Access Code: **000719**

CMP9794M
Advanced Artificial Intelligence

Heriberto Cuayahuitl

UNIVERSITY OF LINCOLN

School of Engineering and Physical Sciences

Last Week

- Introduction to Bayesian Networks
  - Networks encode conditional independence
  - Structure via a Directed Acyclic Graph (DAG)
  - Each node has a conditional prob. table (CPT)
  - CPTs can be learnt via Maximum Likelihood Estimation (MLE) with Laplace / Aditive / Dirichlet smooting to avoid zero probabilities

- Introduction to Probabilistic Reasoning
  - Inference by enumeration
  - Inference by variable elimination

Something to Keep in Mind

We should ask the following when training Bayesian Networks (or graphical models):

1. Where do the probabilities come from?

   A: Usually from training data (e.g., via MLE)

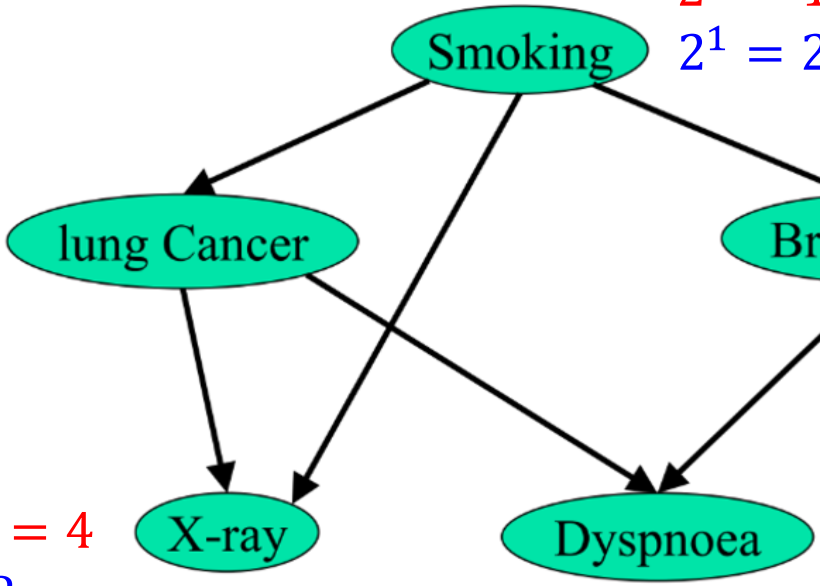
2. Where does the structure come from?

   A: Either from data or from expert/prior knowledge

3. How will probabilistic inference be done?

   A: Using algorithms for exact or approximate inference

How many probabilities in a Bayesian network?



$2^0 = 1$
$2^1 = 2$

$2^0 * 2 = 2$
$2^2 = 4$

$2^0 * 2 = 2$
$2^2 = 4$

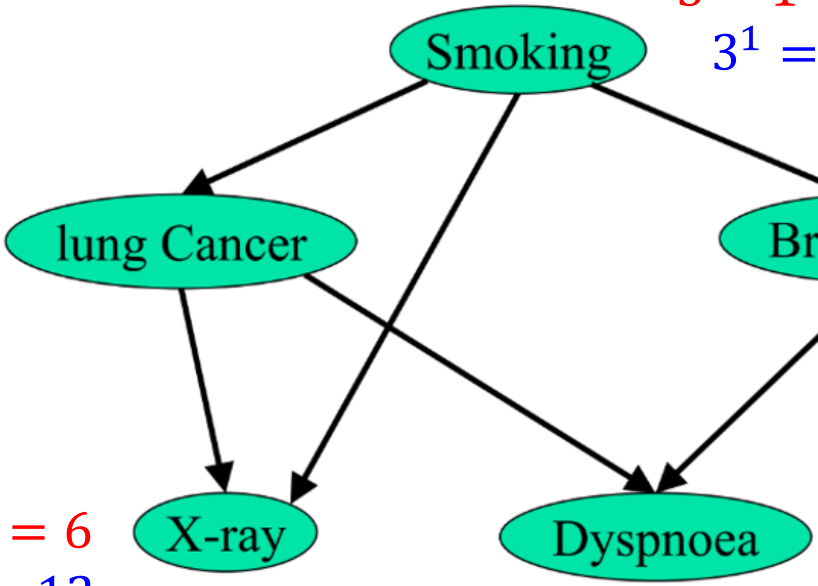
$2^0 * 2 * 2 = 4$
$2^3 = 8$

$2^0 * 2 * 2 = 4$
$2^3 = 8$

All binary random variables:

- Concise version: 1+2+2+4+4=13
- Full enumeration: 2+4+4+8+8=26

How many probabilities in a Bayesian network?



Smoking: $3 - 1 = 2$, $3^1 = 3$

lung Cancer: $2^0 * 3 = 3$, $2 * 3 = 6$

Bronchitis: $2^0 * 3 = 3$, $2 * 3 = 6$

X-ray: $2^0 * 2 * 3 = 6$, $2 * 2 * 3 = 12$

Dyspnoea: $2^0 * 2 * 2 = 4$, $2^3 = 8$

Smoking triple-valued, all others binary random variables:

- Concise version: 2+3+3+6+4=18
- Full enumeration: 3+6+6+12+8=35

Today: Structure Learning

- It is often difficult, if not impossible, to manually specify the structure of a Bayesian network.
  - This difficulty increases according to the number of random variables involved.
  - Due to the lack of agreement between domain experts

- Bayesian network learning is defined as

$$\underbrace{\Pr(\mathscr{B}|\mathscr{D}) = \Pr(G,\Theta|\mathscr{D})}_{\text{learning}} = \underbrace{\Pr(G|\mathscr{D})}_{\text{structure learning}} \cdot \underbrace{\Pr(\Theta|G,\mathscr{D})}_{\text{parameter learning}} .$$

Bayesian network

Data

Directed Acyclic Graph (DAG)

Parameters (probability distributions)

Complexity of Structure Learning

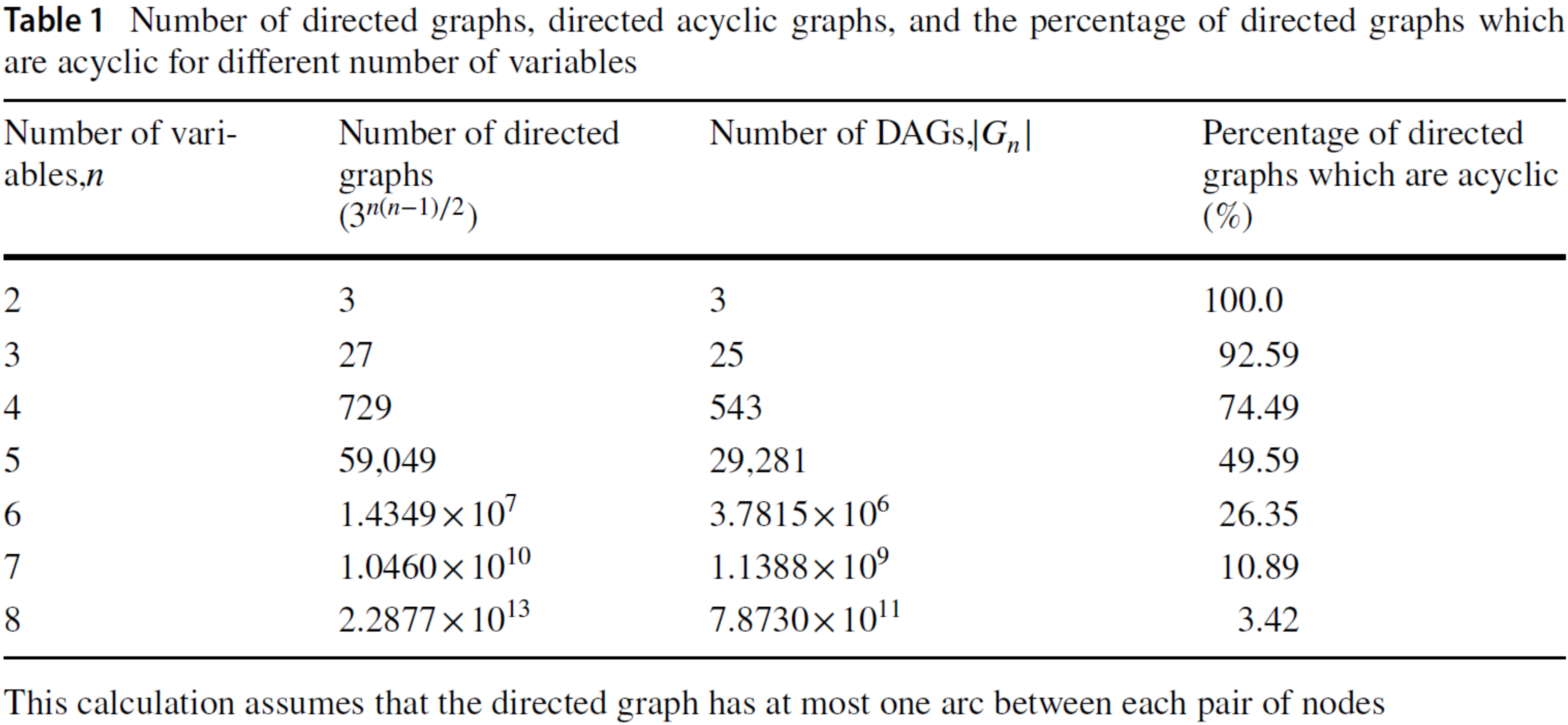


Table 1 Number of directed graphs, directed acyclic graphs, and the percentage of directed graphs which are acyclic for different number of variables

Number of variables,n	Number of directed graphs ($3^{n(n-1)/2}$)	Number of DAGs,$\lvert G_n \rvert$	Percentage of directed graphs which are acyclic (%)
2	3	3	100.0
3	27	25	92.59
4	729	543	74.49
5	59,049	29,281	49.59
6	1.4349×10^7	3.7815×10^6	26.35
7	1.0460×10^{10}	1.1388×10^9	10.89
8	2.2877×10^{13}	7.8730×10^{11}	3.42

This calculation assumes that the directed graph has at most one arc between each pair of nodes

Picture from: A survey of Bayesian Network structure learning | Artificial Intelligence Review (springer.com)

Today

- **Constraint-based structure learning**
  - PC-stable algorithm

- Score-based structure learning
  - Greedy search (Hill Climbing)

- Hybrid algorithms
  - Min-Max Hill Climbing

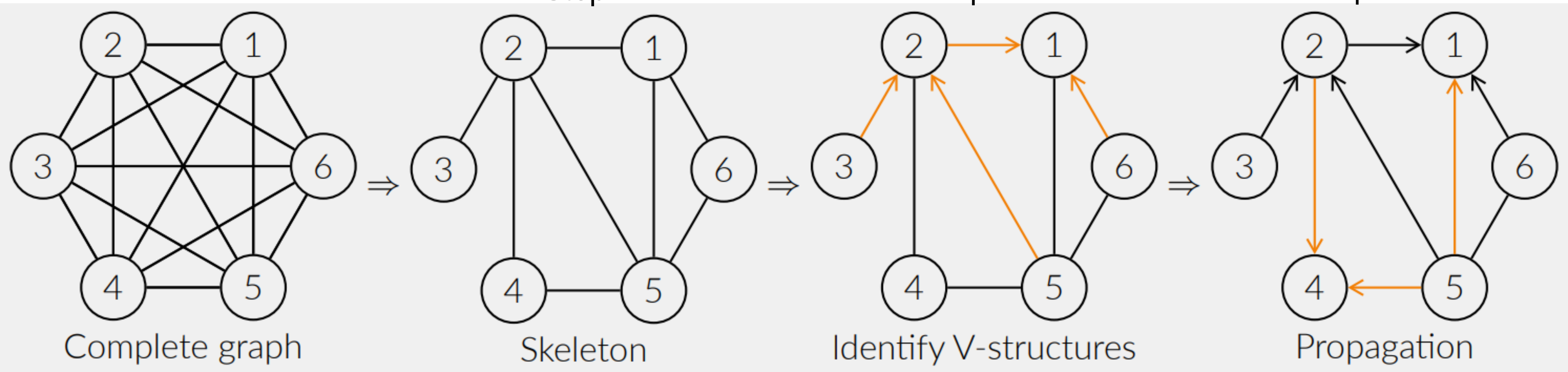
- Evaluation metrics for probabilistic models

PC-Stable Algorithm: Key Steps

Step 1

Step 2

Step 3

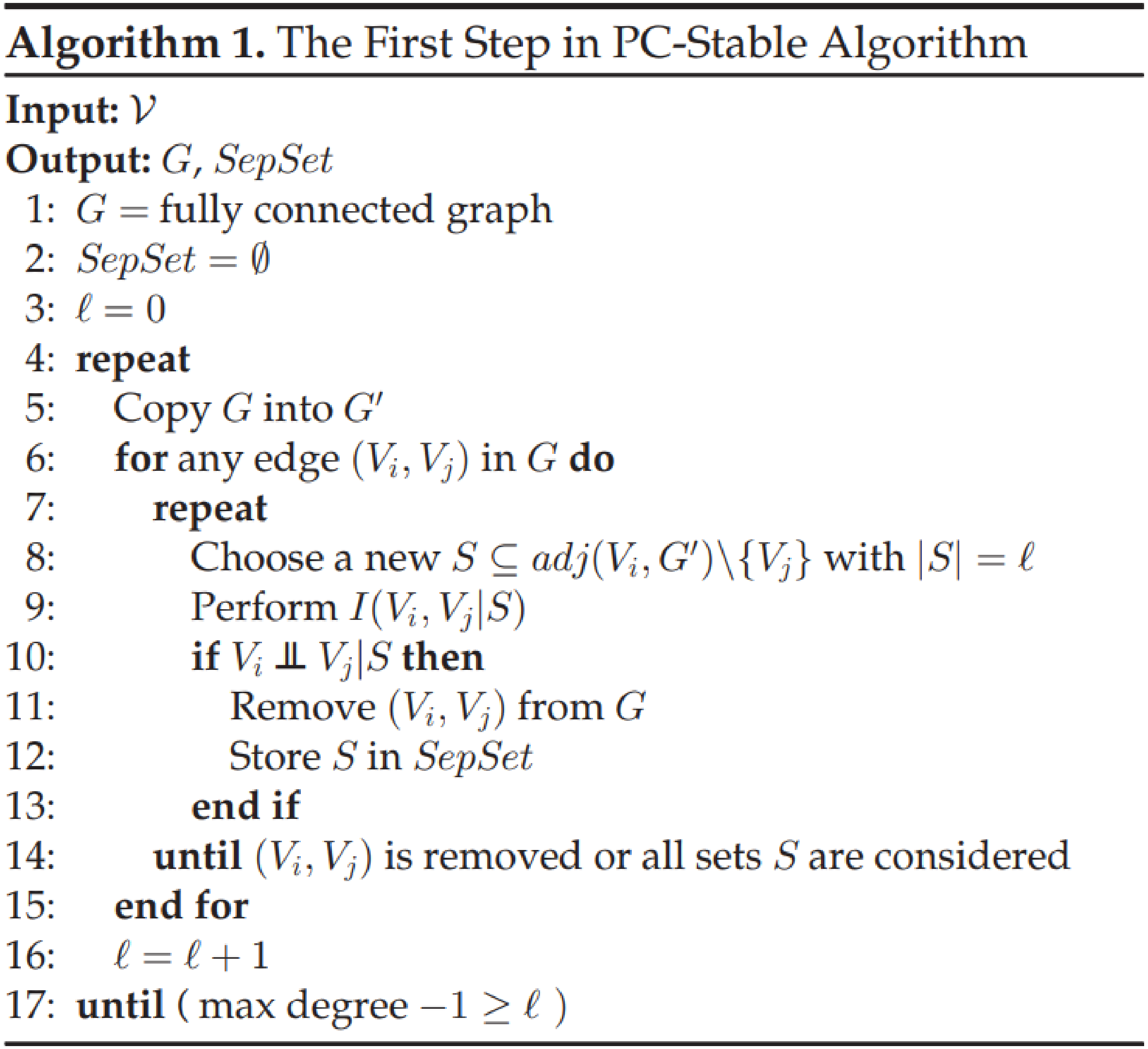


computationally intensive (depending on data size and network complexity)

Steps 2 and 3 use rules not covered in this module

Picture from: https://2019.ds3-datascience-polytechnique.fr/wp-content/uploads/2019/06/DS3-403_2019.pdf

PC-Stable: Find Skeleton



Algorithm 1. The First Step in PC-Stable Algorithm

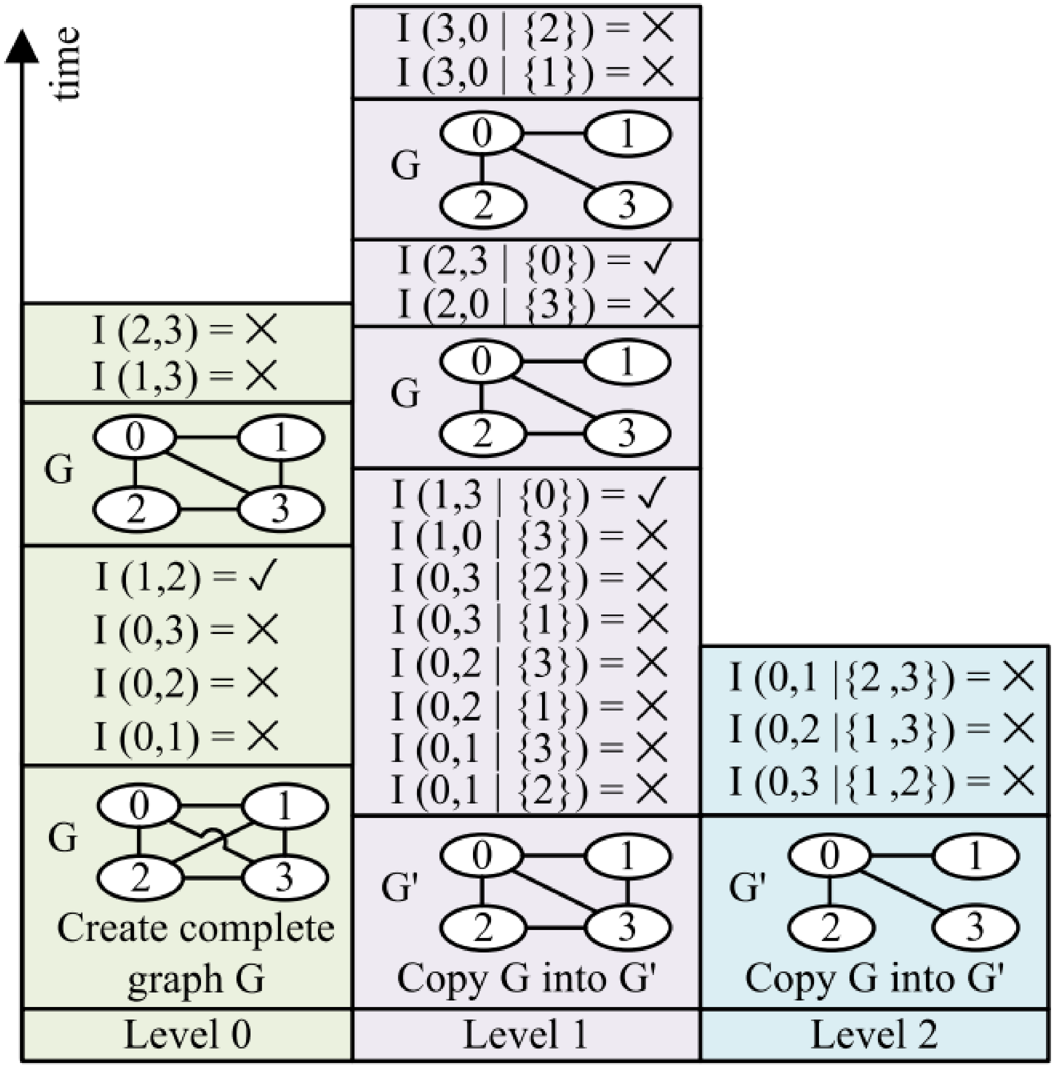
Input: $\mathcal{V}$

Output: G, $SepSet$

```
G = fully connected graph
$SepSet = \emptyset$
$\ell = 0$
repeat
    Copy $G$ into $G'$
    for any edge $(V_i, V_j)$ in $G$ do
        repeat
            Choose a new $S \subseteq adj(V_i, G') \setminus \{V_j\}$ with $|S| = \ell$
            Perform $I(V_i, V_j | S)$
            if $V_i \perp\!\!\!\perp V_j | S$ then
                Remove $(V_i, V_j)$ from $G$
                Store $S$ in $SepSet$
            end if
        until $(V_i, V_j)$ is removed or all sets $S$ are considered
    end for
    $\ell = \ell + 1$
until ( max degree $-1 \geq \ell$ )
```

Zarebavani, B., et al. cuPC: CUDA-Based Parallel PC Algorithm for Causal Structure Learning on GPU. IEEE TPDS, Vol. 31(3), 2020.

PC-Stable: Example Skeleton



Zarebavani, B., et al. cuPC: CUDA-Based Parallel PC Algorithm for Causal Structure Learning on GPU. IEEE TPDS, Vol. 31(3), 2020.

Conditional Independence Tests

Assuming that each arc (in a Bayesian network) encodes a probabilistic dependence, conditional independence tests tell us whether such probabilistic dependence is supported by the data.

Any two variables are associated in a Bayes net if the test's p-value < significance_level.

typically **0.05** or **0.01**
(default=0.05)

If *p*-value < significance level: keep edges

Else: remove edges

Conditional Independence Tests

Null hypothesis H_0: two random variables (e.g., *A*, *B*) are conditionally independent given variable *S*.

Statistical tests produce a test statistic, *p-value*, to estimate how likely the observed data is given H_0.

If *p-value* ≤ significance_level:

  reject $\boldsymbol{H_0}$, i.e., A and B conditionally dependent given S

Else:

  accept $\boldsymbol{H_0}$, i.e., A & B conditionally independent given S

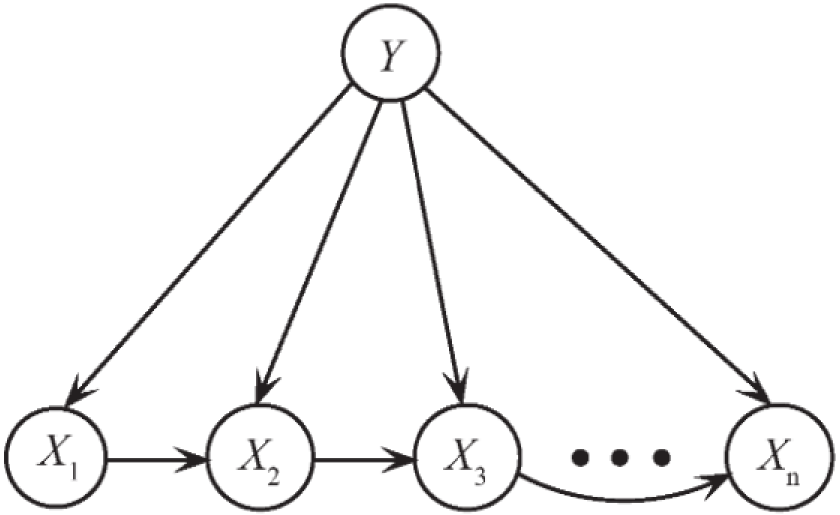
API for Conditional Independence Tests

```
bnlearn.bnlearn.independence_test(model, data
, test='chi_square', alpha=0.05, prune=False,
 verbose=3)
```

The statistical tests:

- chi_square
- g_sq
- log_likelihood
- freeman_tuckey
- modified_log_likelihood
- neyman
- cressie_read

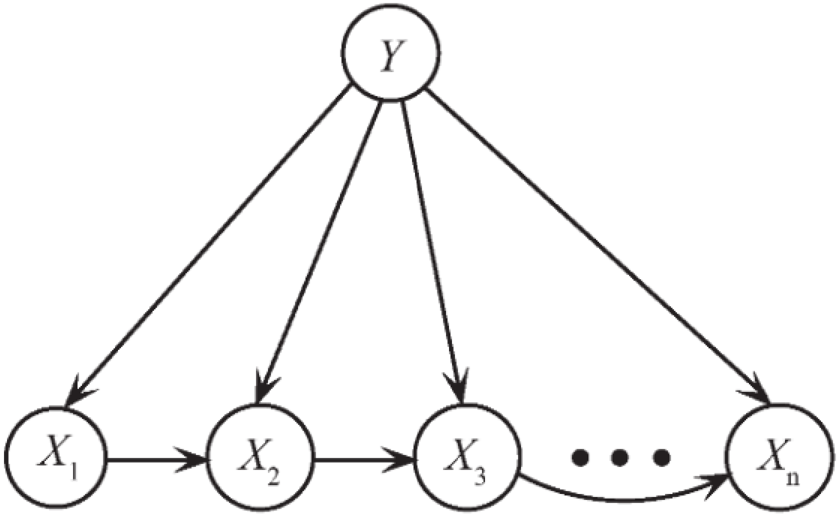
Conditional Independence Tests using the Language Detection data



Is the link between X_1 and X_2 with a p-value of 0.0 needed?

Is the link between X_1 and X_{15} with a p-value of 0.4692 needed?

Conditional Independence Tests using the Language Detection data



Is the link between X_1 and X_2 with a p-value of 0.0 needed?

Yes, keep the link.

Is the link between X_1 and X_{15} with a p-value of 0.4692 needed?

No, remove the link.

Today

- Constraint-based structure learning
  - PC-stable algorithm

- **Score-based structure learning**
  - Greedy search (Hill Climbing)

- Hybrid algorithms
  - Min-Max Hill Climbing

- Evaluation metrics for probabilistic models

Greedy Search: Main Ideas

Explore the search space starting from an empty network and adding/deleting/reversing one arc at a time until the score can't be improved anymore.

Doing the above requires knowledge regarding how good a network is against alternative graphs.

The goal is to search for the network with the *highest score:* $\boldsymbol{B^* = \arg\max_{B} \; Score(B|D)}$

Bayesian network

Data

Scoring Function

The score represents a trade-off:

- How well the network fits the data &
- How complex the network is

Assumption: the scoring function is decomposable

$$Score(B|D) = \sum_{i=1}^{n} Score(X_i|parents(X_i), D)$$

random variable *i*

data

Several scoring functions can been applied to structure learning such as LL, BIC, AIC, etc., ...

Liu, Z., Malone, B., Yuan, C. “Empirical Evaluation of Scoring Functions for BN Model Selection”, BMC Bio, 2012

Scoring Functions: Log-Likelihood (LL)

LL represents the probability of the data (*D*) given a Bayesian network (*B*):

$$
\begin{aligned}
LL(D|B) &= \sum_j^N \log P(D_j|B) \\
&= \sum_i^n \sum_j^N \log P(D_{ij}|PA_{ij})
\end{aligned}
$$

where D_{ij} is the value of random variable X_i in training example D_j, and PA_{ij} are the values of the parents of X_i in D_j.

Liu, Z., Malone, B., Yuan, C. "Empirical Evaluation of Scoring Functions for BN Model Selection", BMC Bio, 2012

LL Example for PlayTennis Data using a Naïve Bayes Structure

$$LL = \sum_{i=1}^{N} \log P(PlayTennis_i) + \sum_{i}^{n} \sum_{j}^{N} \log P(X_{ij}|PA_i)$$

$$\begin{aligned} LL_1 \\ &= \log P(PlayTennis = no) \\ &+ \log P(Outlook = sunny|PlayTennis = no) + \\ &+\log P(Temperature = hot|PlayTennis = no) + \\ &+\log P(Humidity = high|PlayTennis = no) + \\ &+\log P(Wind = weak|PlayTennis = no) \\ &= \log\left(\frac{5}{14}\right) + \log\left(\frac{3}{5}\right) + \log\left(\frac{2}{5}\right) + \log\left(\frac{4}{5}\right) + \log\left(\frac{2}{5}\right) = -3.5961 \end{aligned}$$

and so on for $LL_1 + \cdots + LL_{14} = -54.217$

Scoring Functions: Bayesian Information Criterion (BIC)

BIC is a penalised LL function:

$$BIC(D|B) = LL(D|B) - \sum_{i=1}^{N} Penalty(X_i, B, D)$$

where $LL(.)$ is Log-Likelihood function and the penalty term penalises complex networks as $Penalty(X_i, B, D) = \frac{\log N * p_i}{2}$, p_i is the number of parameters/probabilities of random variable X_i, and N is the number of training examples.

Liu, Z., Malone, B., Yuan, C. "Empirical Evaluation of Scoring Functions for BN model Selection", BMC Bio, 2012

BIC Example for PlayTennis Data using a Naïve Bayes Structure

$$BIC(D|B) = LL(D|B) - \sum_i \frac{\log N * p_i}{2}$$

$$= LL(D|B) - \frac{\log N * \sum_i p_i}{2}$$

$$BIC(D|B) = -54.217 - \frac{\log(14) * 22}{2}$$

$$= -83.246$$

N.B. The above assume CPTs with full enumeration (not concise ones)

Scoring Functions: Aikake Information Criterion (AIC)

AIC is a penalised LL function:

$$AIC(D|B) = LL(D|B) - \sum_{i=1}^{N} Penalty(X_i, B, D)$$

where $LL(.)$ is Log-Likelihood function and $Penalty(X_i, B, D) = p_i$ with p_i being the number of parameters/probabilities of random variable X_i, and N is the number of training examples.

AIC favours more complex networks than BIC.

Liu, Z., Malone, B., Yuan, C. “Empirical Evaluation of Scoring Functions for BN model Selection”, BMC Bio, 2012

AIC Example for PlayTennis Data using a Naïve Bayes Structure

$$AIC(D|B) = LL(D|B) - \sum_{i=1}^{N} p_i$$

$$\begin{aligned} AIC(D|B) &= -54.217 - \underbrace{(2+6+6+4+4)} \\ &= -76.217 \end{aligned}$$

Using full enumeration ->

| CPT(PT)|=2
|CPT(O|PT)|=6
|CPT(T|PT)|=6
|CPT(W|PT)|=4
|CPT(H|PT)|=4

AIC Example for PlayTennis Data using a Naïve Bayes Structure

$$AIC(D|B) = LL(D|B) - \sum_{i=1}^{N} p_i$$

$$\begin{aligned} AIC(D|B) &= -54.217 - \underbrace{(1 + 3 + 3 + 2 + 2)} \\ &= -65.217 \end{aligned}$$

Using concise enumeration ->

| CPT(PT)|=1

|CPT(O|PT)|=3

|CPT(T|PT)|=3

|CPT(W|PT)|=2

|CPT(H|PT)|=2

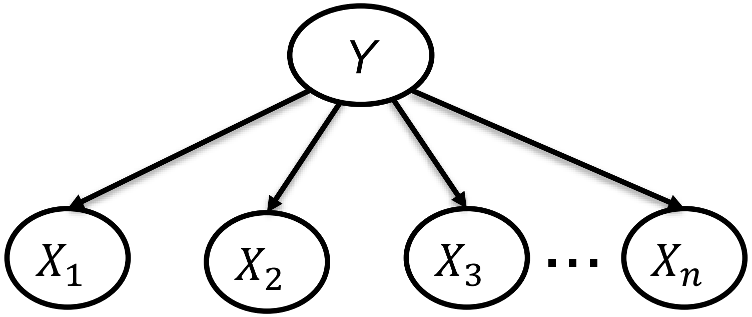
API for Structure Learning with Scoring Functions

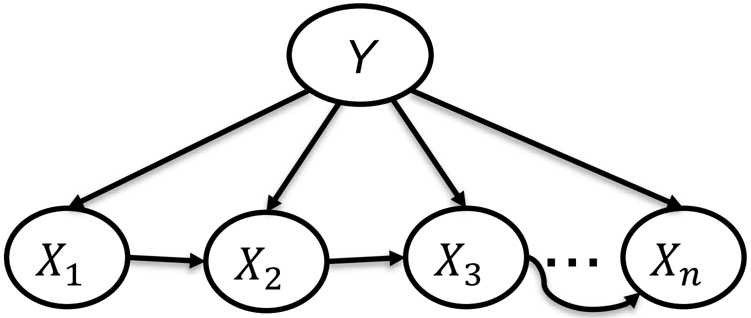
```
bnlearn.structure_learning.fit(data,
methodtype='hc', scoretype='bic',
black_list=None, white_list=None, …,
max_iter=1000000.0, …, n_jobs=-1, verbose=3)
```

The statistical tests:

- ‘bic’
- ‘k2’
- ‘bdeu’
- ‘bds’
- ‘aic’

LL/BIC/AIC for Language Detection



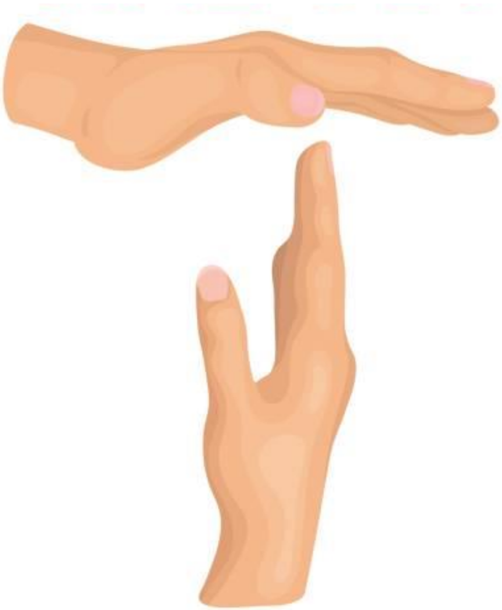


LL=???
BIC=???
AIC=???

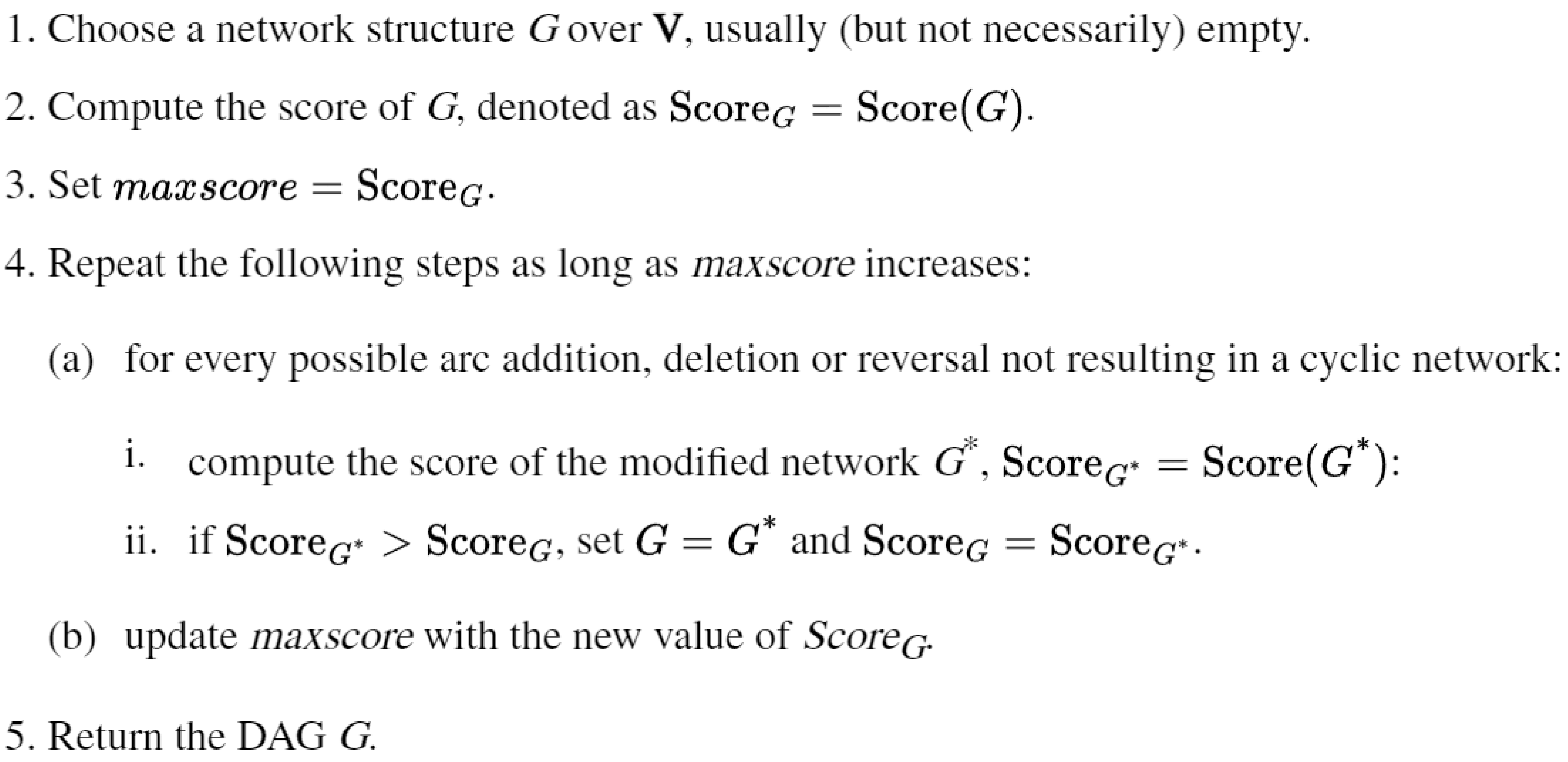
LL=???
BIC=???
AIC=???

Which structure above fits better the data?

Break



Hill Climbing Algorithm



1. Choose a network structure G over $\mathbf{V}$, usually (but not necessarily) empty.
2. Compute the score of G, denoted as $\mathrm{Score}_G = \mathrm{Score}(G)$.
3. Set $maxscore = \mathrm{Score}_G$.
4. Repeat the following steps as long as *maxscore* increases:
   - (a) for every possible arc addition, deletion or reversal not resulting in a cyclic network:
     - i. compute the score of the modified network G^*, $\mathrm{Score}_{G^*} = \mathrm{Score}(G^*)$:
     - ii. if $\mathrm{Score}_{G^*} > \mathrm{Score}_G$, set $G = G^*$ and $\mathrm{Score}_G = \mathrm{Score}_{G^*}$.
   - (b) update *maxscore* with the new value of $Score_G$.
5. Return the DAG G.

Scutari, M. & Denis, JB. “Bayesian Networks with Examples in R”, Routledge Taylor & Francis, 2022, chapter 6.

Hill Climbing Algorithm: Considerations

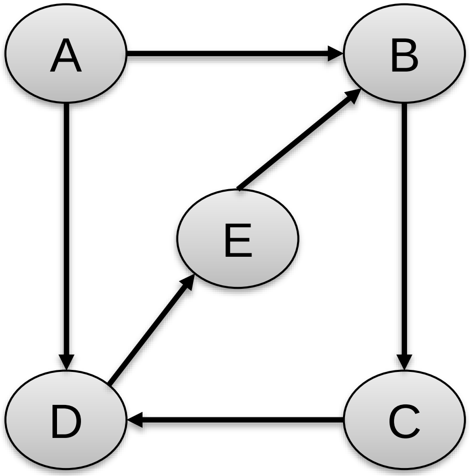
- How many structures to try out?
- How to many changes per iteration, 1 or >1?
- How long (in terms of time) to search for?
- What scoring function to use?
- What structure to start from (empty, non empty)?

Hill Climbing Algorithm: Considerations

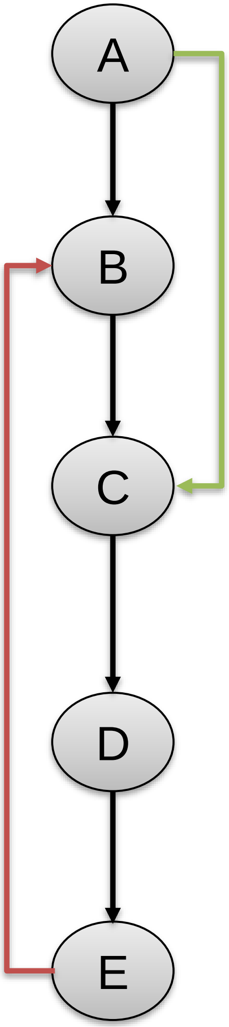
- How many structures to try out?
- How to many changes per iteration, 1 or >1?
- How long (in terms of time) to search for?
- What scoring function to use?
- What structure to start from (empty, non empty)?

Answers depend on your choices, e.g., your own code or publicly available code.

Depth First Search (DFS) for Detecting Cyclic Networks



DAGs do not have back edges



NOTATION:

→ Tree edges

→ Forward edges

→ Back edges

Today

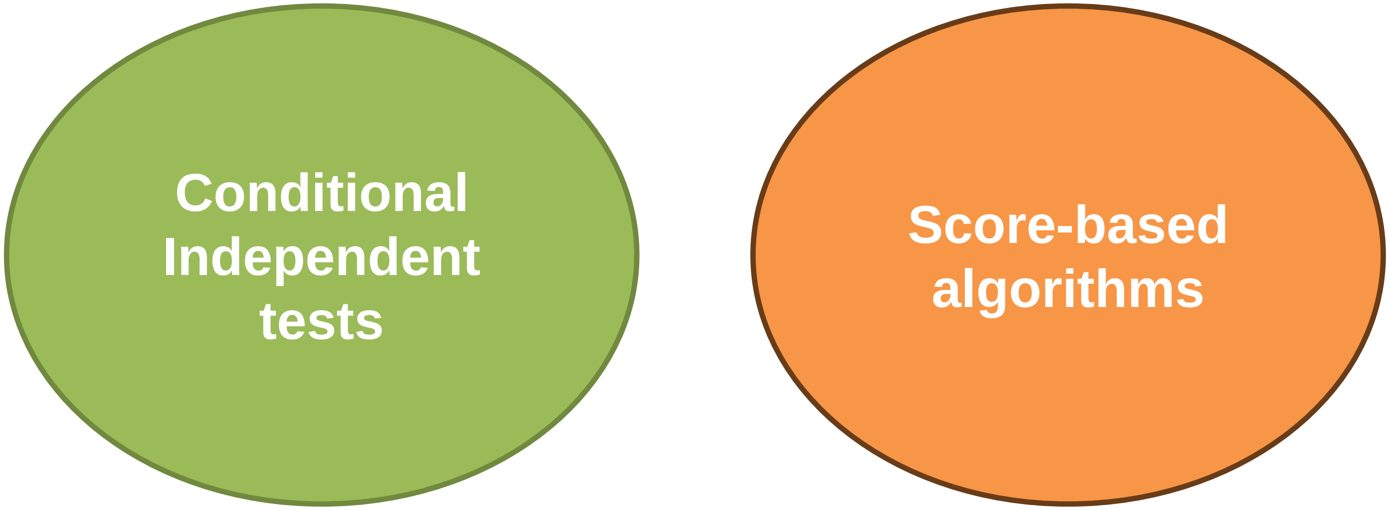
- Constraint-based structure learning
  - PC-stable algorithm

- Score-based structure learning
  - Greedy search (Hill Climbing)

- **Hybrid algorithms**
  - Min-Max Hill Climbing

- Evaluation metrics for probabilistic models

Hybrid Algorithms for Structure Learning in Bayesian Networks

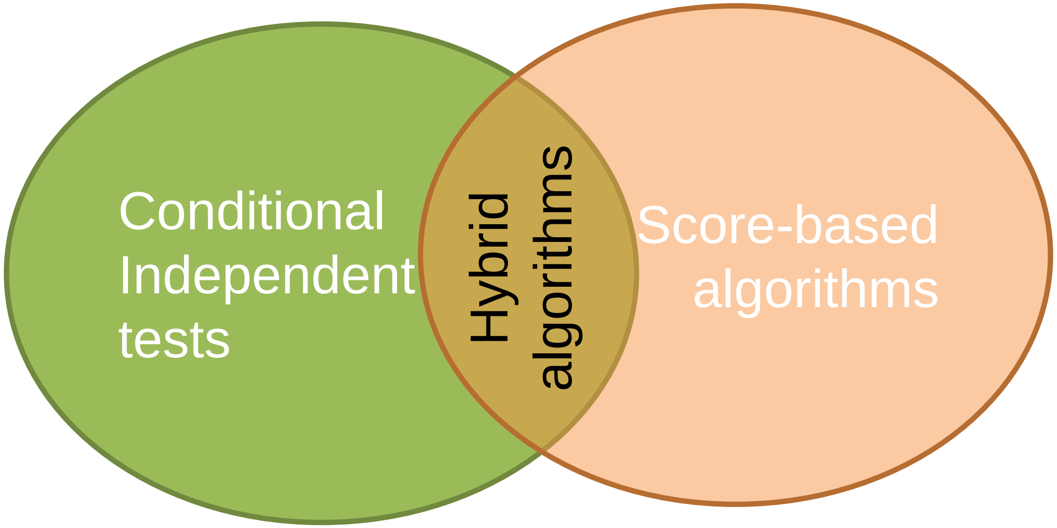


Conditional Independent tests

Score-based algorithms

- Conditional Independent tests are partially rule-based
- Score-based approaches can be search intensive
- Is there any other alternative?

Hybrid Algorithms for Structure Learning in Bayesian Networks



Since Conditional Independent tests and Score-based approaches have pros and cons, hybrid algorithms aim for **the best of both worlds**.

MMHC: Min-Max Hill Climbing

1. Choose a network structure G over V (nodes), usually empty (but not necessarily).
2. **Restrict:** select a set **C**$_i$ of candidate parents for each node $X_i \in V$, which must include the parents of X_i in G.

   via **the PC-stable** algorithm

3. **Maximise:** find the network structure G^* that maximises $Score(G^*)$ among the networks in which the parents of each node X_i are included in the corresponding set **C**$_i$.

   via the **Hill Climbing** algorithm

4. Return DAG G^*.

Today

- Constraint-based structure learning
  - PC-stable algorithm

- Score-based structure learning
  - Greedy search (Hill Climbing)

- Hybrid algorithms
  - Min-Max Hill Climbing

- **Evaluation metrics for probabilistic models**

Metrics for Probabilistic Models

- The higher the better:
  - Balanced accuracy, [0,…,1]
  - F1 Score, [0,…,1]
  - Area under curve, [0,…,1]

- The lower the better
  - Brier score, [0,…,1]
  - KL divergence, [0,…,∞]
  - Training time (in seconds)
  - Inference time (in seconds)

Metrics: Balanced Accuracy

Predicted Label/Condition

Actual Label or Condition

	POSITIVE (PP)	NEGATIVE (PN)
POSITIVE (P)	True Positive (TP)	False Negative (FN)
NEGATIVE (N)	False Positive (FP)	True Negative (TN)

$$BA = \frac{TPR + TNR}{2}, \text{where}$$

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR \qquad FNR = \frac{FN}{P} = \frac{FN}{FN + TP} = 1 - TPR$$

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP} = 1 - FPR \qquad FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - TNR$$

If $FNR = FPR = 0$ then $TPR + TNR = 2$

Metrics: F1 Score

Predicted Label/Condition

Actual Label or Condition

	POSITIVE (PP)	NEGATIVE (PN)
POSITIVE (P)	**True Positive (TP)**	**False Negative (FN)**
NEGATIVE (N)	**False Positive (FP)**	**True Negative (TN)**

$$F_1 = 2 * \left(\frac{\mathit{Precision} * \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}}\right), \text{where}$$

$$\mathit{Precision} = \frac{TP}{PP} = \frac{TP}{TP + FP}$$

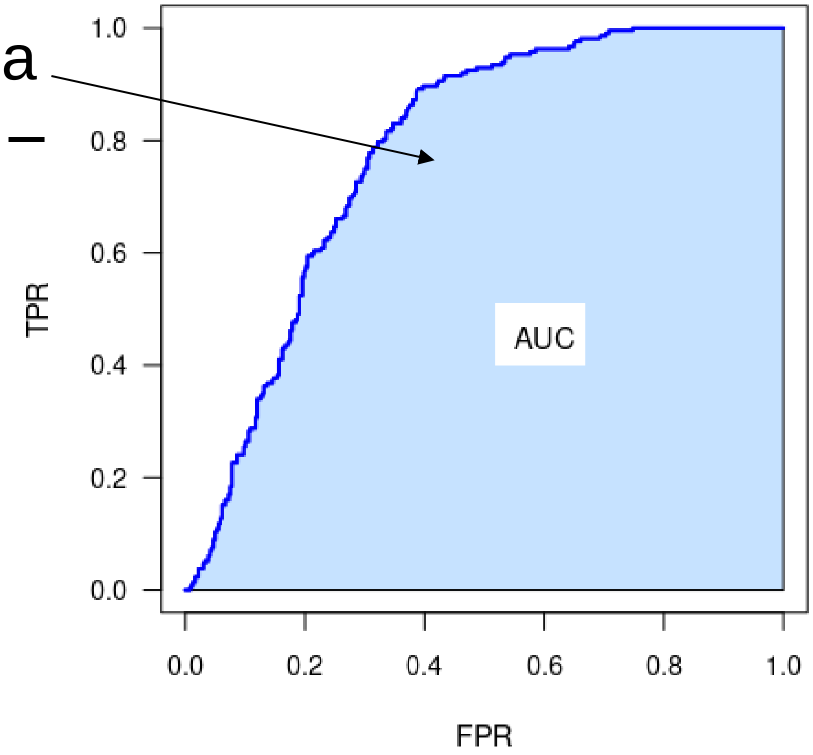
$$\mathit{Recall} = \frac{TP}{P} = \frac{TP}{TP + FN}$$

Metrics: Area Under the ROC Curve (AUC)

Predicted Label/Condition

Actual Label or Condition		POSITIVE (PP)	NEGATIVE (PN)
	POSITIVE (P)	True Positive (TP)	False Negative (FN)
	NEGATIVE (N)	False Positive (FP)	True Negative (TN)

AUC measures the entire area undern the entire ROC curve – the higher the area the better the classifier.



$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - TNR$$

Metrics: Area Under the ROC Curve (AUC)

Predicted Label/Condition

Actual Label or Condition

	POSITIVE (PP)	NEGATIVE (PN)
POSITIVE (P)	True Positive (TP)	False Negative (FN)
NEGATIVE (N)	False Positive (FP)	True Negative (TN)

AUC measures the entire area undern the entire ROC curve – the higher the area the better the classifier.

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} = 1 - FNR$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} = 1 - TNR$$

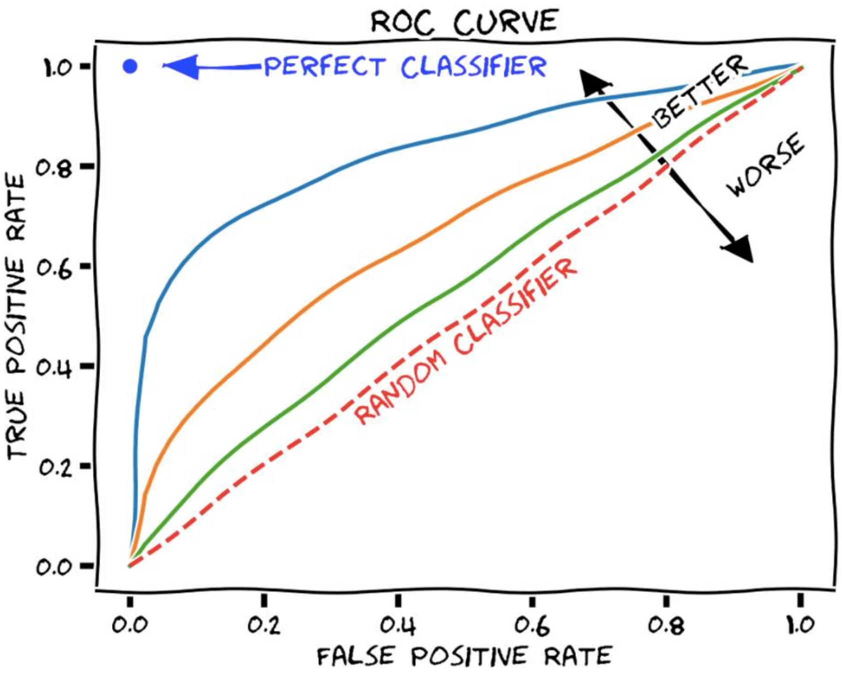


Figure from this link

Metrics for Probabilistic Models

- The higher the better:
  - Balanced accuracy, [0,…,1]
  - F1 Score, [0,…,1]
  - Area under curve, [0,…,1]

Predicted probability

Correct/True probability

- The lower the better
  - Brier score, [0,…,1] $$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$
  - KL divergence, [0,…,∞] $$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$$
  - Training time (in seconds)
  - Inference time (in seconds)

Metrics for Probabilistic Models

- The higher the better:
  - Balanced accuracy, [0,….,1]
  - F1 Score, [0,…,1]
  - Area under curve, [0,…,1]

- The lower the better
  - Brier score, [0,…,1]
  - KL divergence, [0,…,∞]
  - Training time (in seconds)
  - Inference time (in seconds)

$$BS = \frac{1}{N}\sum_{t=1}^{N}(f_t - o_t)^2$$

$$D_{\mathrm{KL}}(\underset{\substack{\uparrow \\ \text{True/Correct} \\ \text{distribution}}}{P} \parallel \underset{\substack{\uparrow \\ \text{Predicted} \\ \text{distrubution}}}{Q}) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$$

KL Divergence: Example (1/2)

- True Distribution P = <1.0, 0.0>
- Predicted distribution Q = <0.6, 0.4>

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

$$\begin{aligned} D_{KL}(P||Q) &= P(x_1)\log\left(\frac{P(x_1)}{Q(x_1)}\right) + P(x_2)\log\left(\frac{P(x_2)}{Q(x_2)}\right) \\ &= 1.0\log\left(\frac{1.0}{0.6}\right) + 0.0\log\left(\frac{0.0}{0.4}\right) \\ &= 0.5108 + 0 \end{aligned}$$

KL Divergence: Example (2/2)

- True Distribution P = <1.0, 0.0>
- Predicted distribution Q = <0.4, 0.6>

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

$$\begin{aligned} D_{KL}(P||Q) &= P(x_1)\log\left(\frac{P(x_1)}{Q(x_1)}\right) + P(x_2)\log\left(\frac{P(x_2)}{Q(x_2)}\right) \\ &= 1.0\log\left(\frac{1.0}{0.4}\right) + 0.0\log\left(\frac{0.0}{0.6}\right) \\ &= 0.9162 + 0 \end{aligned}$$

N.B. This example uses the natural logarithm (base e) – default in Python. Other choices are: $\log_{1o}()$ and $\log_2()$

Metrics for Probabilistic Models

- The higher the better:
  - Balanced accuracy, [0,…,1]
  - F1 Score, [0,…,1]
  - Area under curve, [0,…,1]

- The lower the better
  - Brier score, [0,…,1]
  - KL divergence, [0,…,∞]
  - Training time (in seconds)
  - Inference time (in seconds)

Consider using these metrics in your assignment. Fine if you wish to include any other metric(s).

Today

- Discussion on structure learning
- Algorithms to induce the structure of Bayes nets
- Measuring the performance of probabilistic models/classifiers

Readings:

- Kitson et al. A survey on Bayesian Network structure learning, arXiv, 2023 (main reading)
- Liu et al. Empirical evaluation of scoring functions for Bayesian Network model selection, BMC Bioinformatics, 2012 (optional reading)

Next Week

Workshop (today):

Exercise on Conditional Independent tests

Exercise on scoring functions (BIC, AIC, BDeu)

Python code for structure learning

Lecture (next week):

Bayesian Networks: Approximate Inference

Reading: Russell & Norvig 2016. Section 14.5

Questions?