decides whether an edge is present between every combination of features. Another relevant DL method roughly based on modeling the input data with a Variational Autoencoder for graph recovery is DAG-GNN [51]. These deep architectures have very high number of learnable parameters, which is a significant drawback. Hence, we pursue a different line of research (using inductive biases) which gives similar performance with significantly reduced number of learnable parameters and brings more interpretability as shown in [43].

Deep learning models using inductive biases: Improved performance often results from including domain knowledge in the design/initialization of deep learning architectures. For instance, [42] presents a generic technique to use a probabilistic graphical model as a prior to design a deep model. The authors were able to show enhanced performance on the document classification task by leveraging the Latent Dirichlet Allocation prior. Another way of including prior knowledge about the domain is using an optimization algorithm for a related objective function as a template to design the deep architecture. Unrolling the optimization algorithms and parameterizing the step updates using neural networks have been fairly successful for many tasks [26, 8, 39, 34, 45].

This work focuses on recovering undirected graphs, specifically based on optimizing the graphical lasso objective function. Hence, we skipped discussing the methods developed to recover Directed Acyclic Graphs (DAGs). The work most closely related to ours is GLAD [43] model. Since our algorithm builds upon GLAD's architecture we are going to describe it in detail in Section 3, while pointing out ways in which uGLAD differs from GLAD.

3 The uGLAD model

Given input data $X \in R^{M \times D}$, with M samples each having D features, the task is to recover a sparse graph showing correlations between the D features. Recovering the sparse graph (the adjacency matrix) corresponds to obtaining the precision matrix of the underlying multivariate Gaussian distribution.

3.1 Understanding the GLAD architecture

uGLAD uses the same architecture as GLAD [43] for the function $\Theta = f_{nn}(X)$. The GLAD model uses the Alternating Minimization algorithm updates for the maximum likelihood objective, unrolled to some iterations and uses it as a template for its architecture. Penalty constants (λ_1, λ_2) are replaced by problem dependent neural networks λ_{nn} and γ_{nn}. These neural networks are minimalist in terms of the number of parameters as the input dimensions are fixed to 2 for $f_{nn} \in \{\lambda_{nn}, \gamma_{nn}\}$ and outputs are a single value. This unrolled algorithm with neural network augmentation can be viewed as a highly structured recurrent architecture as illustrated in Fig. 1. Algorithm presented in Fig. 2 summarizes the update equations for the unrolled AM based model, GLAD.

The key features making the GLAD model a suitable choice are: (1) Minimalist learnable parameters (2) inherently maintain permutation invariance w.r.t. the input (3) AM algorithm enforces positive definite constraint throughtout optimization (4) Interpretability and (5) Linear convergence guarantee for faster runtime. For the sake of completeness of understanding uGLAD architecture, we graciously borrow the algorithm (see Alg.1) and neural network design details from [43] here.

3.2 The GLAD loss function

To learn the parameters in GLAD architecture, the authors used supervision in the form of true underlying graphs. They leveraged the interpretable nature of GLAD's deep architecture to define

the loss for training. Speci cally, each iteration of the model will output a valid precision matrix estimation and this allowed them to add auxiliary losses to regularize the intermediate results of GLAD, guiding it to learn parameters which can generate a smooth solution trajectory.

The authors used Frobenius norm in their loss function:

$$L_{GLAD} := \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma^{K-k} \left\| \Theta_k^{(i)} \right\|_F^2 ; \quad (3)$$

where $(\Theta_k^{(i)}, Z_k^{(i)}, \lambda_k^{(i)}) = \text{GLADcell}_\theta(S^{(i)}, \Theta_{k-1}^{(i)}, Z_{k-1}^{(i)}, \lambda_{k-1}^{(i)})$ is the output of the recurrent unit GLADcell at k-th iteration, K is number of unrolled iterations, $\gamma \leq 1$ is a discounting factor and Θ^* is the ground truth precision matrix. Then the stochastic gradient descent algorithm was used to train the parameters θ in the GLADcell.

3.3 The uGLAD loss function

The GLAD model was developed as a supervised learning model, where the supervision is provided in terms of pairs of input data X and the ground truth precision matrices. The parameters were trained by optimizing the mean square error between the predicted and the true precision matrices (see Sec. 3.2 above). In our model uGLAD (unsupervised GLAD), we extend the GLAD model to the setting where the ground truth is unavailable. Note that this is a much more realistic scenario – in most real-world domains the data generating distribution is not known.

In contrast to GLAD, uGLAD cannot take advantage of the underlying true generating distribution in the form of the true covariance matrix. Thus, we need to replace GLAD's MSE loss from Equation 3 with a new loss. Given a function $\Theta = f_{nn}(X)$, we optimize the log likelihood function given by

$$L_{uGLAD}(S; \Theta) = -\log|\det(\Theta)| + \langle S, \Theta \rangle \quad (4)$$

$$L_{uGLAD}(X) = -\log|\det(f_{nn}(X))| + \langle \text{cov}(X), f_{nn}(X) \rangle \quad (5)$$

where $S = \text{cov}(X)$ is the covariance matrix. We take the function f_{nn} as the GLAD model and substitute it in the uGLAD loss function.

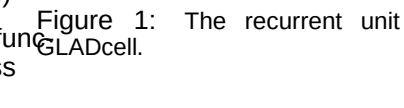

Figure 1: The recurrent unit GLADcell.

Algorithm 2 gives the pseudo code for learning the uGLAD model for doing sparse graph recovery.

3.4 Convergence properties of GLAD and uGLAD

The GLAD paper [43] evaluates convergence properties for the GLAD algorithm using normalized mean square error (NMSE) and probability of success (PS) to evaluate the algorithm performance. NMSE is $\log_{10}(E\|\Theta^p - \Theta^*\|_F^2 / E\|\Theta^*\|_F^2)$ and PS is the probability of correct signed edge-set recovery, i.e., $P(\text{sign}(\Theta^p_{ij}) = \text{sign}(\Theta^*_{ij}), \forall (i,j) \in E(\Theta^*))$, where $E(\Theta^*)$ is the true edge set. Optimization objective always converges. However, errors of recovering true precision matrices measured by NMSE have very different behaviors given different regularity parameter λ, which indicates the necessity of directly optimizing NMSE and hyperparameter tuning. NMSE values are very sensitive to both λ and the quadratic penalty μ of ADMM method. In GLAD and uGLAD, λ and μ are not fixed, but are optimized together with the rest of network parameters, leading to smooth convergence.

In experiments evaluating edge recovery success, GLAD consistently outperforms traditional methods in terms of sample complexity as it recovers the true edges with considerably fewer number of samples. Since in uGLAD we are still using the AM minimization based GLAD architecture which is also based on optimizing the Eq. 1, we expect the linear convergence properties of the AM algorithm will hold for uGLAD as well. The synthetic experiments in Sec. 6 show the results obtained from uGLAD are better than those for block coordinate descent based approach.

3.5 Obtaining Conditional Independence graphs

The CI graph shows the partial correlations between the input features. A non-zero partial correlation between 2 features (f_A, f_B) indicates a direct dependence. So, if all the other features are fixed, a positive partial correlation will indicate that increasing value of f_A will increase the value of f_B and vice-versa for the negative correlation. To obtain this CI graph, we calculate the partial correlation matrix from the precision matrix. Each entry of the partial correlation matrix P_{ij} shows the correlation of the features x_i, x_j given the values of the other features are observed. This helps us obtain the direct dependence of the features $P_{ij} = \frac{p_{ij}}{\sqrt{p_{ii} p_{jj}}}$. We can then visualize the graphs and study the positive and negative correlations, with edge weights corresponding to correlation strengths.

4 Multi-task learning for precision matrix recovery

Most of the work in learning Gaussian graphical models has focused on estimating a single model. In recent years, the framework was extended to jointly fitting a collection of such models, based on data that share the same variables, with dependency structure varying with some external category. For example, in an NLP application, we can encounter different styles, which induce links between some concepts, even as the underlying grammar and semantics of the language stay the same.

There have been extensive studies on the joint estimation of multiple undirected Gaussian graphical models [46, 20, 15, 6, 29, 9, 24, 28, 32, 50, 13, 48]. Most traditional algorithms construct a joint objective for multiple estimation tasks. This objective typically incorporates similarities among various tasks by adding group norms or other regularizations. However, in many practical problems, we only know that multiple tasks are related, without knowing how they are similar to each other quantitatively. Manually constructing the joint objective may not best reflect the actual similarity.

Figure 2: Minimalist neural network architectures designed for GLAD along with the optimizing algorithm.

Algorithm 2 Optimizing uGLAD

Input: Observations $X \in R^{M \times D}$
$S = \text{cov}(X)$
for $e = 1, \ldots, E$ **do**
  $\hat{\Theta}_e = \text{GLAD}(S)$ unrolled for L iterations
  Compute loss $L_{\text{uGLAD}}(S, \hat{\Theta}_e)$
  Backprop to update GLAD params
end for
return $\hat{\Theta}_E$

In contrast to traditional algorithms, in uGLAD, we do not need to assume a specific similarity among different tasks. Instead, we use a single network uGLAD to solve multiple tasks. Since the parameters in uGLAD are shared across different tasks, the similarity among the tasks is automatically learned from data. More specifically, given samples from K different models, $X_K = [X_1; X_2; \ldots; X_K]$, we optimize the following objective

$$L_{\text{uGLAD multitask}}(X_K) = \frac{1}{K} \sum_{k=1}^{K} L_{\text{uGLAD}}(\text{cov}(X_k); f_{nn}(X_k)) \tag{6}$$

Alternatively, we can use the cross-validation split $\{X_K^{\text{train}}; X_K^{\text{val}} \quad X_K\}$ for training.

5 Handling missing values

The missing data problem is ubiquitous in all data problems. uGLAD can easily be extended to handle this problem. If we observe that some specific feature columns have missing values and we have reasons to believe that these values are missing at random, we can run imputation algorithms for

AUPR			
Method	M=10	M=25	M=50
BCD	0.112 0.013	0.132 0.012	0.219 0.085
uGLAD	0.159 0.029	0.174 0.018	0.223 0.062

AUC			
Method	M=10	M=25	M=50
BCD	0.505 0.007	0.532 0.031	0.617 0.083
uGLAD	0.572 0.027	0.595 0.044	0.651 0.021

Table 1: Synthetic data: Gaussian. AUPR and AUC on 20 test graphs for the number of features $D = 25$ and varying number of samples M. Gaussian Random graphs with sparsity $= 0.1$ were chosen and edge values were sampled from $U(-1, 1)$. We can observe that uGLAD (CV mode) significantly outperforms the BCD algorithm (sklearn's graphicalLassoCV) for samples/features ratio « 1 and gives comparable performance as the number of samples increases.

those columns to predict the missing entries. Then, we will have the complete imputed input data $X_{imp} \in R^{M \times D}$ over which we can run the uGLAD model and obtain the underlying precision matrix.

It can often happen that there are technical errors or human mistakes in collecting samples, which can often lead to missing values or noise seeping into the sample. Also, we assume that the samples are independent and identically distributed (IID), so we cannot make use of the imputation techniques discussed in the case above. For this case, we propose a novel multi-task learning technique based on utilizing the uGLAD's ability to optimize over a batch of input samples.

Consensus strategy: Multi-task learning over row-subsampled input

The key idea is to create a batch of row subsamples of the input data $X \in R^{M \times D}$. Since all of these subsamples come from the same underlying distribution, we should ideally recover the same precision matrix for the entire batch. Thus, if we have a model that can be jointly optimized over the entire batch for the uGLAD objective, resulting in the recovered precision matrix being robust against erroneous or noisy samples.

Steps for the multi-task learning approach to train uGLAD model for a dataset with missing data:

1. Statistical imputation for the input: Replace all the missing entries of the input data X with their respective column mean (the mean is calculated ignoring the missing entries): $X[i, c] = \text{mean}(X[:, c])$. Replacing by mean is usually a preferred approach as its contribution zeros out $(X - \bar{X})$ while centering the data for the covariance matrix calculation.
2. Getting the batches: Perform stratified K-fold sampling to distribute the rows with missing values evenly among different batches. Say, we have $X_K = [X_1; X_2; \ldots; X_K]$ batches with each $X_k \in R^{M \times (\frac{K-1}{K}) \times D}$. Thus, the batch input for the uGLAD model is $X_K \in R^{K \times M \times (\frac{K-1}{K}) \times D}$.
3. Optimizing uGLAD: It becomes a multi-task learning setting as we are jointly optimizing over a batch input X_K. The uGLAD model takes in the batch input X_K and outputs the corresponding K precision matrices. Since, the entire batch of data is coming from the same underlying distribution, we use the entire data $X \in R^{M \times D}$ for the uGLAD loss to optimize the parameters of the uGLAD model. Mathematically, we are minimizing the uGLAD loss over the batch as

$$L_{uGLAD-meta}(X_K) = \frac{1}{K} \sum_{k=1}^{K} L_{uGLAD}(cov(X), f_{nn}(X_k)) \tag{7}$$

4. Consensus among the batch to obtain the final precision matrix: After optimizing the uGLAD for the batch input, we will obtain K different precision matrices $\Theta_K \in R^{K \times D \times D}$. Ideally, all the precision matrices should be the same but there will be some discrepancies as we are working with missing values. Our `consensus' strategy to obtain the final precision matrix is to find the common edges with their correlation type (positive or negative) from the batch precision matrices. Mathematically, we can obtain each entry $[i, j]$ of the final precision matrix as

$$\Theta^f_{i,j} = \text{max-count}_{k=1,\ldots,K}(\text{sign}(\Theta^k_{i,j})) \min_{k=1,\ldots,K} |\Theta^k_{i,j}| \tag{8}$$

Here, the `max-count' term determines whether the correlation among the batches for that entry is positive or negative. The 2nd term chooses the minimum absolute value for that entry among the batches as this facilitates sparsity and is conservative in terms of the strength of an edge.

AUPR			
Method	M=20	M=100	M=1000
BCD	0.163 0.028	0.241 0.014	0.523 0.011
uGLAD	0.206 0.035	0.272 0.024	0.569 0.048
AUC			
Method	M=20	M=100	M=1000
BCD	0.670 0.013	0.718 0.014	0.839 0.006
uGLAD	0.774 0.037	0.812 0.049	0.909 0.040

Table 2: GRN data: non-Gaussian. AUPR and AUC on 20 test graphs for $D = 100$ nodes and varying samples M. Graphs were sampled from the SERGIO simulator for the Gene Regulatory network recovery task. We can observe that the uGLAD model is more adaptive in non-Gaussian settings. A post-hoc masking operation was done to remove all the edges not containing a transcription factor. This was done for all the methods.

AUPR			
Method	M=10	M=25	M=50
BCD-avg	0.137 0.099	0.179 0.027	0.241 0.045
uGLAD-multi	0.186 0.028	0.204 0.044	0.279 0.027
AUC			
Method	M=10	M=25	M=50
BCD-avg	0.508 0.024	0.538 0.024	0.597 0.047
uGLAD-multi	0.552 0.048	0.573 0.047	0.626 0.022

Table 3: Multi-task learning. Average AUPR and AUC over $K = 10$ graphs coming from sparsity $[0.05, 0.2]$. The number of nodes $D = 25$ with varying samples $M = [10, 25, 50]$. The BCD-avg considers each instance of the batch as a separate task and reports the average results over the batch. uGLAD-multi is used to recover the graphs jointly using a single model.

AUPR			
Method	dp=0.25	dp=0.50	dp=0.75
BCD-mean	0.583 0.082	0.335 0.012	0.100 0.009
uGLAD-mean	0.605 0.103	0.357 0.034	0.113 0.016
uGLAD-missing	0.612 0.100	0.375 0.043	0.132 0.007
AUC			
Method	dp=0.25	dp=0.50	dp=0.75
BCD-mean	0.792 0.045	0.649 0.005	0.508 0.009
uGLAD-mean	0.806 0.019	0.691 0.025	0.527 0.011
uGLAD-missing	0.815 0.010	0.718 0.002	0.560 0.0.41

Table 4: Missing data: Gaussian. AUPR and AUC on 20 test graphs for $D = 25$ nodes and samples $M = 500$. Gaussian random graphs were generated as described in Sec. 6.1. Increasing fraction of dropouts were introduced to observe the robustness of handling missing data. We can observe that uGLAD-missing model is more robust, especially in high dropout settings.

AUPR			
Method	dp=0.50	dp=0.75	dp=0.90
BCD-mean	0.468 0.015	0.323 0.008	0.042 0.017
uGLAD-mean	0.503 0.011	0.346 0.021	0.090 0.069
uGLAD-missing	0.523 0.004	0.361 0.043	0.117 0.093
AUC			
Method	dp=0.50	dp=0.75	dp=0.90
BCD-mean	0.819 0.005	0.794 0.042	0.510 0.010
uGLAD-mean	0.897 0.009	0.821 0.019	0.598 0.079
uGLAD-missing	0.906 0.007	0.876 0.013	0.706 0.206

Table 5: Missing data: GRN. AUPR and AUC on 20 test graphs for $D = 100$ nodes and samples $M = 1000$. Gene regulatory network data was used as described in Sec. 6.2. Increasing fraction of dropouts were introduced to the observed samples of the microarray expression data. uGLAD-missing model is more robust for high dropout settings. Such high dropout ratios are quite common in collecting samples for microarray gene expression data.

6 Experiments

We believe that our model can be seamlessly integrated with existing pipelines for applications using other graphical lasso methods such as sklearn's GraphicalLassoCV [31]. We are hopeful that it improves results over state-of-the-art methods. Appendix C applies uGLAD to analyse real-world data collected for anaerobic digestion. We use AUC (area under the ROC curve) and AUPR (area under the precision-recall curve) as primary evaluation metrics. The sparsity of the graph leads to very few positive edges. These two metrics account for such imbalance in the data and have the advantage of working without specifically setting a threshold for non-zero entries. Their values reported in this work have the mean and the associated standard deviation values. We primarily compare against the BCD algorithm. It has been shown that the other traditional graphical lasso methods like ADMM and G-ISTA had performance similar to BCD [43]. We are not aware of any unsupervised deep learning approaches that optimize for the graphical lasso objective.

6.1 Performance on synthetically generated Gaussian samples

The synthetic data was generated based on the procedure similar to the one described in [19]. A d-dimensional precision matrix was generated by initializing a $d \times d$ matrix with its off-diagonal entries sampled i.i.d. from a uniform distribution, $\theta_{ij} \sim U(-1, 1)$. These entries were then set to zero based on the sparsity pattern of the corresponding Erdos-Renyi random graph with a certain probability p. Finally, an appropriate multiple of the identity matrix was added to the current matrix, so that the resulting matrix had the smallest eigenvalue of 1. In this way, Θ^* was ensured to be a well-conditioned, sparse and positive definite matrix and was used in the multivariate Gaussian distribution $N(0, \Theta^{*-1})$, to obtain M i.i.d. samples. Table 1 shows the results on this synthetic data.

6.2 Recovery of Gene Regulatory Networks

We conducted an exploratory study to gauge the generalization ability of uGLAD to non-Gaussian distributions. We chose the GRN inference task for this purpose. To this end, we use the SERGIO simulator [11] that provides a list of parameters to simulate cells from different types of biological processes and gene-expression levels with various amounts of intrinsic and technical noise. We provide more details on data generation in Appendix B. Table 2 shows the comparison of the different graph recovery methods on the simulated data generated by SERGIO for cell types $C = 5$ and clean data setting.

6.3 Multi-task learning

This experiment verifies the ability of uGLAD to do multi-task learning. We chose a collection of tasks as a set of data coming from graphs with varying sparsity. For K different tasks, our input data is $X \in R^{K \times M \times D}$. We run the uGLAD model in multi-task learning mode as described in Sec. 4 and recover K different precision matrices $\Theta \in R^{K \times D \times D}$ that are optimized for the loss function $L_{uGLAD}^{multitask}(X, K)$ given by equation 6. Table 3 shows results from a single uGLAD model in multi-task setting which is run on the synthetic data generated as described in Sec. 6.1. uGLAD recovers multiple graphs with varying sparsity and shows promise for multitask learning problems.

6.4 Robustness testing for missing data

We artificially introduced missing values or `dropouts' in the input data $X \in R^{M \times D}$ to create noisy data. Our aim is to study the effectiveness of the `consensus' strategy discussed in Sec. 5. We compare it with the baseline statistical imputation technique that does column-wise (or feature-wise) mean imputation as a preprocessing step. BCD-mean, uGLAD-mean, report the results of running the corresponding methods on the column mean imputed data while uGLAD-missing uses the `consensus' strategy. Tables 4 and 5 show the robustness of the `consensus' strategy introduced in this work for synthetic Gaussian data as well as for the Gene regulatory networks recovery tasks.

7 Software details: Optimizing modes of uGLAD

Our software's function signature is very much akin to the sklearn's GraphicalLassoCV [31]. This was intended to make it easier for the users to try out our method with a minimal change to their

existing code pipeline. To optimize uGLAD, we have introduced 4 different modes of training. In the software package, these modes can be switched from one to the other using an indicator flag.

Direct mode: The input to the uGLAD model is complete data X and the output precision matrix $\Theta = f_{nn}(X)$ is optimized to reduce the uGLAD loss $L_{uGLAD}(X)$, as defined in Eq. 4.

CV mode (recommended): In the k-fold cross validation mode, we split the input samples $X = (X_{train}; X_{valid})$. We use the X_{train} as input and optimize for the uGLAD loss $L_{uGLAD}(X_{train})$. Then, we select the best model that minimizes the $L_{uGLAD}(X_{valid})$ for the X_{valid} samples.

Missing data mode: We give the entire data X as input with a `NaN' indicator for the entries where the values are missing. The software then follows the `consensus' strategy for handling of missing data given in Sec. 5 and outputs the final precision matrix.

Multi-task mode: Given a batch of input data $X \in R^{K \times M \times D}$, we jointly optimize them for the uGLAD loss objective to obtain K different precision matrices $\Theta_K \in R^{K \times D \times D}$, refer to Sec. 4.

8 Potential other applications of the uGLAD model

Listing some applications for which uGLAD can potentially improve the current state-of-the-art:

- Protein structure recovery: PSICOV [22] uses graphical lasso based approach to predict the contact matrix, which then eventually gives the 3D protein structure. uGLAD can be substituted for predicting the contact matrix from the input correlation matrix between the amino acid sequences.
- Finance & Healthcare: Finding correlations between stocks to see how companies compare [16]. Systems for studying important body vitals of ICU patients [44, 5] and Infant Mortality analysis [40].
- Class imbalance handling: We can potentially use uGLAD to find correlation between the features. This correlation graph can be helpful in sampling down useful feature clusters. This will help identify key features and in-turn improve performance in cases where there is less data or imbalanced data (more data points for one class over another). Some of the methods for class imbalance handling on which uGLAD model can act as a preprocessing steps are [35, 41, 4].
- Gaussian processes & time series problems: uGLAD can be extended to this interesting work by [7] on combining graphical lasso with Gaussian processes for learning gene regulatory networks. Similarly, in a recent work on including negative data points for the Gaussian processes [38], uGLAD can be used for narrowing down the relevant features for doing the GP regression. Our model can be used for time-series modeling, refer to [23] for an example.
- Video sequence predictions: uGLAD can be learned to narrow down potential future viable frames for generating unseen future video frames [10, 37].

9 Conclusions

We introduce a novel technique uGLAD to perform sparse graph recovery by optimizing a deep unrolled network for the graphical lasso objective. This is an extension to the previous GLAD model that was designed to use supervision. The key advantages of using our model over the state-of-the-art algorithms for the graphical lasso problems are (1) Sparsity related hyperparameters are modeled using neural networks which are automatically learned during the optimization. We thus address the sensitivity issue of choosing the right sparsity parameters which is usually a tedious task and often manually set for the other algorithms. (2) By design, neural networks in uGLAD enable the sparsity regularization to be adaptive over the iterations of the optimization leading to superior performance. (3) Our software implementation supports GPU based acceleration and thus can be scaled efficiently to meet runtime requirements. (4) The uGLAD framework can do efficient multi-task learning. The proposed `consensus' strategy based on leveraging this property works well to robustly handle missing data. As our experience with anaerobic digestion (see Appendix C) demonstrates, uGLAD can be successfully used as a tool for generating insight into growth dynamics of organisms in a digester and (hopefully) into domain structure in many other applications. We hope that our model becomes one the widely used algorithms to solve the graphical lasso objective.

References

[1] Maneesha Aluru, Harsh Shrivastava, Sriram P Chockalingam, Shruti Shivakumar, and Srinivas Aluru. EnGRaiN: a supervised ensemble learning method for recovery of large-scale gene regulatory networks. Bioinformatics, 2021.

[2] Michelle Badri, Zachary D. Kurtz, Richard Bonneau, and Christian L. Müller. Shrinkage improves estimation of microbial associations under different normalization methods. bioRxiv, 2020. doi: 10.1101/406264. URL https://www.biorxiv.org/content/early/2020/04/04/406264.

[3] Eugene Belilovsky, Kyle Kastner, Gaël Varoquaux, and Matthew B Blaschko. Learning to discover sparse graphical models. In International Conference on Machine Learning, pages 440–448. PMLR, 2017.

[4] Sakyajit Bhattacharya, Vaibhav Rajan, and Harsh Shrivastava. ICU mortality prediction: a classification algorithm for imbalanced datasets. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[5] Sakyajit Bhattacharya, Vaibhav Rajan, and Harsh Shrivastava. Methods and systems for predicting mortality of a patient, November 5 2019. US Patent 10,463,312.

[6] Tony Cai, Weidong Liu, and Xi Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. Journal of the American Statistical Association, 106(494):594–607, 2011.

[7] H Chatrabgoun, AR Soltanian, H Mahjub, and F Bahreini. Learning gene regulatory networks using Gaussian process emulator and graphical lasso. Journal of Bioinformatics and Computational Biology, page 2150007, 2021.

[8] Xinshi Chen, Yu Li, Ramzan Umarov, Xin Gao, and Le Song. RNA secondary structure prediction by learning unrolled algorithms. arXiv preprint arXiv:2002.05810, 2020.

[9] Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society. Series B, Statistical methodology, 76(2):373, 2014.

[10] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In International Conference on Machine Learning, pages 1174–1183. PMLR, 2018.

[11] Payam Dibaeinia and Saurabh Sinha. SERGIO: a single-cell expression simulator guided by gene regulatory networks. Cell Systems, 11(3):252–271, 2020.

[12] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 9(3):432–441, 2008.

[13] André R Gonçalves, Fernando J Von Zuben, and Arindam Banerjee. Multi-task sparse structure learning with Gaussian copula models. The Journal of Machine Learning Research, 17(1): 1205–1234, 2016.

[14] Dominique Guillot, Bala Rajaratnam, Benjamin T Rolfs, Arian Maleki, and Ian Wong. Iterative thresholding algorithm for sparse inverse covariance estimation. arXiv preprint arXiv:1211.2532, 2012.

[15] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Joint estimation of multiple graphical models. Biometrika, 98(1):1–15, 2011.

[16] David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network inference via the time-varying graphical lasso. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 205–213, 2017.

[17] David Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, March 1995. URL https://www.microsoft.com/en-us/research/publication/a-tutorial-on-learning-with-bayesian-networks/ .

[18] David Heckerman and Bharat N. Nathwani. Toward normative expert systems part II. Probability-based representations for efficient knowledge acquisition and inference. *Methods of Information in Medicine*, 31:106–116, August 1992. URL https://www.microsoft.com/en-us/research/publication/toward-normative-expert-systems-part-ii/.

[19] David Heckerman, Eric Horvitz, and Bharat N. Nathwani. Toward normative expert systems part I. The Pathfinder project. *Methods of Information in Medicine*, 31:90–105, June 1992. URL https://www.microsoft.com/en-us/research/publication/toward-normative-expert-systems-part/.

[20] Jean Honorio and Dimitris Samaras. Multi-task learning of Gaussian graphical models. In *ICML*, 2010.

[21] Chenjing Jiang, Miriam Peces, Martin Hjorth Andersen, Sergey Kucheryavskiy, Marta Nierychlo, Erika Yashiro, Kasper Skytte Andersen, Rasmus Hansen Kirkegaard, Liping Hao, Jan Høgh, Aviaja Anna Hansen, Morten Simonsen Dueholm, and Per Halkjær Nielsen. Characterizing the growing microorganisms at species level in 46 anaerobic digesters at Danish wastewater treatment plants: A six-year survey on microbial community structure and key drivers. *Water Research*, 193:116871, 2021. ISSN 0043-1354. doi: https://doi.org/10.1016/j.watres.2021.116871. URL https://www.sciencedirect.com/science/article/pii/S0043135421000695.

[22] David T Jones, Daniel WA Buchan, Domenico Cozzetto, and Massimiliano Pontil. PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[23] Alexander Jung, Gabor Hannak, and Norbert Goertz. Graphical lasso based model selection for time series. *IEEE Signal Processing Letters*, 22(10):1781–1785, 2015.

[24] Mladen Kolar, Le Song, Amr Ahmed, Eric P Xing, et al. Estimating time-varying networks. *Annals of Applied Statistics*, 4(1):94–123, 2010.

[25] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. 2009.

[26] Jialin Liu and Xiaohan Chen. ALISTA: Analytic weights are as good as learned weights in LISTA. In *International Conference on Learning Representations (ICLR)*, 2019.

[27] Thomas Moerman, Sara Aibar Santos, Carmen Bravo González-Blas, Jaak Simm, Yves Moreau, Jan Aerts, and Stein Aerts. GRNBoost2 and Arboreto: efficient and scalable inference of gene regulatory networks. *Bioinformatics*, 35(12):2159–2161, 2019.

[28] Karthik Mohan, Palma London, Maryam Fazel, Daniela Witten, and Su-In Lee. Node-based learning of multiple Gaussian graphical models. *The Journal of Machine Learning Research*, 15(1):445–488, 2014.

[29] Diane Oyen and Terran Lane. Leveraging domain knowledge in multitask Bayesian network structure learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[30] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] Christine Peterson, Francesco C Stingo, and Marina Vannucci. Bayesian inference of multiple Gaussian graphical models. *Journal of the American Statistical Association*, 110(509):159–174, 2015.

[33] Aditya Pratapa, Amogh P Jalihal, Jeffrey N Law, Aditya Bharadwaj, and TM Murali. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature methods*, 17(2):147–154, 2020.

[34] Xingyue Pu, Tianyue Cao, Xiaoyun Zhang, Xiaowen Dong, and Siheng Chen. Learning to learn graph topologies. *Advances in Neural Information Processing Systems*, 34, 2021.

[35] M Mostafizur Rahman and Darryl N Davis. Addressing the class imbalance problem in medical datasets. *International Journal of Machine Learning and Computing*, 3(2):224, 2013.

[36] Pradeep Ravikumar, Martin J Wainwright, Garvesh Raskutti, and Bin Yu. High-dimensional covariance estimation by minimizing l_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

[37] Gaurav Shrivastava and Abhinav Shrivastava. Diverse video generation using a Gaussian process trigger. *arXiv preprint arXiv:2107.04619*, 2021.

[38] Gaurav Shrivastava, Harsh Shrivastava, and Abhinav Shrivastava. Learning what not to model: Gaussian process regression with negative constraints. 2020.

[39] Harsh Shrivastava. *On Using Inductive Biases for Designing Deep Learning Architectures*. PhD thesis, Georgia Institute of Technology, 2020.

[40] Harsh Shrivastava and Urszula Chajewska. Neural Graphical Models. *arXiv preprint arXiv:2210.00453*, 2022.

[41] Harsh Shrivastava, Vijay Huddar, Sakyajit Bhattacharya, and Vaibhav Rajan. Classification with imbalance: A similarity-based method for predicting respiratory failure. In *2015 IEEE international conference on bioinformatics and biomedicine (BIBM)*, pages 707–714. IEEE, 2015.

[42] Harsh Shrivastava, Eugene Bart, Bob Price, Hanjun Dai, Bo Dai, and Srinivas Aluru. Cooperative neural networks (CoNN): Exploiting prior independence structure for improved classification. *arXiv preprint arXiv:1906.00291*, 2019.

[43] Harsh Shrivastava, Xinshi Chen, Binghong Chen, Guanghui Lan, Srinivas Aluru, Han Liu, and Le Song. GLAD: Learning sparse graph recovery. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BkxpMTEtPB`.

[44] Harsh Shrivastava, Vijay Huddar, Sakyajit Bhattacharya, and Vaibhav Rajan. System and method for predicting health condition of a patient, August 10 2021. US Patent 11,087,879.

[45] Harsh Shrivastava, Xiuwei Zhang, Le Song, and Srinivas Aluru. GRNUlar: A deep learning framework for recovering single-cell gene regulatory networks. *Journal of Computational Biology*, 29(1):27–44, 2022.

[46] Le Song, Mladen Kolar, and Eric P Xing. KELLER: estimating time-varying interactions between genes. *Bioinformatics*, 25(12):i128–i136, 2009.

[47] Qiang Sun, Kean Ming Tan, Han Liu, and Tong Zhang. Graphical nonconvex optimization via an adaptive convex relaxation. In *International Conference on Machine Learning*, pages 4810–4817. PMLR, 2018.

[48] Burak Varici, Saurabh Sihag, and Ali Tajer. Learning shared subgraphs in Ising model pairs. In *International Conference on Artificial Intelligence and Statistics*, pages 3952–3960. PMLR, 2021.

[49] Daniela M Witten, Jerome H Friedman, and Noah Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.

[50] Sen Yang, Zhaosong Lu, Xiaotong Shen, Peter Wonka, and Jieping Ye. Fused multiple graphical lasso. *SIAM Journal on Optimization*, 25(2):916–943, 2015.

[51] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

[52] Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. DAGs with NO TEARS: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems*, 31:9472–9483, 2018.

Appendix

A Analysis of the uGLAD architecture

A.1 uGLAD parameter details

We graciously borrow the algorithm (see Alg. 1) and neural network design details from the 'GLAD : Learning Sparse graph recovery' paper. GLAD parameter settings details are: ρ_{nn} was a 4 layer neural network and Λ_{nn} was a 2 layer neural network. Both used 3 hidden units in each layer. The non-linearity used for hidden layers was $\tanh$, while the final layer had sigmoid (σ) as the non-linearity for both, ρ_{nn} and Λ_{nn} (refer to Figure 2). The learnable offset parameter of initial Θ_0 was set to $t = 1$. It was unrolled for $L = 30$ iterations.The optimizer used was adam with the learning rates were chosen between $[0.001, 0.005]$.

B Gene regulatory network recovery: additional experiment details

Additional details on the experiment done in Sec. 6.2. For evaluation purposes in this work, we created random graphs (GRNs) that were used as input to SERGIO and will act as ground truth for evaluation. First, we set the number of Transcription Factors (TFs) or master regulators. Then, we randomly added edges between the TFs and the other genes based on sparsity requirements. Also, we randomly added some edges between the TFs themselves but excluded any self-regulation edges and maintained connectivity of the graph.

When simulating data with no technical noise (clean data), we set the following parameters: sampling-state $= 15$ (determines the number of steps of simulations for each steady state); noise-param $\sim U[0.1, 0.3]$ (controls the amount of intrinsic noise); noise-type = 'dpd' (the type of intrinsic noise is dual production decay noise, which is the most complex out of all types provided); we set genes decay parameter to 1. The parameters required to decide the master regulators' basal production cell rate for all cell types: low expression range of production cell rate $\sim U[0.2, 0.5]$ and high expression range of cell rate $\sim U[0.7, 1]$. We chose K$\sim U[1, 5]$, where K denotes the maximum interaction strength between master regulators and target genes. Positive strength values indicate activating interactions and negative strength values indicate repressive interactions and signs are randomly assigned.

C Case study: Anaerobic Digestion

Our algorithm development was inspired by a practical problem of domain exploration in anaerobic digestion. Anaerobic digestion is a growing field addressing waste management with both environmental benefits (reduced odor and pathogens, improved soil health, reduction in methane emissions) and economic value from use of captured methane gas. Despite numerous studies, the dynamics of organisms' growth in digesters, their dependence on conditions (temperature, pH, feedstock mix, nitrogen to carbon ratio, etc.) and their impact on methane yield are not well understood.

We present findings based on a public dataset from a study of anaerobic digesters at Danish wastewater plants [21]. Data is available at NCBI under bioproject accession number PRJNA637463.

Data comes from a 6-year study of 46 digesters located at 22 Danish treatment plants. We have three types of digesters, operating at different temperatures (mesophilic, mesophilic with thermal hydrolysis pretreatment, thermophilic). Digesters operate continuously with sludge retention rate of 15.8 to 35.6 days. Samples were taken at 3-month and 6-month intervals, so they can be treated as i.i.d. We have a total of 1,010 sludge samples, 418 used to sequence archaea and 592 bacteria, performed using 16S rRNA gene amplicon sequencing. Analysis resulted in identification of 33,047 bacterial and 878 archaeal unique amplicon sequence variants (ASVs). 70% of genera and 93% of the species were determined to be novel or previously unclassified. This presents problems for all approaches attempting to utilize existing databases to determine organisms' function for the purpose of grouping and feature selection. In fact, one of the ways to determine an organism's function is based on checking properties of organisms whose abundance numbers in the digester best correlate with the given organism's numbers.

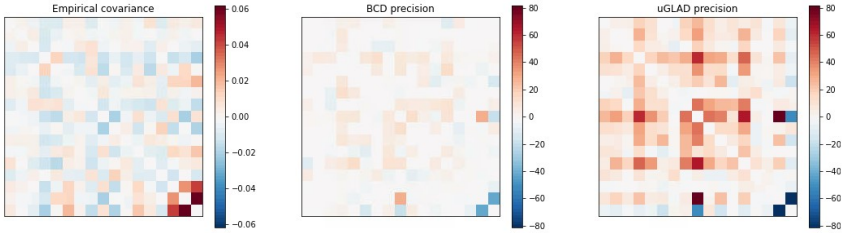

Figure 3: uGLAD recovered precision matrix compared to empirical covariance and precision matrix recovered by the BCD algorithm for archaea at **family** level

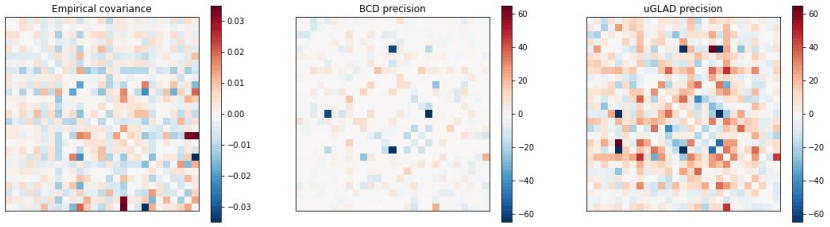

Figure 4: uGLAD recovered precision matrix compared to empirical covariance and precision matrix recovered by the BCD algorithm for archaea at **species** level

Our algorithm works with any input, including: ASVs filtered by frequency, ASVs rolled up to higher taxonomy levels (species, genus, family), ASVs abundance normalized in various ways [2]. We calculate the partial correlation matrix from the precision matrix. Each entry of the partial correlation matrix P_{ij} shows the correlation of the features x_i, x_j given the values of the other features are observed. This helps us obtain the direct dependence of the features.

$$\mathrm{P}_{ij} = -\rho \frac{\Theta_{ij}}{\Theta_{ii}\Theta_{jj}} \tag{9}$$

We use networkx package to visualize the graphs, presenting positive correlations in green and negative in red, with edge weights corresponding to the strength of the correlation.

Figures 3 and 4 present precision matrices recovered by our algorithm and BCD with empirical covariance shown for comparison. Figures 5 and 6 show corresponding graphs for archaea at family and species level.

Figures 7 and 8 show a result of multitask learning based on digester type: thermophilic (operated at temperature 53.6°C) and mesophilic (operating at temperature 38°C). The two graphs' edges are filtered to show only edges common to both graphs, which is a small fraction of all edges. Note that in some cases, the sign of the correlation (and the color of the edge) changes depending on digester's type.

Our model is being used by domain experts to gain insight into the domain of anaerobic digestion. One use case is to understand properties of newly discovered bacteria and archaea by analyzing which known organisms their abundance in digester samples correlates with (positively or negatively). That can lead to focusing attention on a smaller organism set. Another use case centers around understanding the role of digester conditions and feedstock mix on organisms' growth and methane yield. The results presented in recovered graphs lead to new hypotheses and new experiments being designed to test them.

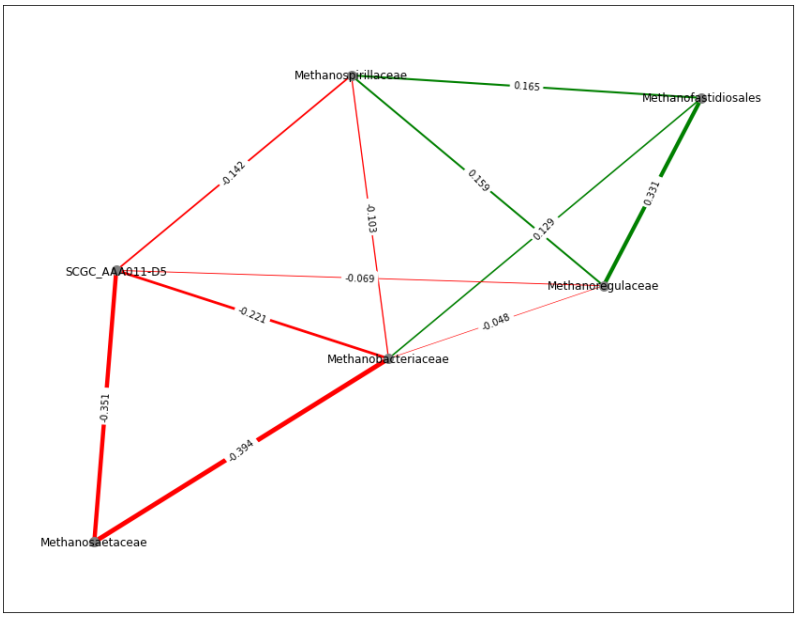

Figure 5: uGLAD graph for archaea at family level. Edge color indicates the sign of the correlation: green - positive, red - negative, edge weight corresponds to correlation's strength.

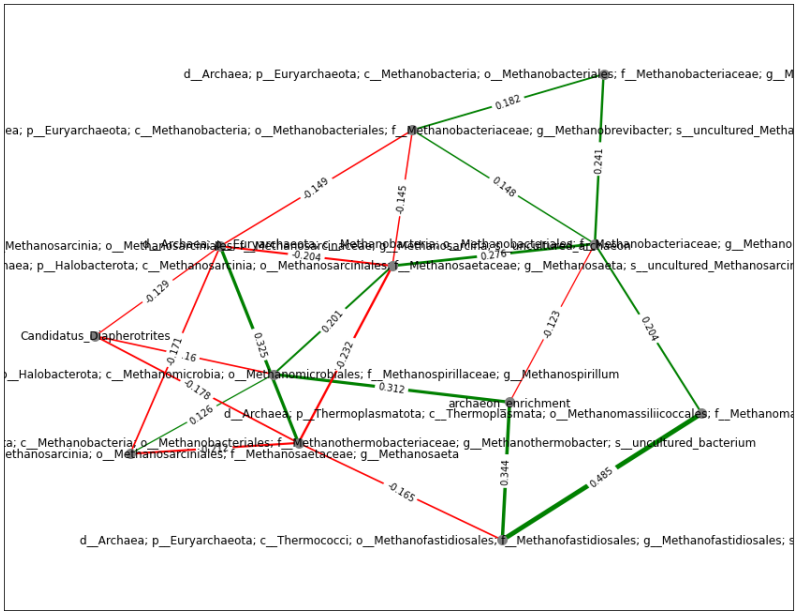

Figure 6: uGLAD graph for archaea at species level. Edge color indicates the sign of the correlation: green - positive, red - negative, edge weight corresponds to correlation's strength.