# Bayesian Networks
## and Their Extensions in Modern Machine Learning

Marco Scutari
scutari@idsia.ch

Dalle Molle Institute for
Artificial Intelligence (IDSIA)

April 8, 2021

# Overview

## Bayesian Networks

- Construction and fundamental properties
- Probability distributions
- Learning from data
- Inference

## Extensions

- Incomplete data
- Latent variables
- Time series
- Causal modelling
- Fairness

# A Graph and a Probability Distribution

A Bayesian network (BN) is defined by:

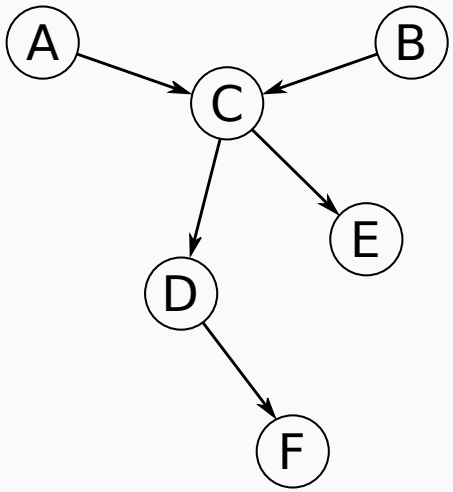
- a network structure, a directed acyclic graph $\mathcal{G}$, in which each node corresponds to a random variable $X_i$;
- a global probability distribution $\mathbf{X}$ with parameters $\Theta$, which can be factorised into smaller local probability distributions according to the arcs present in $\mathcal{G}$.

The main role of the network structure is to express the conditional independence relationships among the variables in the model through graphical separation, thus specifying the factorisation of the global distribution:

$$\mathrm{P}(\mathbf{X}) = \prod_{i=1}^{p} \mathrm{P}(X_i \mid \Pi_{X_i}; \Theta_{X_i}) \quad \text{where} \quad \Pi_{X_i} = \{\text{parents of } X_i\}.$$

# How the DAG Maps to the Probability Distribution

DAG




Graphical separation

$$
\begin{aligned}
&A \perp\!\!\!\perp_G B \\
&A \perp\!\!\!\perp_G D \mid C \\
&B \perp\!\!\!\perp_G D \mid C \\
&A \perp\!\!\!\perp_G E \mid C \\
&B \perp\!\!\!\perp_G E \mid C \\
&D \perp\!\!\!\perp_G E \mid C \\
&C \perp\!\!\!\perp_G F \mid D \\
&\dots
\end{aligned}
$$

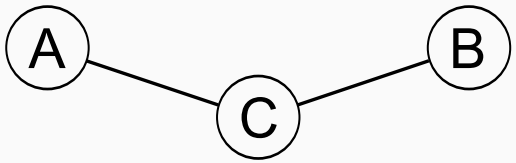
Probabilistic independence

$$
\begin{aligned}
&A \perp\!\!\!\perp_P B \\
&A \perp\!\!\!\perp_P D \mid C \\
&B \perp\!\!\!\perp_P D \mid C \\
&A \perp\!\!\!\perp_P E \mid C \\
&B \perp\!\!\!\perp_P E \mid C \\
&D \perp\!\!\!\perp_P E \mid C \\
&C \perp\!\!\!\perp_P F \mid D \\
&\dots
\end{aligned}
$$

Formally, the DAG is an independence map of the probability distribution of $\mathbf{X}$: graphical separation ($\perp\!\!\!\perp_G$) implies probabilistic independence ($\perp\!\!\!\perp_P$).

# Graphical Separation in DAGs (Fundamental Connections)

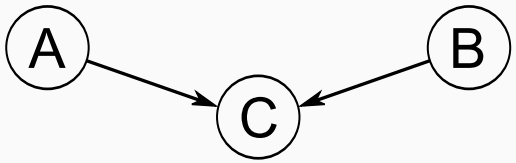
separation (undirected graphs)




$$\mathbf{A} \perp\!\!\!\perp \mathbf{B} \,|\, \mathbf{C}$$

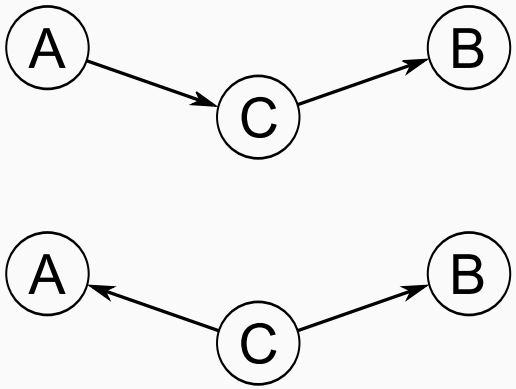
$$\mathrm{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathrm{P}(\mathbf{A} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{B} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{C})$$

d-separation (directed acyclic graphs)




$$\mathbf{A} \not\perp\!\!\!\perp \mathbf{B} \,|\, \mathbf{C}$$

$$\mathrm{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \mathrm{P}(\mathbf{C} \,|\, \mathbf{A}, \mathbf{B})\, \mathrm{P}(\mathbf{A})\, \mathrm{P}(\mathbf{B})$$




$$\mathbf{A} \perp\!\!\!\perp \mathbf{B} \,|\, \mathbf{C}$$

$$\begin{aligned} \mathrm{P}(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= \\ &= \mathrm{P}(\mathbf{B} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{C} \,|\, \mathbf{A})\, \mathrm{P}(\mathbf{A}) \\ &= \mathrm{P}(\mathbf{A} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{B} \,|\, \mathbf{C})\, \mathrm{P}(\mathbf{C}) \end{aligned}$$

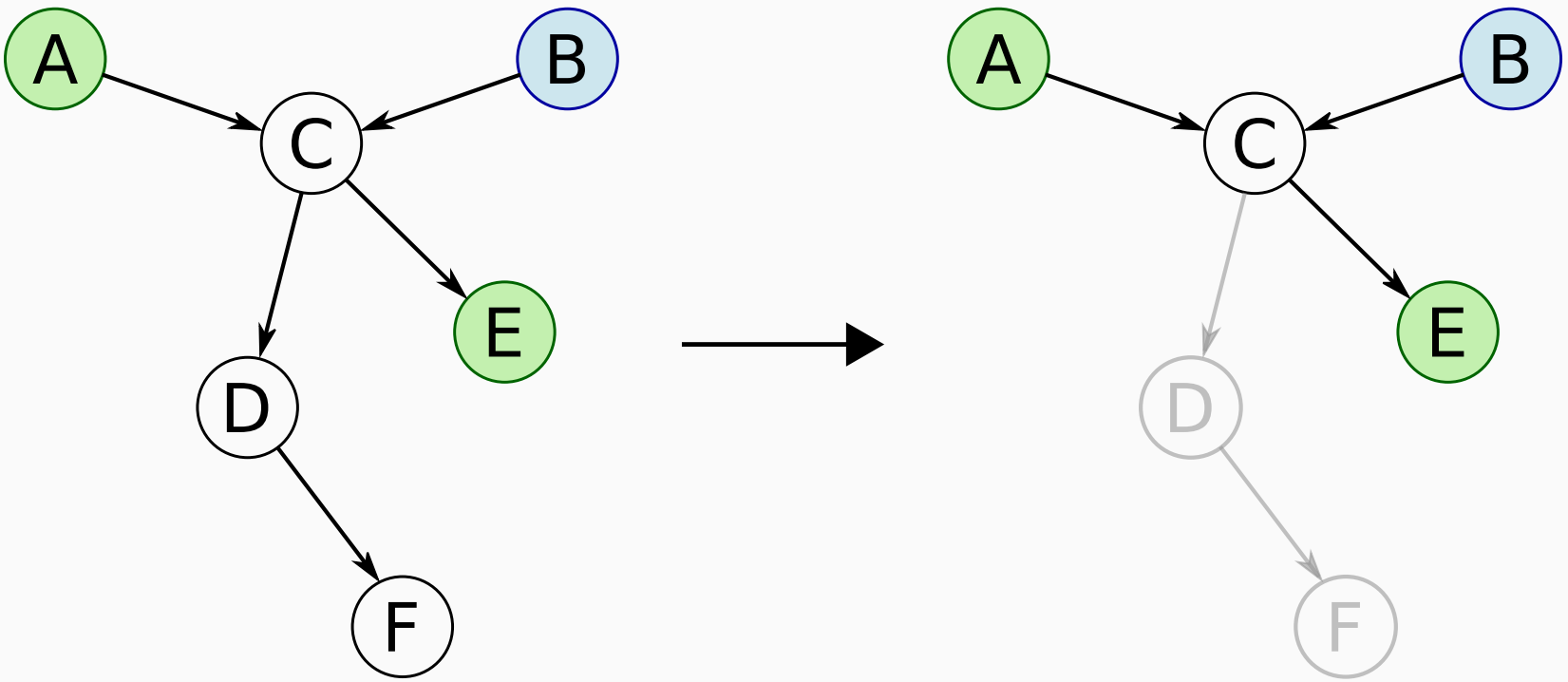
# Graphical Separation in DAGs (General Case)

In the general case we can extend the patterns from the fundamental connections and apply them to every possible path between $\mathbf{A}$ and $\mathbf{B}$ for a given $\mathbf{C}$: this is how d-separation is defined.

> *If $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are three disjoint subsets of nodes in a directed acyclic graph $\mathcal{G}$, then $\mathbf{C}$ is said to d-separate $\mathbf{A}$ from $\mathbf{B}$, denoted $\mathbf{A} \perp\!\!\!\perp_G \mathbf{B} \mid \mathbf{C}$, if along every path between a node in $\mathbf{A}$ and a node in $\mathbf{B}$ there is a node $v$ satisfying one of the following two conditions:*
>
> 1. *$v$ has converging edges (i.e. there are two edges pointing to $v$ from the adjacent nodes in the path) and none of $v$ or its descendants (i.e. the nodes that can be reached from $v$) are in $\mathbf{C}$.*
> 2. *$v$ is in $\mathbf{C}$ and does not have converging edges.*

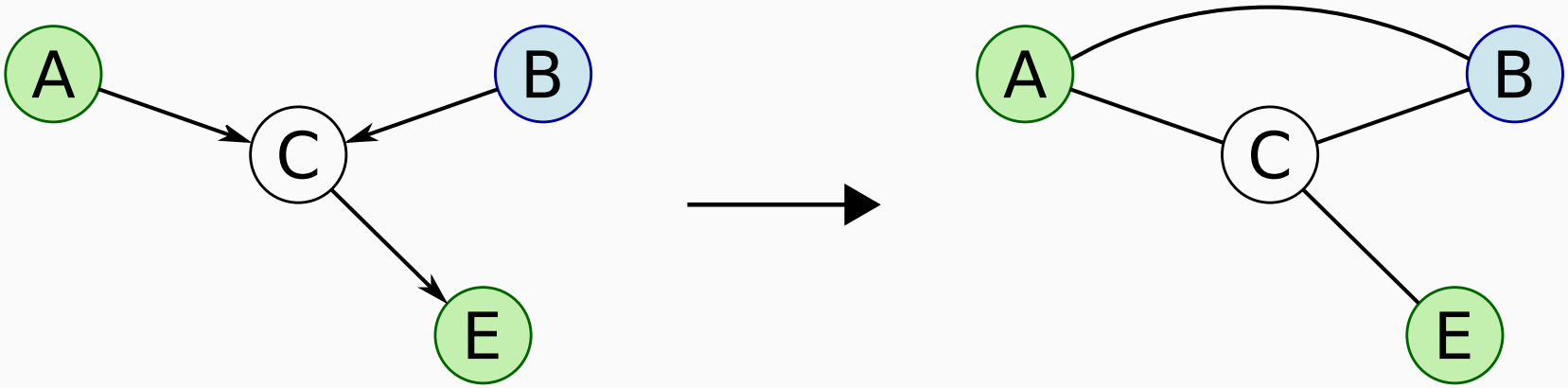
This definition clearly does not provide a computationally feasible approach to assess d-separation; but there are other ways.

# A Simple Algorithm to Check D-Separation (I)




Say we want to check whether $A$ and $E$ are d-separated by $B$. First, we can drop all the nodes that are not ancestors (the parents, parents' parents, etc.) of $A$, $E$ and $B$ since each node only depends on its parents.

# A Simple Algorithm to Check D-Separation (II)

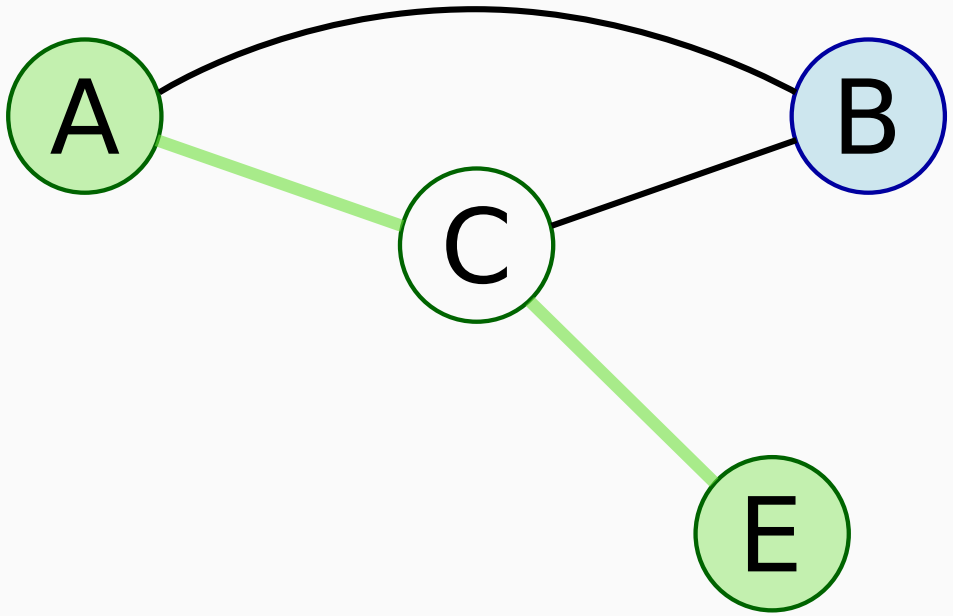



We transform the subgraph into its moral graph by

1. connecting all nodes that have one child in common; and
2. removing all arc directions to obtain an undirected graph.

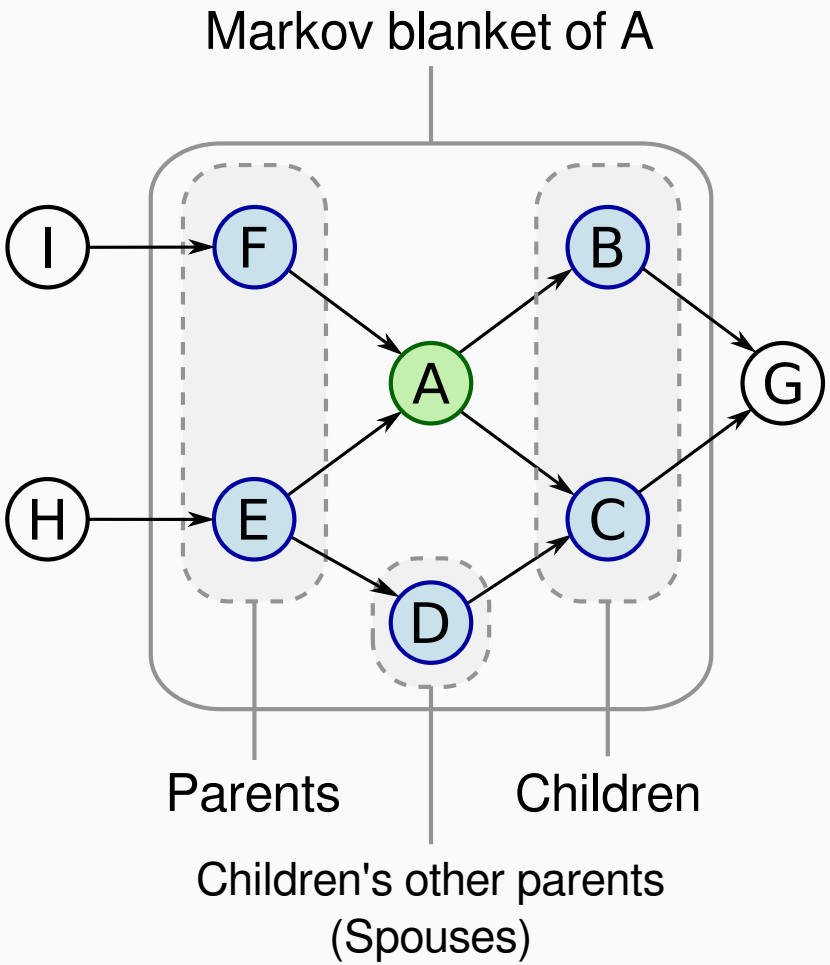
This transformation makes the dependence between parents explicit by "marrying" them and allows us to use the classic definition of graphical separation.

# A Simple Algorithm to Check D-Separation (III)




Finally, we can just perform a depth-first or breadth-first search and see if we can find an open path between $A$ and $B$, that is, a path that is not blocked by $C$.

# Completely D-Separating: Markov Blankets




We can easily use the DAG of a BN to solve the feature selection problem. The set of nodes that graphically isolates a target node from the rest of the DAG is called its Markov blanket and includes:

- its parents;
- its children;
- other nodes sharing a child.

Since $\perp\!\!\!\perp_G$ implies $\perp\!\!\!\perp_P$, we can restrict ourselves to the Markov blanket to perform any kind of inference on the target node, and disregard the rest.

# Different DAGs, Same Distribution: Topological Ordering

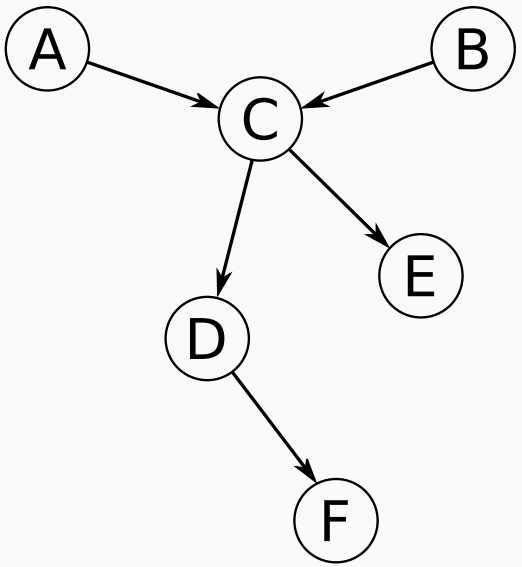
A DAG uniquely identifies a factorisation of $\mathrm{P}(\mathbf{X})$; the converse is not necessarily true. Consider again the DAG on the left:

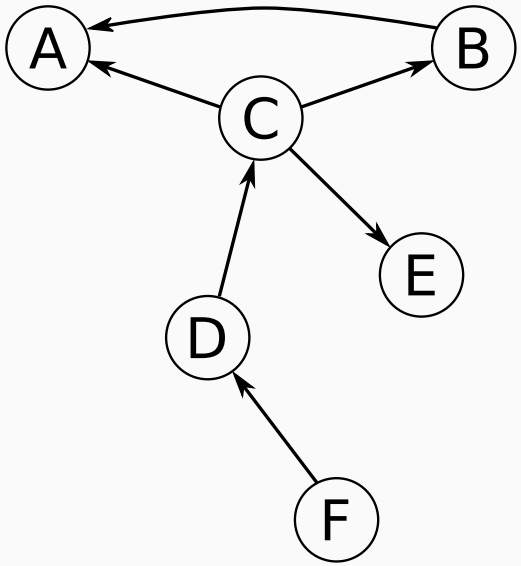
$$\mathrm{P}(\mathbf{X}) = \mathrm{P}(A)\,\mathrm{P}(B)\,\mathrm{P}(C \mid A, B)\,\mathrm{P}(D \mid C)\,\mathrm{P}(E \mid C)\,\mathrm{P}(F \mid D).$$

We can rearrange the dependencies using Bayes theorem to obtain:

$$\mathrm{P}(\mathbf{X}) = \mathrm{P}(A \mid B, C)\,\mathrm{P}(B \mid C)\,\mathrm{P}(C \mid D)\,\mathrm{P}(D \mid F)\,\mathrm{P}(E \mid C)\,\mathrm{P}(F),$$

which gives the DAG on the right, with a different topological ordering.



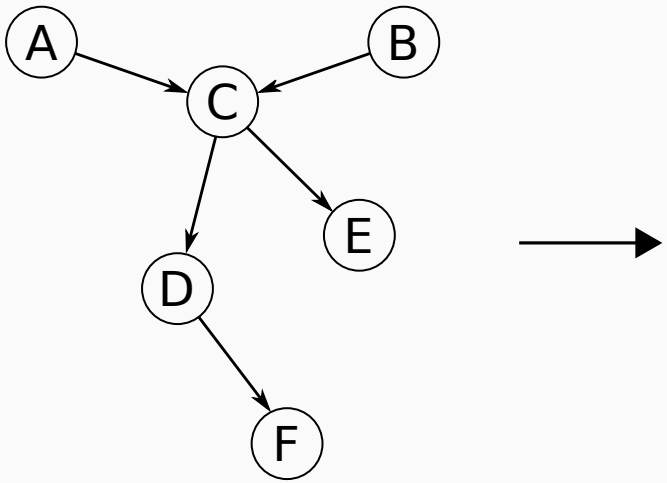
# Different DAGs, Same Distribution: Equivalence Classes

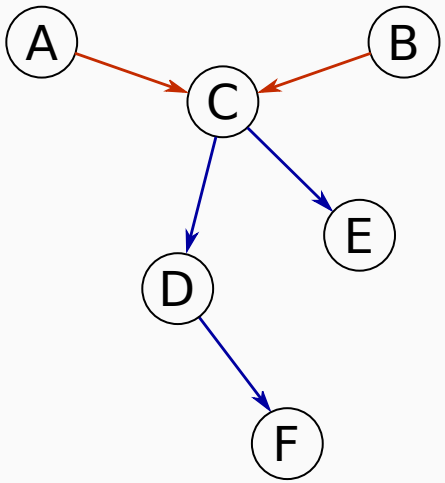
Even keeping the same underlying undirected graph we can reverse a number of arcs without changing the dependence structure of $\mathbf{X}$. Since the triplets $A \to B \to C$ and $A \leftarrow B \to C$ are probabilistically equivalent, we can reverse the directions of their arcs as we like as long as we do not create any new v-structure ($A \to B \leftarrow C$, with no arc between $A$ and $C$).

This means that we can group DAGs into equivalence classes that are uniquely identified by the underlying undirected graph and the v-structures. The directions of other arcs can be either:

- uniquely identifiable because one of the directions would introduce cycles or new v-structures in the graph (compelled arcs);
- completely undetermined.

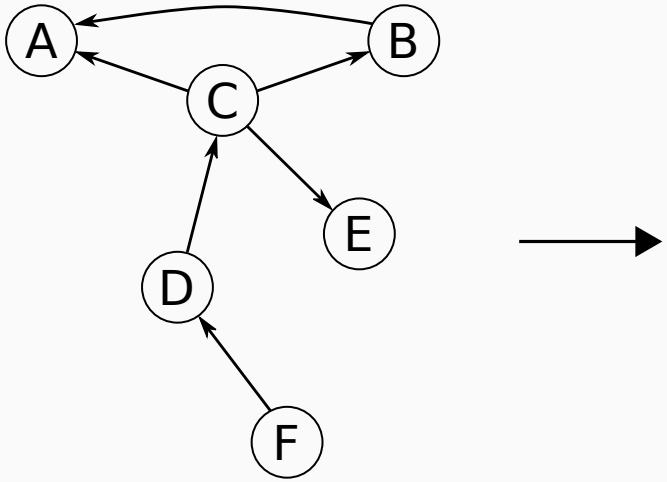
# Completed Partially Directed Acyclic Graphs (CPDAGs)

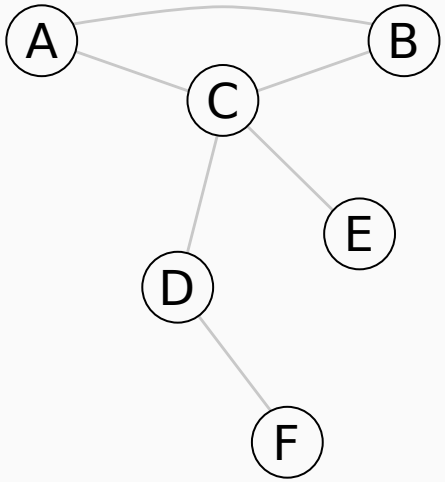



DAG

CPDAG



# What About the Probability Distributions?

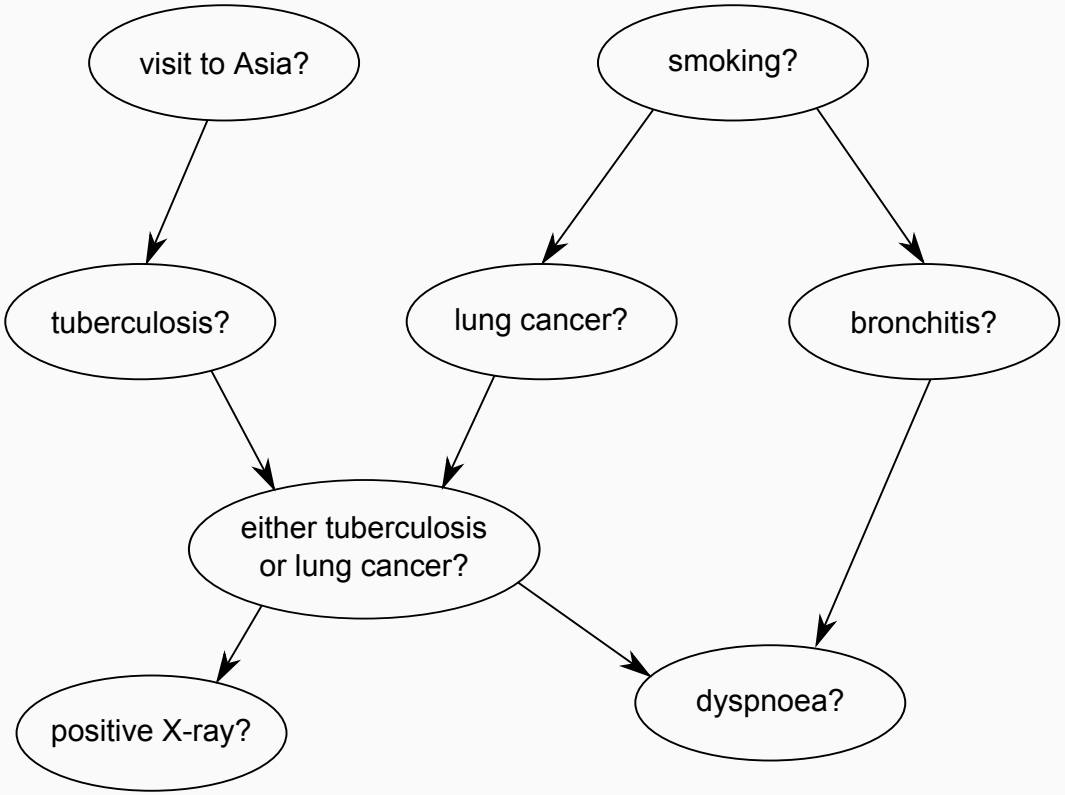
The second component of a BN is the probability distribution $\mathrm{P}(\mathbf{X})$. The choice should such that the BN:

- can be learned efficiently from data;
- is flexible (distributional assumptions should not be too strict);
- is easy to query to perform inference.

The three most common choices in the literature (by far), are:

- discrete BNs (DBNs), in which $\mathbf{X}$ and the $X_i \mid \Pi_{X_i}$ are multinomial;
- Gaussian BNs (GBNs), in which $\mathbf{X}$ is multivariate normal and the $X_i \mid \Pi_{X_i}$ are univariate normal;
- conditional linear Gaussian BNs (CLGBNs), in which $\mathbf{X}$ is a mixture of multivariate normals and the $X_i \mid \Pi_{X_i}$ are either multinomial, univariate normal or mixtures of normals.

It has been proved in the literature that exact inference is possible in these three cases, hence their popularity.

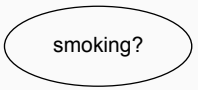



A classic example of DBN is the ASIA network from Lauritzen & Spiegelhalter (1988), which includes a collection of binary variables. It describes a simple diagnostic problem for tuberculosis and lung cancer.

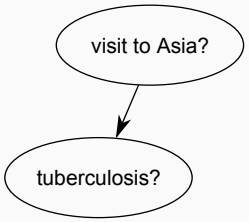
Total parameters of $\mathbf{X}$ :
$2^8 - 1 = 255$

# Conditional Probability Tables (CPTs)

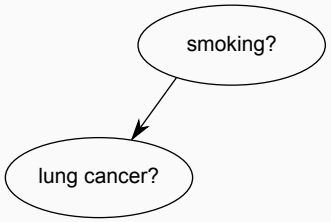

| yes | no |
|---|---|
| 0.01 | 0.99 |



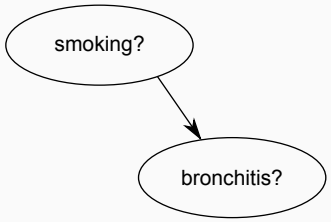

| yes | no |
|---|---|
| 0.50 | 0.50 |



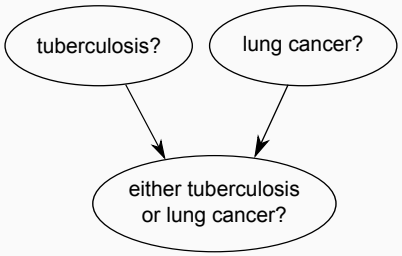

| | yes | no |
|---|---|---|
| yes | 0.05 | 0.95 |
| no | 0.01 | 0.99 |



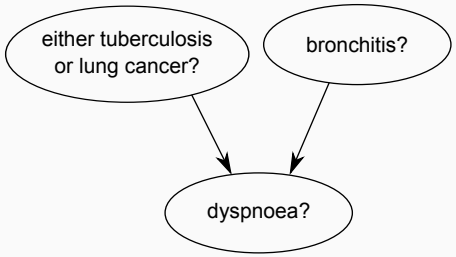

| | yes | no |
|---|---|---|
| yes | 0.10 | 0.90 |
| no | 0.01 | 0.99 |



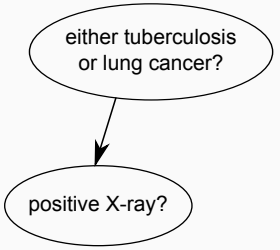

| | yes | no |
|---|---|---|
| yes | 0.60 | 0.40 |
| no | 0.30 | 0.70 |




| | yes | no |
|---|---|---|
| yes:yes | 1 | 0 |
| yes:no | 1 | 0 |
| no:yes | 1 | 0 |
| no:no | 0 | 1 |



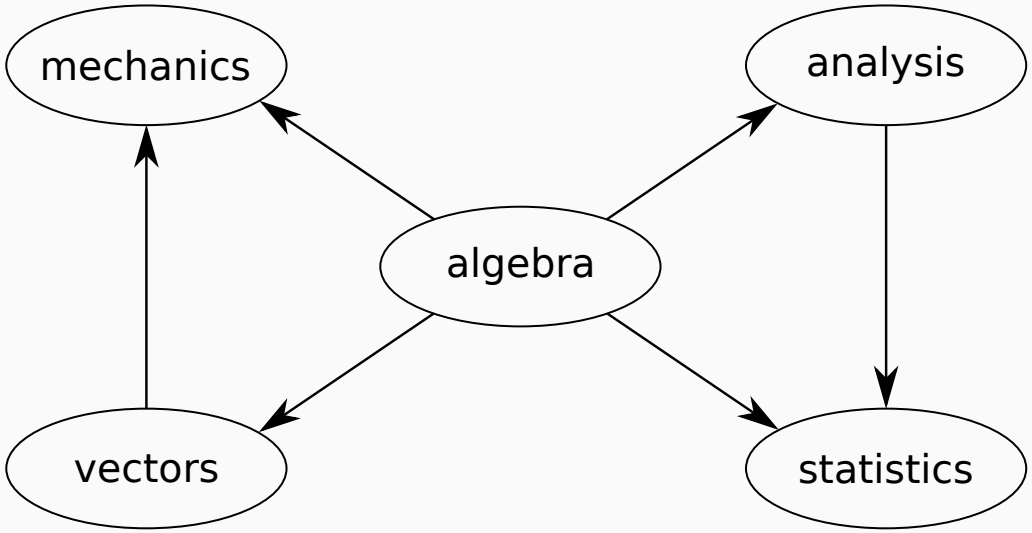

| | yes | no |
|---|---|---|
| yes:yes | 0.90 | 0.10 |
| yes:no | 0.70 | 0.30 |
| no:yes | 0.80 | 0.20 |
| no:no | 0.10 | 0.90 |




| | yes | no |
|---|---|---|
| yes | 0.98 | 0.02 |
| no | 0.05 | 0.95 |

The local distributions $X_i \mid \Pi_{X_i}$ take the form of conditional probability tables for each node given all the configurations of the values of its parents.

Overall parameters of the $X_i \mid \Pi_{X_i}$ : 18




A classic example of GBN is the MARKS networks from Mardia, Kent & Bibby JM (1979), which describes the relationships between the marks on 5 math-related topics.

Assuming $\mathbf{X} \sim N(\mu, \Sigma)$, we can compute $\Omega = \Sigma^{-1}$. Then $\Omega_{ij} = 0$ implies $X_i \perp\!\!\!\perp_P X_j \mid \mathbf{X} \setminus \{X_i, X_j\}$. The absence of an arc $X_i \to X_j$ in the DAG implies $X_i \perp\!\!\!\perp_G X_j \mid \mathbf{X} \setminus \{X_i, X_j\}$, which in turn implies $X_i \perp\!\!\!\perp_P X_j \mid \mathbf{X} \setminus \{X_i, X_j\}$.

Total parameters of $\mathbf{X}$ : $5 + 15 = 20$

# Partial Correlations and Linear Regressions

The local distributions $X_i \mid \Pi_{X_i}$ take the form of linear regression models with the $\Pi_{X_i}$ acting as regressors and with independent error terms.

$$
\begin{aligned}
\mathsf{ALG} &= 50.60 + \varepsilon_{\mathsf{ALG}} \sim N(0, 112.8) \\
\mathsf{ANL} &= -3.57 + 0.99\mathsf{ALG} + \varepsilon_{\mathsf{ANL}} \sim N(0, 110.25) \\
\mathsf{MECH} &= -12.36 + 0.54\mathsf{ALG} + 0.46\mathsf{VECT} + \varepsilon_{\mathsf{MECH}} \sim N(0, 195.2) \\
\mathsf{STAT} &= -11.19 + 0.76\mathsf{ALG} + 0.31\mathsf{ANL} + \varepsilon_{\mathsf{STAT}} \sim N(0, 158.8) \\
\mathsf{VECT} &= 12.41 + 0.75\mathsf{ALG} + \varepsilon_{\mathsf{VECT}} \sim N(0, 109.8)
\end{aligned}
$$

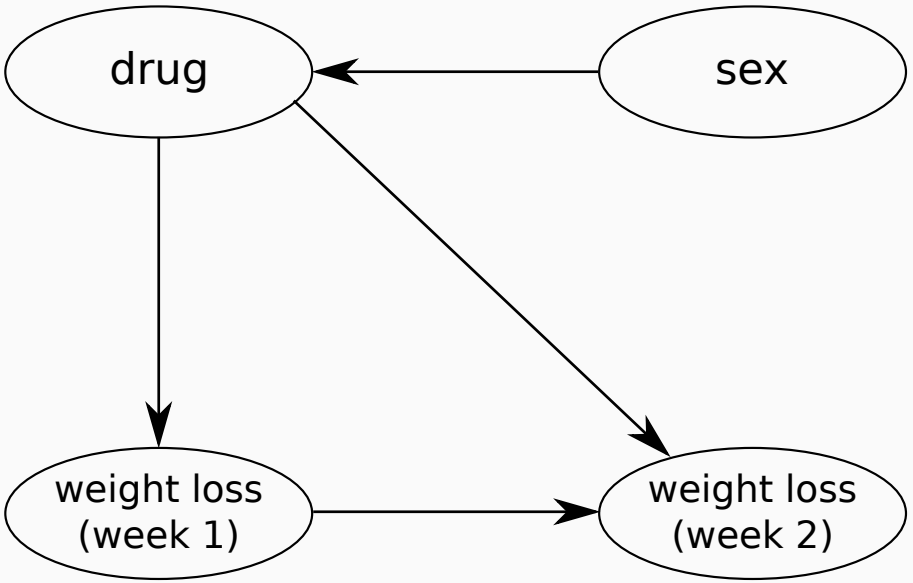
(That is because $\Omega_{ij} \propto \beta_j$ for $X_i$, so $\beta_j \neq 0$ if and only if $\Omega_{ij} \neq 0$. Also $\Omega_{ij} \propto \rho_{ij}$, the partial correlation between $X_i$ and $X_j$, so we are implicitly assuming that all probabilistic dependencies are linear.)

Overall parameters of the $X_i \mid \Pi_{X_i}$ : $11 + 5 = 16$

# Conditional Linear Gaussian Bayesian Networks

CLGBNs contain both discrete and continuous nodes, and combine DBNs and GBNs as follows to obtain a mixture-of-Gaussians network:

- continuous nodes cannot be parents of discrete nodes;
- the local distribution of each discrete node is a CPT;
- the local distribution of each continuous node is a set of linear regression models, one for each configurations of the discrete parents, with the continuous parents acting as regressors.




One of the classic examples is the RATS' WEIGHTS network from Edwards (1995), which describes weight loss in a drug trial performed on rats.

The resulting local distribution for the first weight loss for drugs $\mathrm{D}_1$, $\mathrm{D}_2$ and $\mathrm{D}_3$ is:

$$\begin{aligned}
\mathrm{W}_{1,\mathrm{D}_1} &= 7 + \varepsilon_{\mathrm{D}_1} \sim N(0, 2.5) \\
\mathrm{W}_{1,\mathrm{D}_2} &= 7.50 + \varepsilon_{\mathrm{D}_2} \sim N(0, 2) \\
\mathrm{W}_{1,\mathrm{D}_3} &= 14.75 + \varepsilon_{\mathrm{D}_3} \sim N(0, 11)
\end{aligned}$$

with just the intercepts since the node has no continuous parents. The local distribution for the second loss is:

$$\begin{aligned}
\mathrm{W}_{2,\mathrm{D}_1} &= 1.02 + 0.89\beta_{\mathrm{W}_1} + \varepsilon_{\mathrm{D}_1} \sim N(0, 3.2) \\
\mathrm{W}_{2,\mathrm{D}_2} &= -1.68 + 1.35\beta_{\mathrm{W}_1} + \varepsilon_{\mathrm{D}_2} \sim N(0, 4) \\
\mathrm{W}_{2,\mathrm{D}_3} &= -1.83 + 0.82\beta_{\mathrm{W}_1} + \varepsilon_{\mathrm{D}_3} \sim N(0, 1.9)
\end{aligned}$$

Overall, they look like random effect models with random intercepts and random slopes.

Learning a BN $\mathcal{B} = (\mathcal{G}, \Theta)$ from a data set $\mathcal{D}$ is performed in two steps:

$$\underbrace{\mathrm{P}(\mathcal{B} \mid \mathcal{D}) = \mathrm{P}(\mathcal{G}, \Theta \mid \mathcal{D})}_{\text{learning}} = \underbrace{\mathrm{P}(\mathcal{G} \mid \mathcal{D})}_{\text{structure learning}} \cdot \underbrace{\mathrm{P}(\Theta \mid \mathcal{G}, \mathcal{D})}_{\text{parameter learning}}.$$

In a Bayesian setting structure learning consists in finding the DAG with the best $\mathrm{P}(\mathcal{G} \mid \mathcal{D})$. We can decompose $\mathrm{P}(\mathcal{G} \mid \mathcal{D})$ into

$$\mathrm{P}(\mathcal{G} \mid \mathcal{D}) \propto \mathrm{P}(\mathcal{G})\,\mathrm{P}(\mathcal{D} \mid \mathcal{G}) = \mathrm{P}(\mathcal{G}) \int \mathrm{P}(\mathcal{D} \mid \mathcal{G}, \Theta)\,\mathrm{P}(\Theta \mid \mathcal{G}) d\Theta$$

where $\mathrm{P}(\mathcal{G})$ is the prior distribution over the space of the DAGs and $\mathrm{P}(\mathcal{D} \mid \mathcal{G})$ is the marginal likelihood of the data given $\mathcal{G}$ averaged over all possible parameter sets $\Theta$; and then

$$\mathrm{P}(\mathcal{D} \mid \mathcal{G}) = \prod_{i=1}^{N} \left[ \int \mathrm{P}(X_i \mid \Pi_{X_i}, \Theta_{X_i})\,\mathrm{P}(\Theta_{X_i} \mid \Pi_{X_i}) d\Theta_{X_i} \right].$$

# The Hill-Climbing Algorithm

The most common score-based structure learning algorithm, in which we are looking for the DAG that maximises a score such as the posterior $\mathrm{P}(\mathcal{G} \mid \mathcal{D})$ or BIC, is a greedy search such as hill-climbing:

1. Choose an initial DAG $\mathcal{G}$, usually (but not necessarily) empty.
2. Compute the score of $\mathcal{G}$, denoted as $Score_{\mathcal{G}} = \mathrm{Score}(\mathcal{G})$.
3. Set $maxscore = Score_{\mathcal{G}}$.
4. Repeat the following steps as long as $maxscore$ increases:
   - 4.1 for every possible arc addition, deletion or reversal not introducing cycles:
     - 4.1.1 compute the score of the modified DAG $\mathcal{G}^*$, $Score_{\mathcal{G}^*} = \mathrm{Score}(\mathcal{G}^*)$:
     - 4.1.2 if $Score_{\mathcal{G}^*} > Score_{\mathcal{G}}$, set $\mathcal{G} = \mathcal{G}^*$ and $Score_{\mathcal{G}} = Score_{\mathcal{G}^*}$.
   - 4.2 update $maxscore$ with the new value of $Score_G$.
5. Return the DAG $\mathcal{G}$.

Only one local distribution changes in each step, which makes algorithm computationally efficient and easy to speed up with caching.

# Constraint-Based Algorithms

As an alternative, constraint-based structure learning algorithms use conditional independence tests to learn which $\perp\!\!\!\perp_P$ hold: they assume that the corresponding $\perp\!\!\!\perp_G$ hold as well to determine which arcs are in $\mathcal{G}$.

1. For each pair of nodes $A$ and $B$ in $\mathbf{V}$ search for set $\mathbf{S}_{AB} \subset \mathbf{V}$ such that $A$ and $B$ are independent given $\mathbf{S}_{AB}$ and $A, B \notin \mathbf{S}_{AB}$. If there is no such a set, place an undirected arc between $A$ and $B$.
2. For each pair of non-adjacent nodes $A$ and $B$ with a common neighbour $C$, check whether $C \in \mathbf{S}_{AB}$. If this is not true, set the direction of the arcs $A - C$ and $C - B$ to $A \rightarrow C$ and $C \leftarrow B$.
3. Set the direction of arcs which are still undirected by applying iteratively the following two rules:
   - 3.1 if $A$ is adjacent to $B$ and there is a strictly directed path from $A$ to $B$ then set the direction of $A - B$ to $A \rightarrow B$;
   - 3.2 if $A$ and $B$ are not adjacent but $A \rightarrow C$ and $C - B$, then change the latter to $C \rightarrow B$.
4. Return the resulting CPDAG.

# Bayesian Network Parameter Learning

Once the structure of the model is known, the problem of estimating the parameters of the global distribution can be solved by estimating the parameters of the local distributions, one at a time.

Three common choices are:

- maximum likelihood estimators: just the usual estimators from classic linear models and contingency tables literature.
- Bayesian posterior estimators: posterior estimators, based on conjugate priors to keep computations fast, simple and in closed form.
- regularised estimators: regularised estimators based either on James-Stein or Bayesian shrinkage results.

# Events, Evidence and Queries

A BN represents a working model of the world that a computer can understand; but how does a computer system use it to help and perform its assigned task?

We ask questions, and the computer system performs probabilistic inference to answer them and decide what to do in the process.

Questions that can be asked are called queries and are typically about an event of interest given some evidence. The evidence is the input to the computer system and the event is the output. This is often called belief update: we observe some evidence and we update our beliefs before taking action.

The two most common queries are:

- conditional probability queries ("What is the probability of a particular event after we observe some specific evidence?"); and
- most probable explanation queries ("What is the most probable value of this subset of variables given that we observe some specific values for one or more other variables?")

In both cases the evidence is hard evidence: we set some variables to particular values. Then the computer system checks how the probabilities of other variables change and provides an answer to the query.

No more manual probability computations...

# Exact and Approximate Inference

There are two approaches to answer queries using BNs.

Exact algorithms use the DAG of a BN to schedule and perform repeated applications of Bayes theorem and the probability axioms on the probabilities in the model. In other words, the computer system uses the DAG to perform all the math we would do by hand.

The two best known are

- variable elimination; and
- belief updates based on junction trees.

PROS: they return exact values for the probabilities of interest.

CONS: they do not scale well when BNs have many nodes and many arcs.

Approximate algorithms use the BN as a model of the world in a very literal sense. In the real world to answer some question in a scientific, rigorous way we would perform an experiment and observe the outcome; approximate algorithms imitate this process by generating random observations from the BN, thus running a simulated experiment that approximates reality.

The two best known are

- logic sampling; and
- likelihood weighting.

PROS: they scale really well when BNs have many nodes and many arcs.

CONS: they return approximate, estimated values for the probabilities of interest.

# The Logic Sampling Algorithm

INPUT: a BN, evidence $E$ and query event $Q$.

1. Order the variables in $\mathbf{X}$ according to the topological ordering in the DAG (from top to bottom), so that parents come before children.
2. Set $n_E = 0$ and $n_{E,Q} = 0$.
3. For a suitably large number of samples $\mathbf{x}$:
   - 3.1 generate a random value from each $X_i \mid \Pi_{X_i}$ taking advantage of the fact that, thanks to the topological ordering, by the time we are considering $X_i$ we have already generated the values of all its parents $\Pi_{X_i}$;
   - 3.2 if $\mathbf{x}$ includes $E$, set $n_E = n_E + 1$;
   - 3.3 if $\mathbf{x}$ includes both $Q$ and $E$, set $n_{E,Q} = n_{E,Q} + 1$.
4. The answer to the query is the estimated probability $n_{E,Q}/n_E$.

# The Junction Tree Algorithm

1. **Moralise:** create the moral graph of the BN $\mathcal{B}$.
2. **Triangulate:** break every cycle spanning $4$ or more nodes into sub-cycles of exactly $3$ nodes by adding arcs to the moral graph, thus obtaining a triangulated graph.
3. **Cliques:** identify the cliques $C_1, \ldots, C_k$ of the triangulated graph, *i.e.*, maximal subsets of nodes in which each element is adjacent to all the others.
4. **Junction Tree:** create a tree in which each clique is a node, and adjacent cliques are linked by arcs. The tree must satisfy the running intersection property: if a node belongs to two cliques $C_i$ and $C_j$, it must be also included in all the cliques in the (unique) path that connects $C_i$ and $C_j$.
5. **Parameters:** use the parameters of the local distributions of $\mathcal{B}$ to compute the parameter sets of the compound nodes of the junction tree.

- ✓ Overview
- ✓ An Introduction to Bayesian Networks
- → Advances in Bayesian Networks
- Summary
- References

# The Classic Definition of Bayesian Networks

The classic definition of a BN [13] involves:

- a network structure, a directed acyclic graph $\mathcal{G}$, in which each node corresponds to a random variable $X_i$;
- a probability distribution $\mathbf{X}$ and its factorisation into $X_i \mid \Pi_{X_i}$.

What are we assuming when trying to learn a BN? Typically that:

- observations are independent and there are no missing values;
- all variables are observed, that is, no latent variables introducing confounding into the model;
- we measure probabilistic associations (or rather, independencies) and we cannot necessarily interpret them as causal.

What happens if we relax those assumptions? Many extensions suddenly become possible, see [24] for a recent review.

# RECAP: Bayesian Network Learning

Learning a BN $\mathcal{B} = (\mathcal{G}, \Theta)$ from a data set $\mathcal{D}$ is performed in two steps:

$$\underbrace{\mathrm{P}(\mathcal{B} \mid \mathcal{D}) = \mathrm{P}(\mathcal{G}, \Theta \mid \mathcal{D})}_{\text{learning}} = \underbrace{\mathrm{P}(\mathcal{G} \mid \mathcal{D})}_{\text{structure learning}} \cdot \underbrace{\mathrm{P}(\Theta \mid \mathcal{G}, \mathcal{D})}_{\text{parameter learning}}.$$

In a Bayesian setting structure learning consists in finding the DAG with the best $\mathrm{P}(\mathcal{G} \mid \mathcal{D})$. We can decompose $\mathrm{P}(\mathcal{G} \mid \mathcal{D})$ into

$$\mathrm{P}(\mathcal{G} \mid \mathcal{D}) \propto \mathrm{P}(\mathcal{G})\,\mathrm{P}(\mathcal{D} \mid \mathcal{G}) = \mathrm{P}(\mathcal{G}) \int \mathrm{P}(\mathcal{D} \mid \mathcal{G}, \Theta)\,\mathrm{P}(\Theta \mid \mathcal{G}) d\Theta$$

where $\mathrm{P}(\mathcal{G})$ is the prior distribution over the space of the DAGs and $\mathrm{P}(\mathcal{D} \mid \mathcal{G})$ is the marginal likelihood of the data given $\mathcal{G}$ averaged over all possible parameter sets $\Theta$; and then

$$\mathrm{P}(\mathcal{D} \mid \mathcal{G}) = \prod_{i=1}^{N} \left[ \int \mathrm{P}(X_i \mid \Pi_{X_i}, \Theta_{X_i})\,\mathrm{P}(\Theta_{X_i} \mid \Pi_{X_i}) d\Theta_{X_i} \right].$$

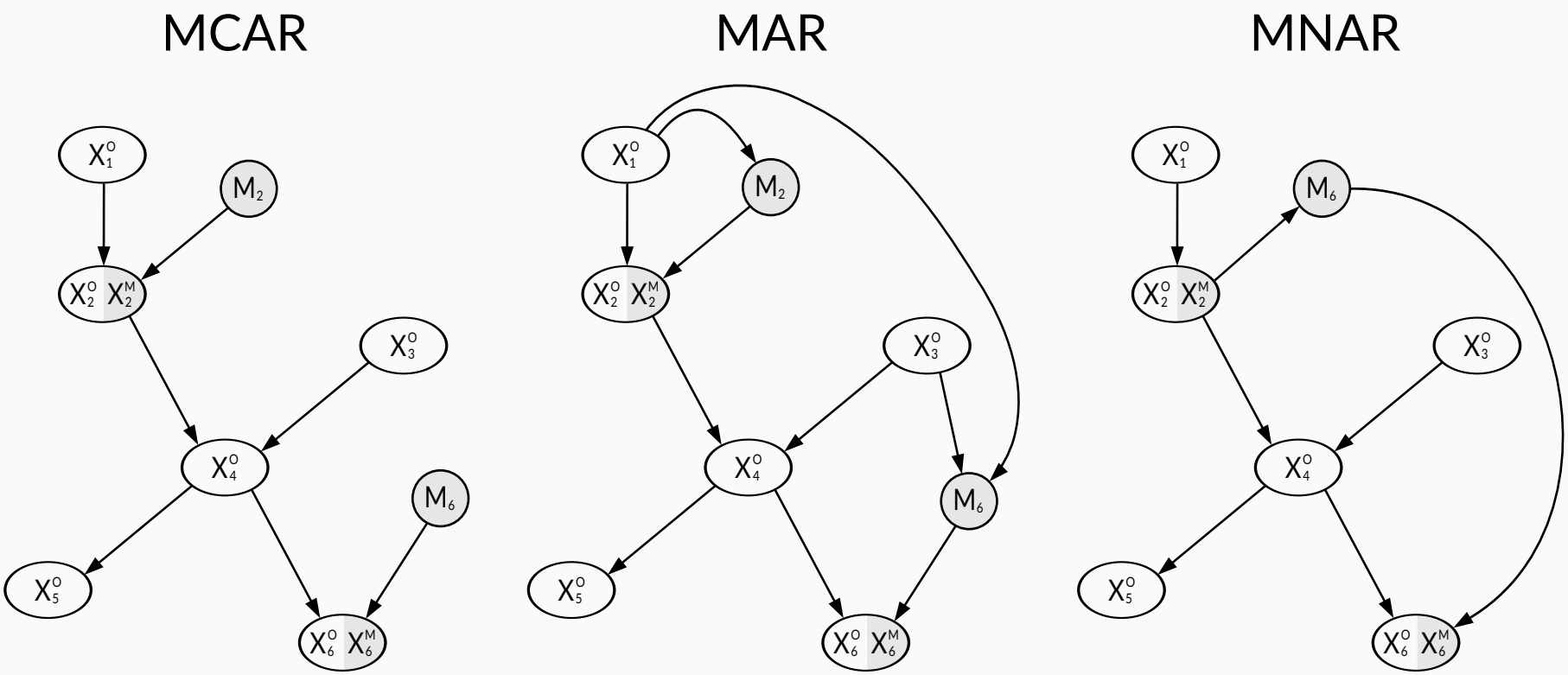
Once $\mathcal{G}$ is known, parameter learning involves estimating the parameters of the local distributions, one at a time.

# Missing Data

Little and Rubin [17, 22] formalised three possible patterns of missingness:

- Missing completely at random (MCAR): complete samples are indistinguishable from incomplete ones. The probability that a value will be missing is independent from both observed and missing values.
- Missing at random (MAR): incomplete samples differ from complete ones, but the pattern of missingness is predictable from other observed variables. The probability that a value will be missing is a function of the observed values.
- Missing not at random (MNAR): the pattern of missingness is not random or it is not predictable from other observed variables; the probability that an entry will be missing depends on both observed and missing values. Common examples are variables that are missing systematically or for which the pattern of missingness depends on the missing values themselves.

# Missing Data as Bayesian Networks



# Ignorable Patterns of Missingness

MCAR and MAR are ignorable. If we denote with $\mathcal{D}^O$ and $\mathcal{D}^M$ the observed and unobserved portions of $\mathcal{D}$, and we group all the binary missingness indicators $M_i$ in $\mathbf{M}$ with parameters $\Xi$, then we can write

$$\mathrm{P}\left(\mathcal{D}, \mathbf{M} \mid \mathcal{G}, \Theta, \Xi\right) = \mathrm{P}\left(\mathcal{D}^O, \mathcal{D}^M, \mathbf{M} \mid \mathcal{G}, \Theta, \Xi\right) = \int \mathrm{P}(\mathcal{D}^O, \mathcal{D}^M \mid \mathcal{G}, \Theta)\, \mathrm{P}\left(\mathbf{M} \mid \mathcal{D}^O, \mathcal{D}^M, \mathcal{G}, \Xi\right)\, d\mathcal{D}^M.$$

If the missing data are MAR then $\mathbf{M}$ only depends on $\mathcal{D}^O$,

$$\mathrm{P}\left(\mathbf{M} \mid \mathcal{D}^O, \mathcal{D}^M, \mathcal{G}, \Xi\right) = \mathrm{P}\left(\mathbf{M} \mid \mathcal{D}^O, \mathcal{G}, \Xi\right);$$

and if the missing data MCAR then $\mathbf{M}$ does not depend on $\mathcal{D}^O$ or $\mathcal{D}^M$,

$$\mathrm{P}\left(\mathbf{M} \mid \mathcal{D}^O, \mathcal{D}^M, \mathcal{G}, \Xi\right) = \mathrm{P}\left(\mathbf{M} \mid \mathcal{G}, \Xi\right).$$

In both cases it is possible to model $\mathbf{M}$ from the available data. However, this is not the case for MNAR since $\mathbf{M}$ depends on the unobserved $\mathcal{D}^M$.

# Parameter Learning with Incomplete Data: EM

A classic approach to handle missing data is the Expectation-Maximisation (EM) algorithm [16]:

- the expectation (E) step consists in computing the expected values of the sufficient statistics (such as the counts $n_{ijk}$ in discrete BNs, partial correlations in GBNs), using exact inference to make use of incomplete as well as complete samples;
- the maximisation (M) step takes the sufficient statistics from the E-step and estimates the parameters of the BN, either using maximum likelihood or Bayesian posterior estimators.

The parameter estimates are then used in the next E-step to update the expected values of the sufficient statistics. Repeated iterations of these two steps will in the limit return the maximum likelihood or maximum a posteriori estimates for the parameters.

The E-step is equivalent to computing $\mathrm{E}(\mathcal{D}^M, \mathcal{D}^O \mid \mathcal{G}, \Theta)$ and the M-step is equivalent to maximising $\mathrm{P}(\Theta \mid \mathcal{G}, \mathcal{D}^O, \mathcal{D}^M)$.

Data Augmentation (DA) [20, 25] is quite similar, but instead of converging iteratively to a single set of parameter estimates it uses Gibbs Sampling to generate values from the posterior distributions of both $\mathcal{D}^M$ and $\Theta$. The two steps are as follows:

- in the imputation (I) step the data are completed with values drawn from the predictive distributions of the missing values;
- in the parameter (P) estimation step a parameter value is drawn from the posterior distribution of $\Theta$ conditional on the completed data from the I-step.

Formally, we define the augmented parameter vector $\{\mathcal{D}^M, \Theta\}$ containing both the missing values and the parameters of the BN. Given an initial set of values, DA updates each element of $\{\mathcal{D}^M, \Theta\}$ by sampling a new value for each missing value from $\mathrm{P}(\mathcal{D}_i^M \mid \mathcal{D}_{-i}^M, \Theta)$, and by sampling a new value for each parameter from $\mathrm{P}(\Theta_i \mid \Theta_{-i}, \mathcal{D}^M)$, each in turn.

Learning the structure of a BN from incomplete data is computationally unfeasible because we need to perform a joint optimisation over the missing values and the parameters to score each candidate network. The maximum a posteriori DAG maximises

$$\mathrm{P}\left(\mathcal{D} \mid \mathcal{G}\right) = \int \mathrm{P}\left(\mathcal{D}^O, \mathcal{D}^M \mid \mathcal{G}, \Theta\right) \mathrm{P}\left(\Theta \mid \mathcal{G}\right)\, d\Theta =$$

$$= \int \underbrace{\mathrm{P}\left(\mathcal{D}^M \mid \mathcal{D}^O, \mathcal{G}, \Theta\right)}_{\text{missing data}} \underbrace{\mathrm{P}\left(\mathcal{D}^O \mid \mathcal{G}, \Theta\right)}_{\text{observed data}} \underbrace{\mathrm{P}\left(\Theta \mid \mathcal{G}\right)\, d\Theta}_{\text{averaging over parameters}} .$$

A full Bayesian approach would require averaging over all the possible configurations of the missing data, leading to

$$\mathrm{P}\left(\mathcal{D} \mid \mathcal{G}\right) = \iint \mathrm{P}\left(\mathcal{D}^M \mid \mathcal{D}^O, \mathcal{G}, \Theta\right) \mathrm{P}\left(\mathcal{D}^O \mid \mathcal{G}, \Theta\right) \mathrm{P}\left(\Theta \mid \mathcal{G}\right)\, d\Theta\, d\mathcal{D}^M.$$

which has one one extra dimension for each missing value. An additional problem is that $\mathrm{P}(\mathcal{D}^M \mid \mathcal{D}^O, \mathcal{G}, \Theta)$ does not factorise in the general case.

# Structure Learning from Incomplete Data: Structural EM

The Structural EM algorithm [6] makes structure learning computationally feasible by searching for the best structure inside of EM, instead of embedding EM inside a structure learning algorithm. It consists of two steps like the classic EM:

- in the E-step, we complete the data by computing the expected sufficient statistics using the current network structure;
- in the M-step, we find the structure that maximises the expected score function for the completed data.

Since the scoring in the M-step uses the completed data , structure learning can be implemented efficiently using standard algorithms. The original proposal by [6] used BIC and greedy search; [7] later extended SEM to a fully Bayesian approach based posterior scores, and proved the convergence of the resulting algorithm.

# Structure Learning from Incomplete Data: Approximations

An alternative is to use scores that approximate $\mathrm{P}(\mathcal{D} \mid \mathcal{G})$ and that are decomposable and efficient to compute even on incomplete data. Some options are:

- The variational-Bayesian EM from [2] that maximises a variational approximation of $\mathrm{P}(\mathcal{D} \mid \mathcal{G})$, which in turn is a lower bound to the true $\mathrm{P}(\mathcal{D} \mid \mathcal{G})$.
- Replacing BIC with the node-average penalised log-likelihood computed from locally complete observations as suggested in [1].
- Using mixtures of truncated exponentials [4] to approximate the distributions of variables that are not completely observed.

# What About Latent Variables?

Note that we can work on latent variables using similar approaches if $\mathcal{G}$ is fixed and we just need to learn $\Theta$. A recent example is given in [27], who shows it is possible to learn the domain of a latent discrete variable as well as the associated parameters as long as it has observed parents and children. Several other examples are discussed in the context of dynamic BNs in [18].

If $\mathcal{G}$ is not fixed, we need to learn:

- how many latent variables we are dealing with;
- their domains;
- how they are connected to the observed variables

at the same time, which is not feasible in general.

Dynamic BNs (DBNs) combine classic BNs and Markov processes to model dynamic data in which each individual is measured repeatedly over time, such as longitudinal or panel data.

Assume we have one set $\mathbf{X}^{(t)}$ of random variables for each of $t = 1, \dots, T$ time points. We can model it as a DBN with a Markov process of the form

$$\mathrm{P}\left(\mathbf{X}^{(0)}, \dots, \mathbf{X}^{(T)}\right) = \mathrm{P}\left(\mathbf{X}^{(0)}\right) \prod_{t=1}^{T} \mathrm{P}\left(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}\right).$$

where $\mathrm{P}(\mathbf{X}^{(0)})$ gives the initial state of the process and $\mathrm{P}(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)})$ defines the transition between times $t-1$ and $t$. When modelling $\mathbf{X}^{(t)}$, the nodes in $\mathbf{X}^{(t-1)}$ only appear in the conditioning; we take them to be essentially fixed and to have no free parameters, so we leave them as root nodes.

We can model this transition with a 2-time BN (2TBN) defined over $(\mathbf{X}^{(t-1)}, \mathbf{X}^{(t)})$, in which we naturally assume that any arc between a node in $t-1$ and a node in $t$ must necessarily be directed towards the node in $t$ following the arrow of time.

We may also want to assume that there are no arcs connecting two nodes in the same $t$ or, in other words, no instantaneous dependencies.

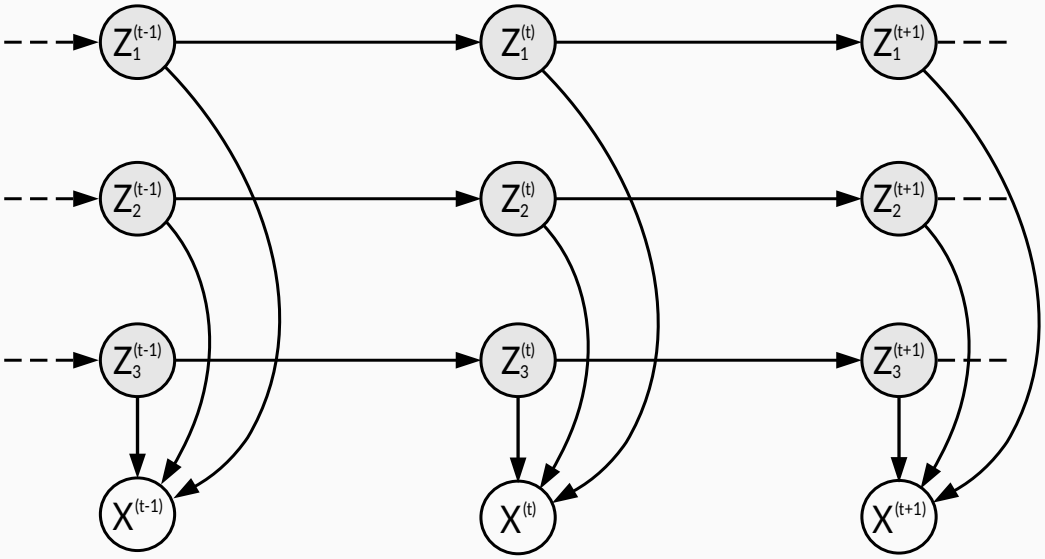
We can then write the decomposition into local distributions

$$\mathrm{P}\left(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}\right) = \prod_{i=1}^{N} \mathrm{P}\left(X_i^{(t)} \mid \Pi_{X_i^{(t)}}\right),$$

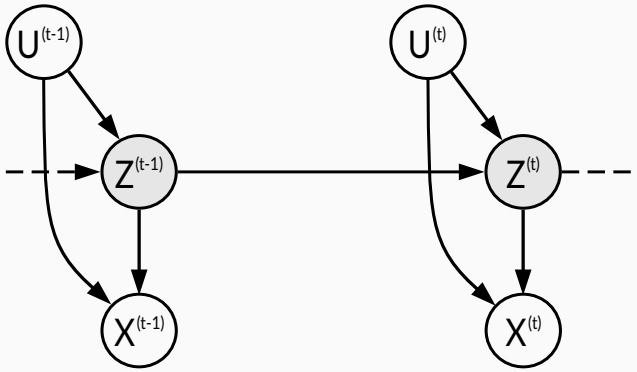
and we usually assume that the parameters associated with the local distributions do not change over time to make the process time-homogeneous.

# Many Models are Dynamic Bayesian Networks

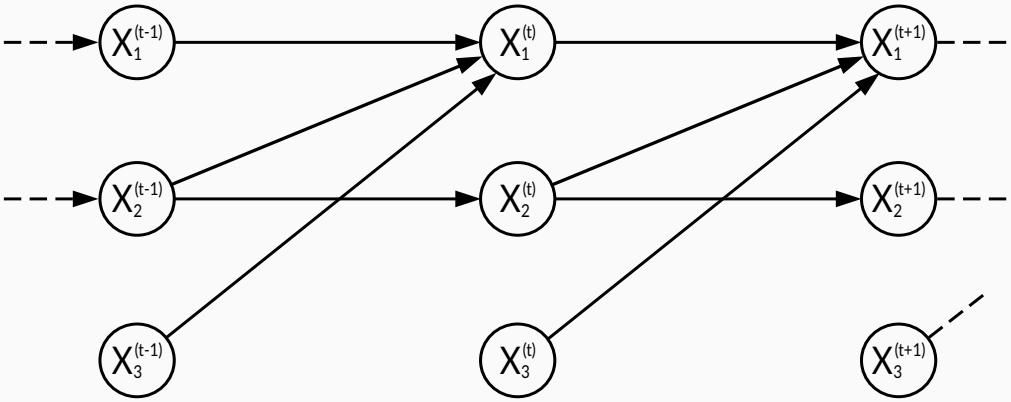
## Factorial HMM



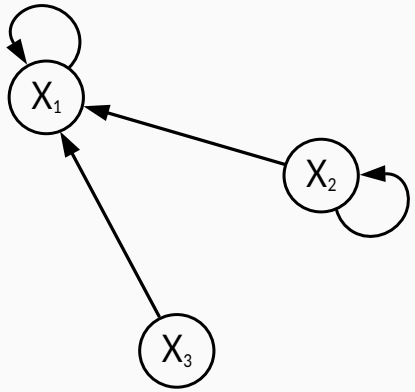

## Kalman Filter




## Unrolled VAR




## Rolled VAR



Hidden Markov models (HMMs) [29] are one of the most widespread approaches to model phenomena with hidden state, that is, in which the behaviour of the observed variables $\mathbf{X}$ depends on that of one or more discrete latent variables $\mathbf{Z}$ as well as on other variables in $\mathbf{X}$.

In DBN terms, an HMM model with $M$ latent variables can be written as

$$\mathrm{P}\left(\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}, \mathbf{Z}^{(t)}\right) = \prod_{i=1}^{N} \mathrm{P}\left(X_i^{(t)} \mid \Pi_{X_i^{(t)}}, \mathbf{Z}^{(t)}\right)$$

$$\mathrm{P}\left(\mathbf{Z}^{(t)}\right) = \prod_{j=1}^{M} \mathrm{P}\left(Z_j^{(t)} \mid \Pi_{Z_j^{(t)}}\right),$$

with the restriction that the parents of $Z_j^{(t)}$ can only be other latent variables. In the vast majority of the literature all variables are assumed to be discrete. Depending on the choice of $\Pi_{Z_j^{(t)}}$, we can obtain various HMM variants such as hierarchical HMMs [5] and factorial HMMs [9].

# Many Models are Dynamic Bayesian Networks: VARs

Vector auto-regressive models (VARs) [3] are a straightforward multivariate extension of univariate auto-regressive time series for continuous variables.

VARs are defined as

$$\mathbf{X}^{(t)} = A_1\mathbf{X}^{(t-1)} + \ldots + A_L\mathbf{X}^{(t-L)} + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \Sigma),$$

for some fixed Markov order $L$. We can rewrite that as

$$\mathbf{X}^{(t)} \mid \mathbf{X}^{(t-1)}, \ldots, \mathbf{X}^{(t-L)} \sim N\left(A_1\mathbf{X}^{(t-1)} + \ldots + A_L\mathbf{X}^{(t-L)}, \varepsilon_t\right)$$

and then restrict the parents of each $X_i^{(t)}$ to those for which the corresponding regression coefficients in $A_1, \ldots, A_L$ are different from zero using the one-to-one correspondence between regression coefficients and partial correlations [26]. Formally, $X_j^{(t-l)} \in \Pi_{X_i^{(t)}}$ if and only if $A_l[i, j] \neq 0$, which makes it possible to write a VAR as a Gaussian DBN.

# Many Models are Dynamic Bayesian Networks: Kalman Filters

Kalman filters [12] combine traits of both HMMs and VARs, as discussed in depth in [8] and [21]: like VARs, they are linear Gaussian DBNs; but they also have latent variables like HMMs.

In their simplest form, Kalman Filters include a layer of one or more latent variables that model the unobservable part of the phenomenon,

$$\mathbf{Z}^{(t)} = A\mathbf{Z}^{(t-1)} + B\mathbf{U}^{(t)} + \zeta_t, \qquad \zeta_t \sim N(0, \Psi)$$

feeding into one or more observed variables

$$\mathbf{X}^{(t)} = C\mathbf{Z}^{(t)} + D\mathbf{U}^{(t)} + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \Sigma)$$

with independent Gaussian noise added in both layers. Both layers often include additional (continuous) explanatory variables $\mathbf{U}$ and can also be augmented with (discrete) switching variables to allow for different regimes as in [10, 11]. If we exclude the latter, the assumption is that the system is jointly Gaussian: that makes it possible to frame Kalman filters as DBNs in the same way we did for VARs.

# Learning Bayesian Networks as Causal Models

Learning causal models [19], especially from observational data, presents significant challenges. In particular, three additional assumptions are needed:

- Causal Markov assumption: each variable $X_i \in \mathbf{X}$ is conditionally independent of its non-effects, both direct and indirect, given its direct causes.
- Faithfulness: there must exist a DAG which is faithful to the probability distribution $\mathbf{P}$ of $\mathbf{X}$, so that the only dependencies in $\mathbf{P}$ are those arising from d-separation in the DAG.
- Causal Sufficiency: there must be no *latent variables* (unobserved variables influencing the variables in the network) acting as *confounding factors*. Such variables may induce spurious correlations between the observed variables, thus introducing bias in the causal network.

These assumptions are difficult to verify in real-world settings, as the set of the potential confounding factors is not usually known.

# Why is Learning Causal Bayesian Networks Important?

From a practical perspective, ensuring that a BN we learn from data correctly models the underlying cause-effect relationships between the variables is important because:

- it improves our understanding of the underlying phenomenon;
- it allows us to target interventions to effect some desirable change to the underlying phenomenon;
- it allows us to reason about counterfactuals using the BN.

Identifying the correct targets for interventions is the foundation upon which quantitative policy decision making is built.

A counterfactual [19] is an "if" statement in which the "if" portion is untrue or unrealised. The "if" portion of a counterfactual is called the hypothetical condition, or more often, the antecedent. We use counterfactuals to emphasise our wish to compare two outcomes under the exact same conditions, differing only in one aspect: the antecedent.

# Counterfactual Fairness

Counterfactual Fairness: a sensitive attribute $A$ a should not be a cause of a target variable in any (statistical) individual [23, 15]. In other words, changing $A$ while holding things which are not causally dependent on $A$ constant will not change the distribution of $Y$.

Formally: a predictor $\hat{Y}$ of $Y$ is counterfactually fair given the sensitive attribute $A = a$ and any observed variables $\mathbf{X}$ if

$$\mathrm{P}(\hat{Y}_{A \leftarrow a} = y \mid \mathbf{X} = \mathbf{x}, A = a) = \mathrm{P}(\hat{Y}_{A \leftarrow a'} = y \mid \mathbf{X} = \mathbf{x}, A = a)$$

for all $y$ and $a' \neq a$.

In graphical terms: $\hat{Y}$ is counterfactually fair if it is a function of the non-descendants of $A$ in causal network.
This condition is too strong for most practical applications because it requires that the distributions are exactly the same for all values of $A$.

# Approximate Counterfactual Fairness

Relaxing the definition of counterfactual fairness by introducing some tolerance in the comparison of the distributions of $\hat{Y}$ under different values of $A$ gives:

Approximate Counterfactual Fairness: a predictor $f(\mathbf{X}, A)$ satisfies $(\varepsilon, \delta)$-approximate counterfactual fairness if, given the sensitive attribute $A = a$ and any instantiation $\mathbf{X} = \mathbf{x}$, we have that:

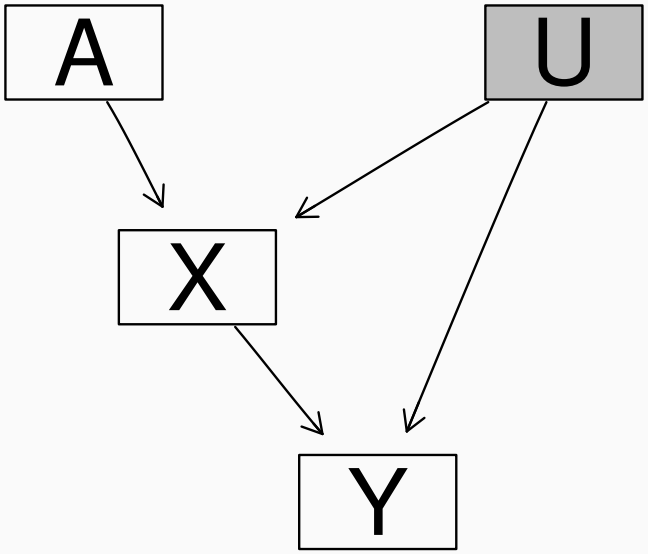
$$\mathrm{P}\left(|f(\mathbf{X}_{A \leftarrow a}, a) - f(\mathbf{X}_{A \leftarrow a'}, a')| \leqslant \varepsilon \mid \mathbf{X} = \mathbf{x}, A = a\right) > 1 - \delta$$

for all $a' \neq a$.

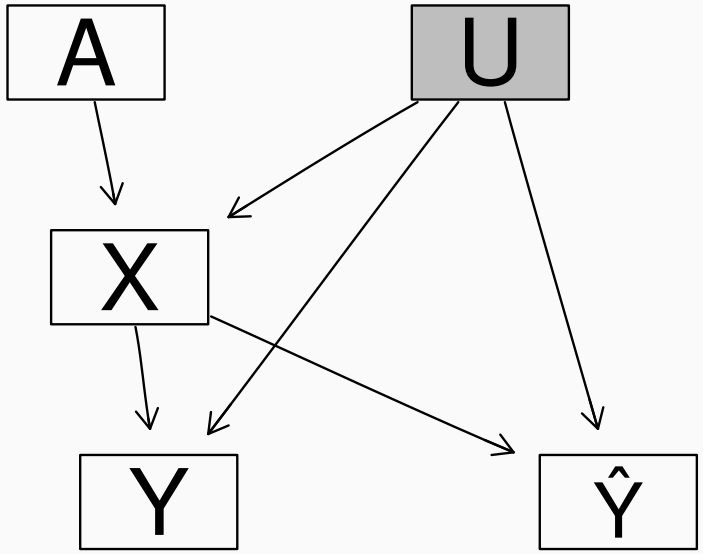
In other words, we allow some degree of unfairness but we assume that the predictor $f(\mathbf{X}, A)$ is mostly fair most of the time. This is common in all fairness literature [14, 28, etc.], regardless of the statistical model.

# Consider: an Example

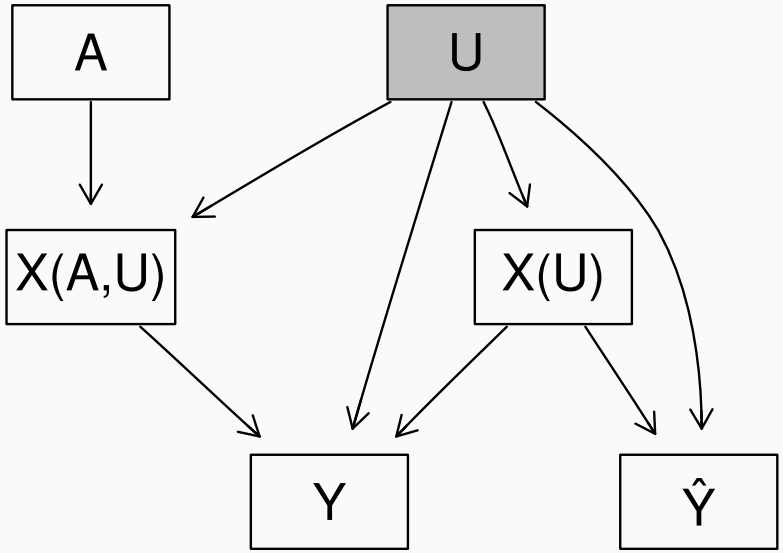
original model



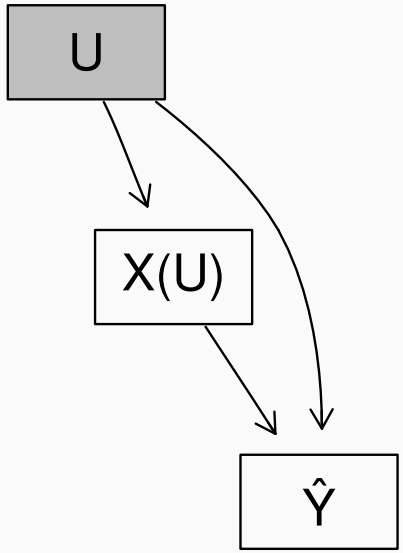

with predictions




split fair and unfair




fair prediction



# Bayesian Networks and Modern Machine Learning

- BNs provide an intuitive representation of the relationships linking heterogeneous sets of variables, which we can use for qualitative and causal reasoning.
- BNs subsume a large number of probabilistic models and thus can readily incorporate other techniques from statistics and computer science.
- BNs can be extended to handle data with complex structure such as multivariate time series and incomplete data, while taking advantage of standard learning and inference approaches.
- BNs also provide a rigorous solution to the problem of producing fair predictions that do not discriminate individuals based on sensitive attributes.

- ✓ Overview
- ✓ An Introduction to Bayesian Networks
- ✓ Advances in Bayesian Networks
- ✓ Summary
- → References

# References I


- N. Balov.
  Consistent Model Selection of Discrete Bayesian Networks from Incomplete Data.
  *Electronic Journal of Statistics*, 7(1):1047–1077, 2013.
- M. J. Beal and Z. Ghahramani.
  The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures.
  In *Proceedings of the 7th Valencia International Meeting*, pages 453–464, 2003.
- G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung.
  *Time Series Analysis: Forecasting and Control.*
  Wiley, 5th edition, 2016.
- A. Fernández, J. D. Nielsen, and A. Salmerón.
  Learning Bayesian Networks for Regression from Incomplete Databases.
  *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 18(01):69–86, 2010.
- S. Fine, Y. Singer, and N. Tibshy.
  The Hierarchical Hidden Markov Model: Analysis and Applications.
  *Machine Laarning*, 32(1):41–62, 1998.
- N. Friedman.
  Learning Belief Networks in the Presence of Missing Values and Hidden Variables.
  In *ICML*, pages 125–133, 1997.
- N. Friedman.
  The Bayesian Structural EM Algorithm.
  In *UAI*, pages 129–138, 1998.
- Z. Ghahramani.
  An Introduction to Hidden Markov Models and Bayesian Networks.
  *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42, 2001.
- Z. Ghahramani and M. Jordan.
  Factorial Hidden Markov Models.
  In *NIPS*, pages 472–278, 1996.

# References II


- M. Grzegorczyk and D. Husmeier.
  Non-Stationary Continuous Dynamic Bayesian Networks.
  In *NIPS*, pages 682–690, 2009.
- M. Grzegorczyk and D. Husmeier.
  Non-homogeneous Dynamic Bayesian Networks for Continuous Data.
  *Machine Learning*, 83(3):355–419, 2011.
- J. D. Hamilton.
  *Time Series Analysis*.
  Princeton University Press, 1994.
- D. Koller and N. Friedman.
  *Probabilistic Graphical Models: Principles and Techniques*.
  MIT Press, 2009.
- J. Komiyama, A. Takeda, J. Honda, and H. Shimao.
  Nonconvex Optimization for Regression with Fairness Constraints.
  *Proceedings of Machine Learning Research*, 80:2737–2746, 2018.
  Proccedings of the of the 35th International Conference on Machine Learning (ICML).
- M. Kusner, J. Loftus, C. Russell, and R. Silva.
  Counterfactual Fairness.
  In *NIPS*, pages 4066–4076, 2017.
- S. L. Lauritzen.
  The EM Algorithm for Graphical Association Models with Missing Data.
  *Computational Statistics and Data Analysis*, 19(2):191–201, 1995.
- R. J. A. Little and D. B. Rubin.
  *Statistical Analysis with Missing Data*.
  Wiley, 1987.

# References III


- K. Murphy.
  *Dynamic Bayesian Networks: Representation, Inference and Learning*.
  PhD thesis, UC Berkeley, Computer Science Division, 2002.
- J. Pearl and M. Glymour and.
  *Causal Inference in Statistics: a Primer*.
  Wiley, 2016.
- C. Riggelsen.
  Learning Parameters of Bayesian Networks from Incomplete Data via Importance Sampling.
  *International Journal of Approximate Reasoning*, 42(1–2):69–83, 2006.
- S. Roweis and Z. Ghahramani.
  A Unifying Review of Linear Gaussian Models.
  *Neural Computation*, 11(2):305–345, 1999.
- D. B. Rubin.
  Inference and Missing Data.
  *Biometrika*, 63:581–592, 1976.
- C. Russell, M. J. Kusner, J. R. Loftus, and R. Silva.
  When Worlds Collide: Integrating Different Countefactual Assumptions in Fairness.
  In *NIPS*, volume 30, pages 6414–6423, 2017.
- M. Scutari.
  Bayesian Network Models for Incomplete and Dynamic Data.
  *Statistica Neerlandica*, 74(3):397–419, 2020.
- M. Tanner and W. Wong.
  The Calculation of Posterior Distributions by Data Augmentation.
  *Journal of the American Statistical Association*, 82(398):528–540, 1987.
- C. E. Weatherburn.
  *A First Course in Mathematical Statistics*.
  Cambridge University Press, 1961.

# References IV


- K. Yamazaki and Y. Motomura.
  Hidden Node Detection between Observable Nodes Based on Bayesian Clustering.
  *Entropy*, 21(1):32, 2019.
- M. B. Zafar, I. Valera, M. Gomez-Rodriguez, and K. P. Gummadi.
  Fairness Constraints: a Flexible Approach for Fair Classification.
  *Journal of Machine Learning Research*, 20:1–42, 2019.
- W. Zucchini and I. L. MacDonald.
  *Hidden Markov Models for Time Series: An Introduction Using R.*
  Chapman & Hall, 2009.

# Thanks!

# Any questions?