Supervised Whole DAG Causal Discovery

Hebi Li, Qi Xiao, Jin Tian
Iowa State University

Presented By:

Nitish Nagesh and Kushagra Pandey
University of California, Irvine

Mar 7, 2023

Outline

Motivation and Problem Setting

Causal Discovery

- Central Assumption: A Causal Graph is given for Causal Inference (Interventions, Counterfactuals etc)

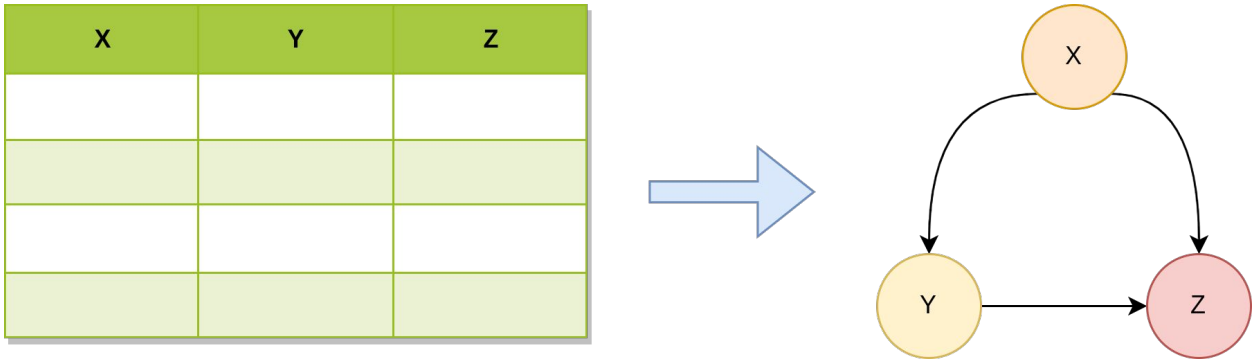
Causal Discovery

- Central Assumption: A Causal Graph is given for Causal Inference (Interventions, Counterfactuals etc)
- However, acquiring a Causal Graph can be expensive! (Can require a lot of domain specific experts, poor scalability)

Causal Discovery

- Central Assumption: A Causal Graph is given for Causal Inference (Interventions, Counterfactuals etc)

Can we learn the structure of the causal graph directly from observational data?



Motivation - Existing Methods and Trends

Existing Methods are expensive during inference and scale poorly

Motivation - Existing Methods and Trends

Existing Methods are expensive during inference and scale poorly

Algorithm 1 Greedy Search

Input: data $\boldsymbol{x}$, initial graph G
Compute $\mathrm{BIC}(G|\boldsymbol{x})$ and set $\mathrm{BIC}_\star = \mathrm{BIC}(G|\boldsymbol{x})$.
repeat
  Initialize $Improvement = false$.
  for all graphs G' reachable from G **do**
    Compute $\mathrm{BIC}(G'|\boldsymbol{x})$.
    if $\mathrm{BIC}(G'|\boldsymbol{x}) < \mathrm{BIC}_\star$ **then**
      Set $G = G'$ and $\mathrm{BIC}_\star = \mathrm{BIC}(G'|\boldsymbol{x})$
      Set $Improvement = true$.
    end if
  end for
until $Improvement$ is $false$
Output: graph G

Remember this?
Ordinal Causal Discovery

Motivation - Existing Methods and Trends

Existing Methods are expensive during inference and scale poorly

Algorithm 1 Greedy Search

Input: data $\boldsymbol{x}$, initial graph G
Compute $\mathrm{BIC}(G|\boldsymbol{x})$ and set $\mathrm{BIC}_\star = \mathrm{BIC}(G|\boldsymbol{x})$.
repeat
  Initialize $Improvement = false$.
  for all graphs G' reachable from G **do**
    Compute $\mathrm{BIC}(G'|\boldsymbol{x})$.
    if $\mathrm{BIC}(G'|\boldsymbol{x}) < \mathrm{BIC}_\star$ **then**
      Set $G = G'$ and $\mathrm{BIC}_\star = \mathrm{BIC}(G'|\boldsymbol{x})$
      Set $Improvement = true$.
    end if
  end for
until $Improvement$ is $false$
Output: graph G

Iterative Inference

Motivation - Existing Methods and Trends

Existing Methods are expensive during inference and scale poorly

Algorithm 1 Greedy Search

Input: data $\boldsymbol{x}$, initial graph G
Compute $\mathrm{BIC}(G|\boldsymbol{x})$ and set $\mathrm{BIC}_\star = \mathrm{BIC}(G|\boldsymbol{x})$.
repeat
  Initialize $Improvement = false$.
  for all graphs G' reachable from G **do**
    Compute $\mathrm{BIC}(G'|\boldsymbol{x})$.
    if $\mathrm{BIC}(G'|\boldsymbol{x}) < \mathrm{BIC}_\star$ **then**
      Set $G = G'$ and $\mathrm{BIC}_\star = \mathrm{BIC}(G'|\boldsymbol{x})$
      Set $Improvement = true$.
    end if
  end for
until $Improvement$ is $false$
Output: graph G

Iterative Inference

Search sensitive to the choice of “score”

Motivation - Existing Methods and Trends

Existing Methods are expensive during inference and scale poorly

Algorithm 1 Greedy Search

Input: data $\boldsymbol{x}$, initial graph G
Compute $\mathrm{BIC}(G|\boldsymbol{x})$ and set $\mathrm{BIC}_\star = \mathrm{BIC}(G|\boldsymbol{x})$.
repeat
  Initialize $Improvement = false$.
  for all graphs G' reachable from G **do**
    Compute $\mathrm{BIC}(G'|\boldsymbol{x})$.
    if $\mathrm{BIC}(G'|\boldsymbol{x}) < \mathrm{BIC}_\star$ **then**
      Set $G = G'$ and $\mathrm{BIC}_\star = \mathrm{BIC}(G'|\boldsymbol{x})$
      Set $Improvement = true$.
    end if
  end for
until $Improvement$ is $false$
Output: graph G

Score-Based Methods:

- Use a predefined score-function, such as AIC, BIC scores.
- Search in the space of all valid DAGs for the highest possible score via search algorithms.

Motivation - Existing Methods and Trends

Existing Methods are expensive during inference and scale poorly

Algorithm 1 Greedy Search

Input: data $\boldsymbol{x}$, initial graph G
Compute $\mathrm{BIC}(G|\boldsymbol{x})$ and set $\mathrm{BIC}_\star = \mathrm{BIC}(G|\boldsymbol{x})$.
repeat
  Initialize $Improvement = false$.
  for all graphs G' reachable from G **do**
    Compute $\mathrm{BIC}(G'|\boldsymbol{x})$.
    if $\mathrm{BIC}(G'|\boldsymbol{x}) < \mathrm{BIC}_\star$ **then**
      Set $G = G'$ and $\mathrm{BIC}_\star = \mathrm{BIC}(G'|\boldsymbol{x})$
      Set $Improvement = true$.
    end if
  end for
until $Improvement$ is $false$
Output: graph G

Pairwise Methods:

- Infer causal relationships between a pair of variables at a time
- Lack Global context!

Motivation - Existing Methods and Trends

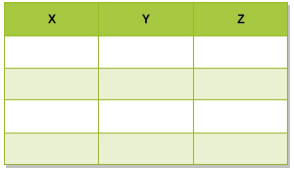
Existing Methods are expensive during inference and scale poorly

Algorithm 1 Greedy Search

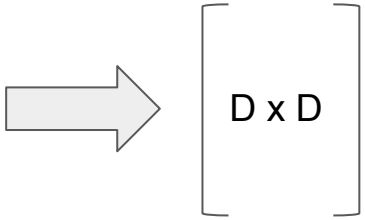
Input: data $\boldsymbol{x}$, initial graph G
Compute $\mathrm{BIC}(G|\boldsymbol{x})$ and set $\mathrm{BIC}_\star = \mathrm{BIC}(G|\boldsymbol{x})$.
repeat
  Initialize $Improvement = false$.
  for all graphs G' reachable from G **do**
    Compute $\mathrm{BIC}(G'|\boldsymbol{x})$.
    if $\mathrm{BIC}(G'|\boldsymbol{x}) < \mathrm{BIC}_\star$ **then**
      Set $G = G'$ and $\mathrm{BIC}_\star = \mathrm{BIC}(G'|\boldsymbol{x})$
      Set $Improvement = true$.
    end if
  end for
until $Improvement$ is $false$
Output: graph G

Need Causal Discovery methods which are scalable and in-expensive during inference

Motivation - Inference in Proposed Method

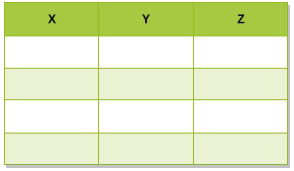


Observational Data (n x D)

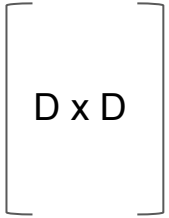


Features

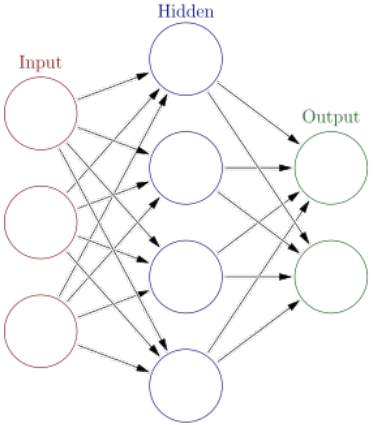
Motivation - Inference in Proposed Method



Observational Data (n x D)

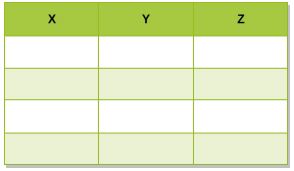


Features



Neural Net Estimator

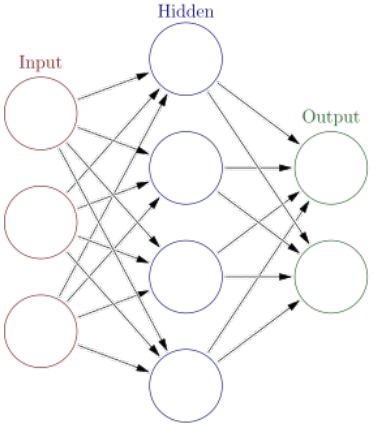
Motivation - Inference in Proposed Method



Observational Data (n x D)

D x D

Features

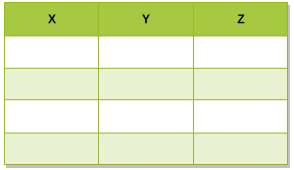


Neural Net Estimator

D x D

DAG Adjacency matrix

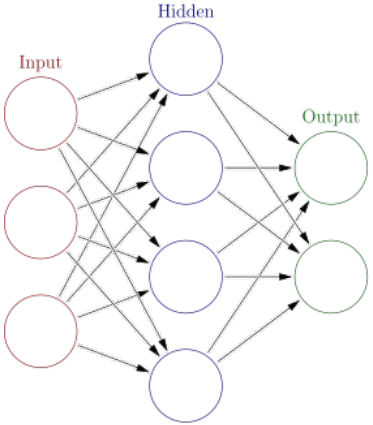
Motivation - Inference in Proposed Method



Observational Data (n x D)

D x D

Features



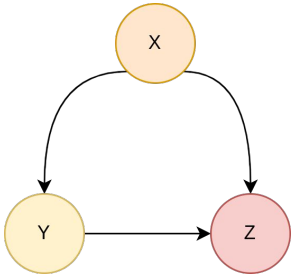
Neural Net Estimator

D x D

DAG Adjacency matrix

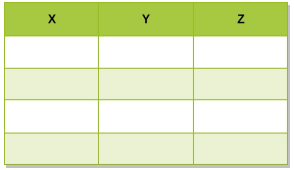
D x D

DAG Adjacency matrix



Causal DAG

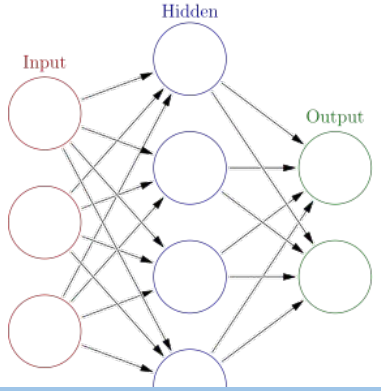
Motivation - Inference in Proposed Method



Observational Data (n x D)

D x D

Features



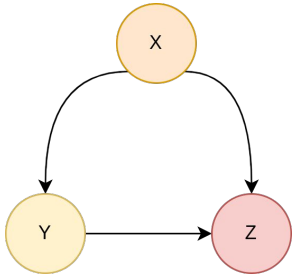
D x D

DAG Adjacency matrix

DAG output can be obtained in only a single forward pass from the neural network predictor. This is both in-expensive and scalable during inference.

D x D

DAG Adjacency matrix



Causal DAG

Formal Problem Setup

For a set of random variables $\{X_1, X_2, \ldots X_d\}$ where d is the number of variables, a functional causal model consists of a set of equations of the form:

$$X_i = f_i(pa_i, \epsilon_i)$$

where pa_i are the parents of X_i and ϵ_i is the contribution due to unobserved factors. We further assume the functions f_i to be linear, hence we consider Linear SCM's

Formal Problem Setup

For a set of random variables $\{X_1, X_2, \ldots X_d\}$ where d is the number of variables, a functional causal model consists of a set of equations of the form:

$$X_i = f_i(pa_i, \epsilon_i)$$

where pa_i are the parents of X_i and ϵ_i is the contribution due to unobserved factors. We further assume the functions f_i to be linear, hence we consider Linear SCM's

$$X_j = \sum_{i \neq j} c_{ji} X_i + \epsilon_j, \qquad j = 1, \ldots d$$

$$c_{ji} = 0 \;\text{ for }\; i < j$$

Formal Problem Setup

For a set of random variables $\{X_1, X_2, \ldots X_d\}$ where d is the number of variables, a functional causal model consists of a set of equations of the form:

$$X_i = f_i(pa_i, \epsilon_i)$$

where pa_i are the parents of X_i and ϵ_i is the contribution due to unobserved factors. We further assume the functions f_i to be linear, hence we consider Linear SCM's

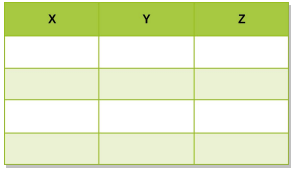
$$X_j = \sum_{i \neq j} c_{ji} X_i + \epsilon_j, \qquad j = 1, \ldots d$$

$$\boxed{c_{ji} = 0 \ \text{ for } \ i < j}$$

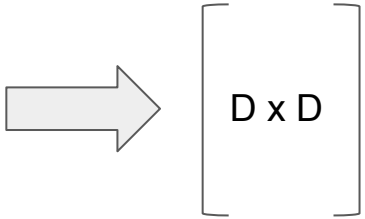
Essentially states that we are dealing with DAGs!

Methodology

Methodology - Feature Extraction



Observational Data (n x D)



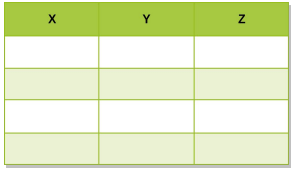
Features

Given a set of N observations, compute the Pearson Correlation coefficients between D variables.

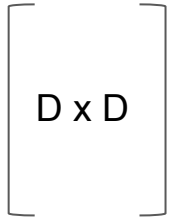
$$\rho_{X_i,X_j} = \frac{\mathrm{cov}(X_i,X_j)}{\sigma_{X_i}\sigma_{X_j}}$$

Proposed: Given a set of observations, compute the Pearson Correlation coefficients between the variables. This results in a D x D matrix which can be used as features for model training

Methodology - Feature Extraction



Observational Data (n x D)



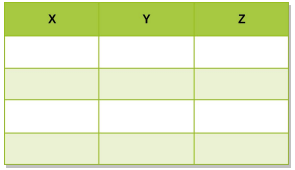
Features

Given a set of N observations, compute the Pearson Correlation coefficients between D variables.

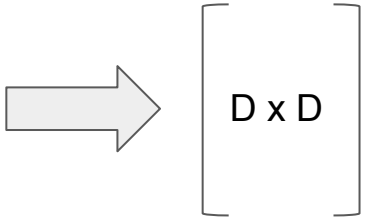
$$\rho_{X_i,X_j} = \frac{\boxed{\mathrm{cov}(X_i,X_j)}}{\sigma_{X_i}\sigma_{X_j}}$$

Using the covariance matrix alone is not a good proposal since cov(X,Y) is susceptible to the observed covariate values and thus can differ between different observed datasets. This limits transfer learning (More on this later!)

Methodology - Feature Extraction



Observational Data (n x D)



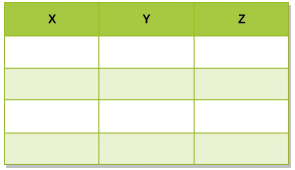
Features

Given a set of N observations, compute the Pearson Correlation coefficients between D variables.

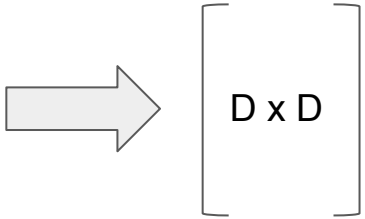
$$\rho_{X_i,X_j} = \frac{\mathrm{cov}(X_i,X_j)}{\boxed{\sigma_{X_i}\sigma_{X_j}}}$$

Normalizes cov(X,Y) and keeps the features between [-1, 1]

Methodology - Feature Extraction



Observational Data (n x D)



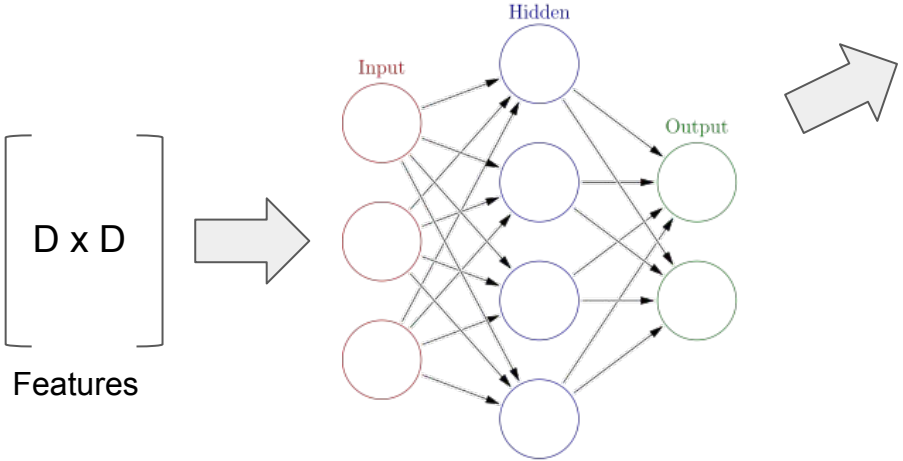
Features

Given a set of n observations, compute the Pearson Correlation coefficients between D variables.

$$\rho_{X_i,X_j} = \frac{\mathrm{cov}(X_i,X_j)}{\sigma_{X_i}\sigma_{X_j}}$$

Overall: From the n x D dimensional observed data we get D x D dimensional features.

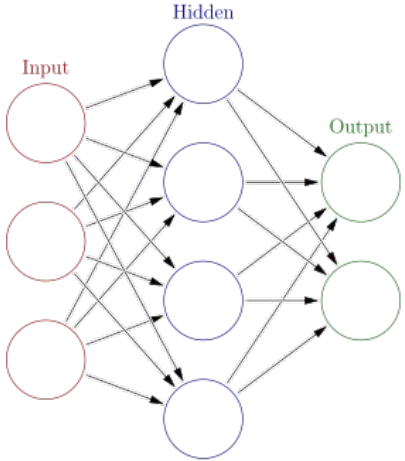
Methodology - Model specification



Multi-Layer Perceptrons
(MLPs)

Methodology - Model specification

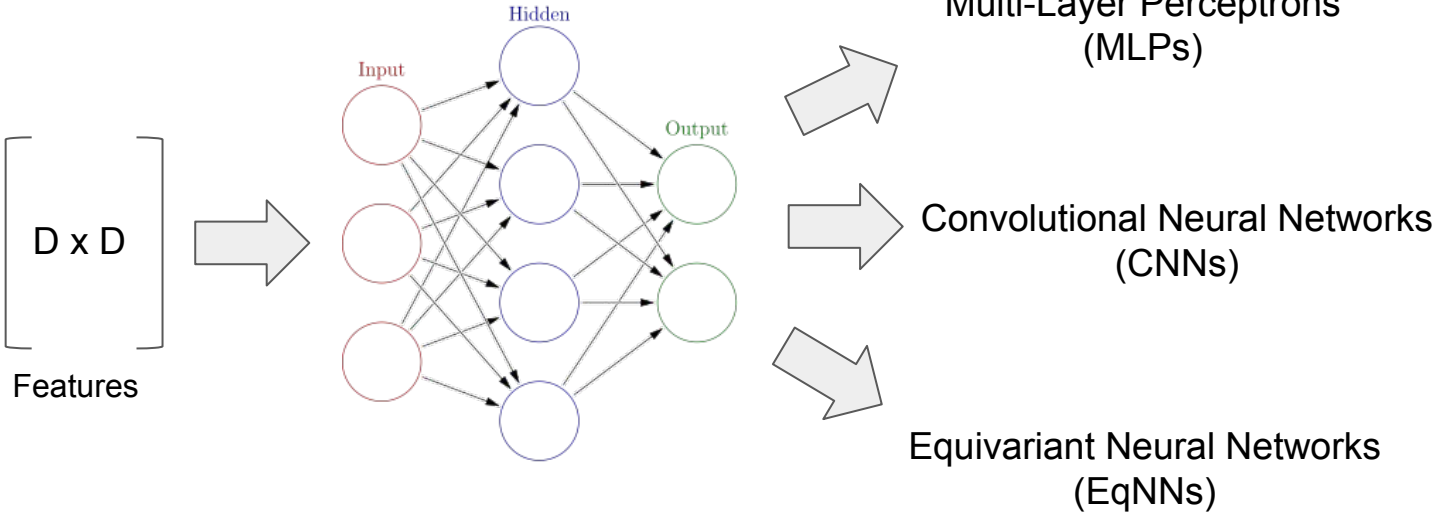
Features



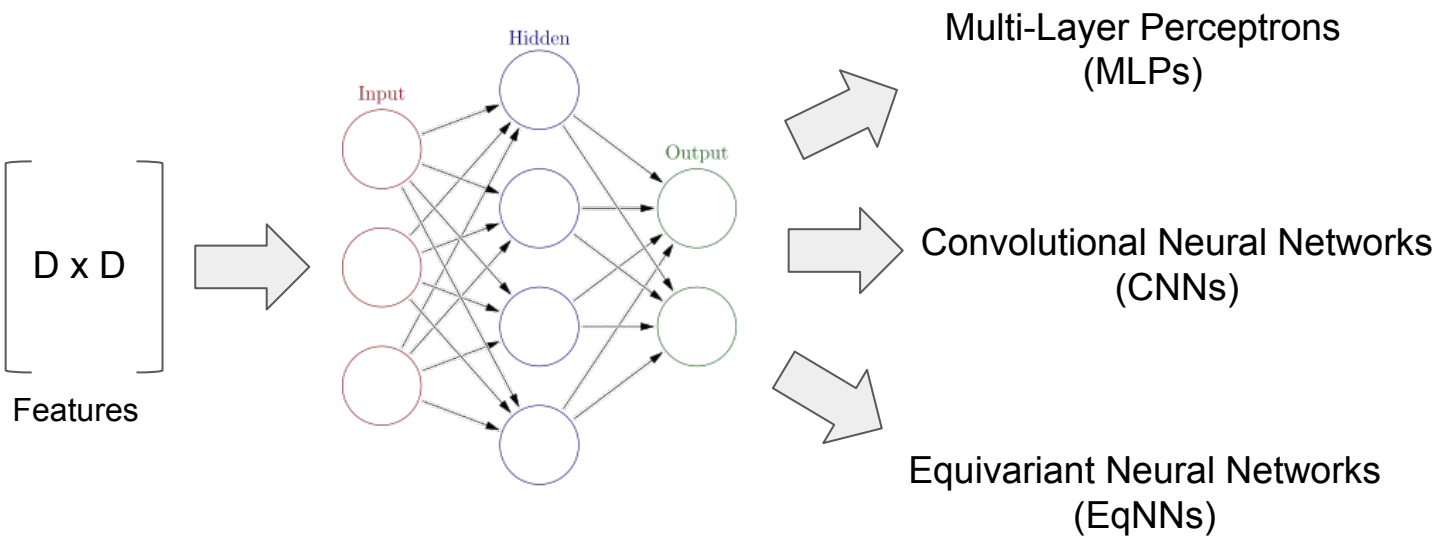
Multi-Layer Perceptrons (MLPs)

Convolutional Neural Networks (CNNs)

Methodology - Model specification

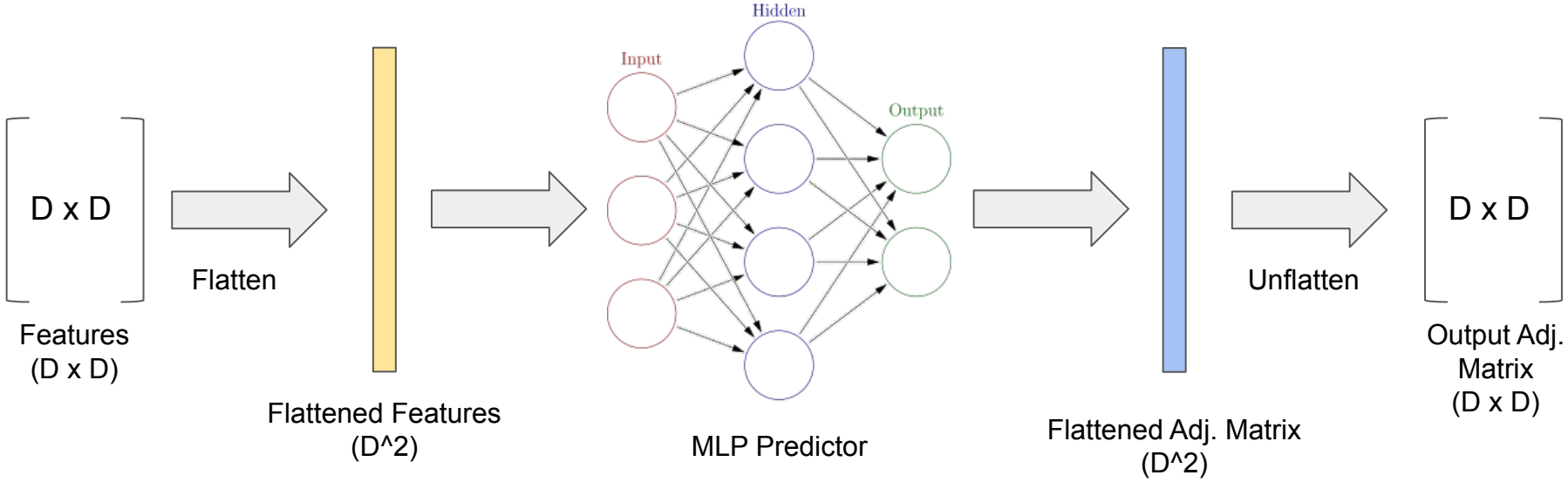


Methodology - Model specification

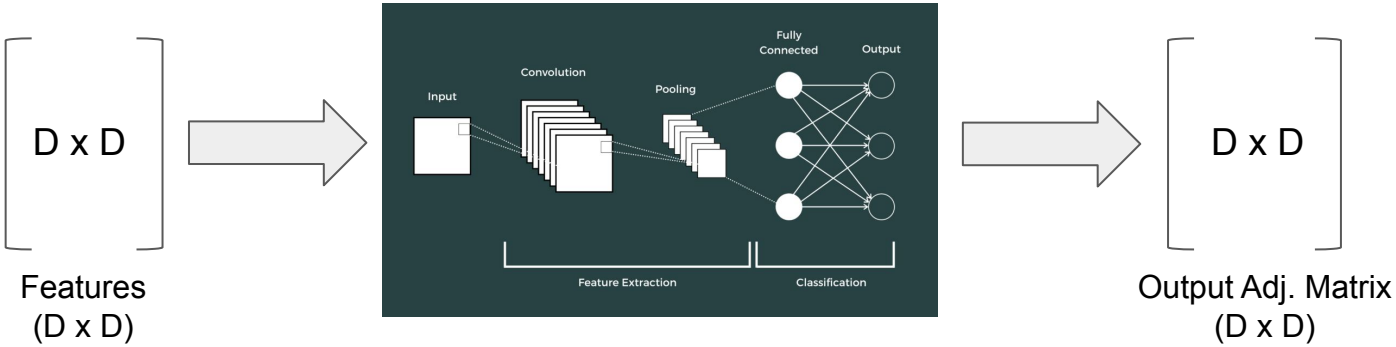


The choice of the neural network architecture impacts performance crucially!
(More on this later when discussing results)

Methodology - Proposal-1 - Using MLPs



Methodology - Proposal-2 - Use CNNs



Convolutional Net

Directly takes the feature matrix as input and outputs a corresponding Adj. Matrix

Image credits: https://www.theclickreader.com/building-a-convolutional-neural-network/

Methodology - Proposal-3 - Use Equivariant NNs

Key Observation: When the input variable order is permuted, the output adjacency matrix should change accordingly with the same permutation applied

Methodology - Proposal-3 - Use Equivariant NNs

Key Observation: When the input variable order is permuted, the output adjacency matrix should change accordingly with the same permutation applied

$$f_\theta(P^T \rho_{X,X} P) = P^T f_\theta(\rho_{X,X}) P$$

Methodology - Proposal-3 - Use Equivariant NNs

Key Observation: When the input variable order is permuted, the output adjacency matrix should change accordingly with the same permutation applied

$$f_\theta(\boxed{P}^T \rho_{X,X} P) = P^T f_\theta(\rho_{X,X}) P$$

Permutation matrix

Methodology - Proposal-3 - Use Equivariant NNs

Key Observation: When the input variable order is permuted, the output adjacency matrix should change accordingly with the same permutation applied

$$f_\theta(P^T \rho_{X,X} P) = P^T f_\theta(\rho_{X,X}) P$$

How to design the neural network layer so that it is permutation equivariant?

Methodology - Proposal-3 - Use Equivariant NNs

Theorem 3.2 *Hartford et al. (2018) Let $f = \sigma(W vec(X))$. f is an exchangeable matrix layer iff the elements of the parameter matrix W are tied together such that the resulting fully connected layer simplifies to*

$$Y = \sigma(w_1 X + w_2 \mathbf{1}\mathbf{1}^T X + w_3 X \mathbf{1}\mathbf{1}^{\mathbf{T}} + w_4 \mathbf{1}\mathbf{1}^T X \mathbf{1}\mathbf{1}^T + b) \tag{8}$$

where $\mathbf{1} = [1, \ldots, 1]^T$ *and* $w_1, \ldots, w_4, b \in \mathbb{R}$

Methodology - Proposal-3 - Use Equivariant NNs

Theorem 3.2 *Hartford et al. (2018) Let $f = \sigma(W vec(X))$. f is an exchangeable matrix layer iff the elements of the parameter matrix W are tied together such that the resulting fully connected layer simplifies to*

$$Y = \sigma(\boxed{w_1 X} + \boxed{w_2 \mathbf{1}\mathbf{1}^T X} + \boxed{w_3 X \mathbf{1}\mathbf{1}^{\mathbf{T}}} + \boxed{w_4 \mathbf{1}\mathbf{1}^T X \mathbf{1}\mathbf{1}^T} + b) \tag{8}$$

where $\mathbf{1} = [1, \ldots, 1]^T$ *and* $w_1, \ldots, w_4, b \in \mathbb{R}$

All these components are Permutation Equivariant. Therefore so is their sum!

Methodology - Proposal-3 - Use Equivariant NNs

Theorem 3.2 *Hartford et al. (2018) Let $f = \sigma(W vec(X))$. f is an exchangeable matrix layer iff the elements of the parameter matrix W are tied together such that the resulting fully connected layer simplifies to*

$$Y = \sigma(\boxed{w_1 X} + \boxed{w_2 \mathbf{1}\mathbf{1}^T X} + \boxed{w_3 X \mathbf{1}\mathbf{1}^T} + \boxed{w_4 \mathbf{1}\mathbf{1}^T X \mathbf{1}\mathbf{1}^T} + b) \tag{8}$$

where $\mathbf{1} = [1, \ldots, 1]^T$ *and* $w_1, \ldots, w_4, b \in \mathbb{R}$

This gives us a roadmap to design full neural networks which are Permutation Equivariant!

Methodology - Proposal-3 - Use Equivariant NNs

Theorem 3.2 *Hartford et al. (2018) Let $f = \sigma(W vec(X))$. f is an exchangeable matrix layer iff the elements of the parameter matrix W are tied together such that the resulting fully connected layer simplifies to*

$$Y = \sigma(\boxed{w_1 X} + \boxed{w_2 \mathbf{1}\mathbf{1}^T X} + \boxed{w_3 X \mathbf{1}\mathbf{1}^T} + \boxed{w_4 \mathbf{1}\mathbf{1}^T X \mathbf{1}\mathbf{1}^T} + b) \tag{8}$$

where $\mathbf{1} = [1, \ldots, 1]^T$ *and* $w_1, \ldots, w_4, b \in \mathbb{R}$

This gives us a roadmap to design full neural networks which are Permutation Equivariant!

Can introduce Permutation Equivariant channels analogous to CNNs

Methodology - Training

$$L = -\frac{1}{N} \boxed{\sum_{n=1}^{N}} \sum_{j=1}^{d} \sum_{i=1}^{d} \left[Y_{i,j}^{n} \cdot \log(\widehat{Y}_{i,j}^{n}) + (1 - Y_{i,j}^{n}) \cdot \log(1 - \widehat{Y}_{i,j}^{n}) \right]$$

N Training graphs

Methodology - Training

$$L = -\frac{1}{N}\sum_{n=1}^{N}\boxed{\sum_{j=1}^{d}\sum_{i=1}^{d}}\left[Y_{i,j}^{n}\cdot\log(\widehat{Y}_{i,j}^{n}) + (1 - Y_{i,j}^{n})\cdot\log(1 - \widehat{Y}_{i,j}^{n})\right]$$

D x D output edges

Methodology - Training

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{d}\sum_{i=1}^{d}\boxed{\left[Y_{i,j}^{n}\cdot\log(\widehat{Y}_{i,j}^{n}) + (1 - Y_{i,j}^{n})\cdot\log(1 - \widehat{Y}_{i,j}^{n})\right]}$$

Binary Cross Entropy loss for each edge

Methodology - Training

$$L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{d}\sum_{i=1}^{d}\left[Y_{i,j}^{n}\cdot\log(\widehat{Y}_{i,j}^{n}) + (1 - Y_{i,j}^{n})\cdot\log(1 - \widehat{Y}_{i,j}^{n})\right]$$

Simple Binary Cross-entropy loss applied to all the edges independently

Empirical Results

Evaluation Platform

Programming Language: Julia

Neural Network Library: Flux (Graph Neural Networks)

Source code and pre-trained models: https://github.com/lihebi/DAG-EQ

Desktop specs: Ryzen 3600 6-core 12-thread processors

Graphics card: Nvidia RTX2060-Super

4.1 Evaluation Setup

Evaluation Metrics

Precision $\frac{|\widehat{E} \cap E|}{|\widehat{E}|}$

$\widehat{E}$ set of predicted edges

Recall $\frac{|\widehat{E} \cap E|}{|E|}$

E set of true edges

Structural Hamming Distance (SHJD) - smallest number of edge additions, deletions and reversals to convert estimated graph into the true DAG

Synthetic Data Generation

Scale Free (SF) and Erdos-Renyi (ER) Graphs
Generate causal graphs close to reality

Number of edges = Number of nodes

Variable settings d = {10, 20, 50, 100}

Sample 1000 random DAGs
80% training, 20% testing

For each DAG, 1000 data points generated based on linear causal model
Linear causal model by generating coefficient matrix C_{ij}

$$C_{ij} = \begin{cases} \text{uniformly from}[-0.5-k, -0.5] \cup [0.5, 0.5+k] & \text{if } E_{i \to j} = \text{true} \\ 0 & \text{otherwise} \end{cases}$$

k = hyperparameter that controls scale of the weight matrix

Y_ij = binary adjacency matrix;
0 → edge not present, 1 → edge present

Model Details

Baseline Model:

- DAG-FC
  - 6 fully connected layers
  - Hidden layer size 1024
  - ReLU non-linear activation after each layer

Baseline Model:

- DAG-CNN
  - 6 CNN layers
  - 3x3 kernels with padding size 1
  - Hidden layer size 1024
  - ReLU non-linear activation and batch normalization after each convolutional layer

Proposed Model:

- DAG-EQ
  - 6 equivariant layers of hidden layer 300 channels
  - Leaky ReLU after each layer
  - Sigmoid activation at final layer for output in range [0,1]

Existing methods in comparison

<u>Pairwise supervised models:</u>
RCC = Randomized Causation Coefficient

1. RCC-RF = RCC with Random Forest Classifier
2. RCC-NN = RCC with fully Neural Network Classifier

<u>Structural learning methods:</u>

3. PC = constraint-based PC algorithm
4. GES = score-based Greedy Equivalence Score
5. CAM = Causal Additive Models

<u>State-of-the-art continuous optimization methods:</u>

6. NOTEARS = Non-combinatorial Optimization via Trace Exponential and Augmented lagRangian for Structure learning
7. DAG-GNN = Generative Models + Graph Neural Network
8. RL-BIC2 = Reinforcement Learning for graph search

4.2 Comparison with literature

DAG-EQ vs DAG-FC & DAG-CNN

Table 1: Comparison with literature, shown scale-free (SF) graph of size d=10,20,50,100

model	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	10	93.0	94.7	1.1	20	92.3	89.2	3.5
DAG-FC	10	81.7	79.8	3.4	20	49.0	41.6	19.3
DAG-CNN	10	88.7	87.7	2.1	20	82.0	78.4	7.4
RCC-RF	10	17.0	96.7	44.4	20	9.4	97.9	192.6
RCC-NN	10	18.6	68.9	33.2	20	11.9	75.3	122.5
PC	10	26.9	35.6	14.3	20	33.9	47.9	26.9
GES	10	19.5	30.0	15.6	20	19.1	26.3	34.3
CAM	10	7.5	25.6	35.5	20	6.6	30.5	94.3
NOTEARS	10	100.0	100.0	0.0	20	92.8	94.7	2.5
DAG-GNN	10	92.3	88.9	1.7	20	84.8	92.1	5.0
RL-BIC2	10	23.3	50.0	15.5	20	12.5	6.6	19.25
DAG-EQ	50	91.1	67.0	19.4	100	82.3	58.4	53.7
DAG-CNN	50	50.0	43.4	49.0	100	51.3	28.4	97.6
RCC-RF	50	3.5	78.8	1113.9	100	2.4	69.5	2800.8
RCC-NN	50	5.6	63.7	554.1	100	3.1	67.1	2130.8
PC	50	33.8	43.3	68.2	100	34.8	45.5	138.0
GES	50	10.7	15.9	104.6	100	6.9	10.9	233.2
CAM	50	7.6	36.7	249.8	100	7.2	34.3	509.0
NOTEARS	50	94.6	97.3	4.2	100	70.6	89.5	49.8
DAG-GNN	50	82.1	91.2	16.0	100	79.3	87.7	37.8

DAG-EQ better especially for larger sample size compared to DAG-FC and DAG-CNN

DAG-EQ vs Pairwise supervised models

Table 1: Comparison with literature, shown scale-free (SF) graph of size d=10,20,50,100

model	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	10	93.0	94.7	1.1	20	92.3	89.2	3.5
DAG-FC	10	81.7	79.8	3.4	20	49.0	41.6	19.3
DAG-CNN	10	88.7	87.7	2.1	20	82.0	78.4	7.4
RCC-RF	10	17.0	96.7	44.4	20	9.4	97.9	192.6
RCC-NN	10	18.6	68.9	33.2	20	11.9	75.3	122.5
PC	10	26.9	35.6	14.3	20	33.9	47.9	26.9
GES	10	19.5	30.0	15.6	20	19.1	26.3	34.3
CAM	10	7.5	25.6	35.5	20	6.6	30.5	94.3
NOTEARS	10	100.0	100.0	0.0	20	92.8	94.7	2.5
DAG-GNN	10	92.3	88.9	1.7	20	84.8	92.1	5.0
RL-BIC2	10	23.3	50.0	15.5	20	12.5	6.6	19.25
DAG-EQ	50	91.1	67.0	19.4	100	82.3	58.4	53.7
DAG-CNN	50	50.0	43.4	49.0	100	51.3	28.4	97.6
RCC-RF	50	3.5	78.8	1113.9	100	2.4	69.5	2800.8
RCC-NN	50	5.6	63.7	554.1	100	3.1	67.1	2130.8
PC	50	33.8	43.3	68.2	100	34.8	45.5	138.0
GES	50	10.7	15.9	104.6	100	6.9	10.9	233.2
CAM	50	7.6	36.7	249.8	100	7.2	34.3	509.0
NOTEARS	50	94.6	97.3	4.2	100	70.6	89.5	49.8
DAG-GNN	50	82.1	91.2	16.0	100	79.3	87.7	37.8

RCC-RF and RCC-NN not suited for predicting whole DAG – lack of distinction between direct and indirect effects

DAG-EQ vs Structural Learning Models

Table 1: Comparison with literature, shown scale-free (SF) graph of size d=10,20,50,100

model	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	10	93.0	94.7	1.1	20	92.3	89.2	3.5
DAG-FC	10	81.7	79.8	3.4	20	49.0	41.6	19.3
DAG-CNN	10	88.7	87.7	2.1	20	82.0	78.4	7.4
RCC-RF	10	17.0	96.7	44.4	20	9.4	97.9	192.6
RCC-NN	10	18.6	68.9	33.2	20	11.9	75.3	122.5
PC	10	26.9	35.6	14.3	20	33.9	47.9	26.9
GES	10	19.5	30.0	15.6	20	19.1	26.3	34.3
CAM	10	7.5	25.6	35.5	20	6.6	30.5	94.3
NOTEARS	10	100.0	100.0	0.0	20	92.8	94.7	2.5
DAG-GNN	10	92.3	88.9	1.7	20	84.8	92.1	5.0
RL-BIC2	10	23.3	50.0	15.5	20	12.5	6.6	19.25
DAG-EQ	50	91.1	67.0	19.4	100	82.3	58.4	53.7
DAG-CNN	50	50.0	43.4	49.0	100	51.3	28.4	97.6
RCC-RF	50	3.5	78.8	1113.9	100	2.4	69.5	2800.8
RCC-NN	50	5.6	63.7	554.1	100	3.1	67.1	2130.8
PC	50	33.8	43.3	68.2	100	34.8	45.5	138.0
GES	50	10.7	15.9	104.6	100	6.9	10.9	233.2
CAM	50	7.6	36.7	249.8	100	7.2	34.3	509.0
NOTEARS	50	94.6	97.3	4.2	100	70.6	89.5	49.8
DAG-GNN	50	82.1	91.2	16.0	100	79.3	87.7	37.8

Poor performance of constraint-based methods in large graphs

DAG-EQ vs STOA optimization methods

Table 1: Comparison with literature, shown scale-free (SF) graph of size d=10,20,50,100

model	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	10	93.0	94.7	1.1	20	92.3	89.2	3.5
DAG-FC	10	81.7	79.8	3.4	20	49.0	41.6	19.3
DAG-CNN	10	88.7	87.7	2.1	20	82.0	78.4	7.4
RCC-RF	10	17.0	96.7	44.4	20	9.4	97.9	192.6
RCC-NN	10	18.6	68.9	33.2	20	11.9	75.3	122.5
PC	10	26.9	35.6	14.3	20	33.9	47.9	26.9
GES	10	19.5	30.0	15.6	20	19.1	26.3	34.3
CAM	10	7.5	25.6	35.5	20	6.6	30.5	94.3
NOTEARS	10	100.0	100.0	0.0	20	92.8	94.7	2.5
DAG-GNN	10	92.3	88.9	1.7	20	84.8	92.1	5.0
RL-BIC2	10	23.3	50.0	15.5	20	12.5	6.6	19.25
DAG-EQ	50	91.1	67.0	19.4	100	82.3	58.4	53.7
DAG-CNN	50	50.0	43.4	49.0	100	51.3	28.4	97.6
RCC-RF	50	3.5	78.8	1113.9	100	2.4	69.5	2800.8
RCC-NN	50	5.6	63.7	554.1	100	3.1	67.1	2130.8
PC	50	33.8	43.3	68.2	100	34.8	45.5	138.0
GES	50	10.7	15.9	104.6	100	6.9	10.9	233.2
CAM	50	7.6	36.7	249.8	100	7.2	34.3	509.0
NOTEARS	50	94.6	97.3	4.2	100	70.6	89.5	49.8
DAG-GNN	50	82.1	91.2	16.0	100	79.3	87.7	37.8

DAG-EQ performance below SOTA NOTEARS for large d = 100

4.3 Transferability with Ensemble Learning

DAG-EQ trained on different noise models

Transferability: Applying results from one domain to another domain

model	train_noise	test_noise	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	Gaussian	Gaussian	10	87.2	96.9	1.6	20	86.4	93.9	4.0
DAG-EQ	Gaussian	Exp	10	92.1	94.8	1.2	20	92.4	87.5	3.7
DAG-EQ	Gaussian	Gumbel	10	92.1	94.4	1.2	20	92.4	87.6	3.7
DAG-EQ	Gaussian	Poisson	10	92.3	94.9	1.2	20	92.4	87.4	3.8

Good performance while testing across different Gaussian and non-Gaussian models

Transferability between causal strengths

model	train/k	test/k	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	1	1	10	87.2	96.9	1.6	20	86.4	93.9	4.0
DAG-EQ	1	2	10	79.4	91.2	2.9	20	75.2	80.9	8.7
DAG-EQ	1	4	10	64.7	72.3	6.1	20	57.9	61.5	15.8

Causal coefficient weight distribution range [-0.5-k, -0.5] U [0.5, 0.5+k]
Limited transferability compared to noise level
Imperfect alignment between training and test distribution

Transferability between different types of graphs

comparable performance

model	train_gtype	test_gtype	d	prec	recall	shd	d	prec	recall	shd
DAG-EQ	SF	SF	10	87.2	96.9	1.6	20	86.4	93.9	4.0
DAG-EQ	ER	ER	10	72.5	84.1	4.5	20	72.1	80.3	9.9
DAG-EQ	ER	SF	10	82.0	95.4	2.3	20	80.9	84.7	6.7
DAG-EQ	SF	ER	10	69.0	70.9	5.8	20	72.7	62.1	11.9

Transferring to very large graphs

model	d	Pred	TP	prec	recall	shd	d	Pred	TP	prec	recall	shd
DAG-EQ	100	70	58	82.3	58.4	53.7	200	404	152	37.5	75.5	302.17
GES	100	169	13	7.6	13.13	242	200	431	19	4.4	9.5	592
DAG-EQ	300	1211	232	19.1	75.5	1058.7	400	2984	330	11.1	73.5	2796.3
GES	300	742	19	3.7	9.2	980.8	400	1148	28	4.1	11.6	1443.2

Training on d=100, testing on d = 200, 300, 400
As sample size increases, DAG-EQ maintains good precision and recall.
For d=400, recall is high but precision is low due to higher number of positive predictions

Overall, GES – lower precision and recall
For d=300 and d=400, GES lower SHD; smaller number of edges predicted compared to DAG-EQ.

Ensemble training

model	test_d	test_gtype	prec	recall	shd	test_gtype	prec	recall	shd
DAG-EQ-Ensemble	10	SF	89.1	95.3	1.5	ER	72.1	66.9	5.6
DAG-EQ-Ensemble	15	SF	88.2	95.7	2.4	ER	74.3	67.5	8.1
DAG-EQ-Ensemble	20	SF	84.8	94.7	4.2	ER	72.0	66.0	11.6
DAG-EQ-Ensemble	30	SF	75.1	93.3	10.9	ER	68.1	68.9	18.7
DAG-EQ-Ensemble	50	SF	58.8	90.4	35.9	ER	56.0	73.3	41.8
DAG-EQ-Ensemble	80	SF	43.4	88.6	100.2	ER	44.5	79.1	95.3

Ensemble training on SF graphs with d = 10, 15, 20
Testing on SF & ER graphs with d = 10, 15, 20, 30, 50, 80
Model trained on small graphs works well while tested on larger graphs

4.4 Real Data Experiment

Protein Signal Network [Sachs et al. 2005]

<u>High level dataset overview:</u>

- 853 samples
- ground truth graph 11 nodes and 17 edges
- Ensemble trained DAG-EQ applied

model	predicted edges	correct edges	SHD
DAG-EQ	10	5	16
NOTEARS	20	6	19
RL-BIC2	10	7	11
CAM	10	6	12
DAG-GNN	15	6	16

5. Our Perspective

Summary

- Supervised DAG-EQ scales well to large sample size by training on smaller size
  - faster compute time
  - higher efficiency
  - storage savings
- Novel transferability with mechanism with ensemble learning for synthetic data
  - scalable
  - aids in validation
- DAG-EQ has slightly lower performance compared to state-of-the-art NOTEARS but better than pairwise supervised models like RCC-RF & RCC-NN
  - model fine tuning
  - exploring graph neural networks
- More extensive performance metric evaluation on real-world dataset
  - precision, recall and SHD analysis
- Transitioning from linear causal models to non-linear causal models

Question for the audience

How do you envision integrating supervised learning and causal discovery for real-world datasets and/or your research?

Reach us

Nitish Nagesh nnagesh1@uci.edu
Kushagra Pandey pandeyk1@uci.edu