Machine Reasoning using Bayesian Network

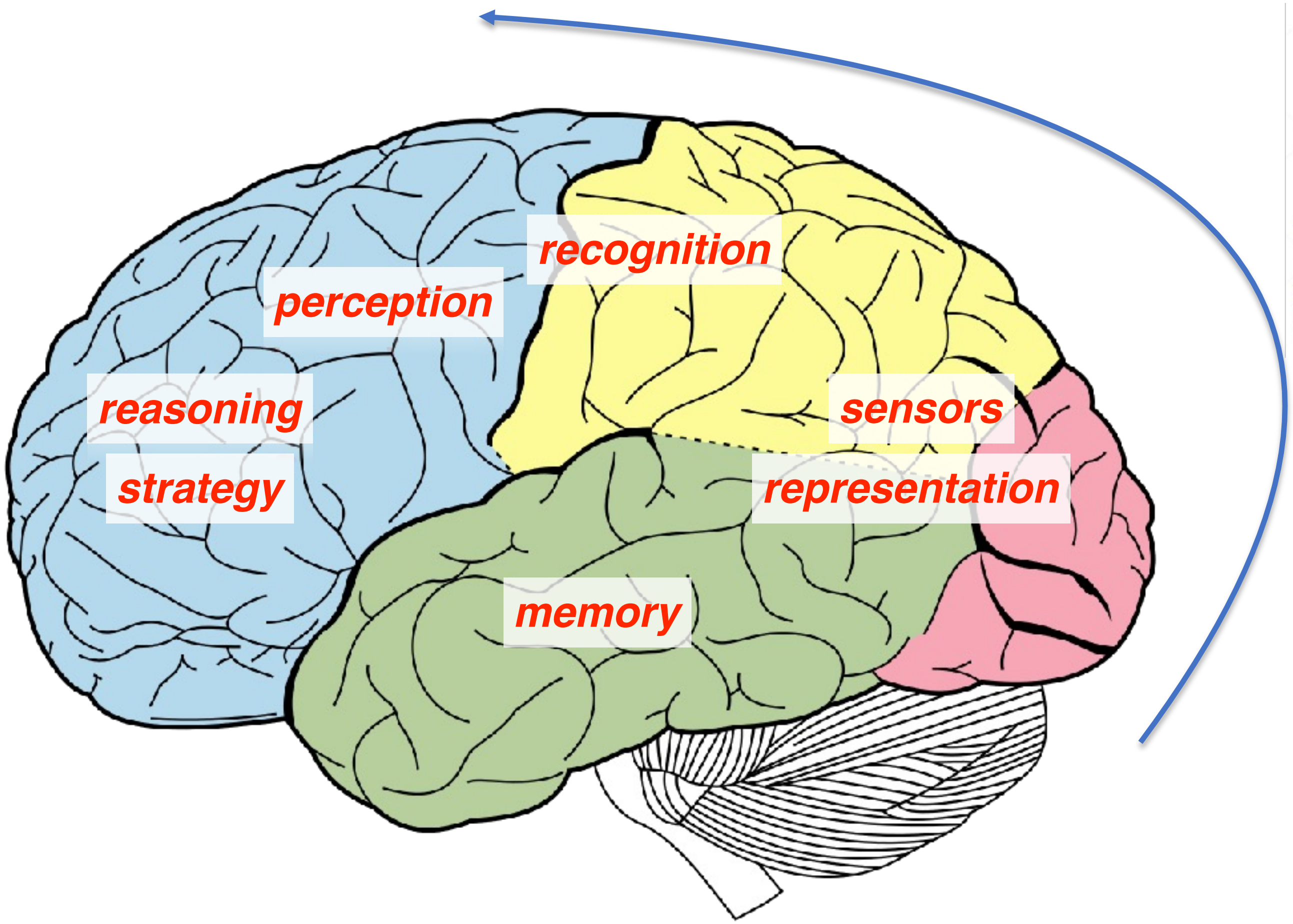
Ching-Yung Lin, Ph.D.
Columbia University
March 22, 2024

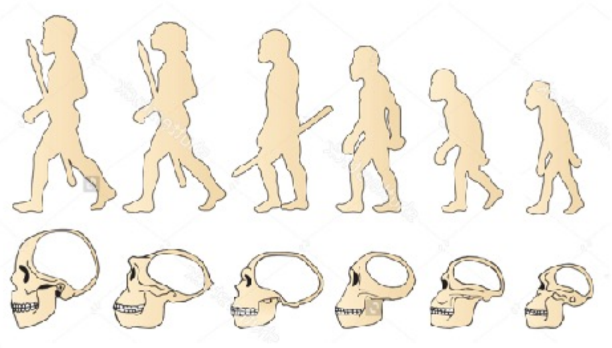
Outline

- **Introduction**
- Probability Review
- Bayesian Network
- Inference Methods
- Network Structure Learning

Evolution of Intelligence

Direction of Evolution





Introduction

Suppose the doctor is trying to determine if a patient has inhalational anthrax. She observes the following symptoms:

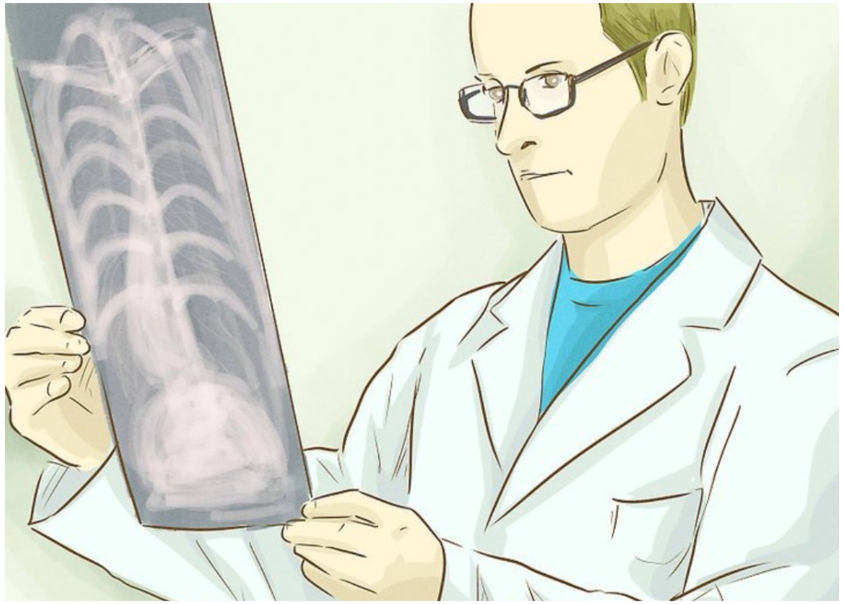
- The patient has a cough
- The patient has difficulty in breathing
- The patient has a fever

Introduction

Dealing with uncertainty:

You would like to determine how likely the patient is infected with inhalational anthrax given that the patient has a cough, a fever, and difficulty breathing

Introduction



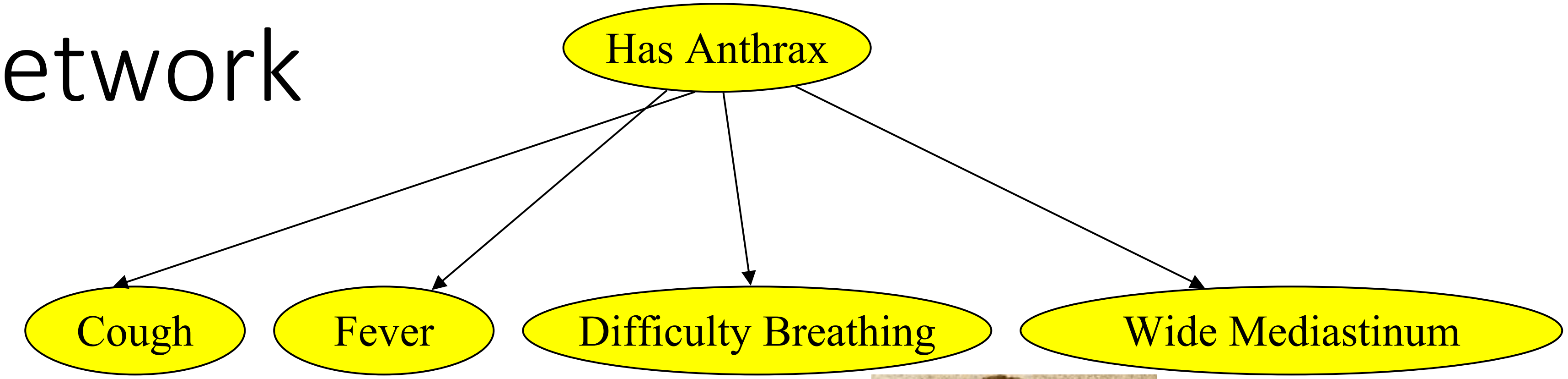
New evidence: X-ray image shows that the patient has a wide mediastinum.

Belief update: your belief that the patient is infected with inhalational anthrax is now much **higher** now.

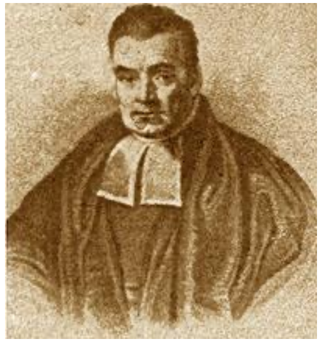
Introduction

- In the previous slides, what you observed affected your belief that the patient is infected with anthrax
- This is called reasoning with uncertainty
- Wouldn't it be nice if we had some tools for reasoning with uncertainty? In fact, we do…

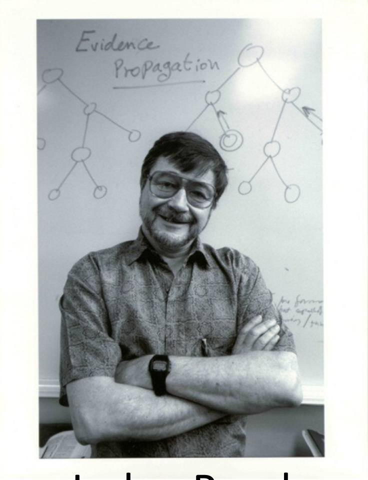
Bayesian Network



- Need a representation and reasoning system that is based on conditional independence
  - Compact yet expressive representation
  - Efficient reasoning procedures
- Bayesian Network is such a representation
  - Named after Thomas Bayes (ca. 1702 –1761)
  - Term coined in 1985 by Judea Pearl (1936 –) ,   2011 winner of the ACM Turing Award
  - Many applications, e.g., spam filtering, speech recognition, robotics, diagnostic systems and even syndromic surveillance



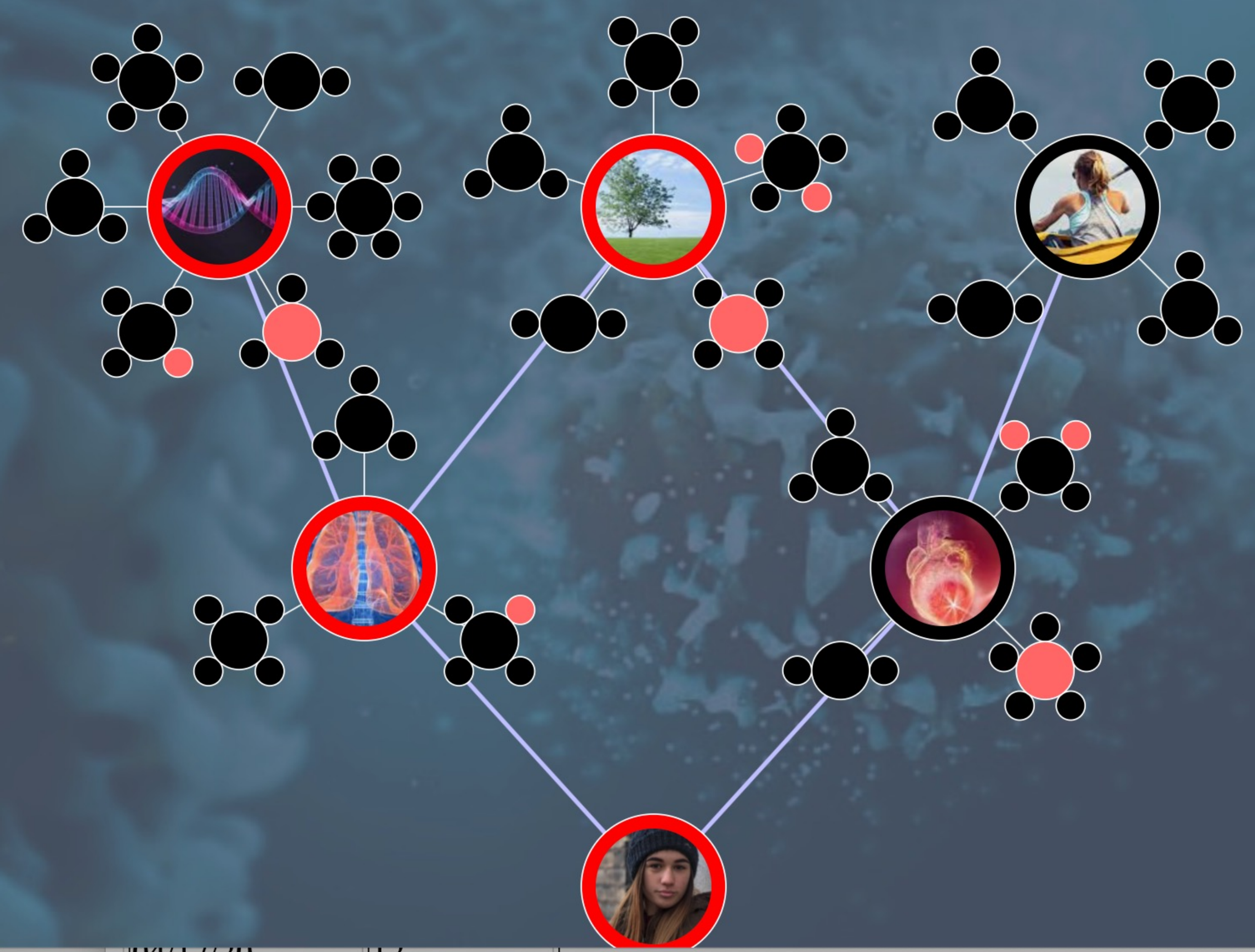
Thomas Bayes



Judea Pearl

Graphen Health Bayesian Network Visualizer

Time: 11



Outline

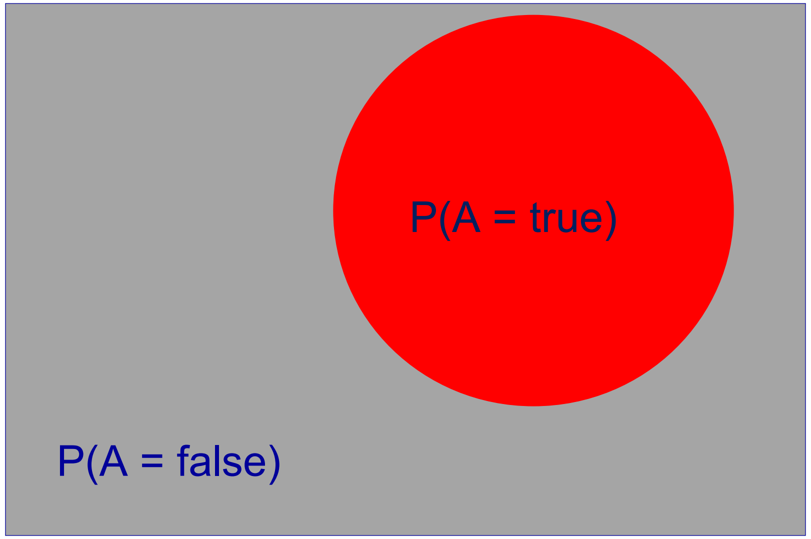
- Introduction
- **Probability Review**
- Bayesian Network
- Inference methods
- Network Structure Learning

Probabilities

We will write *P(A = true)* to mean the probability that *A = true*.

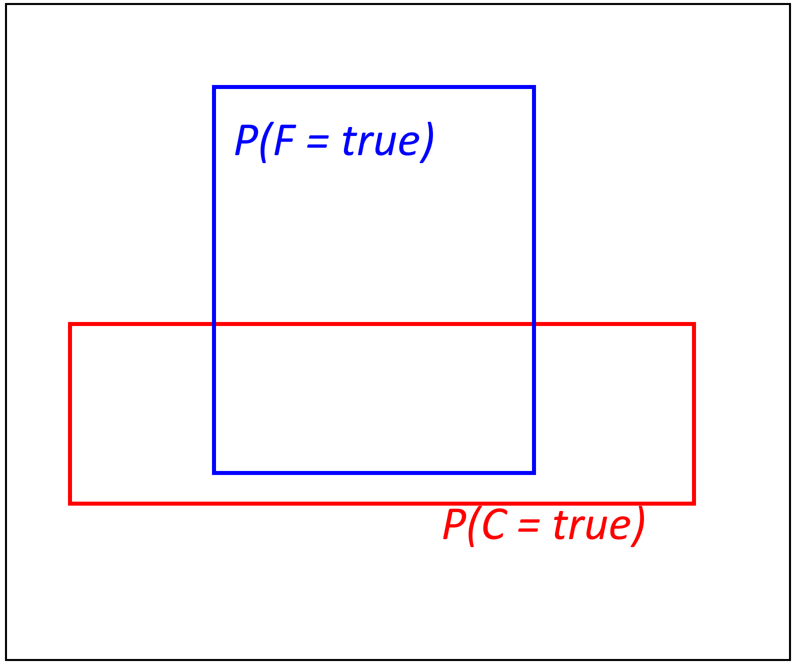
One definition of probability: the relative frequency with which an outcome would be obtained if the process were repeated a large number of times under similar conditions

The sum of the red and blue areas is 1



Conditional Probability

- P(A = *true* | B = *true*) : Out of all the outcomes in which B is true, how many also have A equal to true
- Read as: "Probability of A given B"



F = "Have a fever"
C = "Coming down with cold"

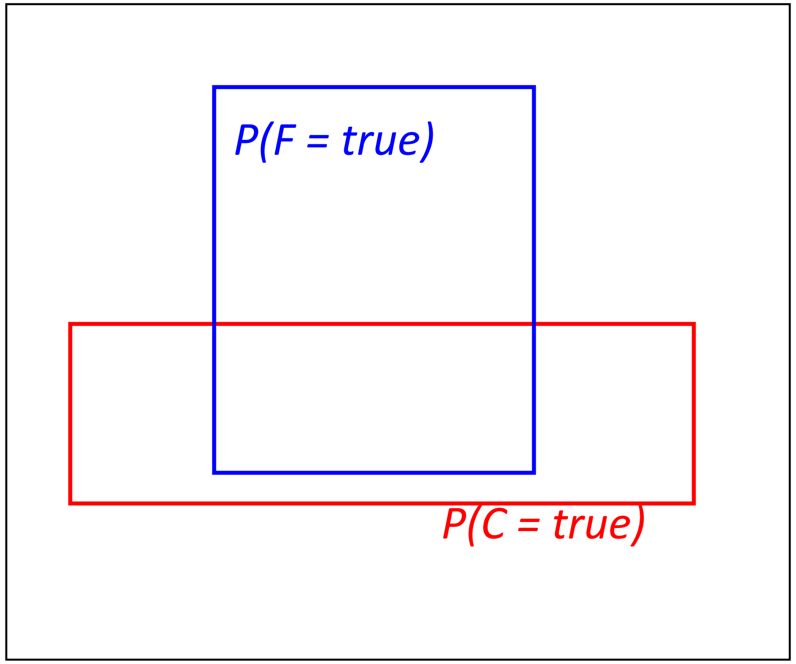
P(*F* = *true*) = 1/10
P(*C* = *true*) = 1/15
P(*F* = *true* | C = *true*) = 1/2

"Fever are rare and cold is rarer, but if you're coming down with cold there's a 50-50 chance you'll have a headache."

The Joint Probability Distribution

- *P(A = true, B = true)* :“the probability of *A = true* **and** *B = true*”
- Notice that:

P(F=true|C=true)



$$P(F = true / C = true) = \frac{\text{Area of "C and F" region}}{\text{Area of "C" region}} = \frac{P(C = true, F = true)}{P(C = true)}$$

The Joint Probability Distribution

- Joint probabilities can be between any number of variables

  e.g. *P(A = true, B = true, C = true)*

- For each combination of variables, we need to say how probable that combination is

A	B	C	P(A,B,C)
false	false	false	0.1
false	false	true	0.2
false	true	false	0.05
false	true	true	0.05
true	false	false	0.3
true	false	true	0.1
true	true	false	0.05
true	true	true	0.15

Sums to 1

The Joint Probability Distribution

- Once you have the joint probability distribution, you can calculate any probability involving *A*, *B*, and *C*
- Note: May need to use marginalization and Bayes rule,

A	B	C	P(A,B,C)
false	false	false	0.1
false	false	true	0.2
false	true	false	0.05
false	true	true	0.05
true	false	false	0.3
true	false	true	0.1
true	true	false	0.05
true	true	true	0.15

Examples of things you can compute:

- $P(A=true)$ = sum of $P(A,B,C)$ in rows with $A=true$
- $P(A=true, B = true \mid C=true) =$

  $P(A = true, B = true, C = true) / P(C = true)$

Independence

Variables *A* and *B* are independent if any of the following hold:

- $P(A,B) = P(A)\,P(B)$
- $P(A \mid B) = P(A)$
- $P(B \mid A) = P(B)$

Independence

How is independence useful?

- Suppose you have n coin flips and you want to calculate the joint distribution $\boldsymbol{P}(C_1, \ldots, C_n)$
- If the coin flips are not independent, you need 2^n values in the table
- If the coin flips are independent, then

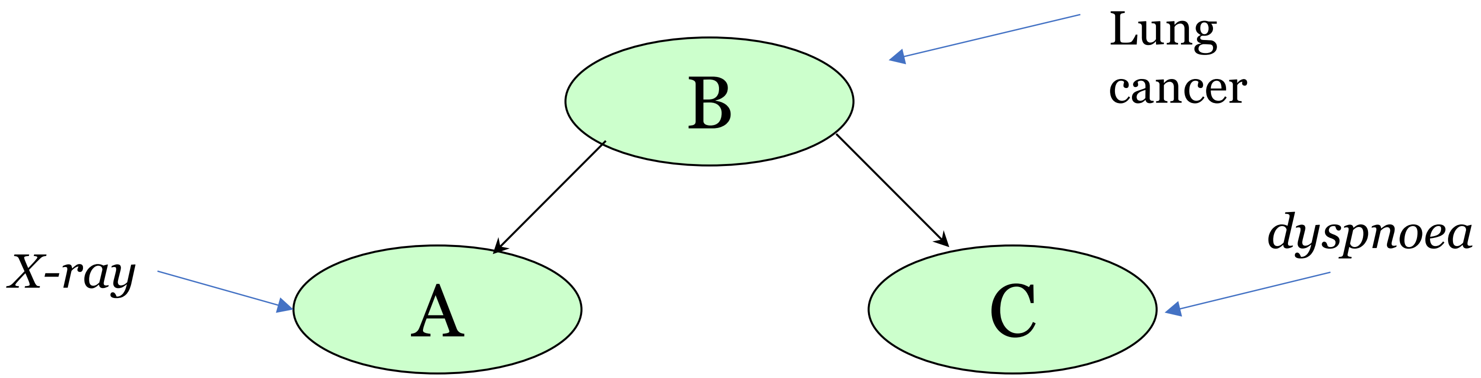
$$P(C_1, \ldots, C_n) = \prod_{i=1}^{n} P(C_i)$$

Conditional Independence

- *C* and *A* are conditionally independent given *B* if the following holds:

  $P(C \mid A, B) = P(C \mid B)$

- *Example: "Cancer is a common cause of the two symptoms: a positive X-ray and dyspnoea"*



- *Joint distribution:*
  $P(A,B,C)=P(C/A,B)P(A,B)=P(C/B)P(A,B)=P(C/B)P(A/B)P(B)$

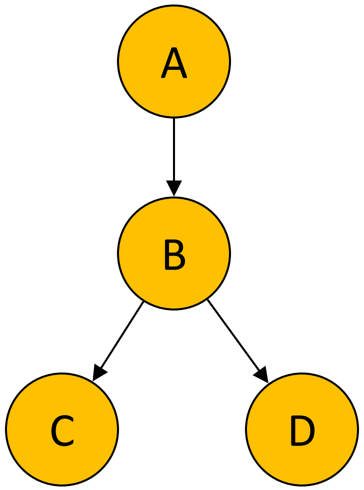
Outline

- Introduction
- Probability Review
- **Bayesian Network**
- Inference methods

A Bayesian Network

A Bayesian network is made up of:

1. A Directed Acyclic Graph



2. A set of tables for each node in the graph: conditional probability table

A	P(A)
false	0.4
true	0.6

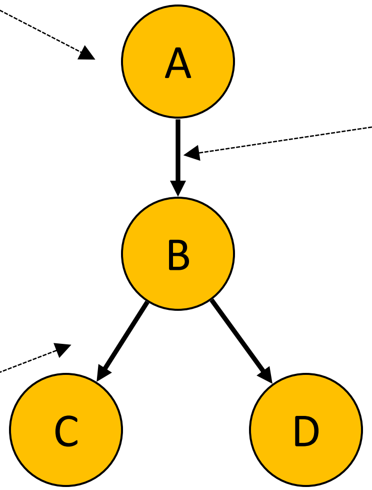
A	B	P(B|A)
false	false	0.03
false	true	0.97
true	false	0.6
true	true	0.4

B	D	P(D|B)
false	false	0.01
false	true	0.99
true	false	0.04
true	true	0.96

B	C	P(C|B)
false	false	0.3
false	true	0.7
true	false	0.8
true	true	0.2

A Directed Acyclic Graph

Each node in the graph is a random variable



A node X is a parent of another node Y if there is an arrow from node X to node Y e.g. A is a parent of B

an arrow from node X to node Y means X has a direct influence on Y

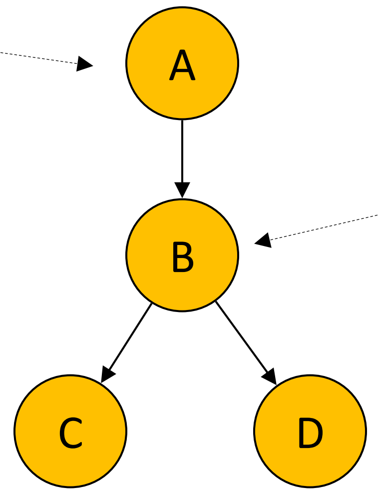
A Set of Tables for Each Node

Each node X_i has a conditional probability distribution

$P(X_i \mid Parents(X_i))$ that quantifies the effect of the parents on the node except the root node

A	P(A)
false	0.4
true	0.6

A	B	P(B|A)
false	false	0.03
false	true	0.97
true	false	0.6
true	true	0.4



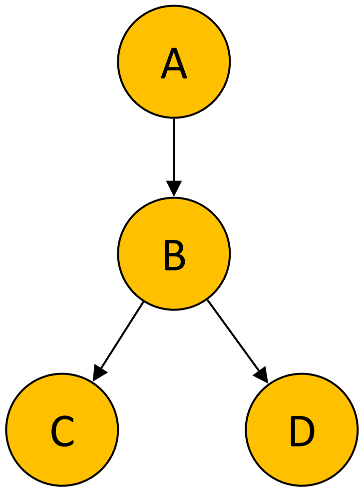
B	C	P(C|B)
false	false	0.3
false	true	0.7
true	false	0.8
true	true	0.2

B	D	P(D|B)
false	false	0.01
false	true	0.99
true	false	0.04
true	true	0.96

Bayesian Networks

Two important properties:

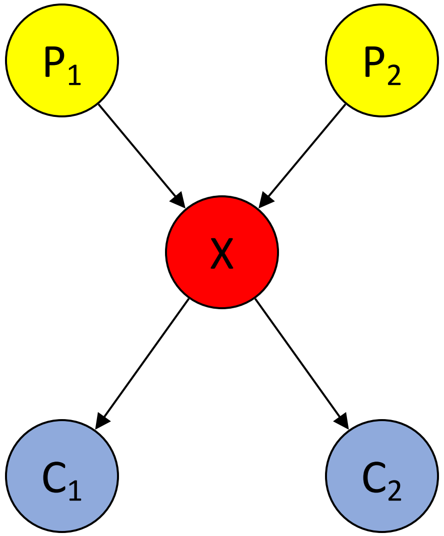
1. Encodes the conditional independence relationships between the variables in the graph structure
2. Is a compact representation of the joint probability distribution over the variables



Conditional Independence

The probability distribution for each node depends only on its parents

C_1 and C_2 are conditionally independent given X



The Joint Probability Distribution

Due to the conditional independence property, the joint probability distribution over all the variables X_1, …, X_n in the Bayesian net can be computed using the formula:

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$

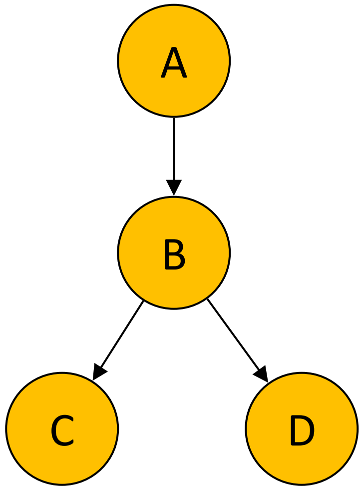
Using a Bayesian Network Example

P(A = true, B = true, C = true, D = true)

= P(A = true) * P(B = true | A = true) *

P(C = true | B = true) *P(D = true | B = true)

= (0.6)*(0.4)*(0.2)*(0.96)



Using a Bayesian Network Example

P(A = true, B = true, C = true, D = true)

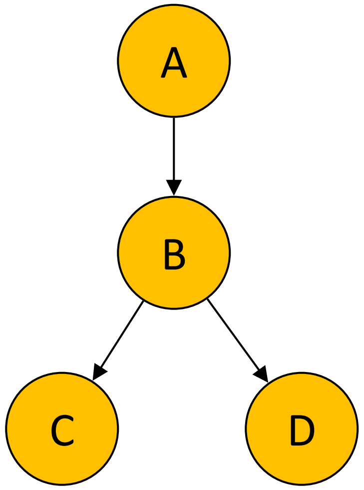
= P(A = true) * P(B = true | A = true) *

P(C = true | B = true) *P(D = true | B = true)

= (0.6)*(0.4)*(0.2)*(0.96)

from the graph structure

From the conditional probability tables



A	P(A)
false	0.4
true	0.6

A	B	P(B|A)
false	false	0.03
false	true	0.97
true	false	0.6
true	true	0.4

B	D	P(D|B)
false	false	0.01
false	true	0.99
true	false	0.04
true	true	0.96

B	C	P(C|B)
false	false	0.3
false	true	0.7
true	false	0.8
true	true	0.2

Another example

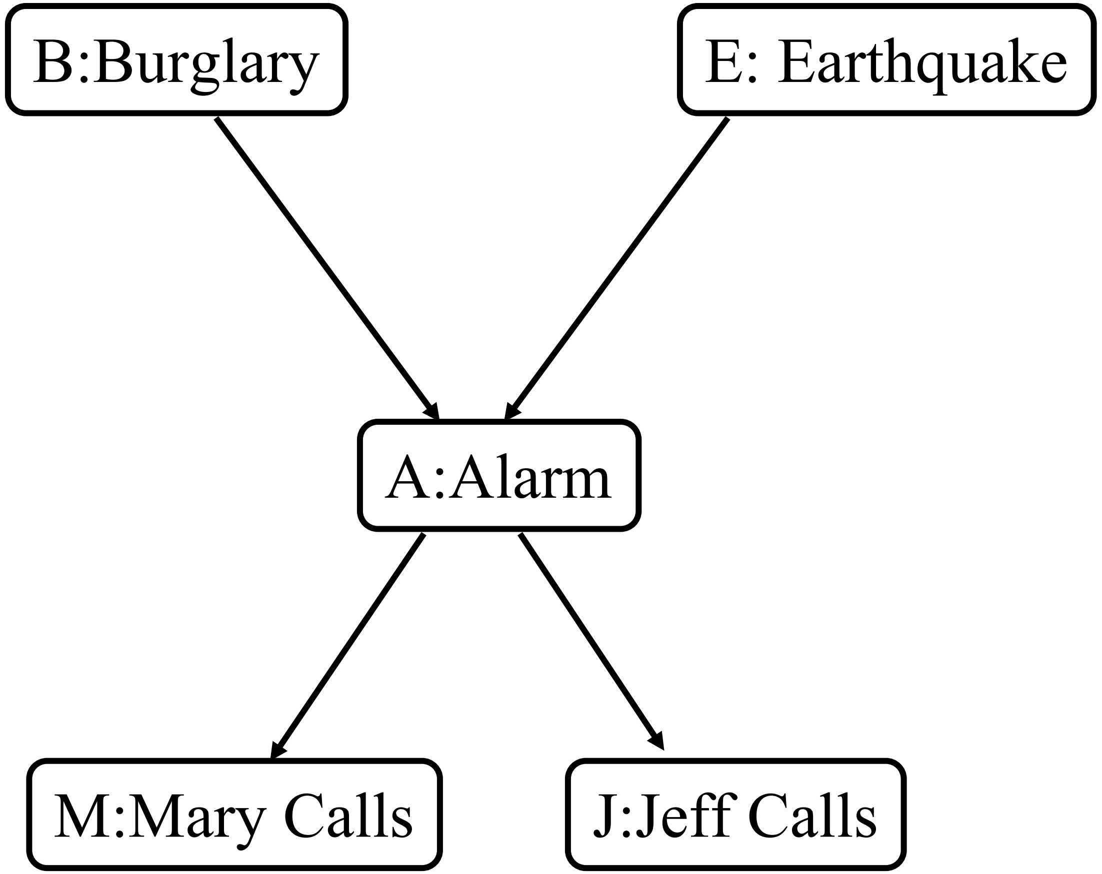
- I'm at work, neighbor Jeff calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: *Burglary*, *Earthquake*, *Alarm*, *JeffCalls*, *MaryCalls*

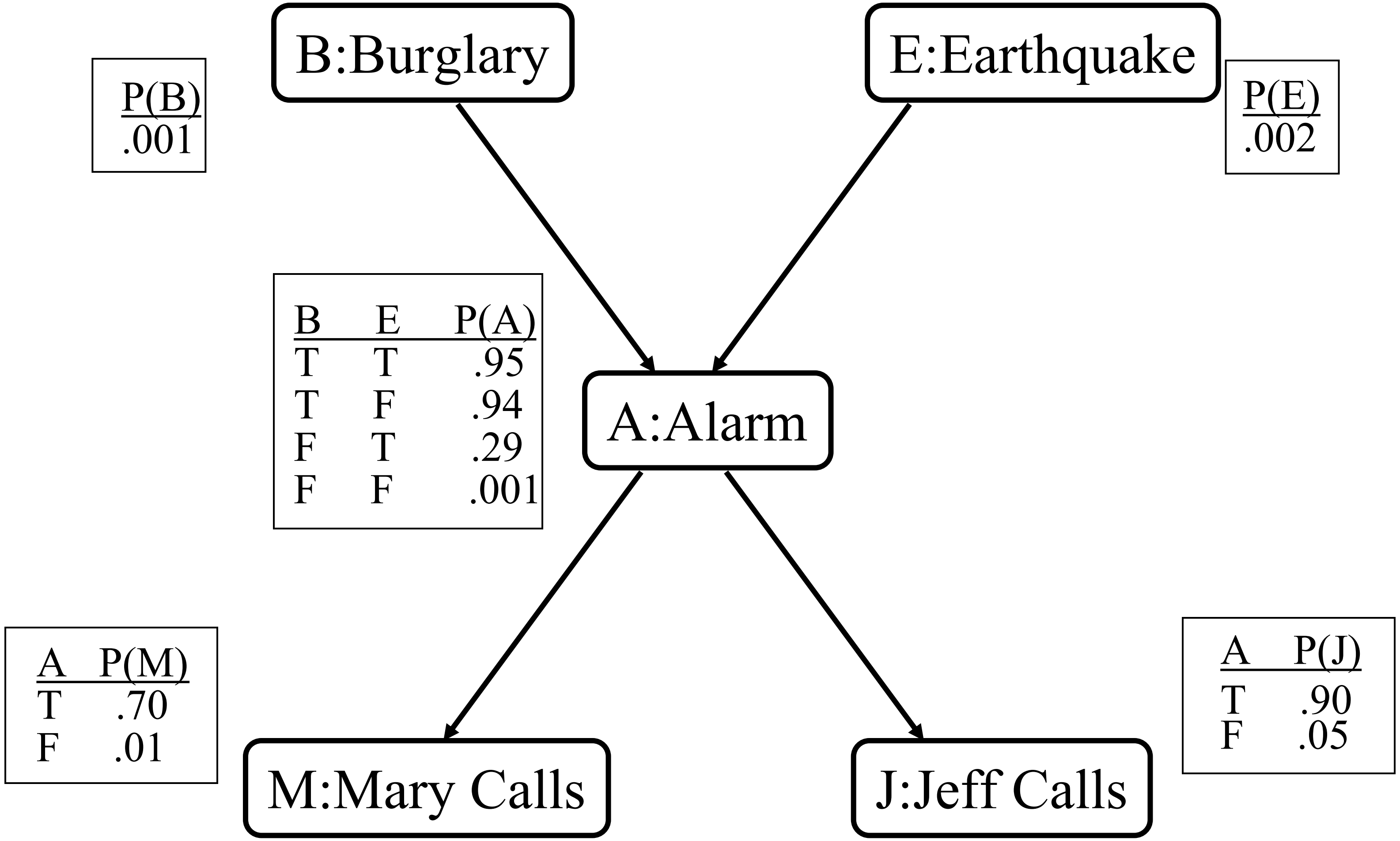
- Network topology reflects "causal" knowledge:

  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause Jeff to call

Another example: Earthquake or Burglar



Bayesian Network for Alarm Domain



P(B)
.001

P(E)
.002

B	E	P(A)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(M)
T	.70
F	.01

A	P(J)
T	.90
F	.05

P(J =true, M=true, A=true, B=false, E=false)

= P(J =true |A =true)P(M =true |A =true)P(A =true |B =false, E =false)P(B =false)P(E =false)

= 0.9 * 0.7 * 0.001 * 0.999 * 0.998 = 0.00062

Outline

- Introduction
- Probability Review
- Bayesian Network
- **Inference methods**
- Network Structure Learning

Inference

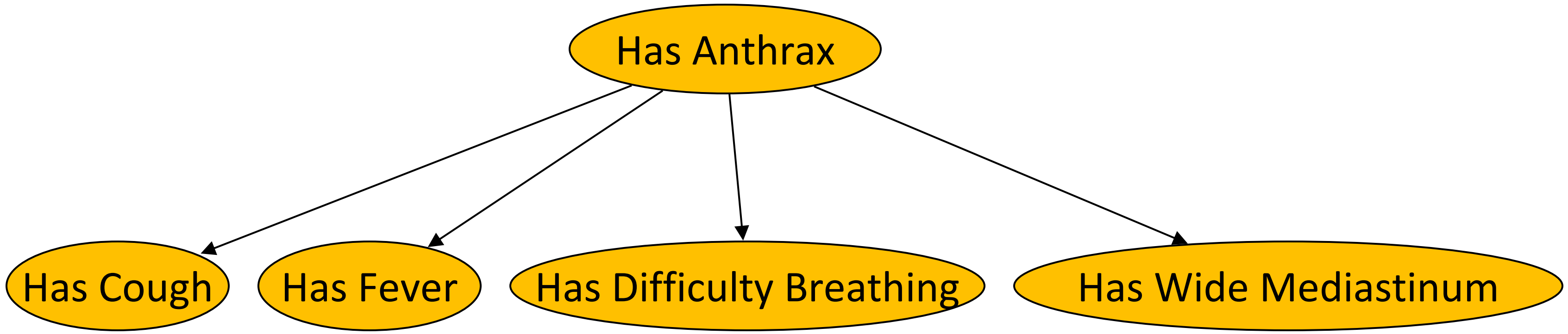
- How can one infer the (probabilities of) values of one or more network variables, given observed values of others?
- P(X | E)

E = The evidence variable(s)

X = The query variable(s)

- Bayes net contains all information needed for this inference
- If only one variable with unknown value, easy to infer it
- In general case, problem is NP hard

Inference: example



- An example of a query would be:

  P(*Anthrax = true* | *Fever = true*, *Cough = true*)

- Note: Even though *HasDifficultyBreathing* and *HasWideMediastinum* are in the Bayesian network, they are not given values in the query
- They are treated as unobserved variables

Inference in Bayesian Network

- Exact inference:

  Variable Elimination

  Junction Tree

- Approximate inference:

  Markov Chain Monte Carlo

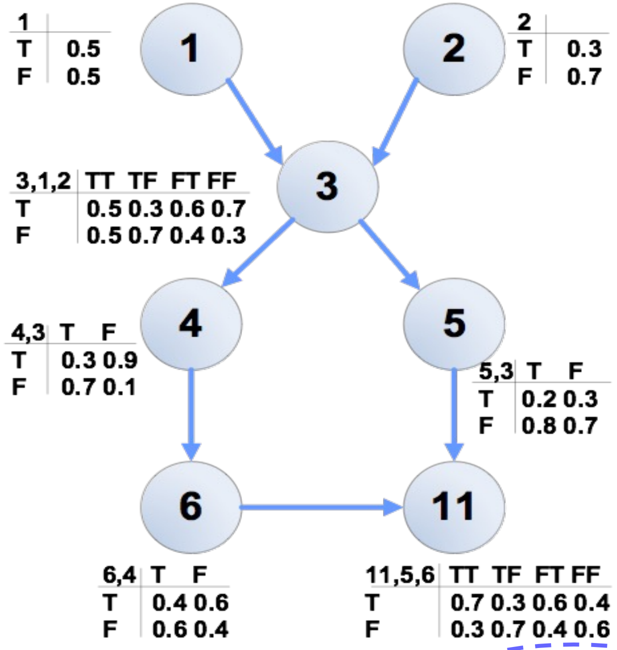
  Variational Methods

From Bayesian Network to Junction Tree – Exact Inference

- Conditional dependence among random variables allows information propagated from a node to another
  ⇨ foundation of probabilistic inference

Not straightforward

node → random variable
edge → precedence relationship
conditional probability table (CPT)



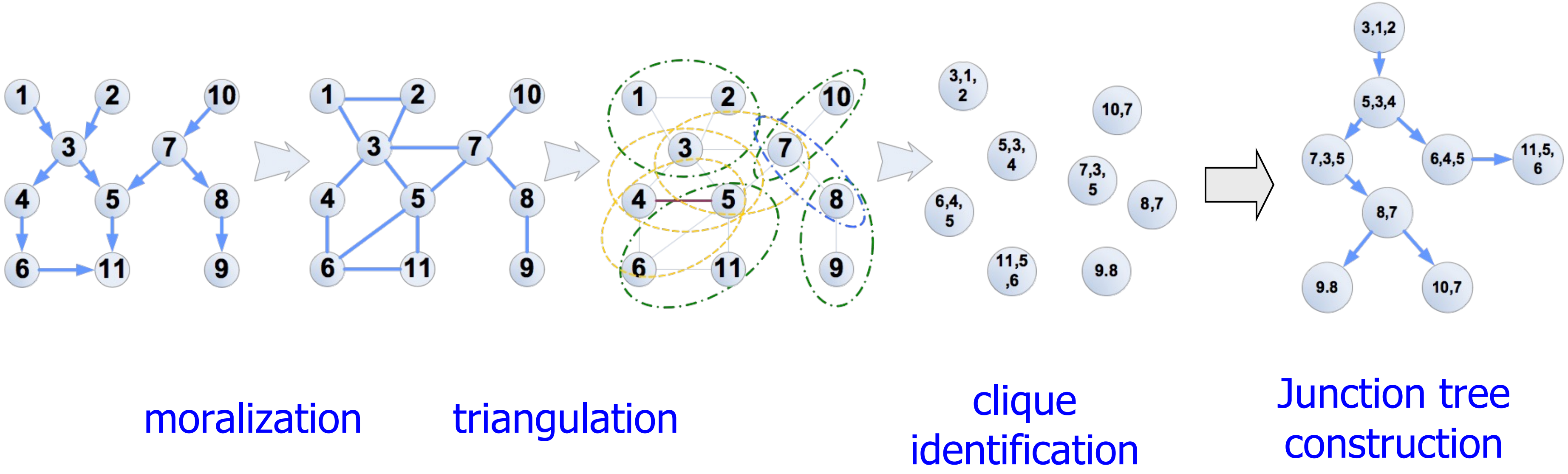
Given evidence (observations) E, output the posterior probabilities of query P(Q|E)

NP hard

Bayes' theorem can not be applied directly to non-singly connected networks, as it would yield erroneous results

Therefore, junction trees are used to implement exact inference

Conversion of Bayesian Network into Junction Tree

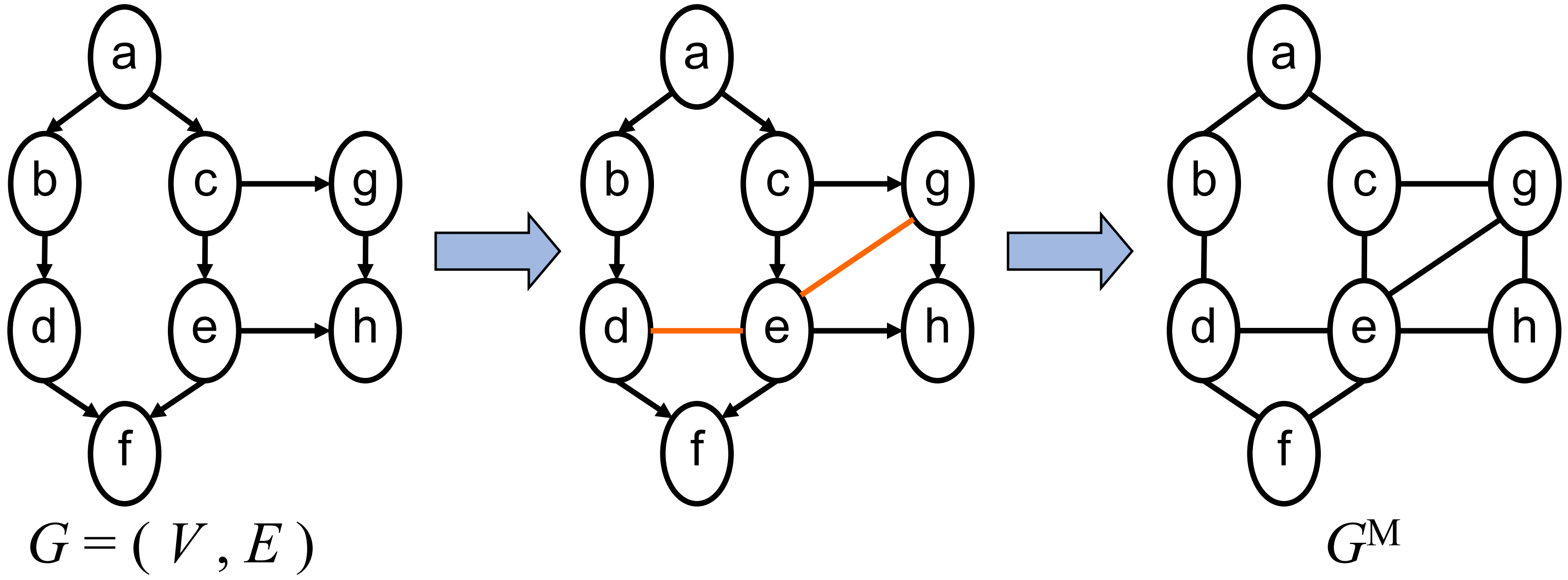


– Parallel Moralization connects all parents of each node
– Parallel Triangulation chordalizes cycles with more than 3 edges
– Clique identification finds cliques using node elimination
  - Node elimination is a step look-ahead algorithms that brings challenges in processing large scale graphs
– Parallel Junction tree construction builds a hypergraph of the Bayesian network based on running intersection property

Constructing Junction Trees

1. Moralization: construct an undirected graph from the DAG
2. Triangulation: Selectively add arcs to the moral graph
3. Build a junction graph by identifying the cliques and separators
4. Build the junction tree by find an appropriate spanning tree

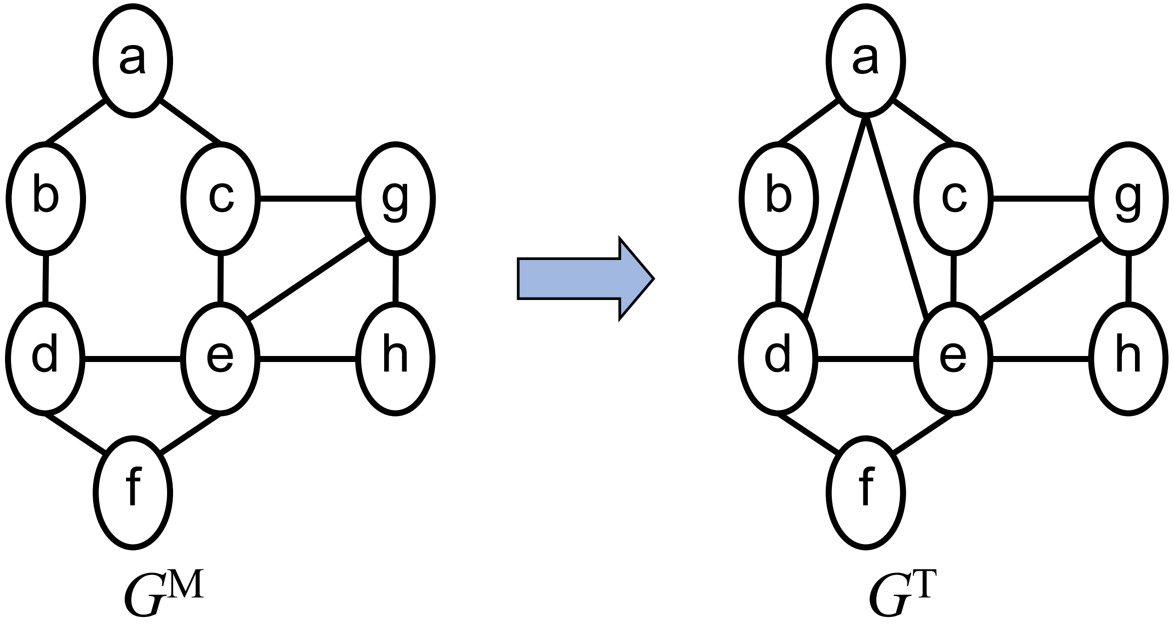
Step 1: Moralization: marry the parents



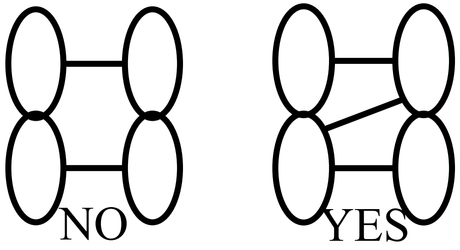
1. For all w $\in V$:
   - For all u,v$\in$*parents*(w) add an edge e=u-v.

2. Undirect all edges.

Step 2: Triangulation

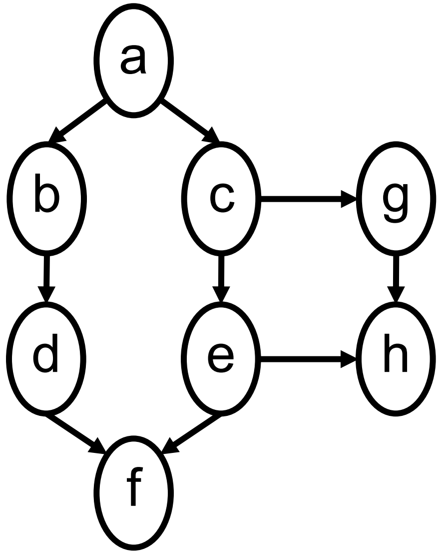


Add edges to G^M such that there is no cycle with length ≥ 4 that does not contain a chord.



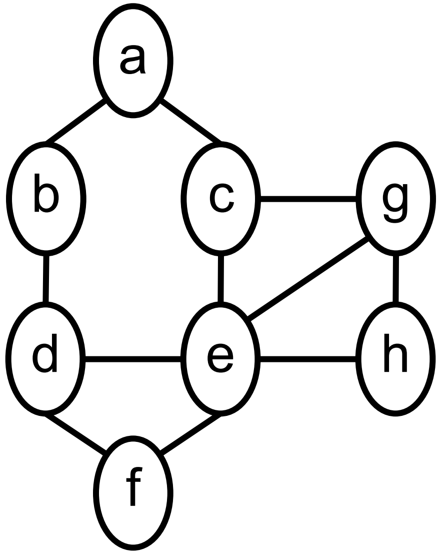
Step 3: Build the junction graph

- A junction graph for an undirected graph G is an undirected, labeled graph.
- Clique: a subgraph that is complete and maximal.
- The nodes are the cliques in G.
- If two cliques intersect, they are joined in the junction graph by an edge labeled with their intersection. (separators)

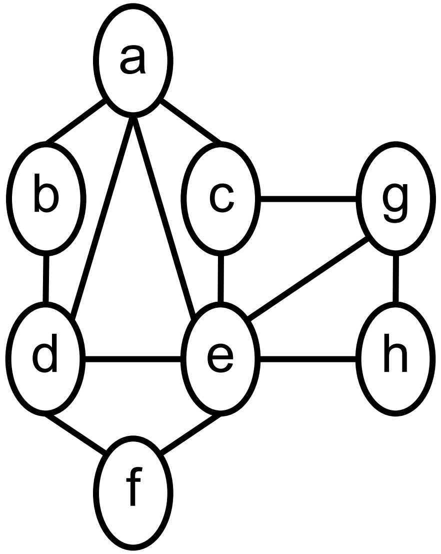


Bayesian Network

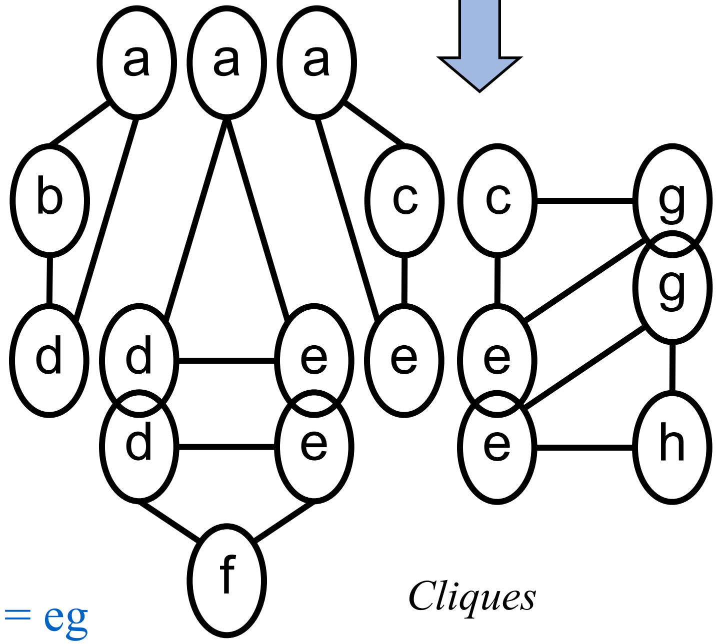
$G = (V, E)$



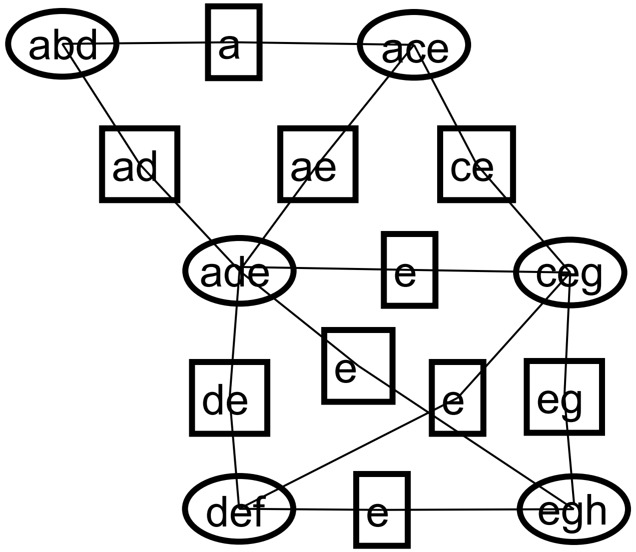
Moral graph G^M



Triangulated graph G^T



Cliques



separators

e.g. ceg ∩ egh = eg

Junction graph G^J *(not complete)*

Step 4: Junction Tree

- A junction tree is a sub-graph of the junction graph that
  - Is a tree
  - Contains all the cliques (spanning tree)
  - Satisfies the *running intersection* property:

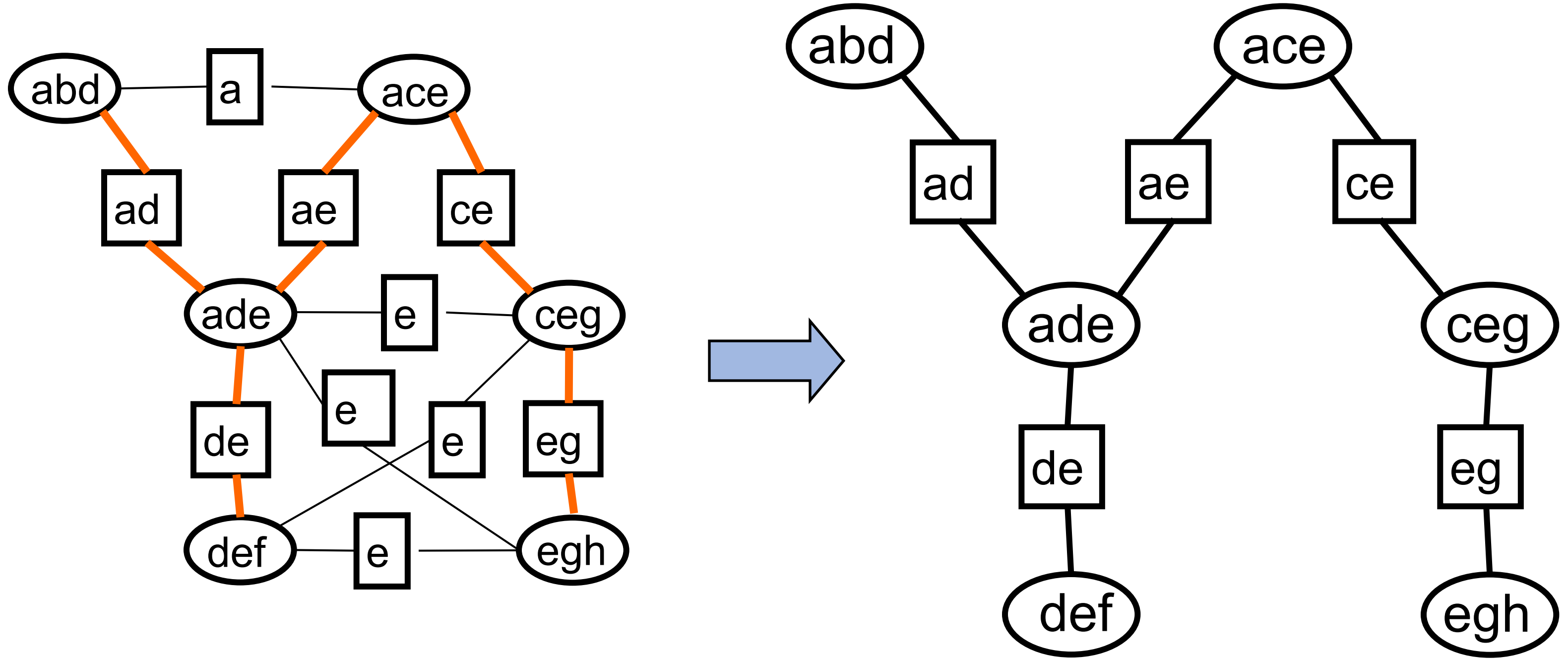
    for each pair of nodes U, V, all nodes on the path between U and V contain $U \cap V$

Step 4: Junction Tree (cont.)

- Theorem: An undirected graph is triangulated if and only if its junction graph has a junction tree
- Definition: The *weight* of a link in a junction graph is the number of variable in the label. The weight of a junction tree is the sum of weights of the labels.
- Theorem: A sub-tree of the junction graph of a triangulated graph is a junction tree if and only if it is a spanning of maximal weight

There are several methods to find MST.

Kruskal's algorithm: choose successively a link of maximal weight unless it creates a cycle.



Junction graph G^J (not complete)

Junction tree G^{JT}

Inference using junction tree

- Potential $\emptyset_X$: a function that maps each instantiation x of a set of variables X into a nonnegative real number
- Marginalization: suppose $X \in Y$, $\emptyset_Y = \sum_{Y \backslash X} \emptyset_Y$
- Constraints on potentials
  1) Consistency property : for each clique X and neighboring separator S, it holds that

  2) The potentials encode the joint distribution P(U) of the network according to

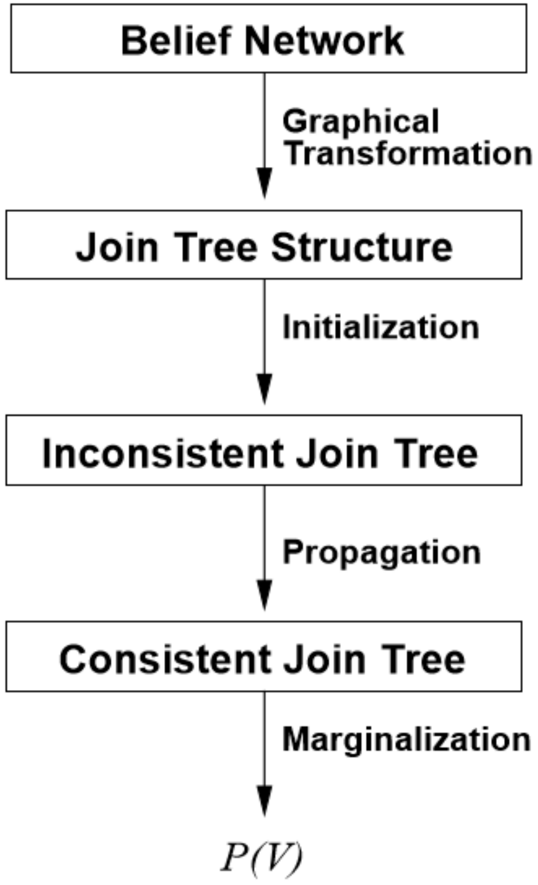
- Property: for each clique/separator, it holds that $\emptyset_X = P(X)$
  That means, for any variable $V \in X$, we can compute its marginal by

$$\sum_{\mathbf{X} \backslash \mathbf{S}} \phi_{\mathbf{X}} = \phi_{\mathbf{S}}.$$

$$P(\mathbf{U}) = \frac{\prod_i \phi_{\mathbf{X_i}}}{\prod_j \phi_{\mathbf{S_j}}}$$

$$P(V) = \sum_{\mathbf{X} \backslash \{\mathbf{V}\}} \phi_{\mathbf{X}}.$$

Inference without evidence



from Huang&Darwiche, 1996

Initialization

The following procedure assigns initial join tree potentials, using the conditional probabilities from the belief network:

1. For each cluster and sepset $\mathbf{X}$, set each $\phi_{\mathbf{X}}(\mathbf{x})$ to 1:

$$\phi_{\mathbf{X}} \longleftarrow 1.$$

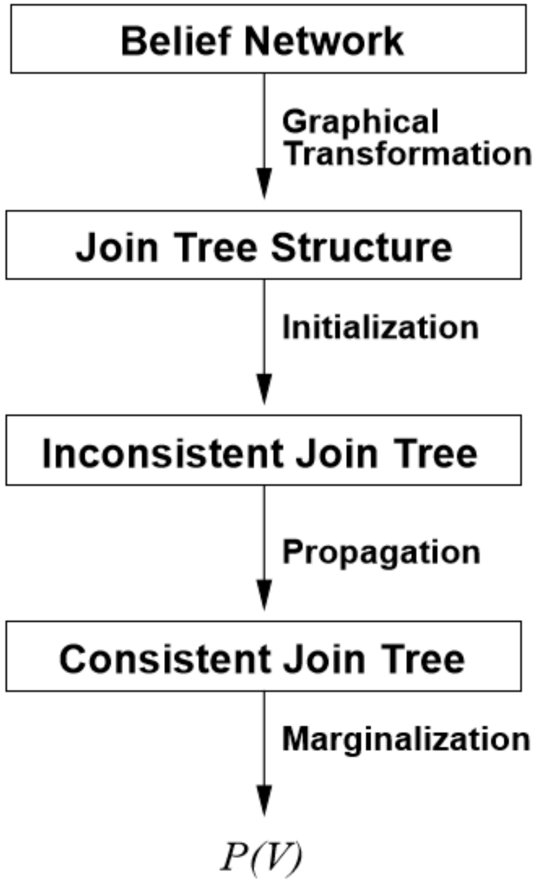
2. For each variable V, perform the following: Assign to V a cluster $\mathbf{X}$ that contains $\mathbf{F_V}$;[11] call $\mathbf{X}$ the **parent cluster of $\mathbf{F_V}$**. Multiply $\phi_{\mathbf{X}}$ by $P(V \mid \mathbf{\Pi}_V)$:

$$\phi_{\mathbf{X}} \longleftarrow \phi_{\mathbf{X}} P(V \mid \mathbf{\Pi}_V).$$

After initialization, the conditional distribution $P(V \mid \mathbf{\Pi}_V)$ of each variable V has been multiplied into some cluster potential. The initialization procedure satisfies Equation (2) as follows:

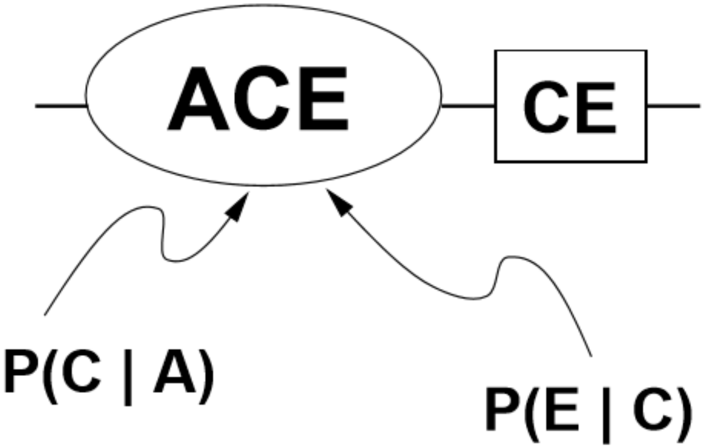
$$\frac{\prod_{i=1}^{N} \phi_{\mathbf{X_i}}}{\prod_{j=1}^{N-1} \phi_{\mathbf{S_j}}} = \frac{\prod_{k=1}^{Q} P(V_k \mid \mathbf{C}_{V_k})}{1} = P(\mathbf{U}),$$

where N is the number of clusters, Q is the number of variables, and $\phi_{\mathbf{X_i}}$ and $\phi_{\mathbf{S_j}}$ are the cluster and sepset potentials, respectively.



from Huang&Darwiche, 1996

Example for initialization



φ_{ACE}

a	c	e	Initial Values
on	on	on	1 × .7 × .3 = .21
on	on	off	1 × .7 × .7 = .49
on	off	on	1 × .3 × .6 = .18
on	off	off	1 × .3 × .4 = .12
off	on	on	1 × .2 × .3 = .06
off	on	off	1 × .2 × .7 = .14
off	off	on	1 × .8 × .6 = .48
off	off	off	1 × .8 × .4 = .32

φ_{CE}

c	e	Initial Values
on	on	1
on	off	1
off	on	1
off	off	1

etc.

from Huang&Darwiche, 1996

Global propagation

- Single message pass

Consider two adjacent clusters X and Y with separator R.

A message pass from X to Y occurs in two steps:

1. **Projection.** Assign a new table to $\mathbf{R}$, saving the old table:

$$\phi_{\mathbf{R}}^{old} \longleftarrow \phi_{\mathbf{R}}.$$

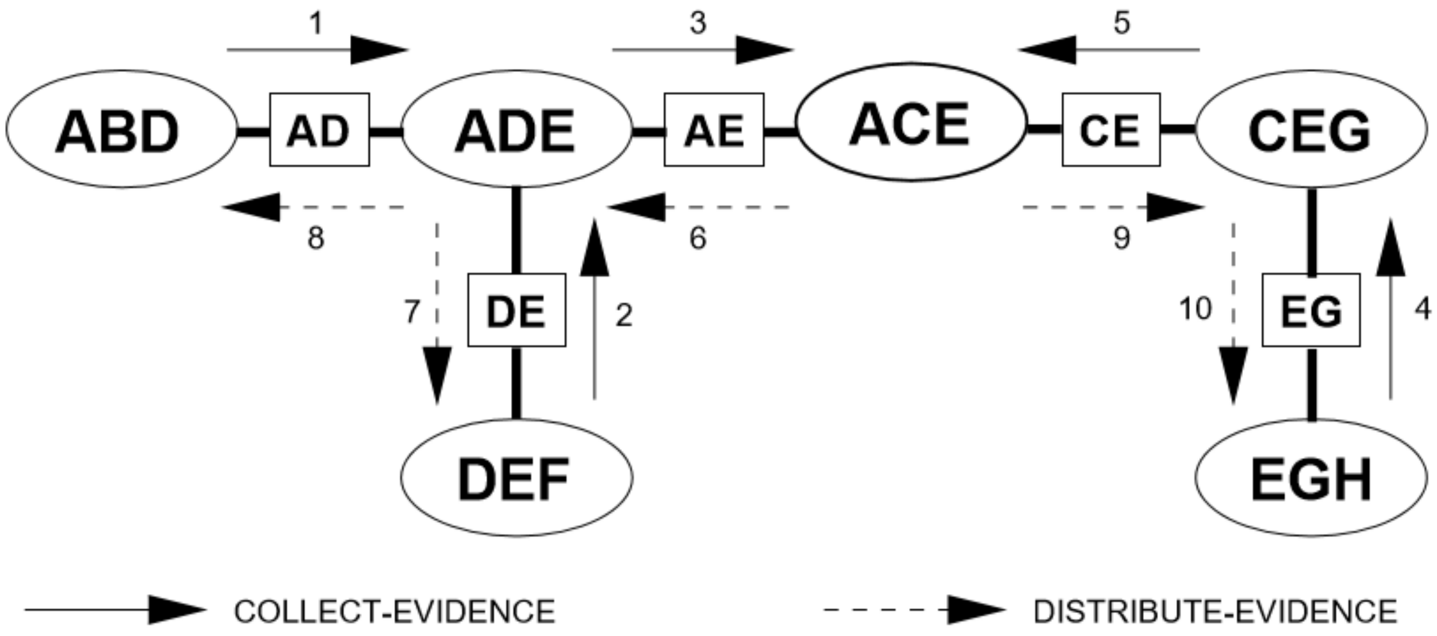
$$\phi_{\mathbf{R}} \longleftarrow \sum_{\mathbf{X}\setminus\mathbf{R}} \phi_{\mathbf{X}}.$$

2. **Absorption.** Assign a new table to $\mathbf{Y}$, using both the old and new tables of $\mathbf{R}$:

$$\phi_{\mathbf{Y}} \longleftarrow \phi_{\mathbf{Y}} \frac{\phi_{\mathbf{R}}}{\phi_{\mathbf{R}}^{old}}.$$

from Huang&Darwiche, 1996

Global propagation



COLLECT-EVIDENCE(X)

1. Mark **X**.
2. Call COLLECT-EVIDENCE recursively on **X**'s unmarked neighboring clusters, if any.
3. Pass a message from **X** to the cluster which invoked COLLECT-EVIDENCE(**X**).

DISTRIBUTE-EVIDENCE(X)

1. Mark **X**.
2. Pass a message from **X** to each of its unmarked neighboring clusters, if any.
3. Call DISTRIBUTE-EVIDENCE recursively on **X**'s unmarked neighboring clusters, if any.

from Huang&Darwiche, 1996

Marginalization

- Once we have a consistent junction tree, we can compute P(V) for each variable of interest V by computing the marginals.

$$P(V) = \sum_{\mathbf{X} \setminus \{\mathbf{V}\}} \phi_{\mathbf{X}}.$$

$\varphi_{ABD} =$

a	b	d	φ_{ABD}(abd)
on	on	on	.225
on	on	off	.025
on	off	on	.125
on	off	off	.125
off	on	on	.180
off	on	off	.020
off	off	on	.150
off	off	off	.150

$$P(A) = \sum_{BD} \varphi_{ABD} =$$

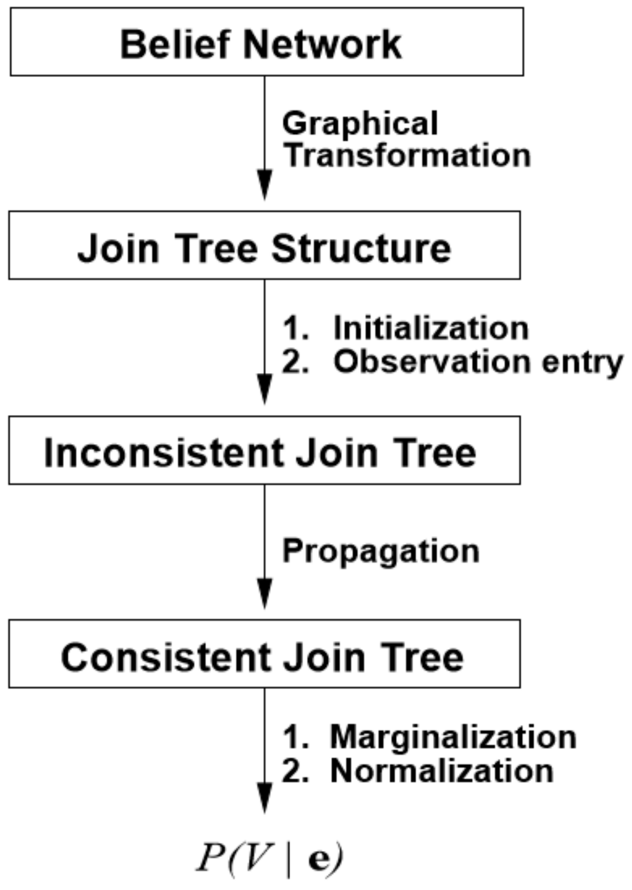
a	P(a)
on	.225 + .025 + .125 + .125 = .500
off	.180 + .020 + .150 + .150 = .500

$$P(D) = \sum_{AB} \varphi_{ABD} =$$

d	P(d)
on	.225 + .125 + .180 + .150 = .680
off	.025 + .125 + .020 + .150 = .320

from Huang&Darwiche, 1996

Inference with evidence



from Huang&Darwiche, 1996

Observations and Likelihoods

- An observation is a statement of the form $V = v$
- Observations are the simplest forms of evidence.
- Collections of observations denoted by E.
- Define likelihood to encode observations:

  - If $V \in \mathbf{E}$—that is, if V is observed—then assign each $\Lambda_V(v)$ as follows:

  $$\Lambda_V(v) = \begin{cases} 1, & \text{when } v \text{ is the observed value of } V \\ 0, & \text{otherwise} \end{cases}$$

  - If $V \notin \mathbf{E}$—that is, if the value of V is unknown—then assign $\Lambda_V(v) = 1$ for each value v.

from Huang&Darwiche, 1996

Example of likelihood encoding

- Suppose we have observations C =on, E = off

Variable V	$\Lambda_V(v)$	
	$v = on$	$v = off$
A	1	1
B	1	1
C	1	0
D	1	1
E	0	1
F	1	1
G	1	1
H	1	1

from Huang&Darwiche, 1996

Initialization with observations

We keep track of observations by maintaining a likelihood for each variable. We initialize these likelihoods by adding step 2b to the initialization procedure below:

1. For each cluster and sepset $\mathbf{X}$, set each $\phi_{\mathbf{X}}(\mathbf{x})$ to 1:

$$\phi_{\mathbf{X}} \longleftarrow 1.$$

2. For each variable V:

(a) Assign to V a cluster $\mathbf{X}$ that contains $\mathbf{F_V}$; multiply $\phi_{\mathbf{X}}$ by $P(V \mid \mathbf{\Pi}_V)$:

$$\phi_{\mathbf{X}} \longleftarrow \phi_{\mathbf{X}} P(V \mid \mathbf{\Pi}_V).$$

(b) Set each likelihood element $\Lambda_V(v)$ to 1:

$$\Lambda_V \longleftarrow 1.$$

Observation entry

Note that upon completion of initialization, the likelihoods encode no observations. We incorporate each observation $V = v$ by encoding the observation as a likelihood, and then incorporating this likelihood into the join tree, as follows:

1. Encode the observation $V = v$ as a likelihood Λ_V^{new}.

2. Identify a cluster $\mathbf{X}$ that contains V.[13]

3. Update $\phi_{\mathbf{X}}$ and Λ_V:

$$\begin{aligned} \phi_{\mathbf{X}} &\longleftarrow \phi_{\mathbf{X}} \Lambda_V^{new}. \\ \Lambda_V &\longleftarrow \Lambda_V^{new}. \end{aligned} \tag{1}$$

By entering a set of observations $\mathbf{e}$ as described above, we modify the join tree potentials, so that *all subsequent probabilities derived from the join tree are probabilities of events that are conjoined with evidence* $\mathbf{e}$. In other words, instead of computing $P(\mathbf{X})$ and $P(V)$, we compute $P(\mathbf{X}, \mathbf{e})$ and $P(V, \mathbf{e})$, respectively. Note also that the join tree encodes $P(\mathbf{U}, \mathbf{e})$ instead of $P(\mathbf{U})$ (see Equation (2)).

from Huang&Darwiche, 1996

Normalization

After the join tree is made consistent through global propagation, we have, for each cluster (or sepset) $\mathbf{X}$, $\phi_{\mathbf{X}} = P(\mathbf{X}, \mathbf{e})$, where $\mathbf{e}$ denotes the observations incorporated into the join tree according to Section 6.4 [2]. When we marginalize a cluster potential $\phi_{\mathbf{X}}$ into a variable V, we obtain the probability of V *and* $\mathbf{e}$:

$$P(V, \mathbf{e}) = \sum_{\mathbf{X} \setminus \{\mathbf{V}\}} \phi_{\mathbf{X}}.$$

Our goal is to compute $P(V \mid \mathbf{e})$, the probability of V *given* $\mathbf{e}$. We obtain $P(V \mid \mathbf{e})$ from $P(V, \mathbf{e})$ by **normalizing** $P(V, \mathbf{e})$ as follows:

$$P(V \mid \mathbf{e}) = \frac{P(V, \mathbf{e})}{P(\mathbf{e})} = \frac{P(V, \mathbf{e})}{\sum_{V} P(V, \mathbf{e})}. \tag{2}$$

The probability of the observations $P(\mathbf{e})$ is often referred to as a **normalizing constant**.

from Huang&Darwiche, 1996

Approximate inference

- Exact inference is feasible in small to medium-sized networks
- Takes a very long time for large networks
- Turn to approximate inference techniques which are much faster and give pretty good results

Sampling

- Input: Bayesian network with set of nodes X
- Sample = a tuple with assigned values

  $s=(X_1=x_1, X_2=x_2, \dots, X_k=x_k)$
- Tuple may include all variables (except evidence) or a subset
- Sampling schemas dictate how to generate samples (tuples)
- Ideally, samples are distributed according to $P(X|E)$

Sampling algorithms

- Gibbs Sampling (MCMC)
- Importance Sampling
- Sequential Monte-Carlo (Particle Filtering) in Dynamic Bayesian Networks
- etc.

A list of Python libraries

Library	Algorithm	Algorithm Type	License
BayesPy	variational message passing	approximate	MIT
pomegranate	loopy belief	approximate	MIT
pgmpy	multiple	approximate/exact	MIT
libpgm	likelihood sampling	approximate	Proprietary
bayesnetinference	variable elimination	exact	None

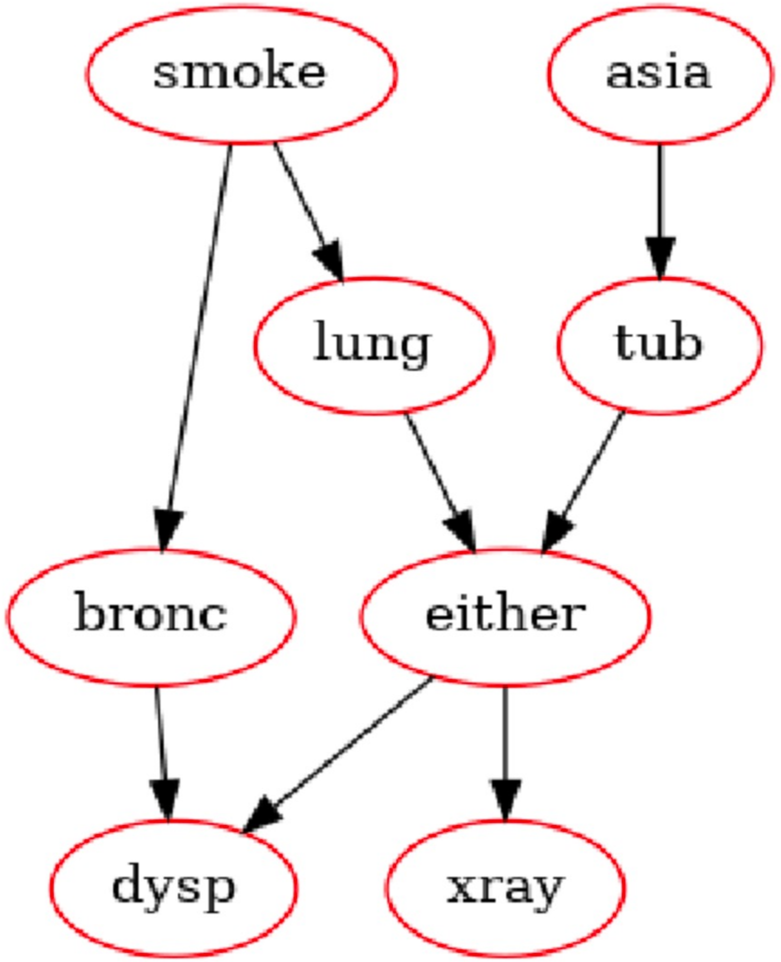
Outline

- Introduction
- Probability Review
- Bayesian Network
- Inference methods
- **Network Structure Learning**

Original Graph – Asia Example

```
In [18]: bnh.get_model_from_bif('/datadrive/masa/bayesian/asia.bif')
```

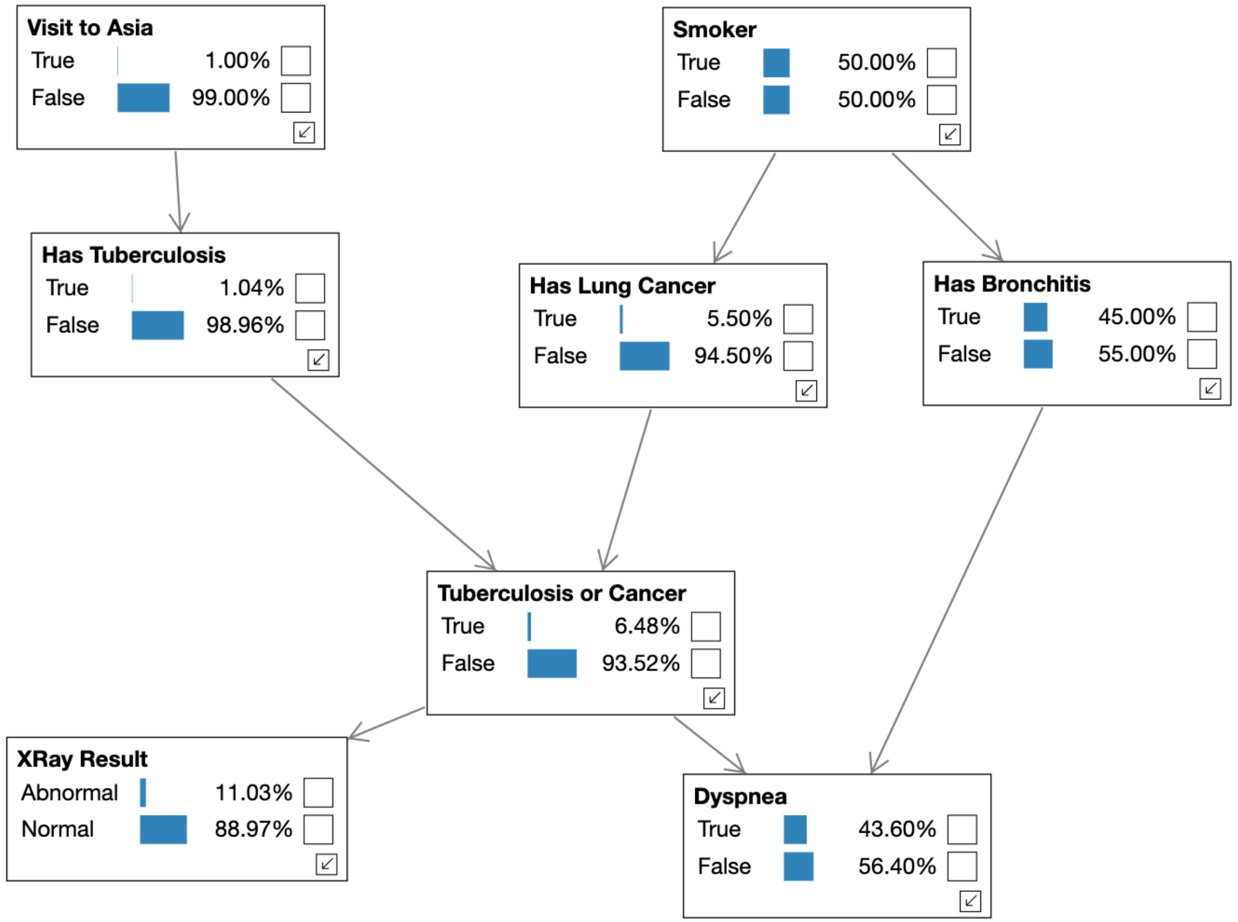
```
In [19]: bnh.plot_model()
```



Asia

This example is the well known Asia Bayesian network.

> The Bayesian network below will update when you click the check boxes to set evidence. The on
> very small subset of the features of the full User Interface and APIs.



Prediction

- Setting one (or more) column to None and let Pomegranate predict the value based on other observations
- Need to already have Bayesian network with conditional probabilities (through .bif file or generated network)
- Right example: original Asia network, 64% accuracy

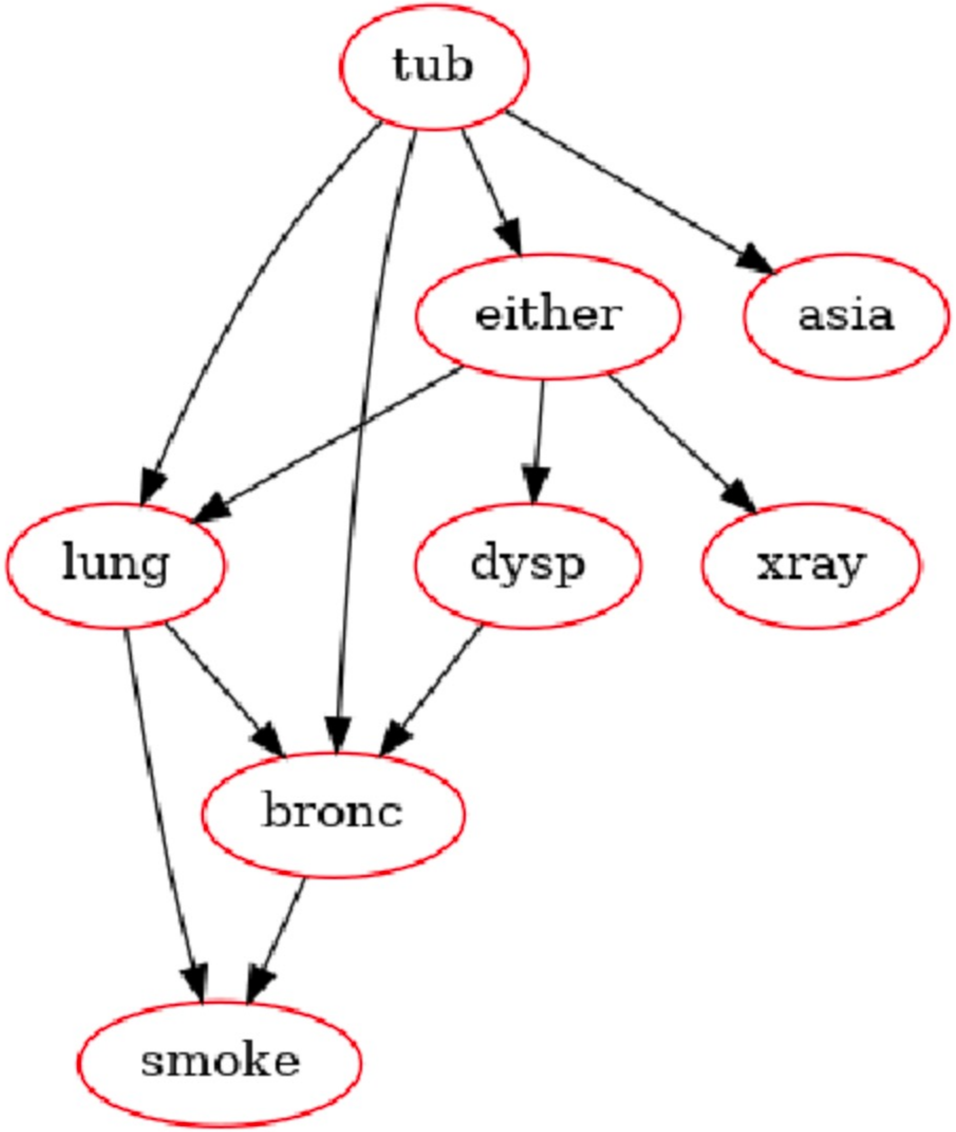
In [98]:
```
X_test_copy
```

Out[98]:

	smoke	lung	bronc	asia	tub	either	xray	dysp
75721	None	1	1	1	1	1	1	1
80184	None	1	0	1	1	1	1	1
19864	None	1	1	1	1	1	1	1
76699	None	1	0	1	1	1	1	0
92991	None	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...
97545	None	1	0	1	1	1	1	0
15490	None	1	1	1	1	1	1	1
62384	None	1	1	1	1	1	1	1
54594	None	1	0	1	1	1	1	1
64709	None	1	0	1	1	1	1	0

33000 rows × 8 columns

- Get random samples from Asia network, then re-feed the samples to generate graph estimate
- Run prediction again on “smoke”
- 49% accuracy
- Takeaway: generated networks and conditional probabilities from samples may generate completely different networks



Generating Bayesian Network = NP-Hard

PGMPY

Exhaustive Search

class `pgmpy.estimators.ExhaustiveSearch.`**`ExhaustiveSearch`**(*data*, *scoring_method=None*, ***kwargs*)

`all_dags`(*nodes=None*) [source][source]

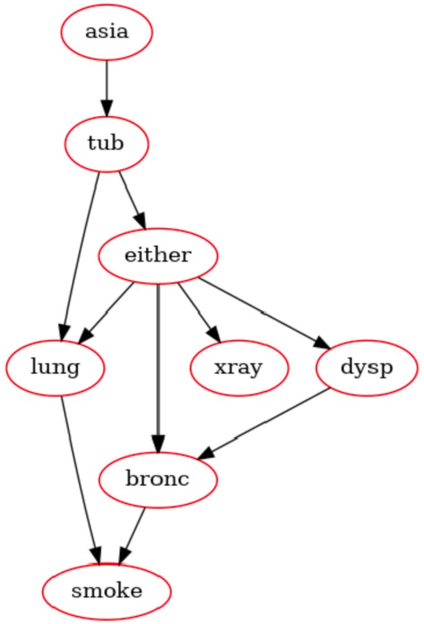
Computes all possible directed acyclic graphs with a given set of nodes, sparse ones first. `2**(n*(n-1))` graphs need to be searched, given n nodes, so this is likely not feasible for n>6. This is a generator.

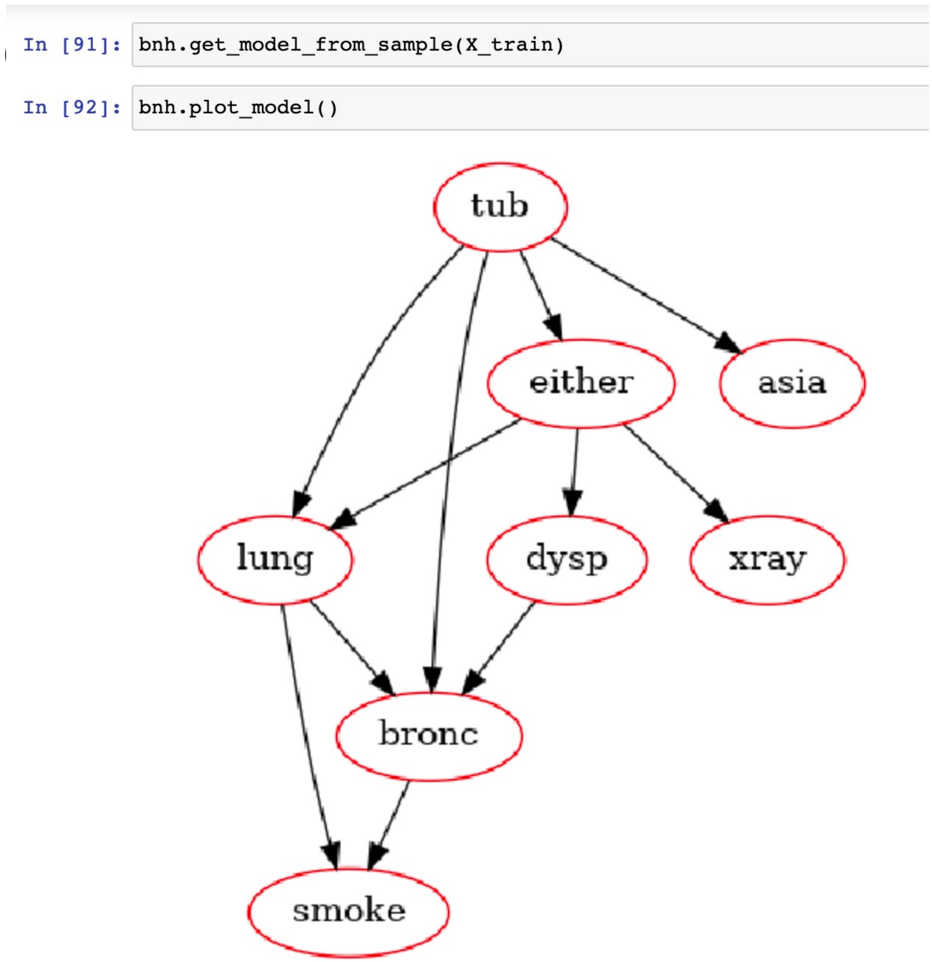
Pomegranate

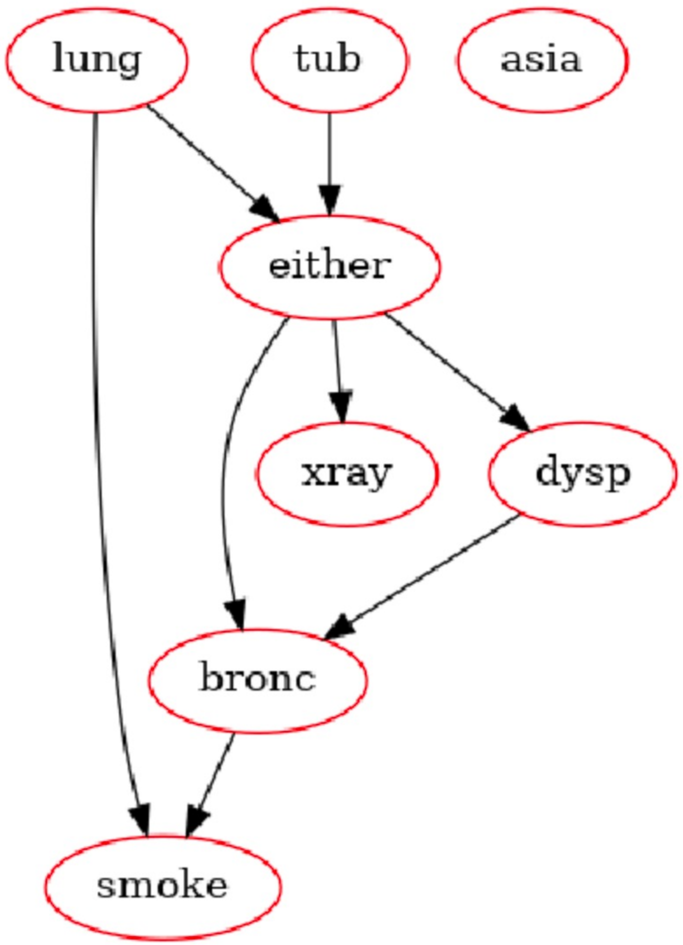
However, one can also initialize a Bayesian network based completely on data. As mentioned before, the exact version of this algorithm takes exponential time with the number of variables and typically can't be done on more than ~25 variables. This is because there are a super-exponential number of directed acyclic graphs that one could define over a set of variables, but fortunately one can use dynamic programming in order to reduce this complexity down to "simply exponential." The implementation of the exact algorithm actually goes further than the original dynamic

Generate Models with Pomegranate (greedy)

- Results can highly dep
  on samples







Generate Acyclic Permutations

- For each node, randomly assign it to level 1, 2, …, K
- Randomly pick two nodes
  - One node from level k
  - Other node from level k+1
  - Add a directed edge from first to second node
  - Edges cannot skip levels or connect nodes in the same level
- This leveling system prevents generating networks with cycles
- Use PGMPY's K2Score to quantify network fit

Bayesian scoring functions

- Compute the posterior probability distribution, starting from a prior probability distribution on the possible networks, conditioned to data T, that is, $P(B|T)$.
- The best network is the one that maximizes the posterior probability.
- Since the term $P(T)$ is the same for all possible networks, in practice, for comparative purposes, computing $P(B, T)$ is sufficient.
- As it is easier to work in the logarithmic space, the scoring functions use the value $\log(P(B, T))$ instead of $P(B, T)$.

http://www.lx.it.pt/~asmc/pub/talks/09-TA/ta_pres.pdf

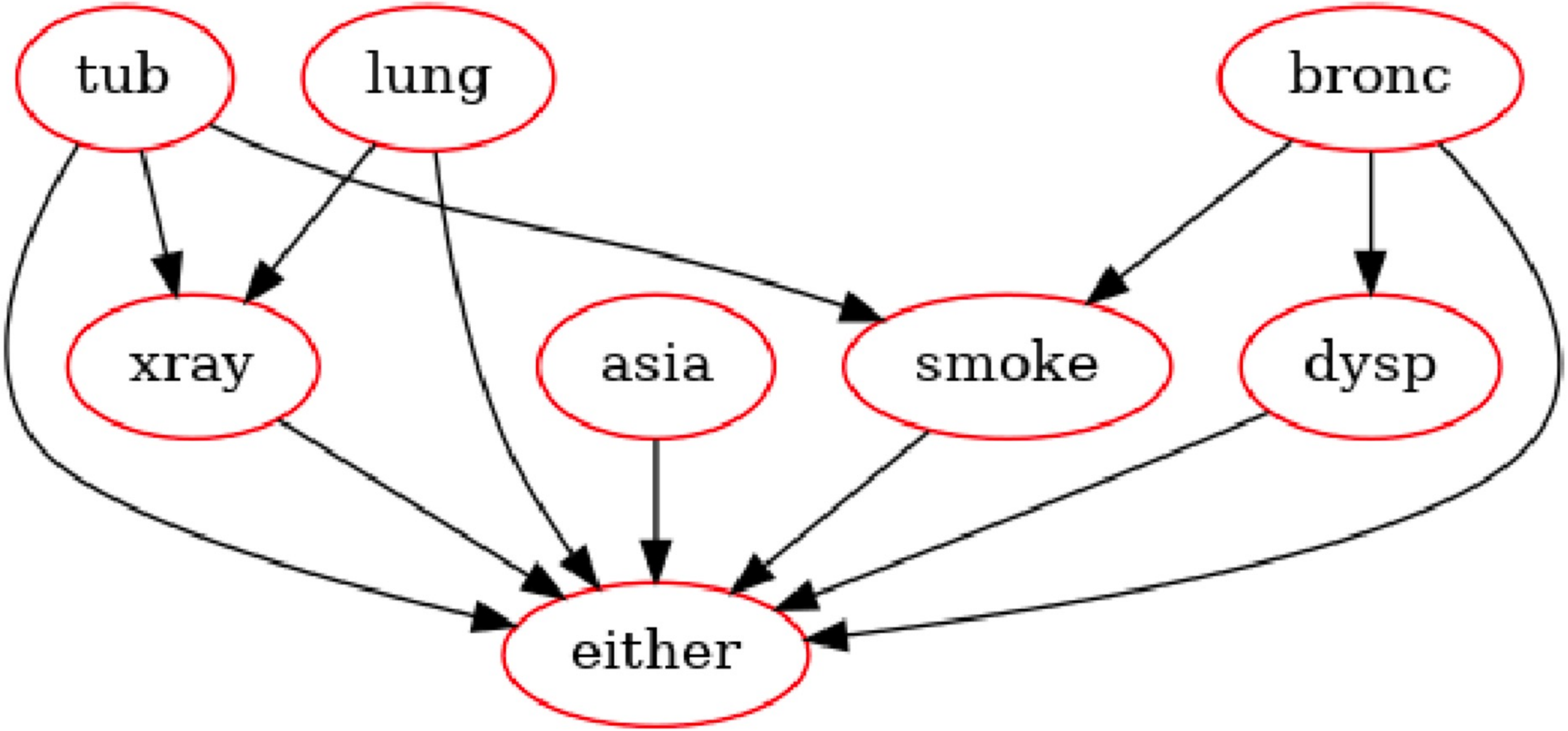
K2 scoring function

Cooper and Herskovits (1992) proposed a particular case of the BD score, called the ***K2* score**,

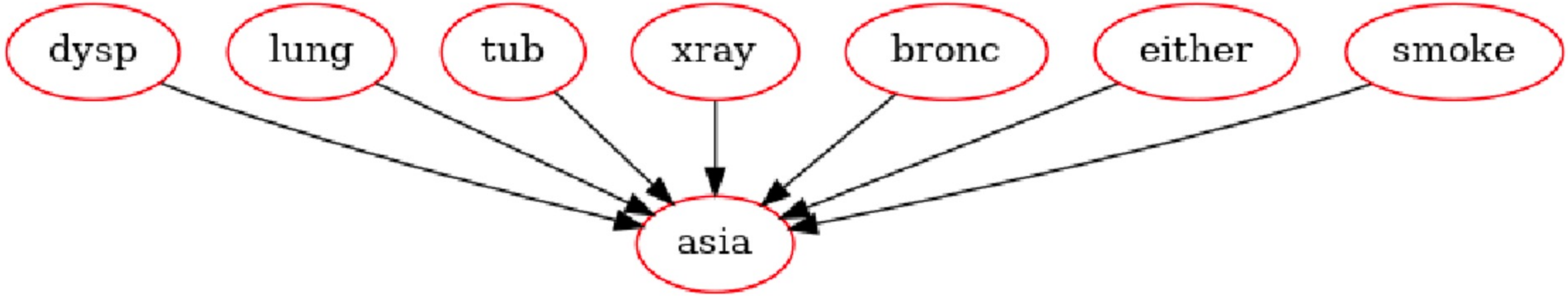
$$\text{K2}(B,T) = \log(P(B)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\log\left(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\right) + \sum_{k=1}^{r_i}\log(N_{ijk}!)\right),$$

with the uninformative assignment $N'_{ijk} = 1$ (corresponding to zero pseudo-counts).

Example where max_level = 3



Network with best score



-29380.702304021157

How to handle larger datasets with many columns/nodes and cardinality?

```
In [2]: csv_file = '/datadrive/masa/bayesian/hidden_rel_new_1.csv'
        df = pd.read_csv(csv_file)
        df
```

Out[2]:

	date	source	target	source_in_degree	source_out_degree	source_all_degree	target_in_degree	target_out_degree	target_all_degree	source_closeness
0	2018/8/30	39	28	6.0	2.0	8.0	6.0	5.0	11.0	1.000000
1	2018/8/30	28	39	6.0	5.0	11.0	6.0	2.0	8.0	2.950000
2	2018/8/30	45	28	0.0	4.0	4.0	6.0	5.0	11.0	2.083333
3	2018/8/30	28	45	6.0	5.0	11.0	0.0	4.0	4.0	2.950000
4	2018/8/30	45	39	0.0	4.0	4.0	6.0	2.0	8.0	2.083333
...	...	...	...	...	...	...	...	...	...	...
276	2018/8/30	25	33	4.0	2.0	6.0	4.0	2.0	6.0	2.616667
277	2018/8/30	33	25	4.0	2.0	6.0	4.0	2.0	6.0	1.500000
278	2018/8/30	23	49	0.0	2.0	2.0	4.0	0.0	4.0	1.000000
279	2018/8/30	38	49	0.0	2.0	2.0	4.0	0.0	4.0	1.000000
280	2018/8/30	23	40	0.0	1.0	1.0	1.0	0.0	1.0	1.000000

281 rows × 40 columns

Use histogram binning to reduce the cardinality

- Instead of having too many cardinality, reduce it using histogram (fixed # bins or percentage).
- Also reduces chance of having value only appear once or twice

```
In [5]: simp = Simplify(df_valid)
```

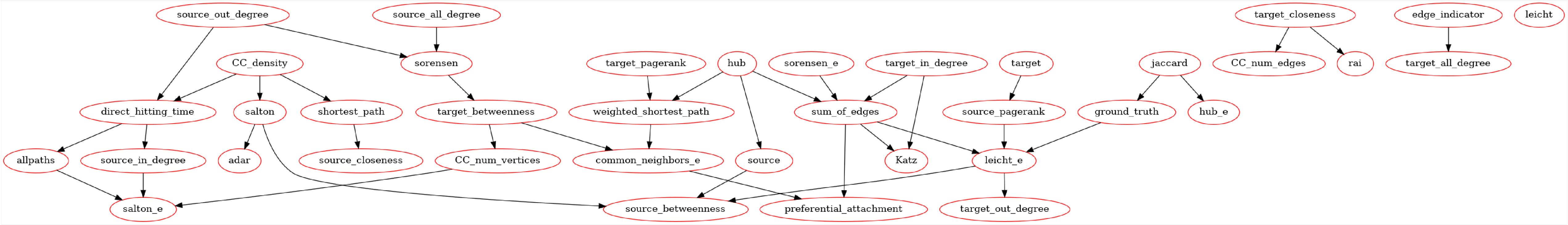
```
In [6]: simp.df_hist
```

Out[6]:

	preferential_attachment	direct_hitting_time	CC_num_vertices	leicht	source	allpaths	edge_indicator	source_all_degree	adar	salton_e	...	target_between
0	7.2	0.231071	6.4	0.0	30.4	50.0	1.0	5.8	-0.5	0.0	...	0.00
1	28.8	0.231071	6.4	0.0	20.6	50.0	1.0	10.6	-0.5	0.0	...	0.00
2	21.6	0.000000	6.4	0.0	40.2	50.0	0.0	3.4	-0.5	1.2	...	0.00
3	0.0	0.231071	6.4	0.0	20.6	50.0	0.0	10.6	-0.5	0.0	...	0.00
4	21.6	0.000000	6.4	0.0	40.2	50.0	1.0	3.4	-0.5	0.0	...	0.00
...	...	...	...	...	...	...	...	...	...	...	...	
276	7.2	0.231071	8.6	0.0	20.6	50.0	1.0	5.8	-0.5	0.6	...	0.00
277	7.2	0.462143	8.6	0.0	30.4	50.0	1.0	5.8	-0.5	0.6	...	0.00
278	7.2	0.462143	2.0	0.0	20.6	0.0	1.0	1.0	-0.5	0.0	...	0.00
279	7.2	0.924286	2.0	0.0	30.4	0.0	1.0	1.0	-0.5	0.6	...	0.00
280	0.0	0.000000	2.0	0.0	20.6	0.0	1.0	1.0	-0.5	0.0	...	0.00

281 rows × 38 columns

Example with max_level = 5

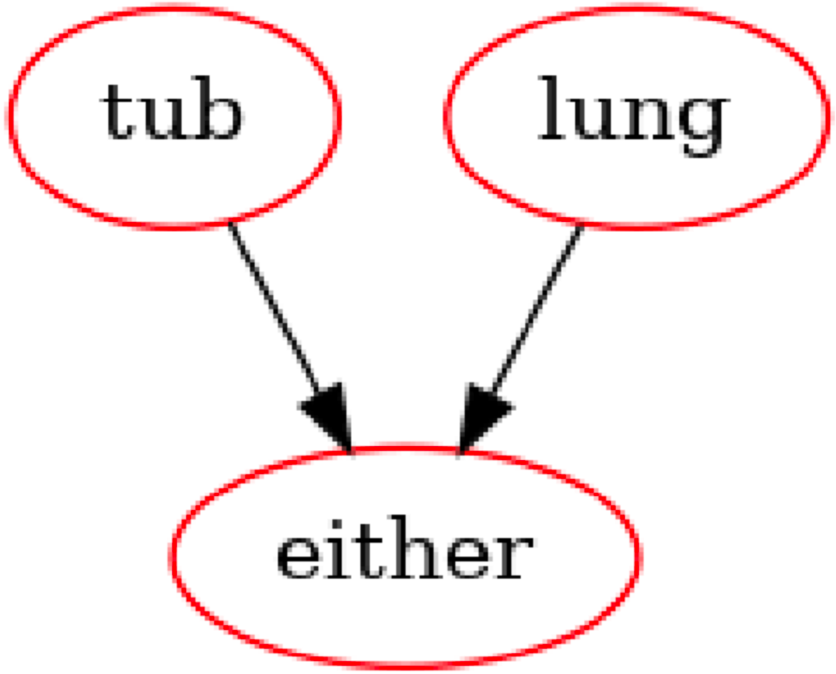


Permutation + Pruning Algorithm

- Have a loop where we choose 1, 2, … tuple of nodes and run through each permutation of node-edge connections with the tuples
- After every loop, look through the permutations that are generated and pick the ones with highest scores
  - Scores are calculated by doing prediction on the leaf nodes (nodes without any children)

Networks with high scores

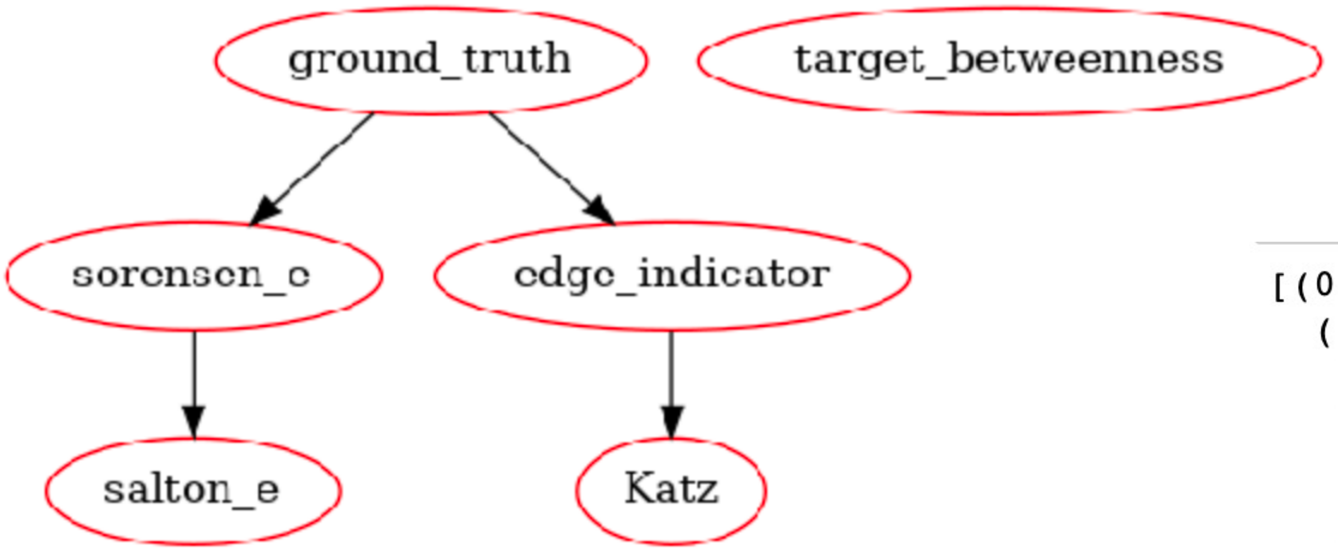
```
',
  'property': {'category': 'Classification', 'trainer': None},
  'model_name': 'my_bayes'},
 'model_metrics': {0: (0.996969696969697, ('tub', 'asia', 'either', 'lung')),
  1: (1.0, ('lung', 'tub', 'either')),
  2: (0.9924242424242424, ('lung', 'asia', 'either')),
  3: (0.9757575757575757, ('lung', 'tub', 'asia')),
  4: (0.9681818181818181, ('tub', 'asia', 'either')),
  5: (0.9681818181818181, ('xray', 'asia', 'either')),
  6: (0.9636363636363636, ('lung', 'xray', 'asia')),
  7: (0.9424242424242424, ('lung', 'tub', 'xray')),
  8: (0.9424242424242424, ('lung', 'xray', 'either')),
  9: (0.9424242424242424, ('tub', 'xray', 'either')),
  10: (0.9393939393939394, ('tub', 'xray', 'asia')),
  11: (0.9924242424242424, ('asia', 'tub')),
  12: (0.990909090909091, ('lung', 'either')),
  13: (0.9681818181818181, ('lung', 'asia')),
  14: (0.9666666666666667, ('lung', 'tub')),
  15: (0.9636363636363636, ('asia', 'either')),
  16: (0.9424242424242424, ('tub', 'either')),
  17: (0.9424242424242424, ('either', 'xray')),
  18: (0.9348484848484848, ('asia', 'xray')),
  19: (0.9333333333333333, ('lung', 'xray'))},
 'log_path': '/datadrive/masa/ardi/machine_learning/client/ml_compute/user_outpu
```



Network with high score

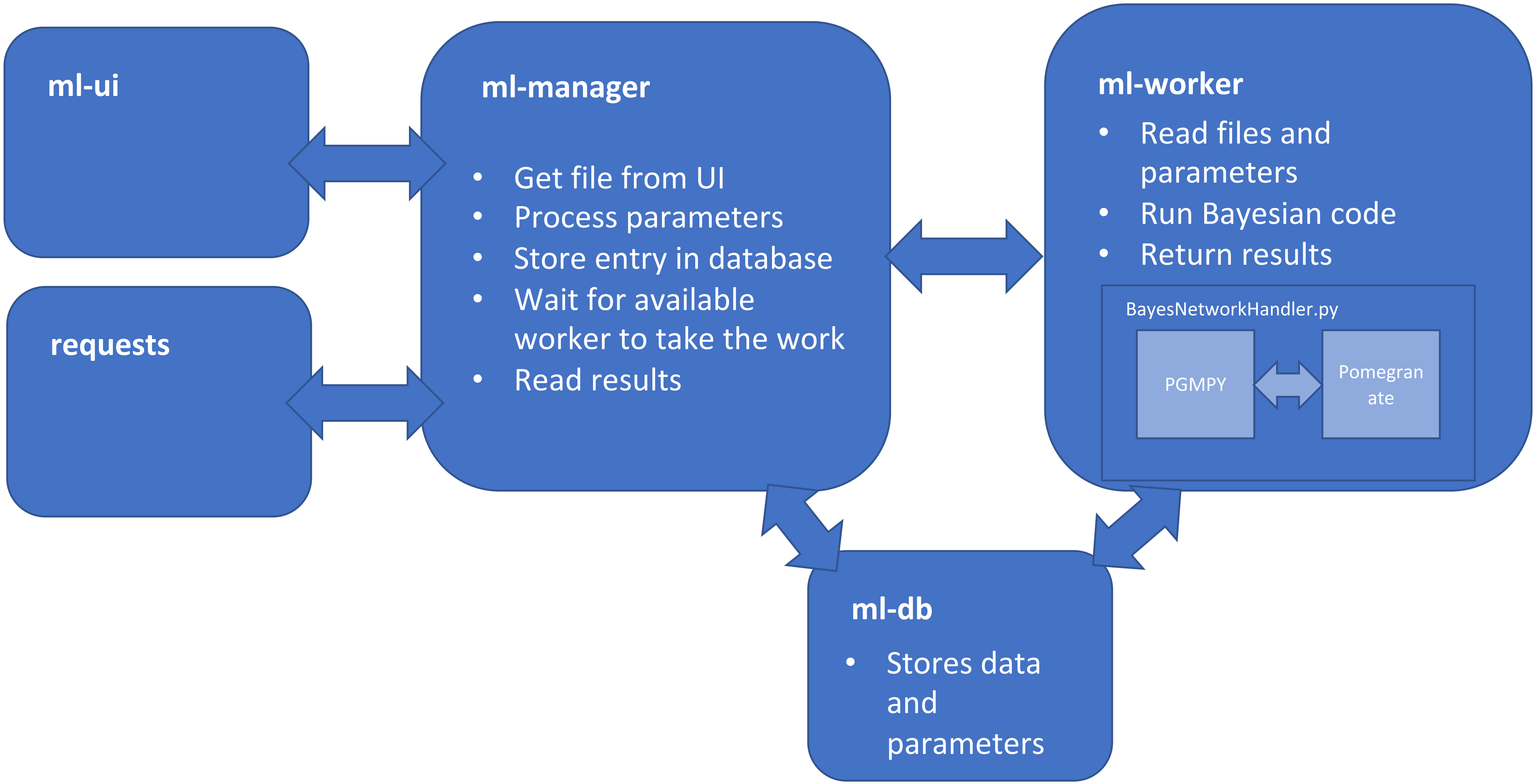
```
In [150]: bayes_analysis.bayes_network_handler.get_model_from_sample(X_train[list(score_dict[6][0][1])])
          bayes_analysis.bayes_network_handler.plot_model()
```

```
[DEBUG] PngImagePlugin.call().146       STREAM b'IHDR' 16 13
[DEBUG] PngImagePlugin.call().146       STREAM b'bKGD' 41 6
[DEBUG] PngImagePlugin._open().592      b'bKGD' 41 6 (unknown)
[DEBUG] PngImagePlugin.call().146       STREAM b'IDAT' 59 8192
```




```
[(0.6103286384976526,
  ('sorensen_e',
   'ground_truth',
   'edge_indicator',
   'Katz',
   'target_betweenness',
   'salton_e'))]
```

Graphen Ardi Bayesian Networks



Acknowledgements

- Some of the materials are based on work by the following:
- Dr.Cheng, Dr. Wong, Dr. Hamo, Dr. Silberstein, Dr. Huang, Mr. Chang-Ogimoto, etc…