

# **Directed Acyclic Graph Structure Estimation using Sparse Bayesian Learning**

---

**Tanvi Nayak**

Mathematics and Computing  
Indian Institute of Technology Dharwad

**Project Supervisor:** Prof. Sundeep Prabhakar Chepuri (Dept. of EECE,  
IISc)

**BTP Mentor:** Prof. B. N. Bharath (Dept. of EECE, IIT Dharwad)

# Motivation

- Many real systems can be described using causal relations between variables.
- DAGs provide a clear and interpretable way to represent these causal dependencies.
- In domains such as biology, neuroscience, economics, and climate science, the true causal structure is typically sparse.
- Recovering these sparse graphs from data is important for understanding the system and supporting meaningful decisions or interventions.

## Problem Statement

- Recover the weighted adjacency matrix of a Directed Acyclic Graph (DAG) from observed data under a Gaussian SEM model.
- Enforce the acyclicity constraint while estimating dependencies among variables.
- Achieve a sparse and interpretable graph structure that suppresses spurious edges and matches real-world network sparsity.
- Integrate Sparse Bayesian Learning (SBL) to adaptively drive unnecessary edges to zero and improve structural recovery.
- Combine SBL-driven sparsity with the NOTEARS acyclicity formulation into a single optimization framework.

## Related Work

Two key works form the foundation for our method:

- **NOTEARS:** X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, “*DAGs with NO TEARS: Continuous Optimization for Structure Learning*”, 2018.
- **Sparse Bayesian Learning (SBL):** M. E. Tipping, “*Sparse Bayesian Learning and the Relevance Vector Machine*”, 2001.

# What is a DAG?

- A directed acyclic graph (DAG) is a directed graph with no directed cycles.
- It can be represented by a matrix whose nonzero entries indicate directed edges.
- In the DAG learning problem, we observe a data matrix  $X \in \mathbb{R}^{n \times d}$  containing  $n$  i.i.d. samples of  $X = (X_1, \dots, X_d)$ .
- Let  $D$  be the discrete space of all DAGs  $G = (V, E)$  on  $d$  nodes.
- The task is to find a DAG  $G \in D$  that represents the joint distribution  $P(X)$ .

## Example: A Small DAG

- Consider a graph with three nodes:  $X_1$ ,  $X_2$ , and  $X_3$ .
- Suppose the directed edges are:

$$X_1 \rightarrow X_2, \quad X_2 \rightarrow X_3.$$

- This forms a simple chain with no cycles.

**Weighted Adjacency Matrix:**

$$W = \begin{pmatrix} 0 & 1.5 & 0 \\ 0 & 0 & -0.8 \\ 0 & 0 & 0 \end{pmatrix}$$

# Structural Equation Models (SEMs)

- We represent a directed graph using a weighted matrix  $W = [w_1 \mid w_2 \mid \cdots \mid w_d] \in \mathbb{R}^{d \times d}$ .
- Each column  $w_j$  contains the weights of all incoming edges to node  $X_j$ .
- Let  $X = (X_1, \dots, X_d)$  be the random vector of variables.
- A linear SEM defines each variable as:

$$X_j = w_j^\top X + z_j,$$

where  $z_j$  is an independent noise term.

- In matrix form, stacking all variables and noise terms:

$$X = XW + Z,$$

where  $Z = (z_1, \dots, z_d)$ .

## Example: SEM for a Simple DAG

### DAG Structure

$$X_1 \rightarrow X_2, \quad X_1 \rightarrow X_3$$

### Corresponding SEM Equations

$$X_1 = z_1,$$

$$X_2 = 1.5 X_1 + z_2,$$

$$X_3 = -0.8 X_1 + z_3.$$

### Weighted Adjacency Matrix

$$W = \begin{pmatrix} 0 & 1.5 & -0.8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

### General Form

$$X_j = w_j^\top X + z_j$$

### Interpretation

- 1.5 : edge  $X_1 \rightarrow X_2$
- -0.8 : edge  $X_1 \rightarrow X_3$
- Column  $w_j$  lists parents of  $X_j$

- Each variable is a linear function of its parents.
- $z_j$  represents independent noise.

## NOTEARS: Acyclicity Constraint

**Key Idea (Zheng et al., 2018):** A DAG can be characterized using a smooth, differentiable function.

- For a weighted adjacency matrix  $W \in \mathbb{R}^{d \times d}$ , define

$$h(W) = \text{tr} \left( e^{W \circ W} \right) - d.$$

- The gradient is simple and fully differentiable:

$$\nabla h(W) = \left( e^{W \circ W} \right)^T \circ (2W).$$

Theorem:

W represents a DAG if and only if  $h(W) = 0$ .

## NOTEARS: Acyclicity Constraint

- The Hadamard product  $W \circ W$  removes signs and keeps only squared edge weights.
- The matrix exponential has the Taylor expansion:

$$e^{W \circ W} = I + (W \circ W) + \frac{1}{2!}(W \circ W)^2 + \frac{1}{3!}(W \circ W)^3 + \dots$$

- Powers of an adjacency matrix count directed walks of length  $k$ .
- If a cycle exists, some power  $(W \circ W)^k$  contributes to the diagonal, making  $\text{tr}(e^{W \circ W}) > d$ .
- If no cycles exist, all diagonal terms remain 1, so  $\text{tr}(e^{W \circ W}) = d$ .

## SEM Likelihood: Single Variable Formulation

- Structural equation for variable  $x_k$ :

$$x_k = Xw_k + e_k, \quad e_k \sim \mathcal{N}(0, \sigma_k^2 I).$$

Example DAG Structure:

$$X_1 \rightarrow X_2, \quad X_1 \rightarrow X_3$$

$$W = \begin{pmatrix} 0 & 1.5 & -0.8 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$X_1 = z_1, \quad X_2 = 1.5 X_1 + z_2, \quad X_3 = -0.8 X_1 + z_3.$$

- Likelihood of a single column:

$$P(x_k | W, \sigma_k^2) = (2\pi\sigma_k^2)^{-N/2} \exp\left(-\frac{1}{2\sigma_k^2} \|x_k - Xw_k\|_2^2\right).$$

## Total Likelihood of the data

- Noise terms  $\{e_k\}$  are independent across variables.
- Full likelihood factorizes into column-wise components:

$$P(X | W, \sigma^2) = \prod_{k=1}^d P(x_k | W, \sigma_k^2).$$

# Sparse Bayesian Learning: Prior Structure

- SBL imposes sparsity via a hierarchical prior:

$$W_{jk} \sim \mathcal{N}(0, \Lambda_{jk}^{-1}), \quad \Lambda_{jk} \sim \text{Gamma}(a, b).^1$$

$$\text{Gamma}(\Lambda_{jk} \mid a, b) = \frac{b^a}{\Gamma(a)} \Lambda_{jk}^{a-1} e^{-b\Lambda_{jk}}.$$

- Large  $\Lambda_{jk}$  shrink  $W_{jk}$  toward zero (automatic sparsity).

---

<sup>1</sup>Tipping, 2001

## Full Joint Posterior: Likelihood + Priors

- Combining SEM likelihood with SBL priors:

$$\begin{aligned}\log P(X, W, \Lambda) &= \log P(X | W) \\ &\quad + \sum_{j,k} \log P(W_{jk} | \Lambda_{jk}) \\ &\quad + \sum_{j,k} \log P(\Lambda_{jk}).\end{aligned}$$

- Maximizing this posterior produces adaptive sparsity.

## Final Optimization Objective: SEM + SBL + NOTEARS

- Objective function:

$$\begin{aligned} J(W, \Lambda) = & \frac{1}{2N} \sum_{k=1}^d \frac{1}{\sigma_k^2} \|x_k - Xw_k\|_2^2 + \frac{1}{2} \sum_{j,k} \Lambda_{jk} W_{jk}^2 \\ & + \lambda h(W) - \frac{1}{2} \sum_{j=1}^d \sum_{k=1}^d \log(\Lambda_{jk}) \end{aligned}$$

- where

$$h(W) = \text{tr}(e^{W \circ W}) - d.$$

## Bayesian Setup (SBL Framework)

- Each column  $w_k$  is modeled as a Bayesian linear regression problem:

$$x_k = Xw_k + e_k, \quad e_k \sim \mathcal{N}(0, \sigma_k^2 I).$$

- Prior on weights:

$$w_k \sim \mathcal{N}(0, \text{diag}(\Lambda_k)^{-1}).$$

- Gaussian likelihood + Gaussian prior gives a Gaussian posterior.

## Posterior Updates

- Posterior covariance:

$$\Sigma_{w_k} = \left( \frac{1}{\sigma_k^2} X^\top X + \text{diag}(\Lambda_k) \right)^{-1}.$$

- Posterior mean:

$$\mu_{w_k} = \frac{1}{\sigma_k^2} \Sigma_{w_k} X^\top x_k.$$

## Hyperparameter Update (SBL Step)

- Precision parameters  $\Lambda_{jk}$  are updated by maximizing the marginal likelihood (Type-II ML) or using the EM M-step.<sup>2</sup>
- Update rule:

$$\Lambda_{jk}^{\text{new}} = \frac{1 - \Lambda_{jk}(\Sigma_{w_k})_{jj}}{(\mu_{w_k})_j^2}.$$

---

<sup>2</sup>Tipping, 2001.

## Update Rules: W-Step and Acyclicity Enforcement (ALM)

- **W-update (Augmented Lagrangian):<sup>3</sup>**

$$W^+ = \arg \min_W \left[ \frac{1}{2N} \sum_k \frac{1}{\sigma_k^2} \|x_k - Xw_k\|_2^2 + \frac{1}{2} \sum_{j,k} \Lambda_{jk} W_{jk}^2 + \alpha h(W) + \frac{\rho}{2} h(W)^2 \right].$$

- **Lagrange multiplier update:**

$$\alpha \leftarrow \alpha + \rho h(W).$$

- **Penalty update (optional):**

$$\rho \leftarrow c\rho, \quad c > 1.$$

---

<sup>3</sup>NOTEARS, Zheng et al., 2018

# Proposed Algorithm

---

**Algorithm 1**

---

1: **Input:** Data matrix  $\mathbf{X}$ , initial weights  $\mathbf{W}^{(0)}$ , regularization parameter  $\lambda$ , initial precision matrix  $\Lambda^{(0)}$ , noise variances  $\sigma^2$ , gamma parameters  $a, b$   
2: **Initialize:** Lagrange multiplier  $\alpha_0$ , penalty parameter  $\rho_0$   
3: **for**  $t = 0, 1, 2, \dots$  until convergence **do**  
4:   **(1) Update SBL hyperparameters:**  
5:     **for**  $k = 1$  to  $d$  **do**  
6:       Compute posterior mean  $\mu_{\mathbf{w}_k}^{(t)}$  and covariance  $\Sigma_{\mathbf{w}_k}^{(t)}$   
7:       **for**  $j = 1$  to  $d$  **do**  
8:         Update precision:  $\Lambda_{jk}^{(t+1)} = \frac{1 - \Lambda_{jk}^{(t)}(\Sigma_{\mathbf{w}_k}^{(t)})_{jj}}{(\mu_{\mathbf{w}_k}^{(t)})_j^2}$   
9:         **end for**  
10:      **end for**  
11:     **(2) Optimize  $\mathbf{W}$  via Augmented Lagrangian:**  
12:       Initialize:  $\mathbf{W}^{(0)}, \alpha_0, \rho_0$   
13:       **for**  $m = 0, 1, 2, \dots$  until inner convergence **do**  
14:         Update weights:  
          
$$\mathbf{W}^{(m+1)} = \arg \min_{\mathbf{W}} \mathcal{L}_{\text{aug}}(\mathbf{W}, \alpha_m, \rho_m)$$
  
15:         Update Lagrange multiplier:  
          
$$\alpha_{m+1} = \alpha_m + \rho_m h(\mathbf{W}^{(m+1)})$$
  
16:       **end for**  
17:       Set  $\mathbf{W}^{(t+1)} = \mathbf{W}^{(m+1)}$   
18:   **end for**  
19: **Thresholding:** Apply element-wise thresholding to the final  $\mathbf{W}$ :  
          
$$W_{jk} \leftarrow \begin{cases} W_{jk} & \text{if } |W_{jk}| \geq \tau \\ 0 & \text{otherwise} \end{cases}$$
  
20: **Output:** Final pruned DAG weight matrix  $\mathbf{W}$ 

---

# Evaluation Metrics and Comparison Setup

We consider three key metrics for causal graph recovery:

- **SHD (Structural Hamming Distance)** – Measures the number of edge additions, deletions, or flips required to transform the estimated graph into the ground truth.
- **FDR (False Discovery Rate)** – Fraction of discovered edges that are incorrect.

$$\text{FDR} = \frac{\text{False Positives} + \text{Reversed Edges}}{\text{Total Predicted Edges}}$$

- **Edge Intersection** – Number of correctly recovered edges.

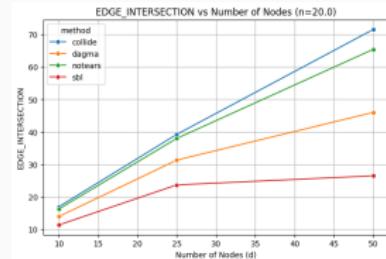
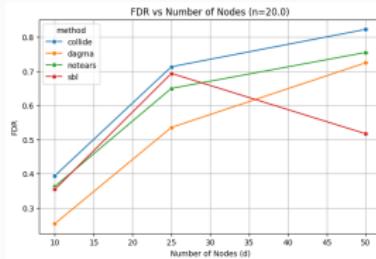
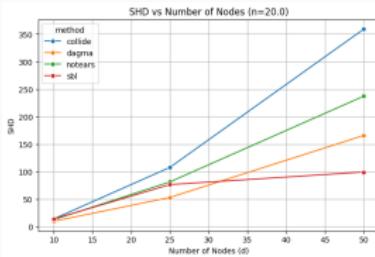
We compare our method (**SBL**) against three baselines:

**NOTEARS**, **DAGMA**, and **CoLiDE**.

---

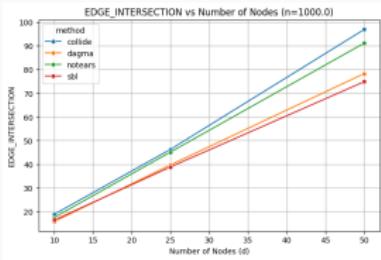
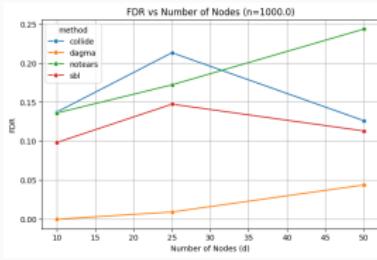
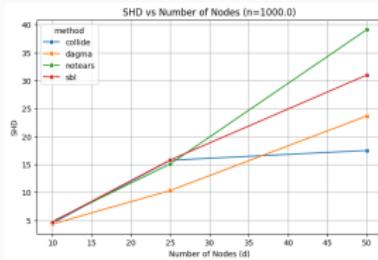
*Note: In all figures that follow, the method CoLiDE is inadvertently labeled as “COLLIDE”; all such instances should be interpreted as referring to CoLiDE.*

# Nodes vs SHD, FDR, Edge Intersection (Samples = 20)



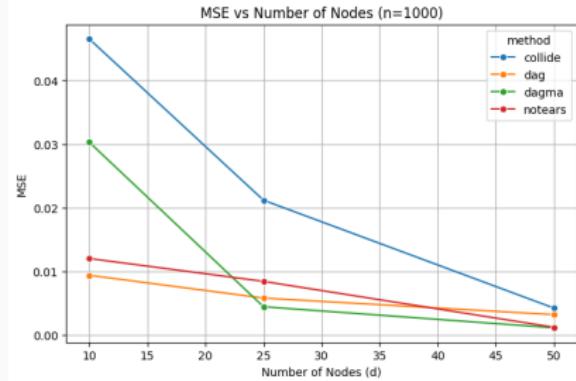
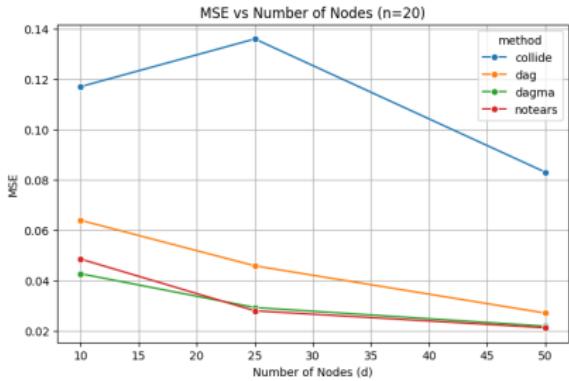
- SBL does not give rise to high SHD or high FDR compared to the other methods at low sample size.
- Consequently, its edge intersection remains on the lower side but follows a stable trend as the number of nodes increases.

# Nodes vs SHD, FDR, Edge Intersection (Samples = 1000)



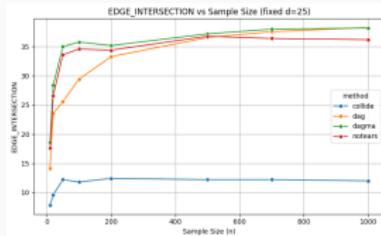
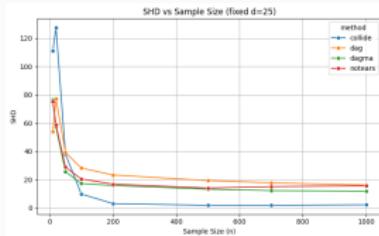
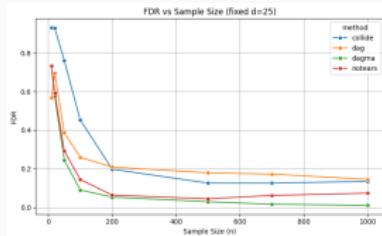
- SBL follows the same overall trend, with SHD and FDR remaining reasonably controlled across node sizes.
- Its edge intersection performance is also quite solid, even if it is not the best among the methods.

# Nodes vs MSE



- The MSE is defined as  $\|W - \widehat{W}\|_2^2$ .
- SBL (labelled as dag in the plot) shows low MSE, not always the lowest but consistently good enough across node sizes.

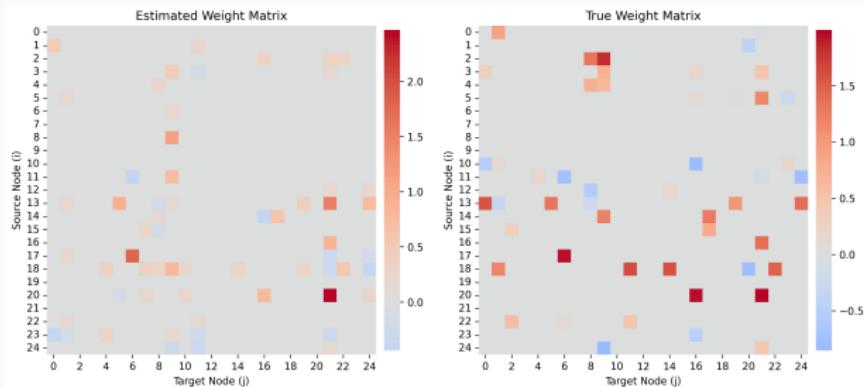
# Sample Size vs SHD, FDR, Edge Intersection



- At low sample sizes, SBL (labelled as DAG) shows lower FDR and SHD compared to the other methods.
- As the sample size increases, its FDR and SHD also rise, but the edge intersection remains reasonably high.

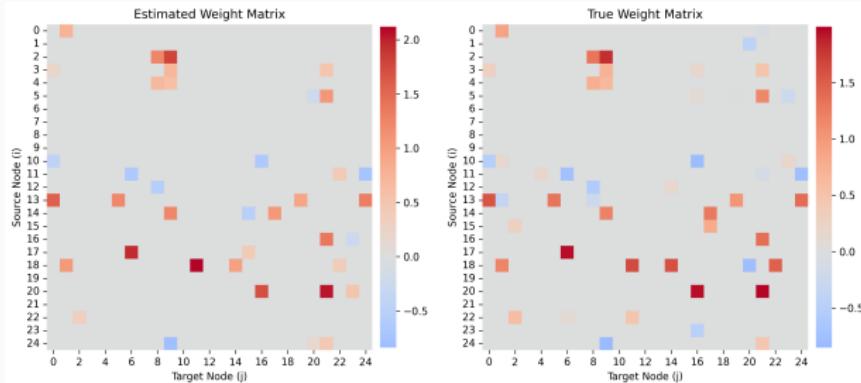
# Effect of Sample Size

Nodes = 25, Edges = 50, Samples = 20



# Effect of Sample Size

Nodes = 25, Edges = 50, Samples = 1000



## Real-World Dataset Performance: Sachs Dataset

We evaluated both SBL and NOTEARS on the Sachs protein-signaling dataset containing a DAG with 11 nodes and 853 samples. The table summarizes three key accuracy metrics: FDR, SHD, and Edge Intersection.

| Method  | FDR    | SHD | Edge Intersection |
|---------|--------|-----|-------------------|
| SBL     | 0.5714 | 14  | 6                 |
| NOTEARS | 0.5714 | 12  | 6                 |

## Conclusion

- We combined sparse Bayesian learning with the NOTEARS acyclicity formulation to develop a DAG learning method.
- The precision updates promoted sparsity, while the acyclicity term ensured a valid graph structure.
- The results were satisfactory: SHD and FDR remained controlled and edge recovery was reasonably accurate.
- Overall, the approach produced stable estimates and a balanced tradeoff between sparsity and structural correctness.

## Future Work

- Improving edge intersection while keeping FDR and SHD within acceptable bounds remains an important direction.
- The pruning threshold is currently selected through trial-and-error; establishing a systematic or theoretically motivated thresholding rule would strengthen the approach.
- Enhancing robustness to noise is essential for applying the method effectively to real-world datasets.

## References

- X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing, *DAGs with NO TEARS: Continuous Optimization for Structure Learning*, 2018.
- M. E. Tipping, *Sparse Bayesian Learning and the Relevance Vector Machine*, 2001.
- S. S. Saboktasir and G. Mateos, *COLIDE: Concomitant Linear DAG Estimation*.
- K. Bello, B. Aragam, and P. Ravikumar, *DAGMA: Learning DAGs via M-matrices and a Log-Determinant Acyclicity Characterization*.