Structural causal models and LiNGAM in practice

M2 GENIOMHE — Lecture 2

Christophe Ambroise
M2 GENIOMHE — Université d'Évry

Lecture 1

- DAG factorisation: $p(x) = \prod_t p(x_t \mid x_{pa(t)})$
- Chain, fork: conditioning blocks; collider: conditioning opens
- d-separation $\Rightarrow$ conditional independence
- Markov equivalence: same skeleton + same v-structures → CPDAG
- Observational data alone: CPDAG at best, some directions unknown

Prior knowledge

Structural causal models

Structural causal model

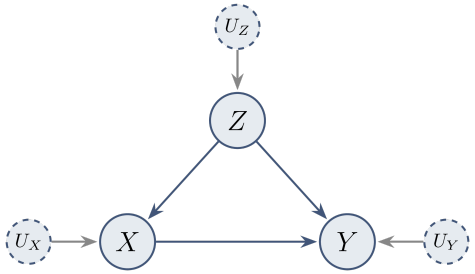
A structural causal model (SCM) generates each endogenous variable by a deterministic mechanism of its direct causes and of an exogenous noise.

$$\mathcal{M} = (U, V, F, P_U): \qquad V_j := f_j\big(V_{pa(j)}, U_j\big), \quad j = 1, \dots, p$$

- V: endogenous variables, modelled
- U: exogenous variables, causes left outside the model
- U_j jointly independent → **Markovian** SCM

SCM example

$$\begin{aligned} Z &:= U_Z \\ X &:= 2Z + U_X \\ Y &:= 3X + Z + U_Y, \end{aligned} \qquad U_Z, U_X, U_Y \overset{iid}{\sim} \mathcal{N}(0,1)$$



Causal DAG: from SCM to graph

- Edge $V_i \to V_j$ iff V_i argument of f_j
- Markovian SCM → observational law factorises along the DAG
- Same DAG as a Bayesian network, **stronger semantics**

Causal model vs graphical model

Graphical model: describes one distribution. Causal model: also predicts the distributions after interventions.

Interventions

Conditioning vs intervening

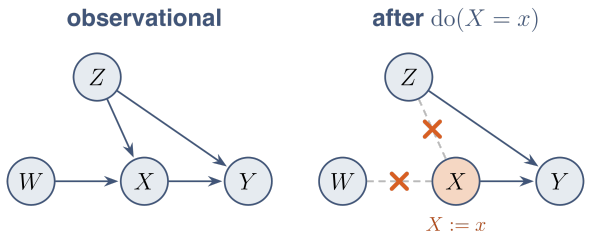
- **Conditioning** $P(Y \mid X = x)$: sub-population with $X = x$, graph intact
- **Intervening** $P(Y \mid do(X = x))$: X forced to x for everyone
- Confounder Z: association through $X \leftarrow Z \rightarrow Y$ only in the first case

$$P(Y \mid X = x) \neq P(Y \mid do(X = x)) \quad \text{in general}$$

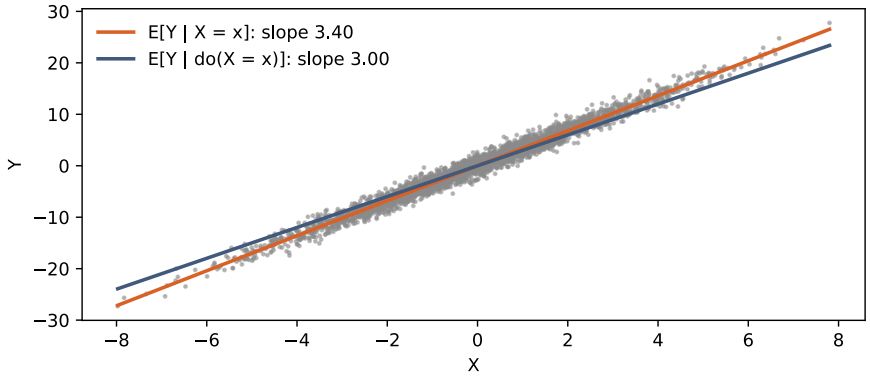
Modularity

Each mechanism f_j, or conditional $P(v_j \mid v_{pa(j)})$, is an autonomous module: an intervention on one variable leaves all other modules unchanged.

- $do(X = x)$: equation of X replaced by $X := x$
- Graph surgery: arrows **into** X deleted → $G_{\overline{X}}$



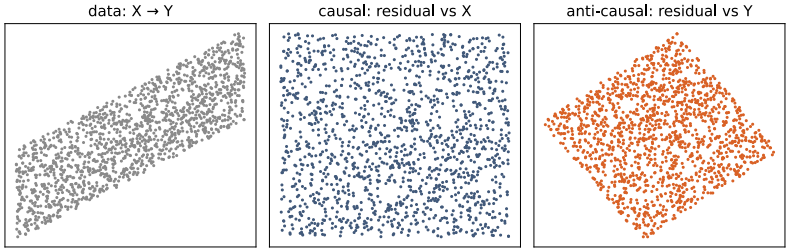
Linear example: regression slope vs causal effect



Theory: $\mathbb{E}[Y \mid X = x] = 3x + \mathbb{E}[Z \mid X = x] = 3.4\,x$,
$\mathbb{E}[Y \mid do(X = x)] = 3x$.

Beyond Markov equivalence: LiNGAM

Two variables, one direction?



$$Y = 0.8X + \varepsilon_Y, \quad X \perp \varepsilon_Y, \quad X \text{ non-Gaussian}$$

- Both residuals **uncorrelated** with the regressor
- Only the causal one **independent**: non-Gaussianity breaks the symmetry

Linear non-Gaussian acyclic model

In LiNGAM, the data follow a linear SEM on a DAG, with mutually independent noises of which at most one is Gaussian.

$$X = BX + \varepsilon, \qquad B \text{ acyclic (strictly lower triangular up to permutation)},$$

Identifiability (Shimizu *et al.*, 2006)

Under these assumptions, B and the causal order are identifiable from the distribution of X alone.

DirectLiNGAM

X_j is exogenous (a source) if and only if every other variable, regressed on X_j, leaves a residual independent of X_j.

$$X_j \text{ source} \iff \forall\, i \neq j: \quad X_i - \frac{\operatorname{Cov}(X_i, X_j)}{\operatorname{Var} X_j} X_j \;\perp\; X_j$$

1. Most independent candidate → next variable of the order
2. Its effect regressed out of all remaining variables
3. Repeat; then B by adaptive lasso along the order

Shimizu *et al.* (2011), *JMLR*. Python package `lingam`.

DirectLiNGAM in practice

```python
import numpy as np, lingam
rng = np.random.default_rng(1); n = 2000
e = rng.laplace(size=(n, 4))                    # non-Gaussian noises
x0 = e[:, 0]
x2 = 0.8 * x0 + e[:, 2]
x1 = -0.7 * x2 + e[:, 1]
x3 = 0.5 * x0 + 0.6 * x1 + e[:, 3]
X = np.c_[x0, x1, x2, x3]
model = lingam.DirectLiNGAM().fit(X)
print("order:", [int(i) for i in model.causal_order_])
print(np.round(model.adjacency_matrix_, 2))     # B[j, k] : effect k -> j
```

```
order: [0, 2, 1, 3]
[[ 0.    0.    0.    0.  ]
 [ 0.    0.   -0.69  0.  ]
 [ 0.79  0.    0.    0.  ]
 [ 0.51  0.6   0.    0.  ]]
```

OT-LiNGAM: non-Gaussianity by optimal transport

2-Wasserstein distance

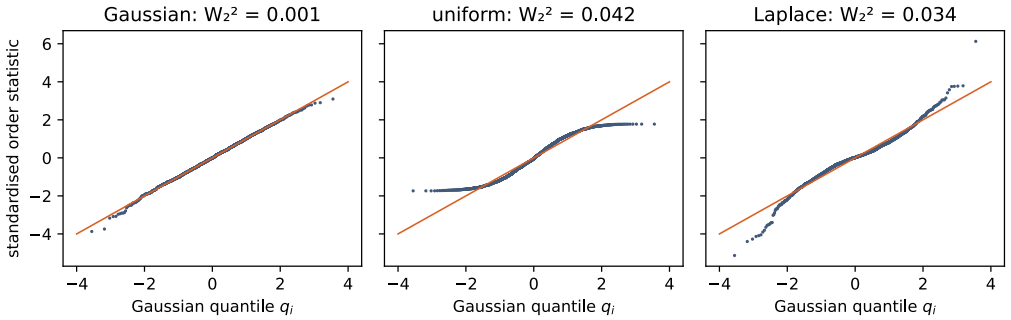
The squared 2-Wasserstein distance is the smallest mean squared displacement among all couplings of the two distributions.

$$\mathcal{W}_2(\mu,\nu)^2 = \inf_{X\sim\mu,\,Y\sim\nu} \mathbb{E}\big[(X-Y)^2\big] \overset{\dim 1}{=} \int_0^1 \big(F_\mu^{-1}(u) - F_\nu^{-1}(u)\big)^2\,du$$

- Dimension 1: optimal coupling = matching of quantiles
- Non-Gaussianity of X: $\mathcal{W}_2\big(\mathrm{std}(X), \mathcal{N}(0,1)\big)^2$, zero iff X Gaussian

Laplante, Ambroise & Humbert, *Contrast-free ICA and causal inference via Wasserstein distances to the Gaussian*, preprint. Package `otlingam`.

Non-Gaussianity of three distributions



Wasserstein order objective

For a candidate order σ, regress each variable on its predecessors and score the non-Gaussianity of the standardised residuals.

$$G(\sigma) = \sum_{j=1}^{d} \mathcal{W}_2\Big(\mathrm{std}\,\big(R_j(\sigma)\big), \mathcal{N}(0,1)\Big)^2, \qquad R_j(\sigma) = X_j - \mathrm{proj}_{\mathrm{span}\{X_k : k \text{ befo}}$$

- Causal order: residuals = structural noises ε_j
- Wrong order: some residual mixes several noises → closer to Gaussian

OT-LiNGAM in practice

```python
import numpy as np
from otlingam import ExhaustiveOTLiNGAM, GreedyOTLiNGAM
from otlingam.utils import disorder
rng = np.random.default_rng(1); n = 2000
e = rng.laplace(size=(n, 4))
x0 = e[:, 0]; x2 = 0.8 * x0 + e[:, 2]
x1 = -0.7 * x2 + e[:, 1]; x3 = 0.5 * x0 + 0.6 * x1 + e[:, 3]
X = np.c_[x0, x1, x2, x3]
B_true = np.zeros((4, 4)); B_true[2, 0], B_true[1, 2], B_true[3, 0], B_true[3, 1] = .8, -.7, .5, .6
for M in (ExhaustiveOTLiNGAM(), GreedyOTLiNGAM()):
    M.fit(X)
    print(type(M).__name__, [int(i) for i in M.causal_order_], "disorder:", disorder(B_true, M.causal_c
```

```
ExhaustiveOTLiNGAM [0, 2, 1, 3] disorder: 0
GreedyOTLiNGAM [0, 2, 1, 3] disorder: 0
```


LiNGAM analysis in practice

Non-Gaussianity check

```python
from scipy import stats
print("excess kurtosis:", np.round(stats.kurtosis(X), 2))
print("Shapiro p-values:", [f"{stats.shapiro(c).pvalue:.1e}" for c in X.T])
```

```
excess kurtosis: [2.72 0.99 1.57 1.21]
Shapiro p-values: ['4.8e-21', '5.6e-09', '2.0e-15', '2.5e-10']
```

- Kurtosis far from 0, tiny Shapiro p-values: non-Gaussian variables
- Near-Gaussian variables: directions unreliable
- Discrete or binary variables: LiNGAM only an approximation

Bootstrap stability of edges

```python
model = lingam.DirectLiNGAM().fit(X)
boot = lingam.DirectLiNGAM().bootstrap(X, n_sampling=100)  # separate instance: refitted in place
P = boot.get_probabilities(min_causal_effect=0.05)  # P[j, k]: frequency of edge k -> j
print(np.round(P, 2))
```

```
[[0.   0.   0.   0.  ]
 [0.01 0.   1.   0.  ]
 [1.   0.   0.   0.  ]
 [1.   1.   0.02 0.  ]]
```

- One fit = one estimate; bootstrap = frequency of each edge over resamples
- Edges reported **with** their frequency

Independence of residuals

```python
print(np.round(model.get_error_independence_p_values(X), 2))
rng = np.random.default_rng(1); h = rng.laplace(size=n)   # h: hidden common cause
a = h + rng.laplace(size=n); b = 0.5 * a + h + rng.laplace(size=n)
D = np.c_[a, b]; m = lingam.DirectLiNGAM().fit(D)
pv = m.get_error_independence_p_values(D)[0, 1]
print(f"a -> b: {m.adjacency_matrix_[1, 0]:.2f}, p-value {pv:.0e}")
```

```
[[0.   0.82 0.12 0.17]
 [0.82 0.   0.35 0.74]
 [0.12 0.35 0.   0.67]
 [0.17 0.74 0.67 0.  ]]
a -> b: 0.97, p-value 3e-06
```

- Model assumption: independent noises → residuals pairwise independent
- Small p-value: hidden confounder, non-linearity or feedback
- Hidden h: effect 0.97 instead of 0.5, residuals clearly dependent

Prior knowledge

```python
pk = lingam.utils.make_prior_knowledge(4, exogenous_variables=[0], sink_variables=[3])
print(pk)                          # pk[j, k]: 1 path k -> j, 0 no path, -1 unknown
print([int(i) for i in lingam.DirectLiNGAM(prior_knowledge=pk).fit(X).causal_order_])
```

```
[[-1  0  0  0]
 [-1 -1 -1  0]
 [-1 -1 -1  0]
 [-1 -1 -1 -1]]
[0, 2, 1, 3]
```

- Exogenous variables: genotype, treatment, time
- Sinks: final phenotype, outcome
- Fewer orders to search, fewer errors

Total causal effect

The total effect of X_k on X_j sums the contributions of all directed paths from X_k to X_j: the change of X_j under an intervention $do(X_k = x + 1)$ instead of $do(X_k = x)$.

```python
print(f"x0 -> x3: {model.estimate_total_effect(X, 0, 3):.2f}")
```

```
x0 -> x3: 0.18
```

- True value: direct 0.5 + path $x_0 \to x_2 \to x_1 \to x_3$: $0.8 \times (-0.7) \times 0.6 \to 0.164$
- Bootstrap version: `boot.get_total_causal_effects()`

Estimators and report

Estimator	Package	Use
`DirectLiNGAM`	`lingam`	bootstrap, prior knowledge, tests
`ExhaustiveOTLiNGAM`	`otlingam`	exact search, few variables
`GreedyOTLiNGAM`	`otlingam`	many variables

Report of a LiNGAM analysis

Non-Gaussianity; orders from two estimators; bootstrap frequencies of the edges; residual tests; prior knowledge; total effects.