causality_04

February 20, 2025

Pupipat Singkhorn, 6532142421

[1]:
```
# !pip install dowhy gcastle
```

[2]:
```
import numpy as np
np.random.seed(42)
import pandas as pd
import networkx as nx
import statsmodels.api as sm
from dowhy import gcm
from dowhy import CausalModel
import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
# Sample Data
S = np.random.normal(loc=0.5, scale=1, size=2000)
X = 2*S + np.random.normal(loc=0, scale=0.5, size=2000)
Y = 3*S + 1.5*X + np.random.normal(loc=1,scale=0.5, size=2000)
```

1 Interventional calculation

[4]:
```
# Estimate causal effect using back-door criterion
# S -> X -> Y
#  \-------^

data = pd.DataFrame({'X': X, 'S': S})
data = sm.add_constant(data, prepend=True)
model = sm.OLS(Y, data) # Ordinary Linear Regression
results = model.fit()
results.summary()
```

[4]:

Dep. Variable:	y	**R-squared:**	0.993
Model:	OLS	**Adj. R-squared:**	0.993
Method:	Least Squares	**F-statistic:**	1.424e+05
Date:	Thu, 20 Feb 2025	**Prob (F-statistic):**	0.00
Time:	13:07:02	**Log-Likelihood:**	-1449.1
No. Observations:	2000	**AIC:**	2904.
Df Residuals:	1997	**BIC:**	2921.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	0.9770	0.013	76.558	0.000	0.952	1.002
X	1.5356	0.022	69.045	0.000	1.492	1.579
S	2.9272	0.046	64.052	0.000	2.838	3.017

Omnibus:	1.385	**Durbin-Watson:**	2.061
Prob(Omnibus):	0.500	**Jarque-Bera (JB):**	1.303
Skew:	-0.036	**Prob(JB):**	0.521
Kurtosis:	3.102	**Cond. No.**	11.8

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[5]:
```
# Calculate P(Y|do(X=1))

X1 = np.array([1.0]*len(X))
Y1 = results.params['X']*X1 + results.params['S']*S + results.params['const']
np.average(Y1)
```

[5]:
4.10815744946464

[6]:
```
# Calculating Intervention using DoWhy GCM Model
# https://www.pywhy.org/dowhy/v0.12/user_guide/causal_tasks/what_if/
 ↪interventions.html
# Hint: you can also use gcm.auto.assign_causal_mechanisms (see: https://www.
 ↪pywhy.org/dowhy/v0.10.1/user_guide/modeling_gcm/draw_samples.html)

# PROBLEM 1 IMPLEMENT INTERVENTION USING DoWHY GCM MODEL
# TODO: [ YOUR CODE HERE ]

data_gcm = pd.DataFrame({"X": X, "S": S, "Y": Y})

# Define the causal model structure (S -> X -> Y and S -> Y)
causal_model = gcm.ProbabilisticCausalModel(
    nx.DiGraph([("S", "X"),
                ("X", "Y"),
                ("S", "Y")])
)
gcm.auto.assign_causal_mechanisms(causal_model, data_gcm)
```

```
gcm.fit(causal_model, data_gcm)

# Perform intervention: set X=1 and generate interventional data
samples = gcm.interventional_samples(causal_model,
                                     {'X': lambda x: 1},
                                     num_samples_to_draw=1000)
print(samples.head())

# Calculate the average Y under intervention
average_y_intervention = samples["Y"].mean()
print(f"Average Y under do(X=1): {average_y_intervention:.4f}")
```

```
Fitting causal mechanism of node Y: 100%|      | 3/3 [00:00<00:00,
333.96it/s]

          S  X         Y
0  0.846448  1  5.320758
1  0.697911  1  4.723199
2 -1.112716  1 -0.973577
3  1.062969  1  4.052086
4  2.139965  1  8.552281
Average Y under do(X=1): 4.1934
```

2 Counterfactual calculation

[7]:
```
data_gcm2 = pd.DataFrame({'X': X, 'S': S, 'Y': Y})
causal_model2 = gcm.InvertibleStructuralCausalModel(nx.DiGraph([('X', 'Y'),␣
 ↪('S', 'X'), ('S', 'Y')]))
gcm.auto.assign_causal_mechanisms(causal_model2, data_gcm2)
gcm.fit(causal_model2, data_gcm2)

gcm.counterfactual_samples(
    causal_model2,
    {'X': lambda x: 1},
    observed_data=pd.DataFrame(data=dict(X=[0], Y=[2], S=[0.5])))
```

```
Fitting causal mechanism of node S: 100%|      | 3/3 [00:00<00:00,
827.12it/s]
```

[7]:
```
     S  X         Y
0  0.5  1  3.535564
```

3 Causal Discovery

[8]:
```
from castle.algorithms import PC, ICALiNGAM
```

```
2025-02-20 13:07:10,200 -
/Users/pupipatsingkhorn/miniconda3/envs/datascience/lib/python3.11/site-
packages/castle/backend/__init__.py[line:36] - INFO: You can use
`os.environ['CASTLE_BACKEND'] = backend` to set the backend(`pytorch` or
`mindspore`).
2025-02-20 13:07:10,221 -
/Users/pupipatsingkhorn/miniconda3/envs/datascience/lib/python3.11/site-
packages/castle/algorithms/__init__.py[line:36] - INFO: You are using
``pytorch`` as the backend.
```

[9]:
```
# Causal Discovery using PC Algorithm
pc = PC()
pc_dataset = np.vstack([X, Y, S]).T
pc.learn(pc_dataset)
print("Causal matrix from PC Algorithm:\n", pc.causal_matrix)
```

```
Causal matrix from PC Algorithm:
 [[0 1 1]
 [1 0 1]
 [1 1 0]]
```

[ ]:
```
# PROBLEM 2 - WHY IS THE CAUSAL METRIX ALMOST COMPLETE?
# CAN YOU USE REFUTION TEST TO CHECK FOR CAUSALITY DIRECTION?
# TODO: IMPLEMENT AND DISCUSS

# Refutation Test for Causality Direction
data_for_refute = pd.DataFrame({'X': X, 'S': S, 'Y': Y})
model_refute = CausalModel(
    data=data_for_refute,
    treatment='X',
    outcome='Y',
    graph="digraph { S -> X; X -> Y; S -> Y; }"
)
identified_estimand = model_refute.identify_effect()
causal_estimate = model_refute.estimate_effect(identified_estimand,
                                               method_name="backdoor.
 ↪linear_regression")
refute_results = model_refute.refute_estimate(identified_estimand,␣
 ↪causal_estimate, method_name="placebo_treatment_refuter")

print("Refutation Test Result:\n", refute_results) # write in markdown below
```

Discussion:

The causal matrix obtained from the PC Algorithm appears almost complete because the dataset

includes strong linear dependencies among the variables (S), (X), and (Y). In this scenario, the PC Algorithm infers edges based on conditional independencies. Given that:

- $(S \to X)$
- $(X \to Y)$
- $(S \to Y)$

The relationships between these variables are direct and robust.
Consequently, the PC Algorithm identifies nearly all possible edges, rendering the causal matrix almost complete.
Additionally, noise levels and the relatively small sample size can impact the identification of weaker or non-existent relationships, leading to an overly dense matrix.

Refutation Test for Causality Direction:

The refutation test using a placebo treatment returned the following results:

Refutation Test Result:

- `Refute: Use a Placebo Treatment`
- `Estimated effect: 1.5355639494717588` (The original estimated causal effect of (X) on (Y) is approximately 1.5356.)
- `New effect (Placebo): 0.0` (When the treatment is replaced with a placebo (a randomly generated variable), the estimated effect drops to 0.0.)
- `p value: 1.0` (the null hypothesis (no causal relationship) cannot be rejected under the placebo scenario.)

Implications for Causality Direction:

- These results suggest that the initially observed causal relationship between (X) and (Y) is not spurious.
- The drastic reduction in the estimated effect when using a placebo treatment supports the conclusion that (X) causally influences (Y).
- Therefore, the directionality $(X \to Y)$ is validated through this refutation test.

Conclusion:

- The near-completeness of the causal matrix is attributed to the strong, direct relationships among the variables and the effectiveness of the PC Algorithm in detecting these links.
- The refutation test further confirms the correct direction of causality from (X) to (Y), adding robustness to the causal inference outcomes.

[11]:
```
N = 2000
a = np.random.uniform(0, 1, N)
b = np.random.uniform(3, 6, N)
c = a + b + .1 * np.random.uniform(-2, 0, N)
d = .7 * c + .1 * np.random.uniform(0, 1, N)
```

```
lingam_dataset = np.vstack([a, b, c, d]).T

lingam = ICALiNGAM(random_state=1)
lingam.learn(lingam_dataset)

print("Weight causal matrix from ICALiNGAM:\n", lingam.weight_causal_matrix)
```

```
Weight causal matrix from ICALiNGAM:
 [[0.    0.    1.006 0.   ]
 [0.    0.    0.999 0.   ]
 [0.    0.    0.    0.699]
 [0.    0.    0.    0.   ]]
```