GLAD: Learning Sparse Graph Recovery

Harsh Shrivastava

Joint work with Xinshi Chen, Binhong Chen, Guanghui Lan, Srinvas Aluru, Han Liu, Le Song
Objective

Recovering sparse conditional independence graph $G$ from data

\[ \Theta_{ij} = 0 \iff X_i \perp X_j \mid \text{other variables} \]
Applications

**Biology**  
Gene Expression data - Microarray experiments

Algorithm

Gene regulatory network

**Finance**  
Time-series features

Algorithm

Relationship between assets
Sparse Graph Recovery Problem Formulation

- Given $M$ samples from a distribution: $X \in \mathbb{R}^{M \times D}$
- Estimate matrix ‘$\Theta$’ corresponding to the sparse graph

Objective function: L1 regularized maximum likelihood estimation

$$
\hat{\Theta} = \arg \min_{\Theta \in S_+^d} - \log(\det \Theta) + \text{tr}(\hat{\Sigma} \Theta) + \rho \| \Theta \|_{1, \text{off}}
$$

- Covariance matrix
  $$\hat{\Sigma} = \frac{X^T X}{M}$$
- Regularization Parameter

- $\rho$
Existing Optimization Algorithms

- **G-ISTA**: Proximal gradient method
- **BCD**: Block coordinate descent method
- **ADMM**: Alternating direction method of multipliers

\[-\log(\det \Theta) + \text{tr}(\hat{\Sigma}\Theta) + \rho \|Z\|_1 + \langle \lambda, \Theta - Z \rangle + \frac{1}{2} \beta \|Z - \Theta\|_F^2.\]

Taking $U := \lambda / \beta$ as the scaled dual variable, the update rules for the ADMM algorithm are

\[
\Theta_{k+1} \leftarrow \left( -Y + \sqrt{Y^TY + (4/\beta)I} \right) / 2, \text{ where } Y = \hat{\Sigma} / \beta - Z_k + U_k
\]

\[
Z_{k+1} \leftarrow \eta \rho / \beta (\Theta_{k+1} + U_k), \quad U_{k+1} \leftarrow U_k + \Theta_{k+1} - Z_{k+1}
\]
Hard to Tune Hyperparameters

‘Grid search’ is tedious and non-trivial

Tuning hyperparameters for Traditional Methods

Outcomes highly sensitive to penalty parameters

Errors of different parameter combinations

\[- \log(\det \Theta) + \text{tr}(\hat{\Sigma}\Theta) + \rho \|Z\|_1 + \langle \lambda, \Theta - Z \rangle + \frac{1}{2}\beta\|Z - \Theta\|_F^2\]
Mismatch in Objectives

\[- \log(\det \Theta) + \text{tr}(\hat{\Sigma} \Theta) + \rho \| \Theta \|_{1,\text{off}}\]

Log-determinant estimator

Mismatch!

Recovery Objective (NMSE)

\[ \| \hat{\Theta} - \Theta^* \|_F^2 \| \Theta^* \|_F^2 \]
Limitations of Existing Optimization Algorithms

Consistency of estimator $\hat{\Theta}$

Based on ‘carefully chosen conditions’ like
1. Lower bound on sample size
2. Sparsity of $\Theta$
3. Degree of graph
4. Magnitude of covariance entries

Limitations of the **convex** formulation

Room for Improvement!

Pradeep Ravikumar, Martin J Wainwright, Garvesh Raskutti, Bin Yu, et al. High-dimensional covariance estimation by minimizing l1-penalized log-determinant divergence. Electronic Journal of Statistics, 5:935–980, 2011.
Big Picture Question

- Given a collection of ground truth precision matrix $\Theta^*$, and the corresponding empirical covariance $\hat{\Sigma}$

- Learn an algorithm $f$ which directly produces an estimate of the precision matrix $\Theta$?

$$\min_f \frac{1}{|\mathcal{D}|} \sum_{(\hat{\Sigma}_i, \Theta^*_i) \in \mathcal{D}} \|\Theta_i - \Theta^*_i\|_F^2, \quad s.t. \quad \Theta_i = f(\hat{\Sigma}_i)$$
Deep Learning Model Example

DeepGraph (DG) architecture. The input is first standardized and then the sample covariance matrix is estimated. A neural network consisting of multiple dilated convolutions (Yu & Koltun, 2015) and a final $1 \times 1$ convolution layer is used to predict edges corresponding to non-zero entries in the precision matrix.

* DeepGraph-39 model from Fig.2 of “Learning to Discover Sparse Graphical Models” by Belilovsky et. al.
Challenges in Designing Learning Models

- Interpretability
- SPD constraint
- Permutation Invariance
- #parameters scale $\text{dim}^2$

Traditional Approaches
- DNNs
- CNNs
- Autoencoders
- VAEs
- RNNs
GLAD: DL model based on Unrolled Algorithm

Alternating Minimization (AM) algorithm: Objective function

\[ \hat{\Theta}_\lambda, \hat{Z}_\lambda := \arg \min_{\Theta, Z \in S_{++}^d} - \log(\det \Theta) + \text{tr}(\hat{\Sigma} \Theta) + \rho \| Z \|_1 + \frac{1}{2} \lambda \| Z - \Theta \|_F^2 \]

AM: Update Equations (Nice closed form updates!)

\[ \Theta_{k+1}^{\text{AM}} \leftarrow \frac{1}{2} \left( - Y + \sqrt{Y^T Y + \frac{4}{\lambda} I} \right), \text{ where } Y = \frac{1}{\lambda} \hat{\Sigma} - Z_k^{\text{AM}} \]

\[ Z_{k+1}^{\text{AM}} \leftarrow \eta_{\rho/\lambda}(\Theta_{k+1}^{\text{AM}}), \text{ where } \eta_{\rho/\lambda}(\theta) := \text{sign}(\theta) \max(|\theta| - \rho/\lambda, 0) \]

**Modifications**

- Unroll to fixed #iterations ‘K’
- Treat it as a deep model
GLAD: Training

**Loss function:** Frobenius norm with discounted cumulative reward

\[
\min_f \text{ loss}_f := \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma^{K-k} \left\| \Theta_k^{(i)} - \Theta^{*(i)} \right\|_F^2
\]

**Optimizer for training:** ‘Adam’.
Learning rate chosen between [0.01, 0.1] in conjunction with Multi-step LR scheduler.

**Gradient Computation through matrix square root in the GLADcell:**

For any SPD matrix $X$: $X = X^{1/2}X^{1/2}$

Solve Sylvester’s equation for $d(X^{1/2})$:

\[
dX = d(X^{1/2})X^{1/2} + X^{1/2}d(X^{1/2})
\]
Use Neural Networks for $(\rho, \lambda)$

\[ \lambda \leftarrow \Lambda_{nn}(\|Z - \Theta\|_F^2, \lambda) \]

\[ \rho_{ij} = \rho_{nn}(\Theta_{ij}, \hat{\Sigma}_{ij}, Z_{ij}) \]

- **# of layers**: 2
  - *Hidden unit size* = 3

- **Minimalist designing of Neural Networks**

- **# of layers**: 4
  - *Hidden unit size* = 3

- **Non-Linearity**:  
  - Hidden layers: ‘tanh’
  - Final layer: ‘sigmoid’
GLAD

Algorithm 1: GLAD

Function GLADcell(\(\hat{\Sigma}, \Theta, Z, \lambda\)):

\[
\begin{align*}
\lambda &\leftarrow \Lambda_{nn}(\|Z - \Theta\|_F^2, \lambda) \\
Y &\leftarrow \lambda^{-1} \hat{\Sigma} - Z \\
\Theta &\leftarrow \frac{1}{2} \left( -Y + \sqrt{Y^\top Y + \frac{4}{\lambda} I} \right)
\end{align*}
\]

For all \(i, j\) do

\[
\begin{align*}
\rho_{ij} &= \rho_{nn}(\Theta_{ij}, \hat{\Sigma}_{ij}, Z_{ij}) \\
Z_{ij} &\leftarrow \eta_{\rho_{ij}}(\Theta_{ij})
\end{align*}
\]

return \(\Theta, Z, \lambda\)

Function GLAD(\(\hat{\Sigma}\)):

\[
\begin{align*}
\Theta_0 &\leftarrow (\hat{\Sigma} + tI)^{-1}, \lambda_0 \leftarrow 1 \\
\text{For } k = 0 \text{ to } K - 1 \text{ do} \\
\Theta_{k+1}, Z_{k+1}, \lambda_{k+1} &\leftarrow \text{GLADcell}(\hat{\Sigma}, \Theta_k, Z_k, \lambda_k)
\end{align*}
\]

return \(\Theta_K, Z_K\)

GLAD

Using algorithm structure as inductive bias for designing unrolled DL architectures
Desiderata for GLAD

- Minimalist Model
- Interpretable
- SPD constraint
- Permutation Invariance

GLAD: Graph recovery Learning Algorithm using Data-driven training

Algorithm 1: GLAD

Function GLADcell(\(\hat{\Sigma}, \Theta, Z, \lambda\)):

\[
\begin{align*}
\lambda &\leftarrow \Lambda_{nn}(\|Z - \Theta\|_F^2, \lambda) \\
Y &\leftarrow \lambda^{-1}\hat{\Sigma} - Z \\
\Theta &\leftarrow \frac{1}{2}(-Y + \sqrt{Y^TY + \frac{4}{\lambda}I})
\end{align*}
\]

For all \(i, j\) do

\[
\begin{align*}
\rho_{ij} &= \rho_{nn}(\Theta_{ij}, \hat{\Sigma}_{ij}, Z_{ij}) \\
Z_{ij} &\leftarrow \eta_{\rho_{ij}}(\Theta_{ij})
\end{align*}
\]

return \(\Theta, Z, \lambda\)

Function GLAD(\(\hat{\Sigma}\)):

\[
\begin{align*}
\Theta_0 &\leftarrow (\hat{\Sigma} + tI)^{-1}, \lambda_0 &\leftarrow 1 \\
\text{For } k = 0 \text{ to } K - 1 \text{ do} &\leftarrow \text{GLADcell}(\hat{\Sigma}, \Theta_k, Z_k, \lambda_k) \\
\text{return } \Theta_K, Z_K
\end{align*}
\]
Experiments: Convergence

Train/finetuning using 10 random graphs
Test on 100 random graphs

Fixed Sparsity level s=0.1

Mixed Sparsity level s \sim U(0.05, 0.15)

GLAD vs traditional methods
Experiments: Recovery probability

Sample complexity for model selection consistency

PS is non-zero if all graph edges are recovered with correct signs

GLAD able to recover true edges with considerably fewer samples
Experiments: Data Efficiency

| Methods      | M=15        | M=35        | M=100       |
|--------------|-------------|-------------|-------------|
| BCD          | 0.578±0.006 | 0.639±0.007 | 0.704±0.006 |
| DeepGraph-39 | 0.664±0.008 | 0.738±0.006 | 0.759±0.006 |
| DG-39+P      | 0.672±0.008 | 0.740±0.007 | 0.771±0.006 |
| GLAD         | 0.788±0.003 | 0.811±0.003 | 0.878±0.003 |

AUC on 100 test graphs with dimension=39, Gaussian random graph sparsity=0.05 and edge values sampled from ~U(-1, 1).

*DeepGraph-39 model from “Learning to Discover Sparse Graphical Models” by Belilovsky et. al.*

`Table 1. of Belilovsky et. al.`
Gene Regulation Data: SynTReN details

Synthetic gene expression data generator creating biologically plausible networks

Models biological & correlation noises

The topological characteristics of generated networks closely resemble transcriptional networks

Contains instances of Ecoli bacteria and other true interaction networks
Gene Regulation Data: Ecoli network predictions

GLAD trained on Erdos-Renyi graphs of dimension=25.

# of train/valid graphs were 20/20.

1 batch of M samples were taken per graph

(a) True graph  (b) M=10, fdr=0.613, tpr=0.913, fpr=0.114  (c) M=100, fdr=0.236, tpr=0.986, fpr=0.024

Recovered graph structures for a sub-network of the E. coli consisting of 43 genes and 30 interactions with increasing samples. All noises sampled ~U(0.01, 0.1). Increasing the samples reduces the fdr by discovering more true edges.
Theoretical Analysis: Assumptions

Assumption 1. Let the set $S = \{(i, j) : \Theta^*_{ij} \neq 0, i \neq j\}$. Then $\text{card}(S) \leq s$.

Assumption 2. $\Lambda_{\min}(\Sigma^*) \geq \epsilon_1 > 0$ (or equivalently $\Lambda_{\max}(\Theta^*) \leq 1/\epsilon_1$), $\Lambda_{\max}(\Sigma^*) \leq \epsilon_2$ and an upper bound on $\|\hat{\Sigma}\|_2 \leq c_\Sigma$.

Assumption 1 just upper bounds sparsity.

Assumption 2 guarantees that $\Theta^*$ exists.
Theoretical Analysis: Linear Convergence of AM

Recalling AM Update Equations

\[ \Theta_{k+1}^{AM} \leftarrow \frac{1}{2} \left( \Theta_{k+1}^{AM} - \frac{1}{\lambda} \sum Z_{k}^{AM} \right) \]

\[ Z_{k+1}^{AM} \leftarrow \eta_{\rho/\lambda}(\Theta_{k+1}^{AM}), \quad \text{where } \eta_{\rho/\lambda}(\theta) := \text{sign}(\theta) \max(\|\theta\| - \rho/\lambda, 0) \]

An adaptive sequence of penalty parameters should achieve a better error bound

Summary

Optimal parameter values depend on the prediction error. Hard to choose manually

AM has linear convergence rate. Can run for fixed iterations with reasonable error margins.
Conclusion

Unrolled DL architecture, GLAD, for sparse graph recovery

Empirical evidence that learning can improve graph recovery

Empirically, GLAD is able to reduce sample complexity

Highlighting the potential of using algorithms as inductive bias for DL architectures
