Causal Machine Learning

Presenter: Zhe Wang

https://qdata.github.io/deep2Read

Zhe Wang

201909

Content

Motivation

Weakness of machine learning algorithm:

- Existence of adversarial samples
- No transfer learning ability
- No fast adaptation ability
- Lack of interpretation

NN should use high-level semantic features to perform the task.

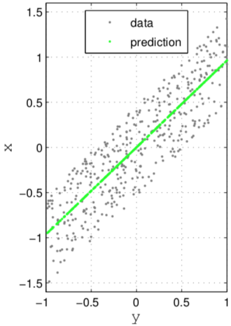
Unfortunately, NN is using redundant features to improve the accuracy.[Arjovsky et.al 2019][Wang et.al 2017]

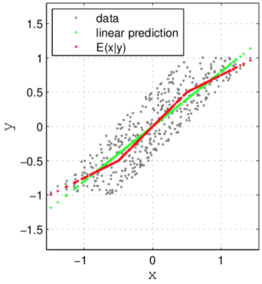
Solution: Use causal features only and rule out all spurious features.

- Avoid Over-Parametrization[Neyshabur et.al 2019].
- Feature selection with expert knowledge: sparsity, low rank.

Possible to explicitly learn causal features?
Yes, via interventions and hypothesis testing.

Example: Suppose we have data sampled from $P(X, Y)$. Given X, it is easy to predict Y. But, what if I ask who is the cause, who is the effect.





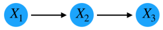
Causal machine learning algorithms require the understanding of dependence and independence.

In deep learning scenario?

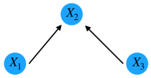
- Learn the causal graph
- Rule out spurious correlations.

Content

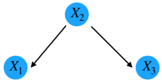
For a joint distribution $P(x_1, x_2, \cdots, x_k)$, there is an underlying (acyclic) causal graph.



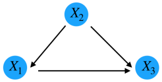
1: chain



2: collider



3: fork



4: confounder

- $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2, p_1)$
- $p(x_1, x_2, x_3) = p(x_1)p(x_3)p(x_2|x_3, p_1)$
- $p(x_1, x_2, x_3) = p(x_2)p(x_1|x_2)p(x_3|x_2)$

Markov property

Given a causal DAG, a variable X is independent of all its non-descendants given its parents.

The property tells us, the joint distribution can be factorized into small groups.
Assumption:

- For human, high level knowledge are organized as small groups. [Bengio 2017]
- When there is a domain shift, only limited numbers of distributions are shifted, most remain invariant. [Schölkopf et.al 2018]

Content

Given n observations of d nodes, then the DAG $|G| = n \times n$. Suppose G is induced by a weighted adjacent matrix W, which encodes the coefficients of SEM. $W_{ij} = 0 \iff G_{ij} = 0$
Because x_i is generated by its own parent nodes, which means $X = WX + \epsilon$, with the constraints $G \in DAG$. Thus the optimization problem is a combinatorial search:

$$\min_{W \in R^{d \times d}} ||X - WX||_F^2 + \lambda ||W||_1 \tag{1}$$

$$s.t. \quad G \in DAGs \tag{2}$$

What does it mean by $G \in DAGs$?

Theorem

[Zhang et.al 2018] A weighted adjacency matrix W induces a DAG if and only if:

$$h(W) = tr(e^{W \cdot W}) - d = 0. \tag{3}$$

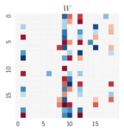
The combinatorial optimization now is equivalent to the continuous problem:

$$\min_{W \in R^{d \times d}} ||X - WX||_F^2 + \lambda ||W||_1 \tag{4}$$

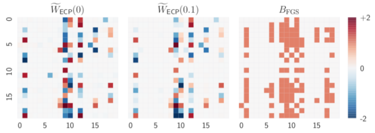
$$s.t. \quad h(W) = 0 \tag{5}$$

With this work, the learning of DAG via NN becomes possible.

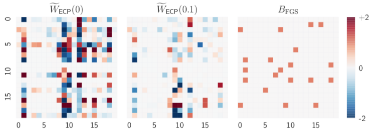
The recovery of weighted adjacency matrix.



(a) true graph



(b) estimate with $n = 1000$



(c) estimate with $n = 20$

Gradient-based neural DAG learning[1]

Use neural network to learn the DAG. Based on Markov property: $P(x) = \prod_{j=1}^{d} p(x_j|Pa(x_j))$ The observation matrix X can be recovered by itself $X = WX$. But this model gives a linear reconstruction.
The work proposes to use a mlp for each node.
For node j, the mlp is noted as the input will be X_{-j}:

$$\theta_j = W_j^{l+1} g(\cdots g(w_j^2 g(W_j^1 X_{-j}))) \quad j = 1, 2, 3, \cdots d \tag{6}$$

With the NN, the distribution of each node is modeled as $p(x_j|x_{-j}, \phi_j)$, but $P(x) = \prod_{j=1}^{d} p(x_j|x_{-j}, \phi_j)$ is not a distribution.
Also how to force the constraint?

[1] Lachapelle et.al ICLR 2020

Suppose there are only two hidden layers, and $(W^1_{ih}, W^2_{hk}, W^3_{km})$ will be a message passing path from node i to node j. It is zero iff the path is inactive.

All paths will be:

$$C_{im} = \sum_h \sum_k |W^1_{ih}||W^2_{hk}||W^3_{km}| \tag{7}$$

C_{im} defines the strength of dependence between m_{th} component of x_j and x_i.
In order to let x_j be independent of x_i:

$$\sum_m C_{im} = 0 \tag{8}$$

Now the weight adjacency matrix can be defined as :

$$A_{ij} = \begin{cases} \sum_m C_{im}^{j}, & i \neq j, \\ 0, & i = j \end{cases} \tag{9}$$

With the adjacency matrix, the constraints is:

$$h(W) = tr(e^{W \cdot W}) - d = 0. \tag{10}$$

Table 1: Results for ER and SF graphs of 10 nodes with Gauss-ANM data

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**1.7±2.5**	**1.7±3.1**	**8.3±2.8**	**21.8±8.9**	**1.2±1.1**	**4.1±6.1**	**9.9±4.0**	**16.4±6.0**
DAG-GNN	11.4±3.1	37.6±14.4	35.1±1.5	81.9±4.7	9.9±1.1	29.7±15.8	20.8±1.9	48.4±15.6
NOTEARS	12.2±2.9	36.6±13.1	32.6±3.2	79.0±4.1	10.7±2.2	32.0±15.3	20.8±2.7	49.8±15.6
CAM	**1.1±1.1**	**1.1±2.4**	**12.2±2.7**	**30.9±13.2**	**1.4±1.6**	**5.4±6.1**	**9.8±4.3**	**19.3±7.5**
GSF	6.5±2.6	[6.2±10.8 17.7±12.3]	21.7±8.4	[37.2±19.2 62.7±14.9]	**1.8±1.7**	**[2.0±5.1 6.9±6.2]**	**8.5±4.2**	**[13.2±6.8 20.6±12.1]**
RANDOM	26.3±9.8	25.8±10.4	31.8±5.0	76.6±7.0	25.1±10.2	24.5±10.5	28.5±4.0	47.2±12.2

Table 2: Results for ER and SF graphs of 50 nodes with Gauss-ANM data

	ER1		ER4		SF1		SF4	
	SHD	SID	SHD	SID	SHD	SID	SHD	SID
GraN-DAG	**5.1±2.8**	**22.4±17.8**	**102.6±21.2**	**1060.1±109.4**	**25.5±6.2**	**90.0±18.9**	**111.3±12.3**	**271.2±65.4**
DAG-GNN	49.2±7.9	304.4±105.1	191.9±15.2	2146.2±64	49.8±1.3	182.8±42.9	144.9±13.3	540.8±151.1
NOTEARS	62.8±9.2	327.3±119.9	202.3±14.3	2149.1±76.3	57.7±3.5	195.7±54.9	153.7±11.8	558.4±153.5
CAM	**4.3±1.9**	**22.0±17.9**	**98.8±20.7**	**1197.2±125.9**	**24.1±6.2**	**85.7±31.9**	**111.2±13.3**	**320.7±152.6**
GSF	25.6±5.1	[21.1±23.1 79.2±33.5]	**81.8±18.8**	**[906.6±214.7 1030.2±172.6]**	**31.6±6.7**	[85.8±29.9 147.3±49.9]	**120.2±10.9**	[284.7±80.2 379.9±98.3]
RANDOM	535.7±401.2	272.3±125.5	708.4±234.4	1921.3±203.5	514.0±360.0	381.3±190.3	660.6±194.9	1198.9±304.6

A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms[2]

For two variables X, Y, analyzing who is the effect and who is the cause.

- $P(X, Y) = P(X)P(Y|X)$
- $P(X, Y) = P(Y)P(X|Y)$

Two distributions are independent.

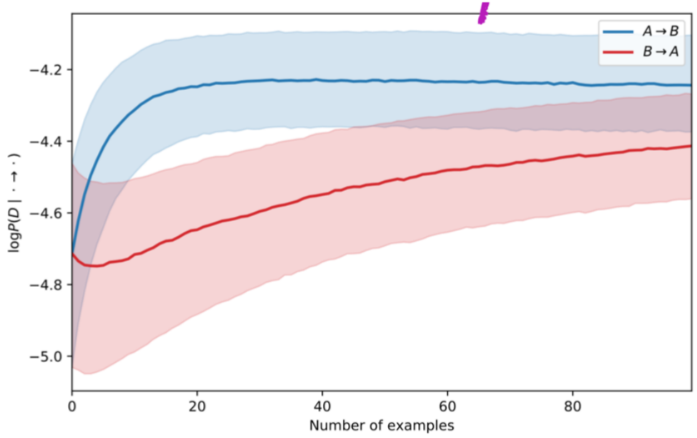
Transfer learning setting:

- source domain (X_s, Y_s)
- target domain (X_t, Y_t)

Suppose the true graph is $X \rightarrow Y$, and $P(Y|X)$ remains invariant for these two domains.

[2]Bengio et.al ICLR 2020

Observation: if model f in source domain learns $X \rightarrow Y$, it can fast adapted to the target domain.



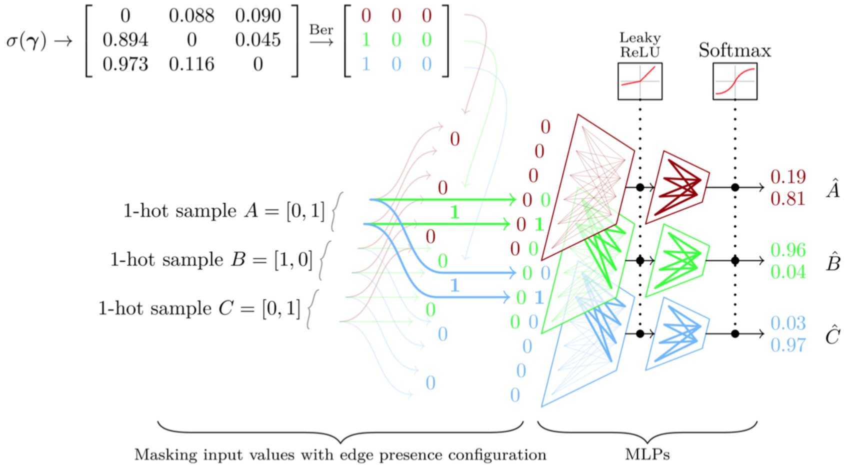
A→B is the correct causal structure: faster online adaptation to modified distribution = lower NLL regret

Meta learning objective:

$$\mathcal{R} = -\log[\sigma(\gamma)L_{A\to B} + (1-\sigma(\gamma))L_{B\to A}]$$

The inner loop update model parameters, the outer loop learns γ. And finally, $\sigma(\gamma) \to 1$.

For multiple variables, it is impossible to enumerate all possible causal structures and compare adaptation speed.
How to parametrize all possible graphs efficiently? For graph structure, there is a belief distribution controlled by $Ber(\sigma(\gamma_{ij}))$



$\sigma(\gamma) \rightarrow \begin{bmatrix} 0 & 0.088 & 0.090 \\ 0.894 & 0 & 0.045 \\ 0.973 & 0.116 & 0 \end{bmatrix} \xrightarrow{\text{Ber}} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
Leaky ReLU
Softmax
1-hot sample $A = [0, 1]$
1-hot sample $B = [1, 0]$
1-hot sample $C = [0, 1]$
0.19
0.81
$\hat{A}$
0.96
0.04
$\hat{B}$
0.03
0.97
$\hat{C}$
Masking input values with edge presence configuration
MLPs

Experiments

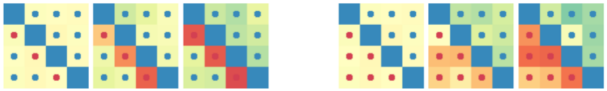


Figure 2: Learned edges at three different stages of training. **Left**: Chain graph with 4 variables. **Right**: Fully-connected DAG graph with 4 variables.

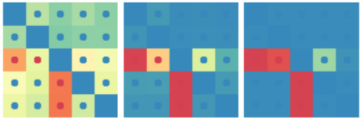


Figure 3: Earthquake: Learned edges at three different stages of training.

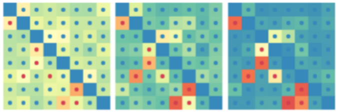
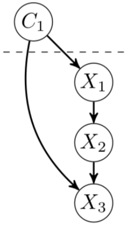


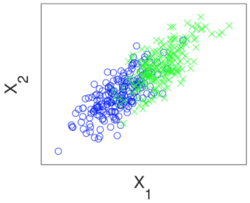
Figure 4: Asia: Learned edges at three different stages of training.

Content

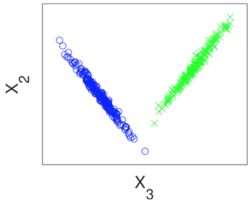
Invariant features will be more reliable in transfer learning.[Magliacane et.al 2018]
Suppose we use variable c to represent the training and test envs.



(a) Causal graph



(b) No distribution shift for $\{X_1\}$: $\mathbb{P}(Y \mid X_1, C_1 = 0) = \mathbb{P}(Y \mid X_1, C_1 = 1)$



(c) Strong distribution shift for $\{X_3\}$: $\mathbb{P}(Y \mid X_3, C_1 = 0) \neq \mathbb{P}(Y \mid X_3, C_1 = 1)$

Which variable will be used to predict x_2 in blue environment?
Given the causal graph, which variable should be used?

separation set

Suppose V is the set of nodes, invariant features $X \in V$ are defined as:

$$C \perp\!\!\!\perp Y | X \tag{11}$$

If such set exist, distribution of Y conditional on A is invariant under transferring from the source domains to the target domains, i.e., $P(Y|X, C=0) = P(Y|X, C=1)$

Suppose l_2 loss is used:

$$Y_A^1(a) = E(Y|A=a, C=1) \tag{12}$$
$$Y_A^0(a) = E(Y|A=a, C=0) \tag{13}$$

Total bias when using Y_A^0 to do prediction on $C=1$ can be decomposed as:

$$Y^1_{V-\{C,Y\}} - Y^0_A = (Y^1_A - Y^0_A) + (Y^1_{V-\{C,Y\}} - Y^1_A). \tag{14}$$

Thus in order to do transfer learning, we should always choose from the subset of invariant features.
How to identify invariant features:

- Statistical testing
- IRM and some extension work.

Statistical Testing

Goal: $P(Y|X, C = 0) = P(Y|X, C = 1)$

If we know the true causal DAG, then it is easy to identify causal features.

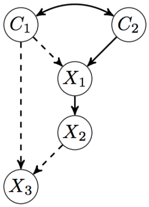
A more challenging setting: DAG and target variable in $C = 1$ are both unknown.

Example: Two context variables C_1, C_2, and three system variables X_1, X_2, X_3.If

$$C_2 \perp\!\!\!\perp X_2|X_1[C_1 = 0],\ C_2X_2[C_1 = 0],\ C_2 \perp\!\!\!\perp X_3|X2[C_1 = 0]$$

.

Then, $\{X_1\}$ is a separation set for X_2 and C_1.



Invariant risk minimization

How to find invariant features $p(y|x, c = 0) = p(y|x, c = 1)$.
Assumption: If $p(y|x, c = 0) = p(y|x, c = 1)$, they should have the same optimal classifier.
Optimization target:

$$\arg\min_{\Phi, w} \sum_e R_e(w^T \Phi(X^e)) \tag{15}$$

$$s.t. \quad w \in \arg\min_{\hat{w}} R^e(\hat{w}^T \phi(X^e)), \forall e \tag{16}$$

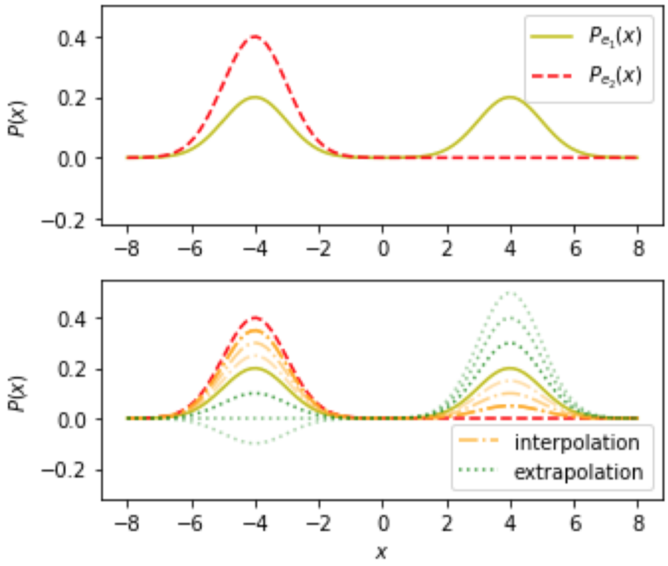
For simplicity, they use a fixed dummy classifier w

$$\arg\min_{\phi} \sum_e R_e(w_0^T \Phi(X^e)) + \lambda ||\nabla_{w_0} R_e(w_0^T \Phi(X^e))||_2^2 \tag{17}$$

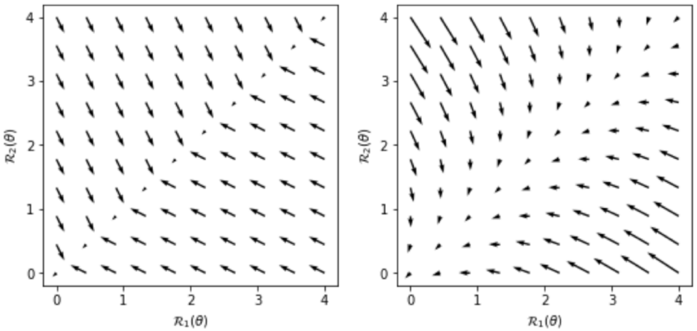
One extension work recently is to bridge robust optimization with IRM. IRM minimize the sum of all loss, robust optimization proposes to optimize the worst case only. Thus the loss function becomes:

$$(1 + m\beta) \max_e R_e(\theta) - \beta \sum_e R_e(w_0^T \Phi(X^e)) \tag{18}$$

If $\beta \rightarrow \infty$, it is forcing each environment to have the same loss.
If $\beta \rightarrow 0$, it degenerates to robust optimization.


$P_{e_1}(x)$
$P_{e_2}(x)$
interpolation
extrapolation
$P(x)$
x
0.4
0.2
0.0
−0.2
−8
−6
−4
−2
0
2
4
6
8

$\nabla \mathcal{R}_{\text{total}}$


$\mathcal{R}_2(\theta)$
$\mathcal{R}_1(\theta)$
$\mathcal{R}_2(\theta)$
$\mathcal{R}_1(\theta)$