Learning semi-Markovian DAGs with flow-based VAE

Dangchan Kim, Byungguk Kang, Jaeseok Kim, Minchan Kim

Seoul National University

December 11, 2023

Table of Contents

Overview

- **Problem**: Learning semi-Markovian DAGs
- **Solution**: Normalizing flow based VAE
- **Contribution**
  - A new method for learning semi-Markovian DAGs
  - A new method for learning DAGs with non-Gaussian noise

DAG and SEM

- Directed Acyclic Graph(DAG) $\mathcal{G} = (V, E)$ with $V = \{1, \ldots, m\}$ is used to represent a causal structure [4]
- Linear Structural Equation Model(SEM) is defined as

$$X_i = \sum_{j \in \mathrm{Pa}(i)} \beta_{ij} X_j + \epsilon_i$$

  where ϵ_i is a noise variable.
- Matrix form can be written as

$$\mathbf{X} = \mathbf{A}^T \mathbf{X} + \epsilon$$

  where $\mathbf{A}$ is an adjacency matrix of $\mathcal{G}$ and ϵ is a noise vector.

DAG and SEM (cont'd)

- If $(\mathbf{I} - \mathbf{A}^T)$ is nonsingular, SEM can be written as

$$\mathbf{X} = (\mathbf{I} - \mathbf{A}^T)^{-1}\epsilon$$

- Usually we assume $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ and

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, (\mathbf{I} - \mathbf{A}^T)^{-1}\mathbf{\Sigma}(\mathbf{I} - \mathbf{A})^{-1})$$

- In most cases, we assume $\mathbf{\Sigma} = \mathbf{I}$ or $\mathbf{\Sigma} = \mathbf{D}$ where $\mathbf{D}$ is a diagonal matrix.
- DAG-GNN [9] generalizes linear SEM using a deep neural network.

$$f_2^{-1}(X) = A^T f_2^{-1}(X) + f_1(Z),$$

Semi-Markovian DAG

- DAGs have many convenient properties (e.g. Markov property, d-separation, etc.)
- More general class of DAGs is acyclic directed mixed graphs (ADMGs) i.e. semi-Markovian DAGs
- Semi-Markovian DAG is a DAG where the noise variables have dependencies [6]
- The noise vector ϵ is a semi-Markovian noise vector if

  $$\Sigma \in \mathcal{W}(G) := \{\Sigma \in \mathbb{R}^{m \times m} : \Sigma_{ij} = \Sigma_{ji} = 0 \text{ if } i \notin \mathrm{sib}(j)\}$$

  where $\mathrm{sib}(j) = \{i : i \sim j \text{ in } G\}$ [8]
- Learn cyclic adjacency matrix [8] or acyclic adjacency matrix

DAG Learning

- DAG learning is a problem of estimating $\mathcal{G}$ from $\mathbf{X}$
- NP-hard problem [2]
- Score-based methods [7] and constraint-based methods [1] are used
- NOTEARS[10]: Combinatorial optimization to Continuous optimization problem

$$\min_{\mathbf{A}} F(A) \text{ subject to } G(A) \in \mathbb{D} \Rightarrow \min_{\mathbf{A}} F(A) \text{ subject to } h(A) = 0$$

where $\mathbb{D}$ denotes the discrete space of DAGs on m nodes and $h(A) = 0$ iff $G(A)$ is acyclic.

DAG Learning (cont'd)

Theorem

A matrix $\mathbf{A}$ is the adjacency matrix of a DAG if and only if

$$h(A) = \mathrm{tr}(\exp(\mathbf{A} \circ \mathbf{A})) = \mathrm{m}$$

where $\circ$ denotes the Hadamard product.

Theorem

A matrix $\mathbf{A}$ is the adjacency matrix of a DAG if and only if

$$\mathrm{tr}[(\mathrm{I} + \alpha\mathbf{A} \circ \mathbf{A})^{\mathrm{m}}] = \mathrm{m}$$

for any $\alpha > 0$.

VAE

- Variational Autoencoder(VAE) is a generative model that learns a latent variable model
- It is trained by maximizing the evidence lower bound(ELBO)

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

- $q_\phi(\mathbf{z}|\mathbf{x})$ is an approximate posterior and $p_\theta(\mathbf{x}|\mathbf{z})$ is a generative model

Normalizing Flow

- Normalizing flow transforms a simple distribution to a complex distribution [5]
- It is a sequence of invertible transformations $f_t : \mathbb{R}^d \to \mathbb{R}^d$

$$z_T = f_T \circ f_{T-1} \circ \cdots \circ f_1(z_0)$$

- The probability density function of z_T is

$$p(z_T) = p(z_0) \left| \det \left(\frac{\partial f_T \circ f_{T-1} \circ \cdots \circ f_1(z_0)}{\partial z_0} \right) \right|$$

Linear IAF

- Inverse Autoregressive Flow(IAF) is a normalizing flow that uses autoregressive transformation [3]
- Linear IAF is a special case of IAF where f_t is a linear transformation

$$\mathbf{z_t} = \mu_\mathbf{t} + \sigma_\mathbf{t} \odot \mathbf{z_{t-1}}.$$

- Transformation of the linear IAF can be written as

$$\mathbf{z}_T = \mathbf{L}(\mathbf{x}) \cdot \mathbf{z}_0.$$

  where $\mathbf{L}(\mathbf{x})$ is a lower triangular matrix from encoder network.
- We can use more flexible prior distribution by using normalizing flow.
- When $\mathbf{z}_0$ has diagonal covariance, $\mathbf{z}_T$ then has full covariance matrix.

Linear IAF (cont'd)

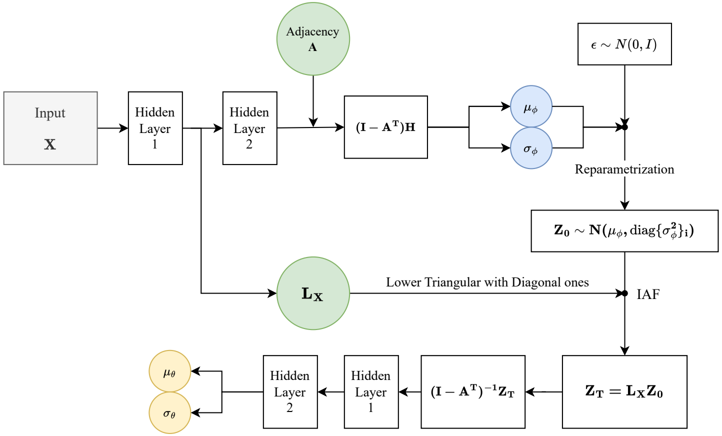
- The KL divergence term of ELBO can be written as

$$\begin{aligned} D_{KL}(q_\phi(\mathbf{z}_0|\mathbf{x}) \| p(\mathbf{z}_T)) &= \log q_\phi(\mathbf{z}_0|\mathbf{x}) - \log p(\mathbf{z}_T) \\ &= -\frac{1}{2}(\mathbf{z}_0 - \mu_\phi)^T \Sigma_\phi^{-1}(\mathbf{z}_0 - \mu_\phi) + \frac{1}{2}\mathbf{z}_T^T \mathbf{z}_T \end{aligned}$$

where μ_ϕ and Σ_ϕ are the mean and covariance matrix from the encoder network.

Model

Overall architecture of our model is shown in Figure below.



Adjacency $\mathbf{A}$
Input $\mathbf{X}$
Hidden Layer 1
Hidden Layer 2
$(\mathbf{I}-\mathbf{A}^{\mathbf{T}})\mathbf{H}$
μ_ϕ
σ_ϕ
$\epsilon \sim N(0, I)$
Reparametrization
$\mathbf{Z_0} \sim \mathbf{N}(\mu_\phi, \mathrm{diag}\{\sigma_\phi^2\}_\mathbf{i})$
$\mathbf{L_X}$
Lower Triangular with Diagonal ones
IAF
$\mathbf{Z_T} = \mathbf{L_X Z_0}$
$(\mathbf{I}-\mathbf{A}^{\mathbf{T}})^{-1}\mathbf{Z_T}$
Hidden Layer 1
Hidden Layer 2
μ_θ
σ_θ

Figure: Overall architecture of our model

Learning

- The optimization procedure is minimizing the following loss function:

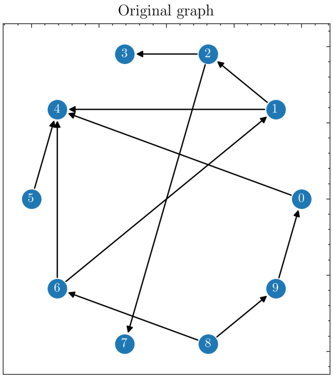
$$\mathcal{L}(A, W, \lambda) = -\mathcal{L}_{\mathrm{ELBO}} + \tau \|A\|_1 + \lambda h(A) + \frac{c}{2}|h(A)|^2.$$

- The second term is the L1 regularization term, which encourages sparsity of the adjacency matrix.
- $\tau = 0.1 \times (\frac{m}{50})^2$ works well in our experiment setting.
- The third and fourth terms are the augmented Lagrangian terms.
- Gradually increasing the value of c during the training process results the acyclicity constraint to zero.

Experiment

- We compare our method with the DAG-GNN [9] with random graph datasets.
- thresholding value of extracting graph as 0.3
- Erdős–Rényi random graph with 5000 samples
- Node size : 10, 20, 30, and 50 nodes
- Noise type
  - Independent : $\epsilon_i \sim \mathcal{N}(0, 1)$, $\epsilon_i \sim \text{Laplace}(0, 1)$, $\epsilon_i \sim \text{Exp}(1)$
  - Dependent : $\epsilon_i \sim \mathcal{N}(0, \mathbf{\Sigma})$ where $\mathbf{\Sigma}$ is a covariance matrix of semi-Markovian DAG

Experiment (cont'd)



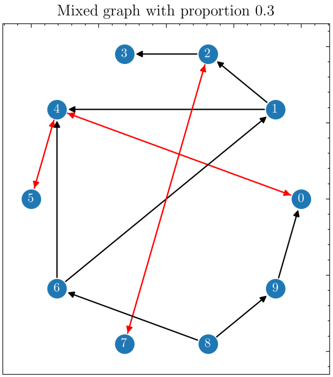


Figure: Example of mixed graph

Experiment (cont'd)

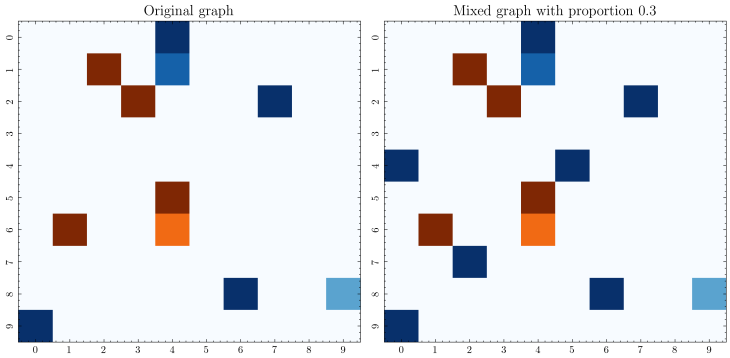


Figure: Adjacency matrix of previous example graph

Experiment(Cont'd)

- We evaluate the performance of our method using two metrics: Structural Hamming Distance (SHD), False Discovery Rate (FDR) with respect to the number of predicted edges.
- SHD measures the number of edge additions, deletions, and reversals required to transform the estimated graph into the true graph.
- FDR represents the ratio of false positives to the total number of predicted edges.
- These metrics are calculated by comparing the estimated graph with the true DAG.
- For each combination, we generate at least 5 random graphs and calculate the average metrics.

Independent Noise

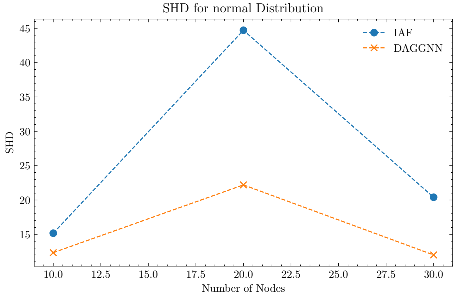


Figure: SHD with normal

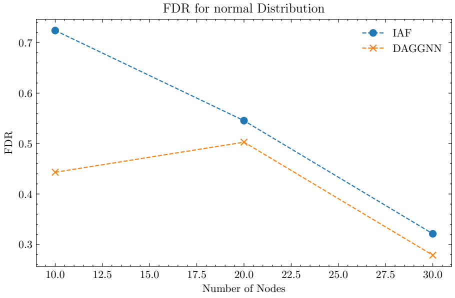


Figure: FDR with normal

- In the case of independent Gaussian noise, DAG-GNN(orange) performs better than our method(blue).

Independent Noise (cont'd)

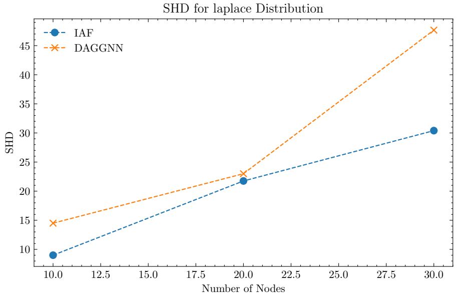


Figure: SHD with Laplace

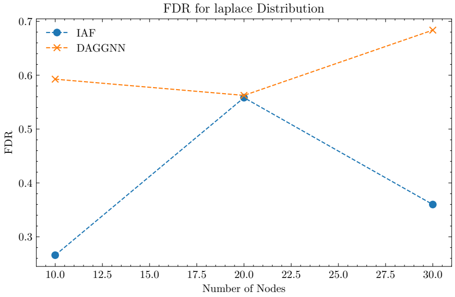


Figure: FDR with Laplace

In the case of independent Laplace noise, our method(blue) performs better than DAG-GNN(orange).

Independent Noise (cont'd)

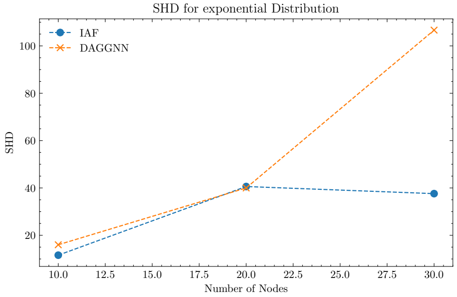


Figure: SHD with Exponential

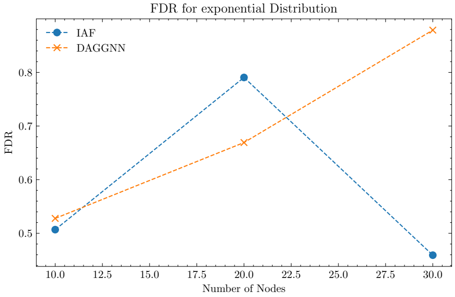


Figure: FDR with Exponential

In the case of independent Exponential noise, our method(blue) performs better than DAG-GNN(orange).

Independent Noise (cont'd)

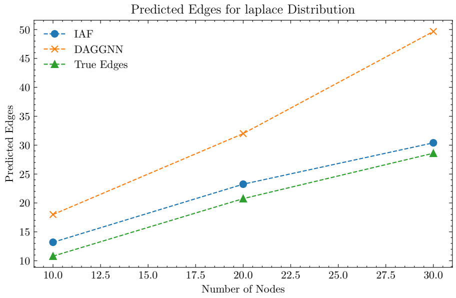


Figure: Number of predicted edges with Laplace

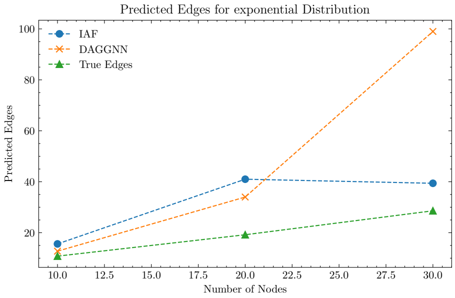


Figure: Number of predicted edges with Exponential

Considering the number of predicted edges with the SHD and FDR, our method(blue) performs better than DAG-GNN(orange).

Independent Noise (cont'd)

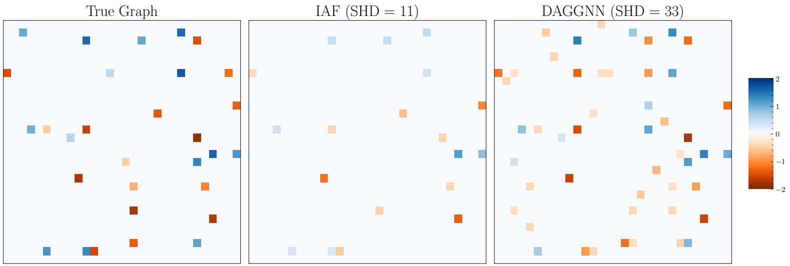


Figure: Comparison of estimated graphs with Laplace noise

Independent Noise (cont'd)

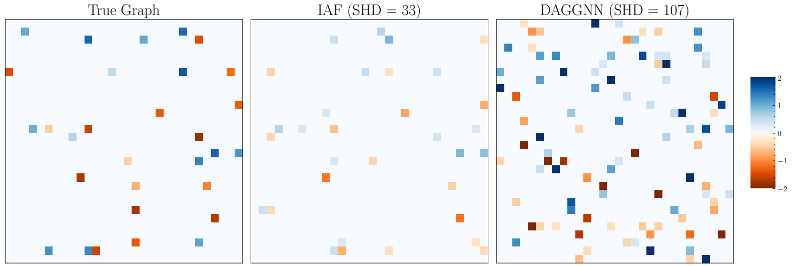


Figure: Comparison of estimated graphs with Exponential noise

Dependent Noise

Proportion of bidirectional edges fixed to 0.3.

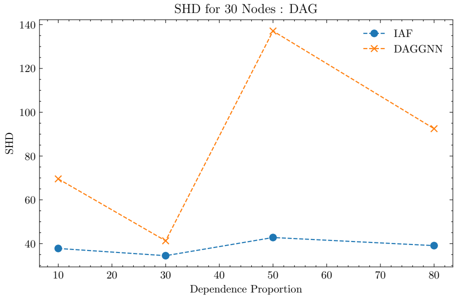


Figure: SHD with dependent Gaussian

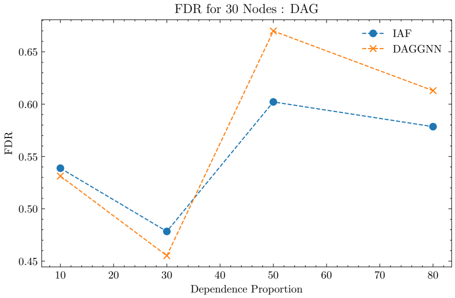


Figure: FDR with dependent Gaussian

Proportion of bidirectional edges fixed to 0.3.

Dependent Noise (cont'd)

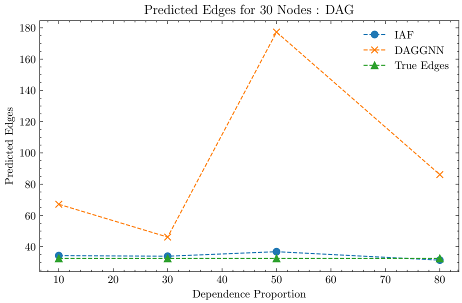


Figure: Number of predicted edges with dependent Gaussian

- In the case of dependent Gaussian noise, our method(blue) performs better than DAG-GNN(orange).
- Considering the stable number of predicted edges with the SHD and FDR, our method(blue) outperforms.

Dependent Noise (cont'd)

Node size $m = 10$: similar performance between DAG-GNN(orange) and our method(blue).

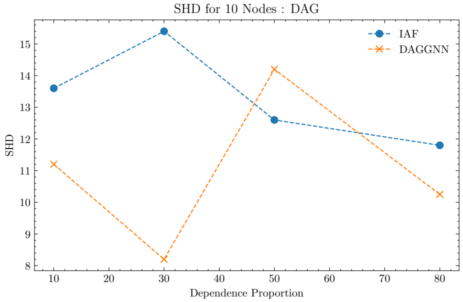


Figure: SHD with dependent Gaussian

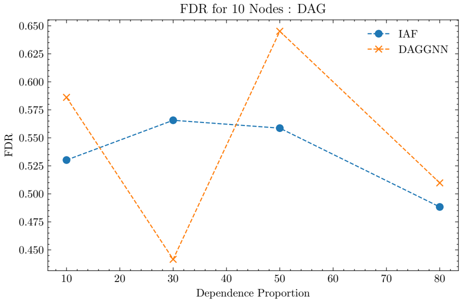


Figure: FDR with dependent Gaussian

Node size $m = 20$: our method(blue) performs better than DAG-GNN(orange).

Dependent Noise (cont'd)

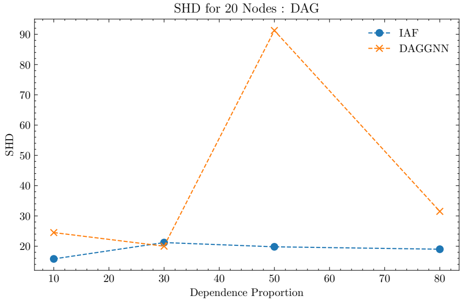


Figure: SHD with dependent Gaussian

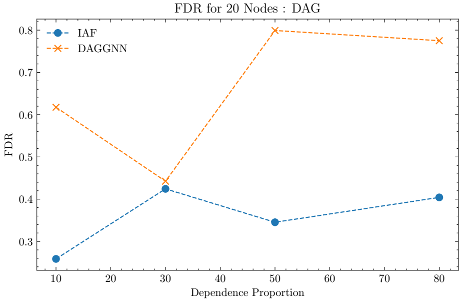


Figure: FDR with dependent Gaussian

Node size $m = 30$: our method(blue) performs better than DAG-GNN(orange).

Dependent Noise (cont'd)

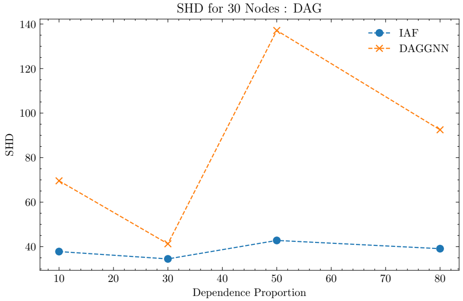


Figure: SHD with dependent Gaussian

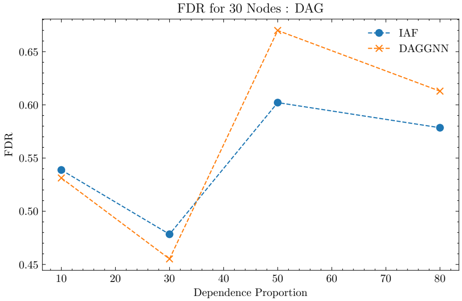


Figure: FDR with dependent Gaussian

Node size $m = 50$: our method(blue) performs better than DAG-GNN(orange).

Dependent Noise (cont'd)

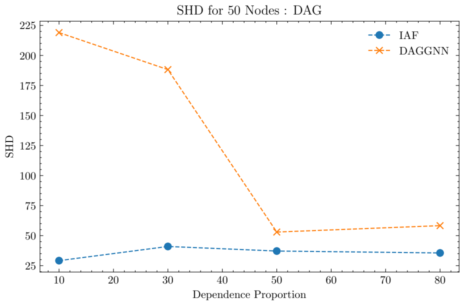


Figure: SHD with dependent Gaussian

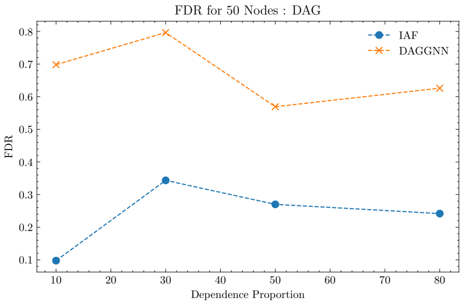


Figure: FDR with dependent Gaussian

Dependent Noise (cont'd)

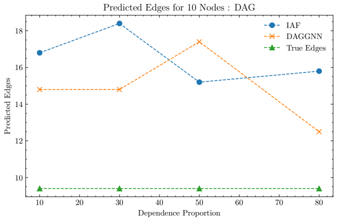


Figure: $m = 10$

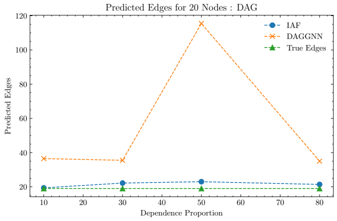


Figure: $m = 20$

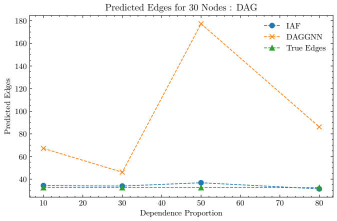


Figure: $m = 30$

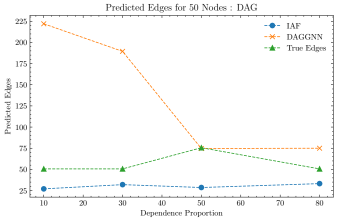


Figure: $m = 50$

Dependent Noise (cont'd)

- FPR : our method performs better than DAG-GNN.
- TPR : similar performance

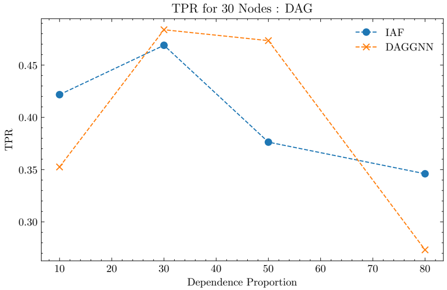


Figure: TPR at $m = 30$

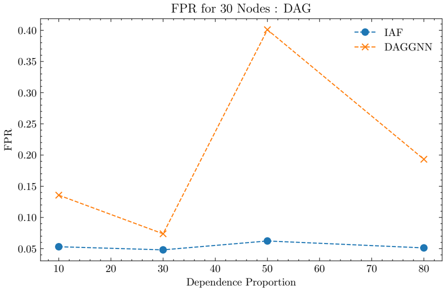


Figure: FPR at $m = 30$

Conclusion

- Proposed a method to learn the structure of a semi-Markovian DAG using a flow-based VAE.
- Conducted experiments on simulated data and compared the performance of our method with that of DAG-GNN.
- Our method outperforms DAG-GNN in terms of both SHD and FDR metrics, especially when the noise variables have a dependent structure and when the size of the graph is large.
- In terms of the number of predicted edges, our method results in fewer edges than DAG-GNN while maintaining the same or better level of SHD and FDR.

Conclusion

- However, contrary to our initial expectations, the lower triangular matrix $\mathbf{L}$ learned in the IAF layer did not capture the covariance matrix of the actual noise variables.
- This seems to be due to the entanglement problem in the latent space, and resolving this issue remains a future research task.
- If the full covariance matrix of the noise variables could be found, it could be possible to directly identify bidirectional edges in the graph, enabling more accurate learning of the semi-Markovian graph.

References

[1] David Maxwell Chickering. "Optimal Structure Identification with Greedy Search". In: *J. Mach. Learn. Res.* 3.null (2003), pp. 507–554. ISSN: 1532-4435. DOI: 10.1162/153244303321897717. URL: https://doi.org/10.1162/153244303321897717.

[2] David Maxwell Chickering, Christopher Meek, and David Heckerman. "Large-Sample Learning of Bayesian Networks is NP-Hard". In: *CoRR* abs/1212.2468 (2012). arXiv: 1212.2468. URL: http://arxiv.org/abs/1212.2468.

[3] Durk P Kingma et al. "Improved Variational Inference with Inverse Autoregressive Flow". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

[4] Judea Pearl. *Causality*. 2nd ed. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511803161.

References (cont'd)

[5] Danilo Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1530–1538. URL: https://proceedings.mlr.press/v37/rezende15.html.

[6] Ilya Shpitser and Judea Pearl. "Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models". In: *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*. AAAI'06. Boston, Massachusetts: AAAI Press, 2006, 1219–1226. ISBN: 9781577352815.

[7] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. 2nd. MIT press, 2000.

References (cont'd)

[8] Y. Samuel Wang and Mathias Drton. "Empirical likelihood for linear structural equation models with dependent errors". In: *Stat* 6.1 (2017), pp. 434–447. DOI: https://doi.org/10.1002/sta4.169. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sta4.169. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sta4.169.

[9] Yue Yu et al. "DAG-GNN: DAG Structure Learning with Graph Neural Networks". In: (2019). arXiv: 1904.10098 [cs.LG].

[10] Xun Zheng et al. *DAGs with NO TEARS: Continuous Optimization for Structure Learning*. 2018. arXiv: 1803.01422 [stat.ML].