Causal Inference – Theory and Applications

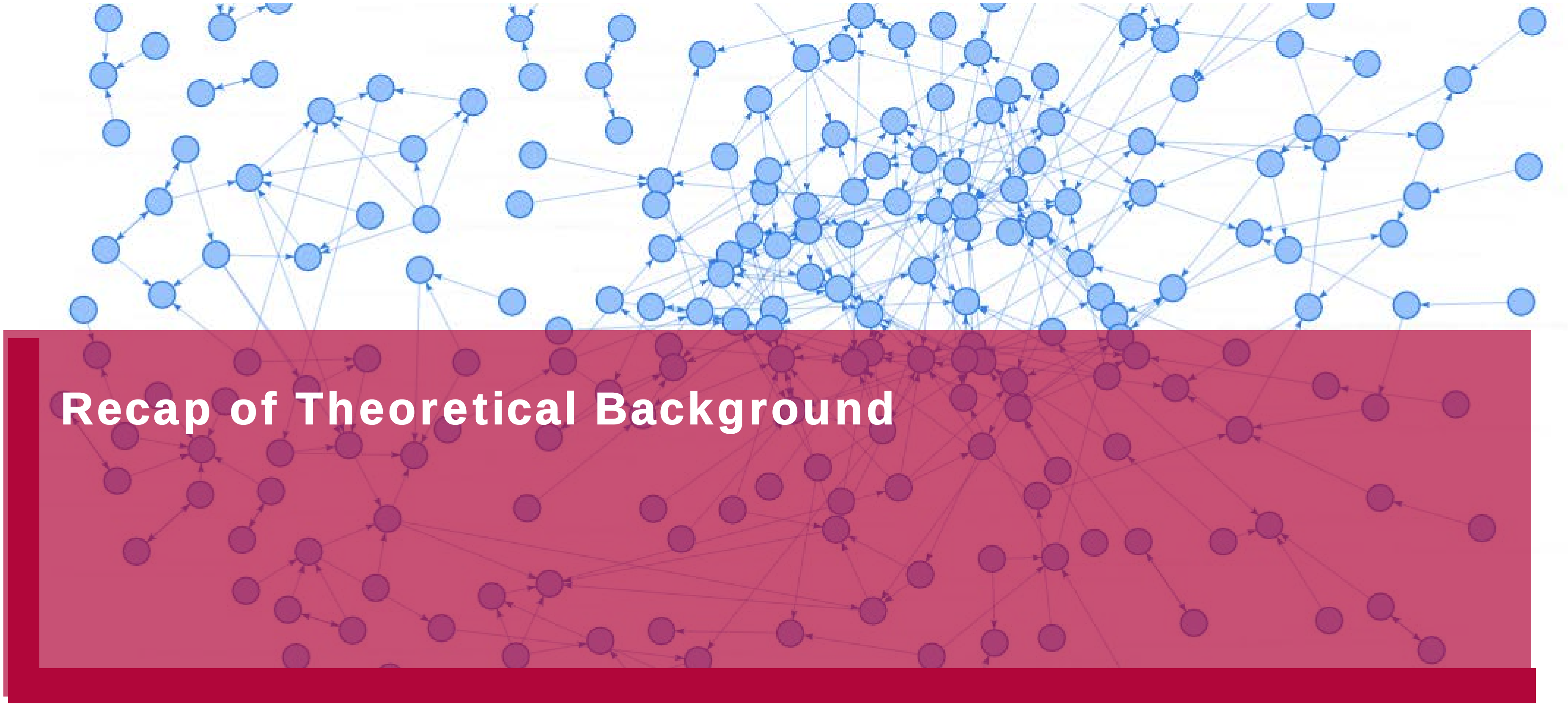
Dr. Matthias Uflacker, Johannes Huegle, Christopher Schmidt

May 03, 2018

Agenda
May 03, 2018

- **Recap of Theoretical Background**
- **Constraint-Based Causal Structure Learning**
  1. Introduction
  2. Constraint-Based Causal Structure Learning
  3. PC Algorithm
  4. PC Algorithm in Application
  5. Extensions of the PC Algorithm
  6. Excursion: Other Causal Structure Learning Algorithms
- **Jupyter Notebook**

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

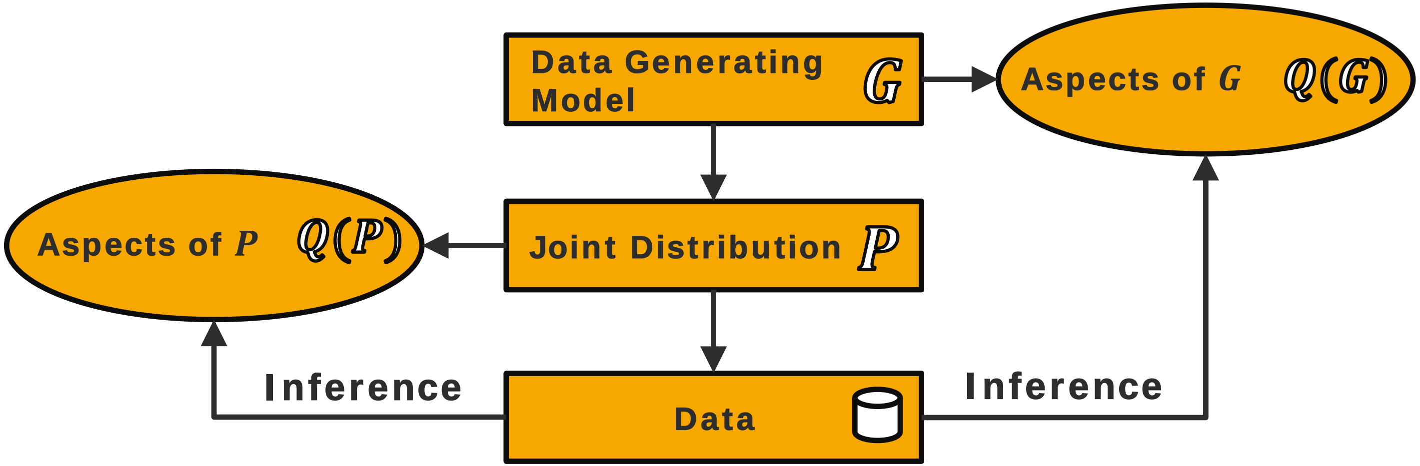


Recap of Theoretical Background

Recap of Theoretical Background
Causal Inference in a Nutshell

Traditional Statistical Inference Paradigm

Paradigm of Structural Causal Models



E.g., what is the sailors' probability of recovery when **we see** a treatment with lemons?

$$Q(P) = P(recovery|lemons)$$

E.g., what is the sailors' probability of recovery if **we do** treat them with lemons?

$$Q(G) = P(recovery|do(lemons))$$

Recap of Theoretical Background
Causal Graphical Models

- Causal Structures formalized by *DAG (directed acyclic graph)* G with random variables $V_1, \dots, V_n$ as vertices.

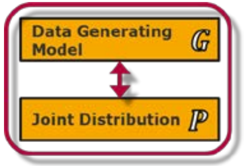
- *Causal Sufficiency*, *Causal Faithfulness* and *Global Markov Condition* imply

$$(X \perp Y | Z)_G \Leftrightarrow (X \perp Y | Z)_P.$$

- *Local Markov Condition* states that the density $p(v_1, \dots, v_n)$ then factorizes into

$$p(v_1, \dots, v_n) = \prod_{i=1}^{n} p(v_i | Pa(v_i)).$$

- Causal conditional $p(v_j | Pa(v_j))$ represent *causal mechanisms*.

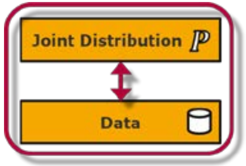


Recap of Theoretical Background
Statistical Inference

- *Null Hypothesis* H_0 is the claim that is initially assumed to be true
- *Alternative Hypothesis* H_1 is a claim that contradicts the H_0
- How to test a hypothesis?
  - Approximate T under H_0 by a known distribution
  - Different distributions yield to different tests, e.g., T-test, χ^2-test, etc.
  - Derive rejection criteria for H_0
    - *c-value:* reject H_0 if $T(x_n) > c$ for a $c \in \mathbb{R}$
    - *p-value:* reject H_0 if $P_{H_0}\big(T(X) > T(x)\big) < \alpha$

    } are equivalent
- *(Conditional) Independence Test*

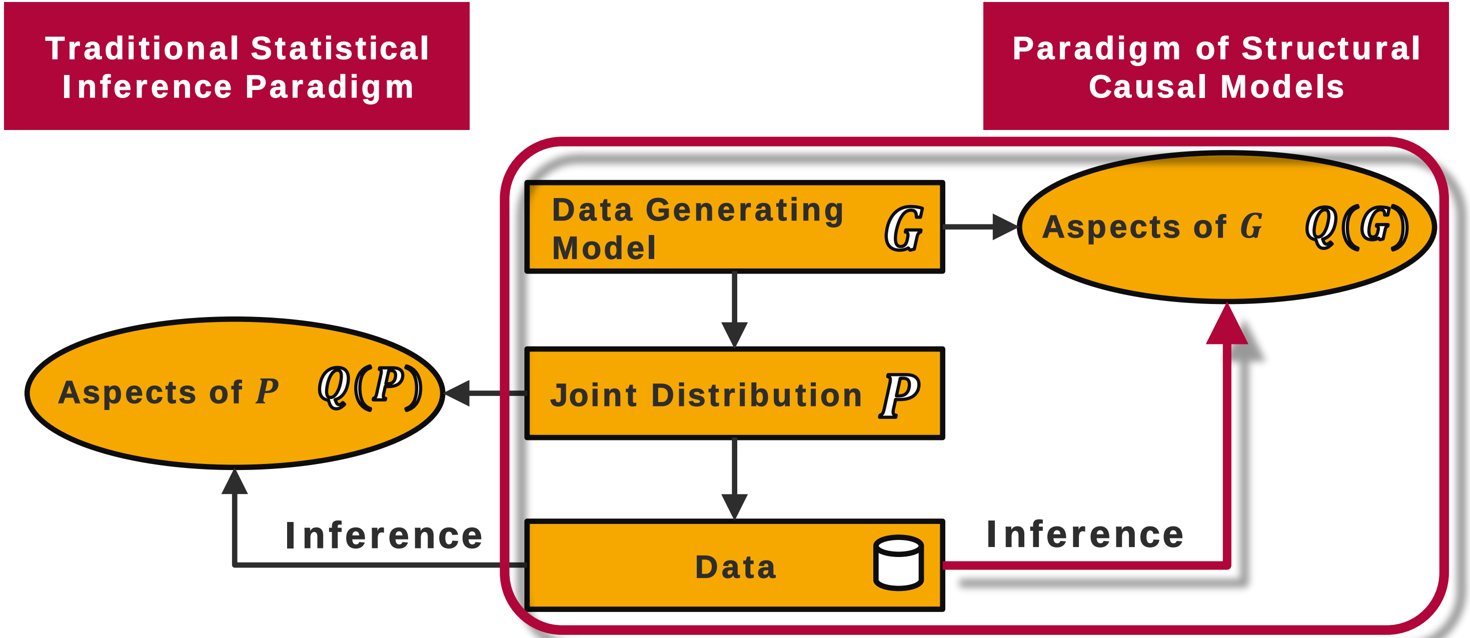
  Distribution of $V_1, \dots, V_N \Rightarrow$ dependence measures $T(V_i, V_j, \boldsymbol{S}) \Rightarrow$ test $H_0: t = 0$
- Allows for *constraint-based causal structure learning*



HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Constraint-Based Causal Structure Learning

1. Introduction
The Concept



E.g., what is the sailors' probability of recovery when **we see** a treatment with lemons?

$$Q(P) = P(recovery|lemons)$$

E.g., what is the sailors' probability of recovery if **we do** treat them with lemons?

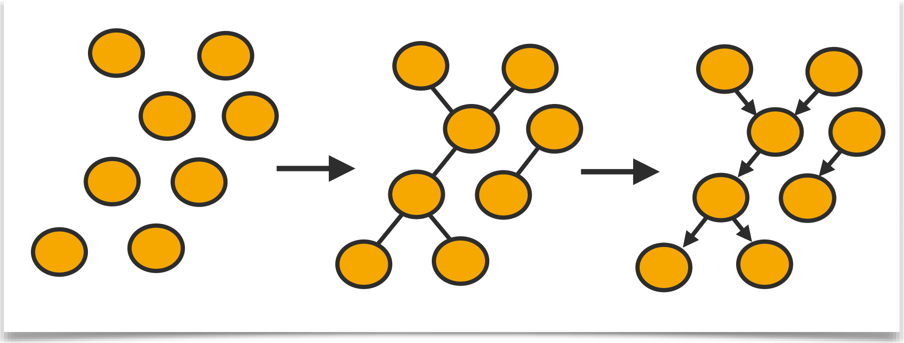
$$Q(G) = P(recovery|do(lemons))$$

1. Introduction
Recap: Basis of Causal Structure Learning (Pearl et al.)

- **Assumptions:**
  - Causal Sufficiency
  - Global Markov Condition
  - Causal Faithfulness

- **Causal Structure Learning:**
  - Accept only those DAG's G as causal hypothesis for which
    $$(X \perp Y | Z)_G \Leftrightarrow (X \perp Y | Z)_P.$$
  - Identifies causal DAG up to *Markov equivalence class* (DAGs that imply the same conditional independencies)
  - The Markov equivalence class of a DAG G includes all DAGs G' that have the same *skeleton* C and the same *v-structures*
  - Markov equivalence class of the true DAG G that can be uniquely described by a *completed partially directed acyclic graph (CPDAG)*

2. Constraint-Based Causal Structure Learning
Algorithmic Construction (I/II)



Idea:

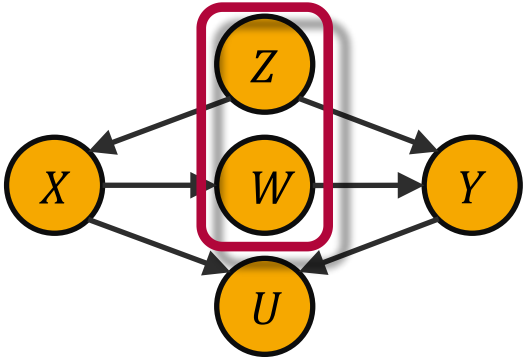
1. Construct skeleton C
2. Find v-structures
3. Direct further edges that follow from
   - Graph is acyclic
   - All v-structures have been found in 2

➡ IC algorithm by Verma and Pearl (1990) to reconstruct CPDAG G from P

2. Constraint-Based Causal Structure Learning
Algorithmic Construction (II/II)

Theorem
Assume Markov condition and faithfulness holds. Then X and Y are linked by an edge if and only if there is no set $S(X,Y)$ such that
$$(X \perp Y | S(X,Y))_{P.}$$

- I.e., dependence mediated by other variables can be screened off by conditioning on an *appropriate* set



- $X \perp Y | \{Z, W\}$
- But not:
  - $X \perp Y | U$
  - $X \perp Y | \{Z, W, U\}$

...but not by conditioning on all other variables!

- $S(X,Y)$ is called *separation set of X and Y*

3. PC Algorithm
The Idea

Question:
How to find the appropriate separation sets $S(V_i, V_j)$ for all variables V_i and V_j?

- Check $V_i \perp V_j \mid S(V_i, V_j)$ for all possible separation sets $S(V_i, V_j) \subseteq V \setminus \{V_i, V_j\}$
  - Computationally infeasible for large V

- Efficient construction of the skeleton C

  Iteration over size of the separation sets S:
  1. Remove all edges $X - Y$ with $X \perp Y$
  2. Remove all edges $X - Y$
     for which there is an adjacent $Z \neq Y$ of X with $X \perp Y \mid Z$
  3. Remove all edges $X - Y$
     for which there are two adjacent $Z_1, Z_2 \neq Y$ of X with $X \perp Y \mid \{Z_1, Z_2\}$
  4. ...

➡ PC algorithm by Spirtes et al. (1993) to reconstruct CPDAG G from P

3. PC Algorithm

Skeleton Discovery: Pseudocode

Algorithm 1 The PC_{pop}-algorithm

INPUT: Vertex Set V, Conditional Independence Information

OUTPUT: Estimated skeleton C, separation sets S (only needed when directing the skeleton afterwards)

Form the complete undirected graph $\tilde{C}$ on the vertex set V.

$\ell = -1; \quad C = \tilde{C}$

repeat

    $\ell = \ell + 1$

    repeat

        Select a (new) ordered pair of nodes i, j that are adjacent in C such that $|adj(C,i) \setminus \{j\}| \geq \ell$

        repeat

            Choose (new) $\mathbf{k} \subseteq adj(C,i) \setminus \{j\}$ with $|\mathbf{k}| = \ell$.

            if i and j are conditionally independent given $\mathbf{k}$ **then**

                Delete edge i, j

                Denote this new graph by C

                Save $\mathbf{k}$ in $S(i,j)$ and $S(j,i)$

            end if

        until edge i, j is deleted or all $\mathbf{k} \subseteq adj(C,i) \setminus \{j\}$ with $|\mathbf{k}| = \ell$ have been chosen

    until all ordered pairs of adjacent variables i and j such that $|adj(C,i) \setminus \{j\}| \geq \ell$ and $\mathbf{k} \subseteq adj(C,i) \setminus \{j\}$ with $|\mathbf{k}| = \ell$ have been tested for conditional independence

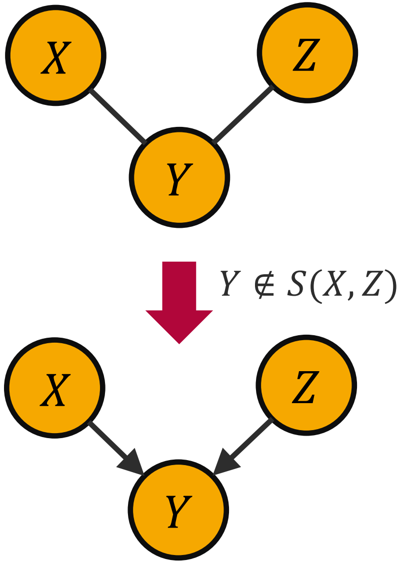
until for each ordered pair of adjacent nodes i, j: $|adj(C,i) \setminus \{j\}| < \ell$.

3. PC Algorithm
Edge Orientation: v-Structures

- Assume the skeleton is given by:
  - Given $X - Y - Z$ with X and Z nonadjacent
  - Given $S(X,Z)$ with $X \perp Z \mid S(X,Z)$

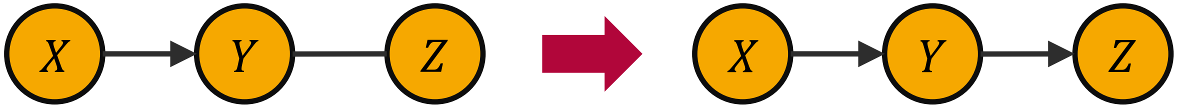
- A priori, there are 4 possible orientations
  - $X \to Y \to Z$
  - $X \leftarrow Y \to Z$
  - $X \leftarrow Y \leftarrow Z$
  - $X \to Y \leftarrow Z$

  The first three: $Y \in S(X,Z)$; the last: $Y \notin S(X,Z)$



v-Structures:
If $Y \notin S(X,Z)$ then replace $X - Y - Z$ by $X \to Y \leftarrow Z$.

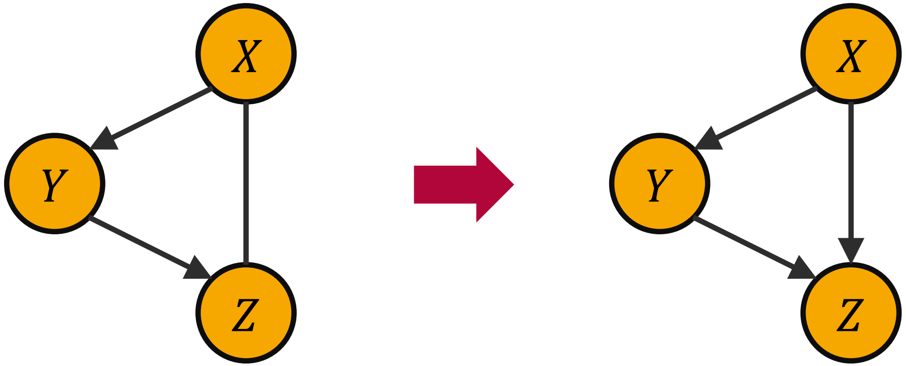
3. PC Algorithm
Edge Orientation: Rule 1



(Otherwise we get a new v-structure)

Rule 1:
Orient $Y - Z$ to $Y \rightarrow Z$ whenever
there is an arrow $X \rightarrow Y$ s.t. X and Z are nonadjacent

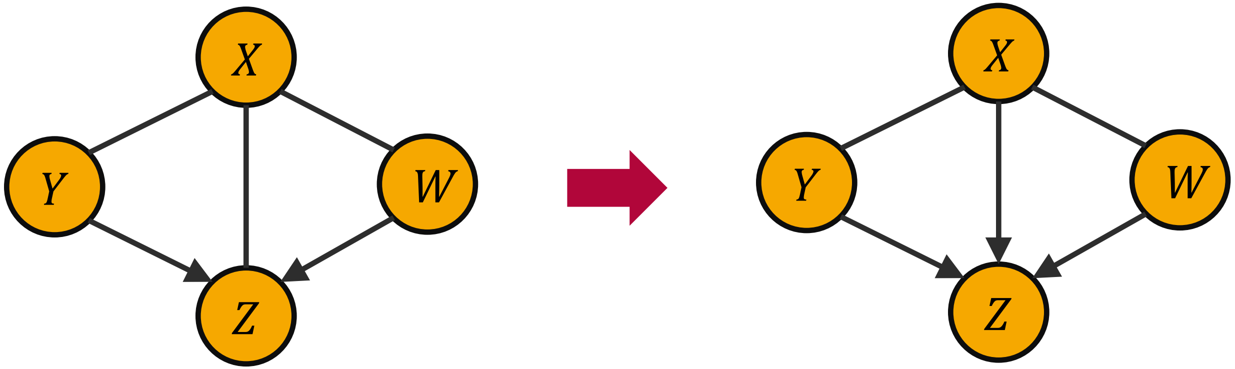
3. PC Algorithm
Edge Orientation: Rule 2



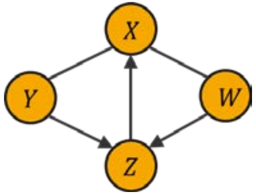
(Otherwise we get a cycle)

Rule 2:
Orient $X - Z$ to $X \rightarrow Z$ whenever
there is a chain $X \rightarrow Y \rightarrow Z$

Edge Orientation: Rule 3

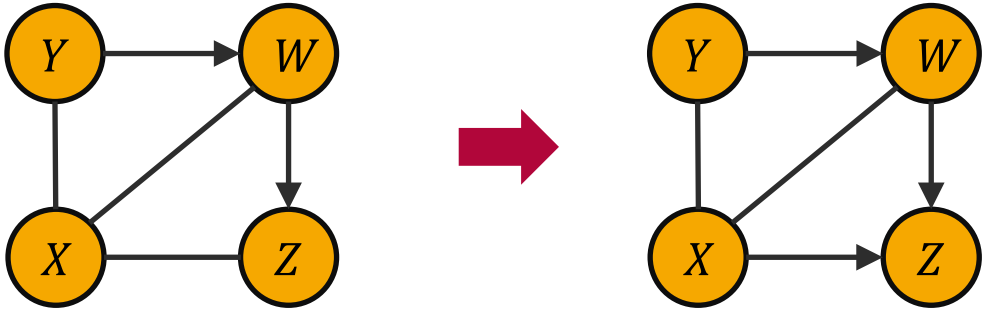


(Could not be completed without creating a cycle or a new v-structure)

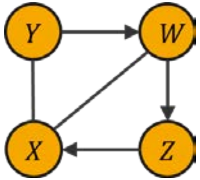


Rule 3:
Orient $X - Z$ to $X \rightarrow Z$ whenever
there are two chains $X - Y \rightarrow Z$ and $X - W \rightarrow Z$ s.t. Y and W are nonadjacent

3. PC Algorithm
Edge Orientation: Rule 4



(Could not be completed without creating a cycle or a new v-structure)



Rule 4:
Orient $X - Z$ to $X \rightarrow Z$ whenever
there are two chains $X - Y \rightarrow W$ and $Y \rightarrow W \rightarrow Z$ s.t. Y and Z are nonadjacent

3. PC Algorithm
Edge Orientation: Pseudocode

Algorithm 2 Extending the skeleton to a CPDAG

INPUT: Skeleton G_{skel}, separation sets S

OUTPUT: CPDAG G

for all pairs of nonadjacent variables i, j with common neighbour k **do**

  if $k \notin S(i, j)$ **then**

    Replace $i - k - j$ in G_{skel} by $i \rightarrow k \leftarrow j$

  end if

end for

In the resulting PDAG, try to orient as many undirected edges as possible by repeated application of the following three rules:

R1 Orient $j - k$ into $j \rightarrow k$ whenever there is an arrow $i \rightarrow j$ such that i and k are nonadjacent.

R2 Orient $i - j$ into $i \rightarrow j$ whenever there is a chain $i \rightarrow k \rightarrow j$.

R3 Orient $i - j$ into $i \rightarrow j$ whenever there are two chains $i - k \rightarrow j$ and $i - l \rightarrow j$ such that k and l are nonadjacent.

R4 Orient $i - j$ into $i \rightarrow j$ whenever there are two chains $i - k \rightarrow l$ and $k \rightarrow l \rightarrow j$ such that k and j are nonadjacent.

3. PC Algorithm
A Review

Advantages

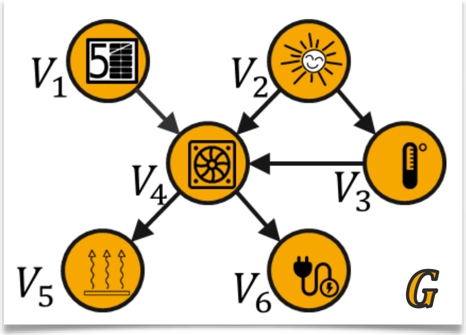
- Testing all sets $S(X,Y)$ containing the adjacencies of X is sufficient
- Many edges can be removed already for small sets
- Depending on sparseness, the algorithm only requires independence tests with small conditioning sets $S(X,Y)$
- Polynomial complexity for graph of N vertices of bounded degree k, i.e.,
$$\frac{N^2(N-1)^{k-1}}{(k-1)!}$$
- Asymptotic consistency (under technical assumptions), i.e.,
$$\Pr\left(\hat{G} = G\right) \to 1 \quad (n \to \infty)$$

Disadvantages

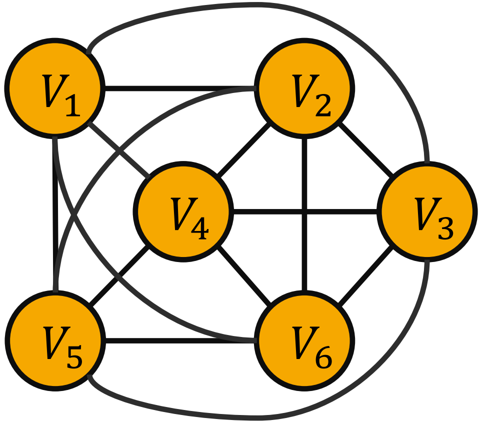
- In the worst case, complexity exponential to number of vertices N
- Assumes causal sufficiency, faithfulness and Markov conditions

4. PC Algorithm in Application
Cooling House Example (I/V)

- Assume the true DAG G is given by:

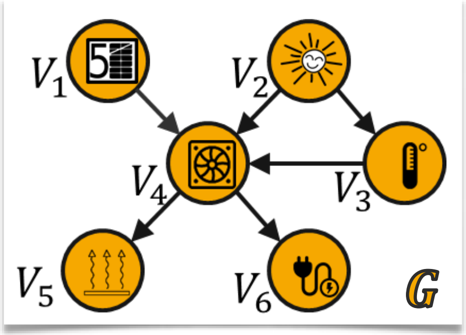


- We start with a fully connected undirected graph:

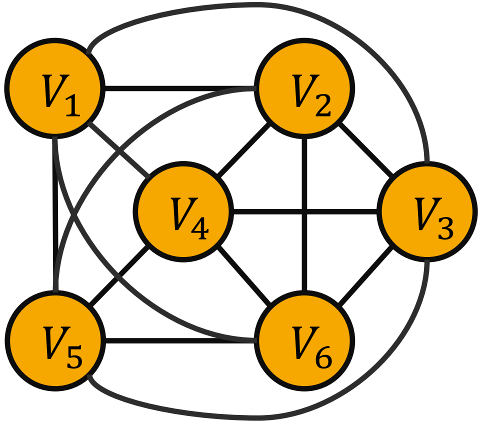


4. PC Algorithm in Application
Cooling House Example (II/V)

- Assume the true DAG G is given by:

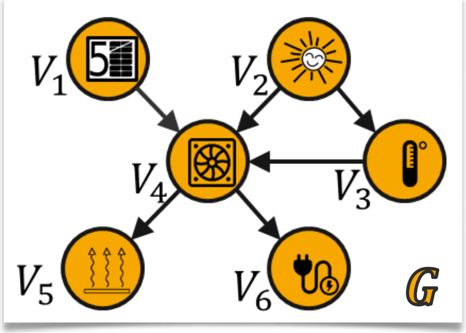


- Remove all edges $X - Y$ that are directly independent, i.e., $X \perp Y \mid \emptyset$
  - $V_1 \perp V_2$
  - $V_1 \perp V_3$

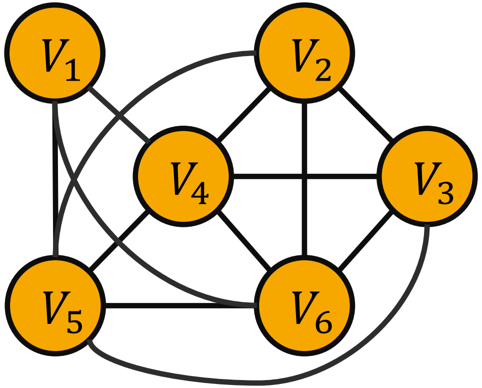


4. PC Algorithm in Application
Cooling House Example (III/V)

- Assume the true DAG G is given by:

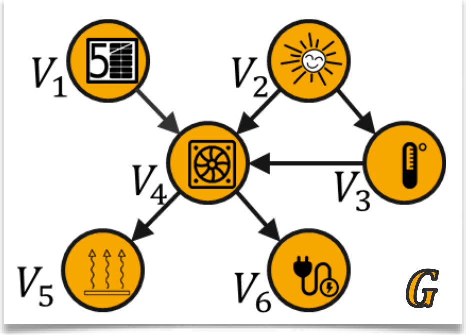


- Remove all edges $X - Y$ having separation sets of size 1, i.e., $X \perp Y \mid Z$
  - $V_1 \perp V_5 \mid V_4$
  - $V_1 \perp V_6 \mid V_4$
  - $V_2 \perp V_5 \mid V_4$
  - $V_2 \perp V_6 \mid V_4$
  - $V_3 \perp V_5 \mid V_4$
  - $V_3 \perp V_6 \mid V_4$
  - $V_5 \perp V_6 \mid V_4$

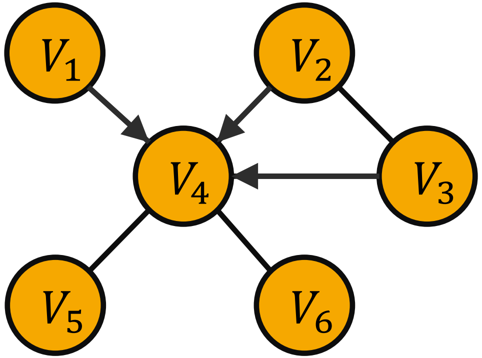


4. PC Algorithm in Application
Cooling House Example (IV/V)

- Assume the true DAG G is given by:

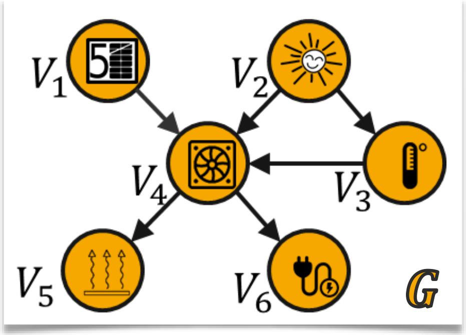


- Find v-structures, i.e., orient $X - Y - Z$ to $X \rightarrow Y \leftarrow Z$ if $Y \notin S(X, Z)$
    - $V_4 \notin S(V_1, V_2)$
    - $V_4 \notin S(V_1, V_3)$

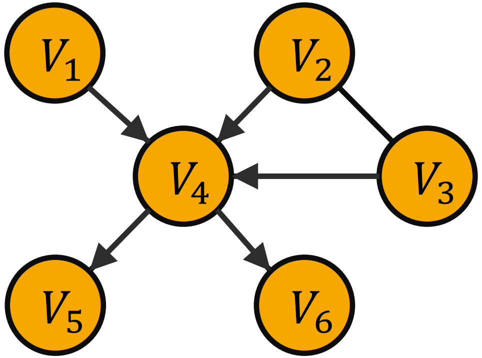


4. PC Algorithm in Application
Cooling House Example (V/V)

- Assume the true DAG G is given by:



- Orient further edges (such that no further v-structures arise)
  - $V_1 \rightarrow V_4 - V_5$ (Rule 1)
  - $V_1 \rightarrow V_4 - V_6$ (Rule 1)



- No further edges can be oriented, i.e., $V_2 - V_3$ remain undirected

5. Extensions of the PC Algorithm
Order Independence (Colombo et al. 2014)

PC algorithm

Order of $V_1, \ldots, V_N$ affects estimation of

1. Skeleton C
2. Separating sets $S(V_i, V_j)$
3. Edge orientation

PC-stable algorithm

For each level l

- Compute and store the adjacency set $a(V_i)$ of all vertices V_i
- Use $a(V_i)$ for search of separation sets

⇨ Edge deletion longer affects which conditional independencies are checked for other pairs of variables at this level l

Algorithm 4.1 Step 1 of the PC-stable algorithm (oracle version)

Require: Conditional independence information among all variables in $\mathbf{V}$, and an ordering order($\mathbf{V}$) on the variables

Form the complete undirected graph $\mathcal{C}$ on the vertex set $\mathbf{V}$
Let $\ell = -1$;
repeat
  Let $\ell = \ell + 1$;
  for all vertices X_i in $\mathcal{C}$ **do**
    Let $a(X_i) = \text{adj}(\mathcal{C}, X_i)$
  end for
  repeat
    Select a (new) ordered pair of vertices (X_i, X_j) that are adjacent in $\mathcal{C}$ and satisfy $|a(X_i) \setminus \{X_j\}| \geq \ell$, using order($\mathbf{V}$);
    repeat
      Choose a (new) set $\mathbf{S} \subseteq a(X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$, using order($\mathbf{V}$);
      if X_i and X_j are conditionally independent given $\mathbf{S}$ **then**
        Delete edge $X_i - X_j$ from $\mathcal{C}$;
        Let sepset(X_i, X_j) = sepset$(X_j, X_i) = \mathbf{S}$;
      end if
    until X_i and X_j are no longer adjacent in $\mathcal{C}$ or all $\mathbf{S} \subseteq a(X_i) \setminus \{X_j\}$ with $|\mathbf{S}| = \ell$ have been considered
  until all ordered pairs of adjacent vertices (X_i, X_j) in $\mathcal{C}$ with $|a(X_i) \setminus \{X_j\}| \geq \ell$ have been considered
until all pairs of adjacent vertices (X_i, X_j) in $\mathcal{C}$ satisfy $|a(X_i) \setminus \{X_j\}| \leq \ell$
return $\mathcal{C}$, sepset.

5. Extensions of the PC Algorithm

Parallelization (Le et al. 2016)

PC algorithm

Limitations:

1. Order-dependent (→*PC-stable*)
2. Inefficient

⇨ Hinders its application on high dimensional datasets

parallelPC algorithm

PC-stable allows for easy parallelization at each level l, i.e.,

1. CI tests are distributed evenly among the cores
2. Each core performs its own sets of CI tests in parallel with the others
3. Synchronize test results into the global skeleton C

⇨ Efficient in high dimensional datasets and consistent with PC-stable algorithm

Algorithm 2: The parallel-PC algorithm

```
Input: Dataset D, significant level α, P cores,
       memory-efficient indicator s, number of edges
       per batch t_b
Output: The undirected graph G with a set of edges E
Assume all nodes are connected in graph G
Let depth d = 0
repeat
    Query and fix the adjacent set adj(X,G) of each
    node X in G
    Compute the set J of unordered pairs of adjacent
    vertices (X,Y) in G
    // Parallelisation Step
    for each batch of t_b edges (t_b = |J| if s = FALSE)
    do
        Distribute the edges in the batch evenly into P
        cores, each with J_p edges
        for each core p = 1...P in parallel do
            for each pair (X,Y) ∈ J_p do
                Let k^p_{X,Y} indicate if (X,Y) is adjacent,
                initialize k^p_{X,Y} = TRUE
                // On X's neighbours
                if |adj(X,G)\{Y}| ≥ d then
                    for each subset Z_X ⊆ adj(X,G)\{Y}
                    and |Z_X| = d do
                        if I(X,Y|Z_X) then
                            k^p_{X,Y} = FALSE
                            break
                        end
                    end
                end
                // On Y's neighbours
                if |adj(Y,G)\{X}| ≥ d then
                    for each subset Z_Y ⊆ adj(Y,G)\{X}
                    and |Z_Y| = d do
                        if I(X,Y|Z_Y) then
                            k^p_{X,Y} = FALSE
                            break
                        end
                    end
                end
            end
        // Synchronisation Step
        for each core p = 1...P do
            for each pair (X,Y) ∈ J_p do
                if k^p_{X,Y} = FALSE then
                    Remove the edge between X and Y
                    and update G and E
                end
            end
        end
    Let d = d + 1
until |adj(X,G)\{Y}| < d for every pair of adjacent
vertices in G;
```

5. Extensions of the PC Algorithm
Theoretical Extensions (A Selection)

- **Weaker form of faithfulness**
  - Learn a Markov equivalence class of DAGs under a weaker-than-standard causal faithfulness assumption
  - Assumes Adjacency-Faithfulness to justify the step of recovering adjacencies in constraint-based algorithms
  - ⇨ Conservative PC (CPC) by Ramsey et al. (1995)
- **Allow for latent and selection variables**
  - Learn a Markov equivalence class of DAGs with latent and selection variables
  - Follows maximal ancestral graph (MAG) models
  - ⇨ Fast causal inference (FCI) by Spirtes et al. (1999)
- **Allow for cycles**
  - Learn Markov equivalence classes of directed (not necessarily acyclic) graphs under the assumption of causal sufficiency.
  - ⇨ Cyclic causal discovery (CCD) by Richardson (1996)

6. Excursion:
Other Causal Structure Learning Algorithms

Score-based methods

- *"search-and-score approach"*, i.e.,
  1. Assume causal structure G and functional restrictions (e.g., linear relations and independent Gaussian noise)
  2. Optimize some score (e.g., likelihood or BIC) given these restrictions
  3. Change G and compute new optimal score value
  4. Repeat this for many G and return G^{opt} with the best (optimized) score

➡ E.g., Greedy-Equivalent-Search (GES) by Chickering (2002)

Hybrid methods

- Combines constraint-based and search-and-score methods, i.e.,
  1. Constraint-based search to find skeleton
  2. Score-based approach to orient edges

➡ E.g., Max-Min Hill-Climbing (MMHC) by Tsamardinos et al. (2006)

References
Literature

- Pearl, J. (2009). *Causal inference in statistics: An overview*. Statistics Surveys.
- Pearl, J. (2009). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.
- Spirtes et al. (2000). *Causation, Prediction, and Search*. The MIT Press.
- Kalisch et al. (2007). *Estimating high-dimensional directed acyclic graphs with the PC-algorithm*. Journal of Machine Learning Research.
- Colombo et al. (2014). *Order-independent constraint-based causal structure learning*. The Journal of Machine Learning Research.
- Le et al. (2016). *A fast PC algorithm for high dimensional causal discovery with multi-core PCs*. IEEE/ACM transactions on computational biology and bioinformatics.
- Kalisch et al. (2014). *Causal structure learning and inference: a selective review*. *Quality Technology & Quantitative Management*

References
Implementations

R

- Kalisch et al. (2017), R Package ‘pcalg’.
- Le et al. (2015), R Package ‘ParallelPC’.

Python

- Kobayashi (2015), CPDAG Estimation using PC-Algorithm.

Other

- Carneggie Mellon University, The Tetrad Project

Causal Inference in Application

Causal Inference in Application
Jupyter Notebook

Causal Inference - Theory and Applications

In our lecture Causal Inference - Theory and Applications, we look at the mathematical concepts that build the basis of causal inference.

Causal Inference in Application

We now look how these concepts are applied on observational data to derive causal relationships and how to use the do-operator to receive an estimation of the causal effect. In order to give you an overview on therelated procedure, this notebook gives a step by step approach in the context of a simple cooling house example.

Table of Contents

Thank you
for your attention!