DAG-GNN Structure Learning

ECSE4810
Donggyu Kim

1. Introduction

Directed Acyclic Graphs (DAGs) serve as a foundational representation in numerous domains, including causal inference, Bayesian networks, and probabilistic graphical models. A DAG encapsulates the conditional dependencies among a set of variables through its directed edges, making it invaluable for applications ranging from medicine and genetics to economics and machine learning. Despite its utility, learning the structure of a DAG from data remains a daunting challenge due to the computational complexity involved.

The task of DAG structure learning involves identifying a graph that faithfully represents the dependencies in the data while ensuring the absence of cycles. Traditional approaches, such as score-based and constraint-based methods, frame this as an optimization problem, often with a combinatorial constraint to maintain acyclicity. However, these methods suffer from scalability issues, as the search space grows superexponentially with the number of graph nodes.

Recent advances, such as the NOTEARS method proposed by Zheng et al. (2018), have revolutionized the field by reformulating the acyclicity constraint as a continuous function. This innovation enabled the use of gradient-based optimization techniques, significantly improving the efficiency of DAG learning. While effective, these methods often rely on linear assumptions, which limit their applicability to real-world datasets characterized by complex, nonlinear relationships.

DAG-GNN (Directed Acyclic Graph Neural Networks) builds upon these advancements by integrating deep generative models and graph neural networks (GNNs) to capture intricate data distributions faithfully. By employing a variational autoencoder (VAE) framework, DAG-GNN extends the capabilities of linear Structural Equation Models (SEMs) to handle nonlinear, discrete, and vector-valued data. This approach not only preserves the acyclic nature of the learned graph but also enhances its representational power, making it suitable for diverse applications.

Directed Acyclic Graph

A Directed Acyclic Graph (DAG) is a graphical representation consisting of nodes connected by directed edges, where the edges indicate a directional relationship between the nodes. Importantly, a DAG does not contain any cycles, meaning there is no path that starts and ends at the same node following the direction of the edges.

DAGs are widely used to model hierarchical or dependency-based systems.

- **Causal Inference**: DAGs depict cause-effect relationships among variables, making them essential for understanding causality.
- **Bayesian Networks**: DAGs serve as the backbone for representing probabilistic dependencies among random variables.
- **Workflow Systems**: In computational and organizational workflows, DAGs describe tasks and their precedence constraints.

Mathematically, a DAG can be defined as a pair, where there is a set of nodes and a set of directed edges such that for every directed edge, there is no path from back to. This property ensures that DAGs represent acyclic structures, which are critical for modeling processes with inherent directionality or order.

This foundational concept underpins many algorithms and methodologies in machine learning, particularly in the context of learning dependency structures from data.

Graph Neural Network

A Graph Neural Network (GNN) is a type of neural network designed to operate on graph-structured data, where the data is represented as nodes connected by edges. The primary goal of a GNN is to learn node, edge, or graph-level representations by aggregating information from the graph structure.

Basic Structure of GNN

The core operation of a GNN involves the following steps:

1. **Node Representation Initialization**: Each node in the graph is initialized with a feature vector, typically derived from input data.
2. **Message Passing and Aggregation**: Nodes exchange information with their neighbors by aggregating messages. The aggregated message for node at layer is: where denotes the neighbors of node, and is a function, such as summation, mean, or maximum.
3. **Node Update**: The node features are updated based on the aggregated message and the node's previous state:
   where is typically implemented as a neural network layer (a feedforward or recurrent network).
4. **Readout**: After a fixed number of layers, a readout function combines the node-level features into a graph-level representation (if required).

Mathematical Framework

Let denote the matrix of node features at layer, where is the number of nodes and is the feature dimension. The adjacency matrix represents the graph structure. A typical GNN layer can be expressed as:

where:

- is a normalized adjacency matrix (e.g., where is the identity matrix and is the degree matrix),
- is a learnable weight matrix for layers,
- is a non-linear activation function (e.g., ReLU).

By stacking multiple layers, the GNN allows each node to incorporate information from increasingly larger neighborhoods, enabling the model to learn higher-order dependencies in the graph structure.

Directed Acyclic Graph Neural Network

A Directed Acyclic Graph Neural Network (DAG-GNN) extends the principles of GNNs to learn the structure of Directed Acyclic Graphs (DAGs). Unlike standard GNNs, which primarily operate on general graph structures, DAG-GNNs are tailored to capture dependencies specific to DAGs while maintaining the acyclicity constraint.

Core Idea

DAG-GNN combines a Graph Neural Network (GNN) with a Variational Autoencoder (VAE) framework. The VAE consists of:

1. **Encoder**: Maps input data into a latent representation that reflects the underlying DAG structure.
2. **Decoder**: Reconstructs the input data from the latent representation, ensuring that the reconstructed data adheres to the learned DAG structure.

The adjacency matrix representing the DAG is treated as a learnable parameter, optimized during training to satisfy both the data reconstruction objective and the acyclicity constraint.

Acyclicity Constraint

To ensure the learned adjacency matrix corresponds to a valid DAG, DAG-GNN imposes an acyclicity constraint. This constraint is mathematically formulated to prevent cycles in the graph by leveraging continuous optimization techniques, such as matrix-based exponential or polynomial functions.

Loss Function

The loss function in DAG-GNN integrates three components:

1. **Reconstruction Loss**: Measures how well the decoder reconstructs the input data.
2. **KL Divergence**: Regularizes the latent representation by aligning it with a prior distribution.
3. **Acyclicity Penalty**: Ensures that the learned graph structure is acyclic.

By combining these elements, DAG-GNN learns a faithful representation of the DAG structure while effectively modeling complex, nonlinear relationships in the data.

Constraint-Based Structure Learning: Acyclicity Constraint

In the context of structure learning for Directed Acyclic Graphs (DAGs), constraint-based methods focus on ensuring that the learned graph adheres to specific properties, such as acyclicity. The Acyclicity Constraint is a critical component in DAG learning, ensuring that the graph does not contain any cycles. This property is essential for the DAG to represent causal or dependency relationships accurately.

Mathematical Formulation

To enforce acyclicity, the adjacency matrix representing the DAG must satisfy:

where:

- The adjacency matrix is the Hadamard (element-wise) square of the adjacency matrix.
- The adjacency matrix represents the matrix exponential, which computes a weighted sum of all powers of the matrix.
- The adjacency matrix calculates the sum of the diagonal elements and is the number of nodes.

This formulation ensures that any nonzero diagonal element in higher powers (indicating cycles) contributes to the penalty, preventing cycles in the learned graph.

Practical Approximation

While the matrix exponential is mathematically elegant, it can be computationally expensive. As a practical alternative, DAG-GNN uses a polynomial approximation for the acyclicity constraint, such as:

where:

- is the identity matrix,
- is a scaling parameter,
- is a sufficiently large power.

This approximation is more suitable for implementation in modern deep-learning frameworks and maintains numerical stability during optimization.

Integration into Loss Function

The acyclicity constraint is incorporated into the overall loss function as a penalty term:

where a hyperparameter controls the weight of the penalty. By minimizing this loss term, the optimization process ensures that the learned graph remains acyclic while satisfying the data reconstruction and latent regularization objectives.

The acyclicity constraint is a cornerstone of DAG learning, enabling the discovery of valid graph structures that accurately capture the underlying dependencies in the data.

Review of Paper: DAG-GNN

1. Linear Structural Equation Model (SEM)

The Linear SEM represents a system of equations modeling dependencies among variables. The equation:

$$X = A^T X + Z$$

encodes relationships such that each variable in X is expressed as a linear combination of its parents (connected nodes in the DAG), with an added noise component. Key details:

- **Adjacency Matrix (A)**:
  - Encodes the graph structure.
  - A strictly upper triangular matrix when variables are arranged in a topological order, ensures that no directed cycles exist.
- **Noise (Z)**:
  - Often assumed to be independent and identically distributed (i.i.d.).
  - Introduces stochasticity to the data generation process.

When the nodes of a Directed Acyclic Graph (DAG) are sorted in topological order, the adjacency matrix A becomes strictly upper triangular. Consequently, ancestral sampling from the DAG can be performed by first generating a random noise vector Z and then applying a triangular solution to obtain the desired samples. The solution for X uses the matrix inversion of (I−A^T).

$$X = (I - A^T)^{-1} Z$$

- (I−A^T) is invertible for acyclic graphs because the determinant is non-zero.

This formulation highlights the hierarchical flow of information, where cycles are inherently avoided by the acyclicity of A.

2. Generalizing SEM with Graph Neural Networks

To extend the linear SEM to nonlinear settings, a Graph Neural Network (GNN) is employed. The general form becomes:

$$X = f_A(Z)$$

where:

- fA(Z) is a parameterized function incorporating graph structural information from A.
- Z is the input node feature matrix.

The architecture for DAG-GNN introduces layers of nonlinear transformations via two functions f1 and f2:

$$X = f_2\left((I - A^T)^{-1} f_1(Z)\right)$$

Key points:

1. **f1**:
    - Encodes features Z into a latent representation suitable for DAG learning.
    - Implemented as a neural network, e.g., a Multilayer Perceptron (MLP).
2. **(I−AT)^−1**:
    - Propagates information across the graph using the adjacency structure.
    - Ensures global dependency modeling while respecting acyclic constraints.
3. **f2**:
    - Decodes latent features into reconstructed variables X.
    - Adds flexibility for nonlinear mappings.

If f2 is invertible, the nonlinear SEM becomes:

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

This highlights the model's ability to reconstruct dependencies from data while maintaining a DAG structure.

3. Variational Autoencoder (VAE) Framework

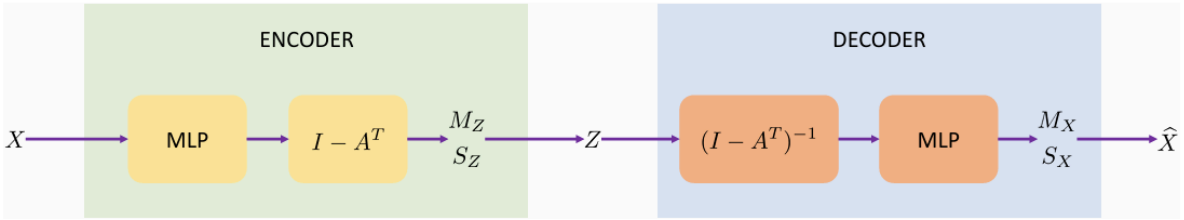

Figure1. Architecture(for continuous variables) In the case of discrete variables,the decoder output is changed from MX SX to PX

The DAG-GNN employs a VAE to optimize the model. The goal is to maximize the log-likelihood of the observed data X:

$$\log p(X) = \sum_{k-1}^{n} \log \int p(X_k|Z)p(Z)\,dZ$$

This integral is intractable for high-dimensional Z, so the Evidence Lower Bound (ELBO) is optimized:

$$L_{\text{ELBO}} = \frac{1}{n}\sum_{k-1}^{n}\left[-D_{\text{KL}}(q(Z|X_k)\|p(Z)) + \mathbb{E}_{q(Z|X_k)}[\log p(X_k|Z)]\right]$$

- **q(Z|Xk)**: Approximates the true posterior p(Z|Xk), typically modeled as a Gaussian with learnable mean and variance.
- **p(Z)**: The prior, often a standard Gaussian N(0,I).
- **p(Xk|Z)**: The likelihood of reconstructing Xk given latent features Z.

The ELBO balances two objectives:

1. **Reconstruction Term**: Measures how well X is reconstructed from Z.
2. **KL Divergence**: Ensures that the latent space Z follows the prior distribution.

4. Encoder and Decoder Details

The encoder maps input data X to latent features Z:

$$Z = f_4\left((I - A^T)f_3(X)\right)$$

- **f3**: An MLP that transforms X into a representation capturing local node features.
- **f4**: Typically identity mapping, ensures the interpretability of the learned features.

The variational posterior q(Z∣X) is modeled as a Gaussian with parameters computed as:

$$[M_Z|\log S_Z] = (I - A^T) \cdot \text{MLP}(X, W_1, W_2)$$

- **MZ**: Mean of the latent Gaussian distribution.
- **SZ**: Standard deviation (encoded as logSZ for numerical stability).

The decoder reconstructs X from Z:

$$[M_X|\log S_X] = \text{MLP}\left((I - A^T)^{-1}Z, W_3, W_4\right)$$

The likelihood p(X∣Z) is modeled as a Gaussian distribution with the decoded mean and variance.

5. Acyclicity Constraint

A critical part of DAG learning is ensuring the adjacency matrix A represents an acyclic graph. The acyclicity constraint is formulated as:

$$h(A) = \text{tr}[(I + \alpha A \circ A)^m] - m$$

- **tr**: Trace operator, summing diagonal elements of the matrix.
- **A∘A**: Elementwise square of A.
- **α>0**: Scaling factor for numerical stability.
- **m**: Number of nodes in the graph.

This formulation ensures that cycles (nonzero entries in higher powers of A) are penalized during optimization. The constraint can also be written using the matrix exponential for exact computation:

$$h(A) = \text{tr}(\exp(A \circ A)) - m$$

6. Loss Function with Augmented Lagrangian

The learning process combines the ELBO and acyclicity constraint into a unified loss function using an augmented Lagrangian approach:

$$\mathcal{L}_c(A, \theta, \lambda) = f(A, \theta) + \lambda h(A) + \frac{c}{2}|h(A)|^2$$

where:

- f(A,θ)=−L(ELBO): Objective to maximize data likelihood.
- h(A): Enforces the acyclicity constraint.
- λ: Lagrange multiplier, updated iteratively.
- c: Penalty parameter to penalize constraint violations.

This formulation ensures that the learned graph is a valid DAG while optimizing the reconstruction and representation of learning objectives.

This chapter delves into Neural DAG Structure Learning, focusing on how DAG-GNN integrates Graph Neural Networks (GNNs) with a Variational Autoencoder (VAE) framework to learn Directed Acyclic Graphs (DAGs) from data. The model generalizes the linear Structural Equation Model (SEM) to nonlinear settings by parameterizing dependencies using graph-based transformations. Key components include the encoder, which maps input data to latent features, and the decoder, which reconstructs the data while maintaining acyclic constraints. The acyclicity constraint is enforced through a trace-based formulation to ensure that the learned adjacency matrix represents a valid DAG. The overall objective is optimized using an augmented Lagrangian method, which combines the evidence lower bound (ELBO) and the acyclicity penalty.

Experimental Result

The experimental evaluation of the DAG-GNN involved training on predefined graph data(ASIA dataset) which is binary to recover the structure of a Directed Acyclic Graph (DAG). The training process was analyzed through three key loss metrics: KL Divergence Loss, Negative Log-Likelihood (NLL) Loss, and the Acyclicity Constraint.

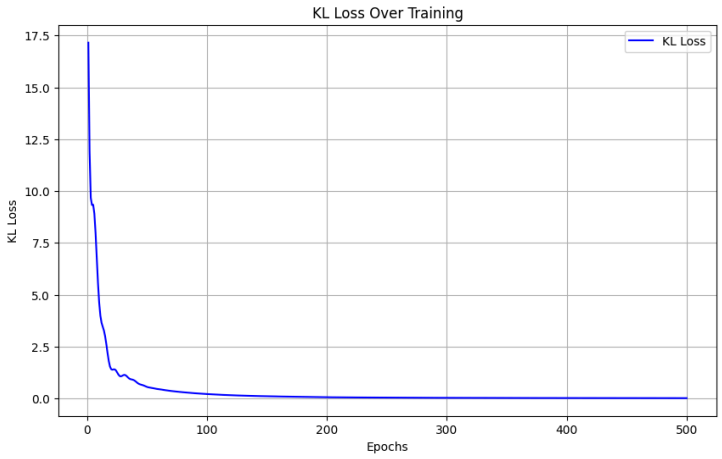

Figure2. KL loss over Training

KL Divergence Loss

The KL Divergence Loss measures how closely the variational posterior aligns with the prior distribution. The graph indicates that the KL Loss rapidly decreases during the initial training epochs, eventually converging to near-zero values. This suggests that the latent space representation aligns well with the imposed prior distribution, ensuring stability in the learned generative model.

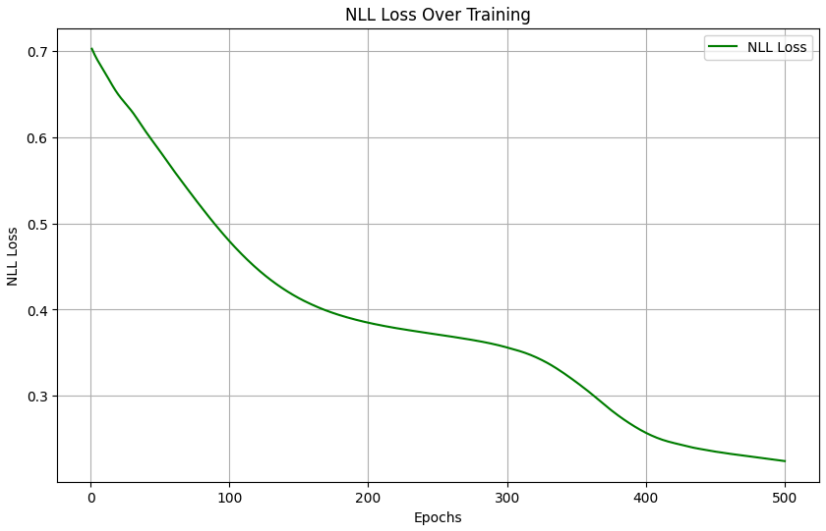

Figure3 NLL loss over Training

NLL Loss

The NLL Loss evaluates the quality of the reconstructed data against the original input. The NLL Loss graph shows a steady decline, indicating continuous improvement in reconstruction quality throughout training. However, the slower convergence in later epochs suggests potential room for optimization in the encoder and decoder structures.

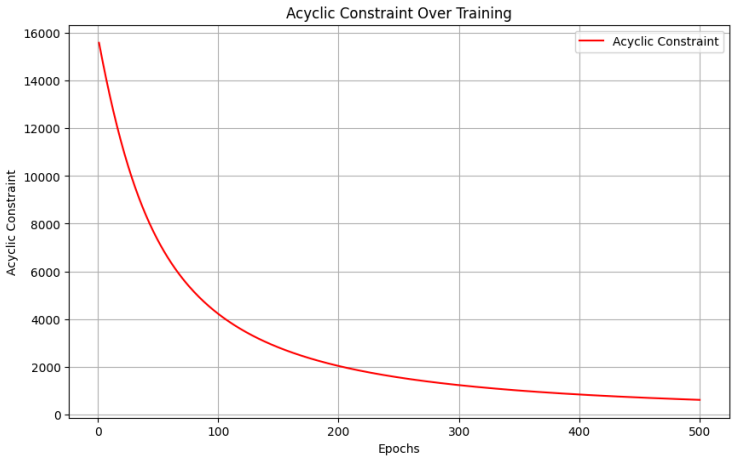

Figure4 Acyclicity Constraint over Training

Acyclicity Constraint

The Acyclicity Constraint graph reveals a substantial reduction in the penalty over training, demonstrating that the model effectively learns to enforce the DAG's acyclic structure. Despite this improvement, minor residual values indicate challenges in achieving perfect acyclicity due to limitations in the threshold selection for adjacency matrix values.

Challenges

Adjacency Matrix Recovery:

Although the DAG-GNN successfully minimized the acyclicity constraint and reconstruction errors, the exact adjacency matrix could not be perfectly recovered. This limitation arose due to difficulties in selecting an optimal threshold to interpret adjacency matrix values as binary (edges present or absent).

Encoder and Decoder Limitations:

Minor structural issues in the encoder and decoder in my code contributed to discrepancies in reconstructing the exact graph structure. Refining these modules could improve the alignment between the learned and ground-truth graphs.

Conclusion

This report provided an in-depth exploration of the Directed Acyclic Graph Neural Network (DAG-GNN) framework, a cutting-edge approach to learning DAG structures from data. By integrating Graph Neural Networks (GNNs) with a Variational Autoencoder (VAE), DAG-GNN extends traditional Structural Equation Models (SEMs) to handle nonlinear dependencies and complex relationships in data. The key strength of DAG-GNN lies in its ability to combine generative modeling with structural learning, enabling the discovery of directed acyclic structures that can be used for causal inference and other applications.

The experimental results highlighted the effectiveness of the DAG-GNN framework in minimizing key loss metrics during training. The KL Divergence Loss demonstrated rapid convergence, indicating that the latent space representation effectively aligns with the prior distribution. Similarly, the steady decrease in Negative Log-Likelihood (NLL) Loss showed that the encoder and decoder were able to accurately capture and reconstruct the underlying relationships within the data. Furthermore, the consistent reduction in the Acyclicity Constraint penalty validated the model's ability to enforce the structural requirements of DAGs, ensuring the learned graph adheres to the necessary acyclic properties.

However, the experiments also revealed certain limitations. The model struggled to recover the exact adjacency matrix due to challenges in thresholding continuous values to determine binary edges. This difficulty in selecting an appropriate threshold resulted in minor discrepancies between the learned graph and the true underlying structure. Additionally, small errors in the encoder and decoder architecture impacted the precision of the reconstructed graph, suggesting areas for refinement in the model's design.

Despite these challenges, DAG-GNN demonstrates significant promise as a framework for DAG structure learning. Its ability to enforce acyclicity while handling nonlinear and complex data relationships makes it a powerful tool for discovering meaningful graph structures. Improvements in thresholding strategies and encoder-decoder design will be crucial for enhancing the model's accuracy and applicability to real-world datasets. Overall, DAG-GNN represents a robust foundation for future research in structure learning, offering new possibilities for applications in causal inference, probabilistic modeling, and other domains requiring the analysis of complex dependency structures.

Reference

Yu, Y., Chen, J., & Yu, T. (2019). DAG-GNN: DAG structure learning with graph neural networks. Proceedings of the 36th International Conference on Machine Learning (ICML), 97, 7154–7163.

Zheng, X., Aragam, B., Ravikumar, P. K., & Xing, E. P. (2018). DAGs with NO TEARS: Continuous optimization for structure learning. *Advances in Neural Information Processing Systems, 31*, 9472–9483.