1 Bayesian Networks

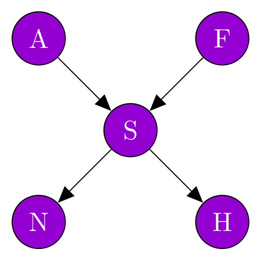
Bayesian Networks are a simple graphical notation for **conditional independence** assertions, hence for **compact specifications of full joint distributions**.

A Bayesian Network is **directed, acyclic graph** with

- **Nodes:** One node per variable
- **Edges:** A directed edge from node N_i to node N_j indicates that the corresponding variable X_i has a direct influence on X_j

Set of random variables $\{X_1, \ldots, X_n\}$

Directed Acyclic Graph (DAG)



Conditional Probability Distribution (CPD)

- Each randome variable X_i in the network is associated with a CPD given its parents $(Pa(X_i))$
- $\boxed{P(X_i|Pa(X_i))}$
- Each variable is probabilistically dependent on its parents

Joint Distribution:

$$\boxed{P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i))}$$

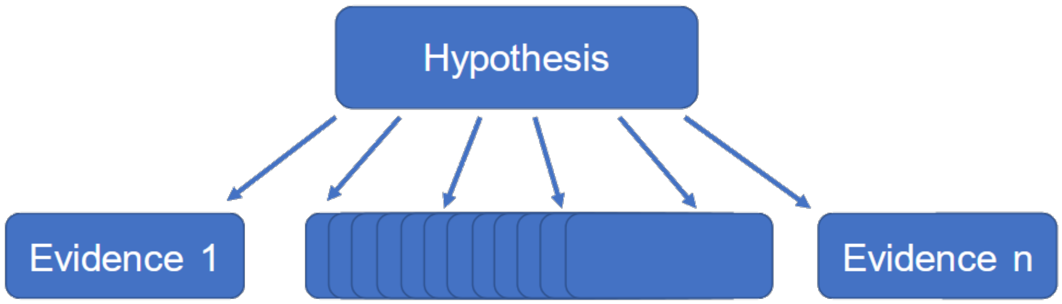
Local Markov Assumption:
Each random variable X_i is conditionally independent of its non-descendants, given its parents.

$$\boxed{X_i \perp \text{nonDescendants} | Pa(X_i)}$$

1.1 Naïve Bayes

A **naïve Bayes** model assumes that all effects are independent given the cause:

$$P(\text{hypothesis}, \text{evidence}_1, \ldots, \text{evidence}_n) = P(\text{hypothesis}) \cdot \prod_{i=1}^{n} P(\text{evidence}_i | \text{hypothesis})$$



1.2 Inference in Bayesian Networks

Query P(X|e)

Definition of conditional probability: $P(X|e) = \dfrac{P(X,e)}{P(e)}$

Up to normalization: $P(X|e) \propto P(X,e)$

Can be rewritten as:

$$P(Y) = \underbrace{\sum_{X_i \notin Y}}_{\text{Marginalization}} \underbrace{\prod_{i=1}^{n} P(X_i | Pa(X_i))}_{\text{BN Semantics}}$$

1.2.1 Variable Elimination

Given a Bayesian Network and a query $P(X|e)/P(X,e)$.

Instantiate evidence e.

Choose an elimination order over the variables $X_1, \ldots, X_n$.

Initial factors of probability distribution comprised of: $f_1, \ldots, f_n$.

For i = 1 to n, if $X_i \notin \{X, E\}$:

Collect factors $f_1, \ldots, f_k$ that contain X_i.

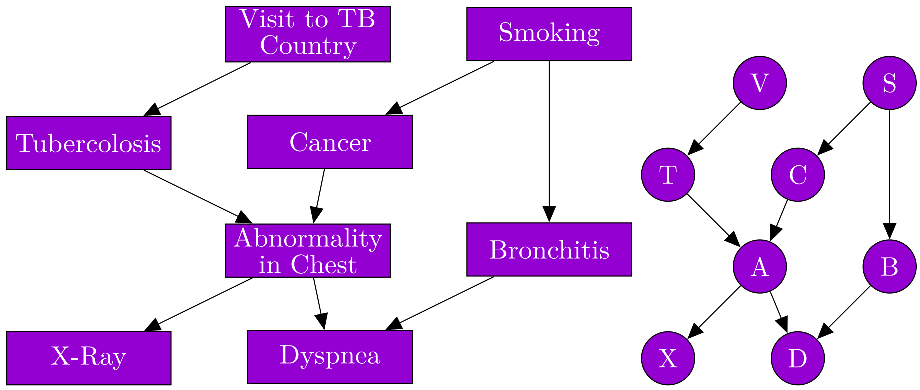
Generate a new factor by eliminating X_i from $f_1, \ldots, f_k$:

$$g = \sum_{X_i} \prod_{j=1}^{k} f_j$$

Remove all factors $f_1, \ldots, f_k$ and add new factor g to the network.

Normalize $P(X,e)$ to obtain $P(X|e)$.

Example



Assume we want to compute P(d), so we need to **eliminate** v,s,t,c,a,b,x.

The **probability distribution** is given as the product of multiple factors:

$$P(v,s,t,c,a,b,x,c) = P(v)P(s)P(t|v)P(c|s)P(b|s)P(a|c,l)P(x|a)P(d|a,b)$$

Lets choose the elimination order: v,s,x,t,c,a,b

From that we get:

$$\begin{array}{c|crl} & & P(v,s,x,t,c,a,b,d) & = P(v)P(s)P(t|v)P(c|s)P(b|s)P(a|c,l)P(x|a)P(d|a,b) \\ v & \Rightarrow & P(s,x,t,c,a,b,d) & = f_v(t)P(s)P(c|s)P(b|s)P(a|c,l)P(x|a)P(d|a,b) \\ s & \Rightarrow & P(x,t,c,a,b,d) & = f_v(t)f_s(b,c)P(a|t,c)P(x|a)P(d|a,b) \\ x & \Rightarrow & P(t,c,a,b,d) & = f_v(t)f_s(b,c)f_x(a)P(a|t,c)P(d|a,b) \\ t & \Rightarrow & P(c,a,b,d) & = f_s(b,c)f_x(a)f_t(a,c)P(d|a,b) \\ c & \Rightarrow & P(a,b,d) & = f_x(a)f_c(a,b)P(d|a,b) \\ a & \Rightarrow & P(b,d) & = f_a(b,d) \\ b & \Rightarrow & P(d) & = f_b(d) \end{array}$$

This unfortunately is not efficient.

Theorem

Inference (even approximate in Bayesion networks is NP-Hard)

1.2.2 Approximate Inference by Stochastic Sampling

Basic Idea:

1. Draw N samples from a sampling distribution S
2. Compute an approximate posterior probability $\hat{P}$
3. Show this converges to the true probability P

Draw samples

Given:

- Random Variable $X|D(X) = \{0,1\}$
- $P(X) = \{0.3, 0.7\}$ (P(X=0) = 0.3, P(X=1) = 0.7)

Sample X = P(X)

- Get a random number $r \in [0, 1]$
- If $r < 0.3$ then $X = 0$
- Else $X = 1$

Can be generalized to any domain size.

Sampling from "Empty Network"

Ergo, **generating samples from a network that has no evidence associated with it.**

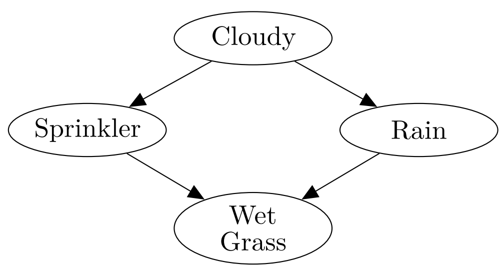
Basic Idea:

- Sample a value for each variable in topological (in respect to dependencies) order
- Using the specified conditional probabilities

```
// belief network specifies joint distribution P(X_1, ..., X_n)
Function prior_sample(belief_network) → event sampled from belief network:
    x = event with n elements;
    For i = 1 to n do
        x_i = random sample from P(X_i|Pa(X_i)) given the values of Pa(X_i) in x;
    return x
```

Example:

P(C)
0.50



C	P(S|C)
T	0.10
F	0.50

C	P(R|C)
T	0.80
F	0.20

S	R	P(W|S,R)
T	T	0.99
T	F	0.90
F	T	0.90
F	F	0.01

Bayesian Network for Weather and Wet Grass

Probability Estimation using Sampling

Calculating a probability estimation:

- Sample many points using the algorithm above
- Count how often each possible combination $x_1, \ldots, x_n$ occurs

- Estimate the probability by the observed percentages

$$\hat{P}(x_1, \ldots, x_n) = N_{PS}(x_1, \ldots, x_n)/\text{number of samples}$$

This converges towards the joint probability function.

Markov Chain Monte Carlo (MCMC) Sampling

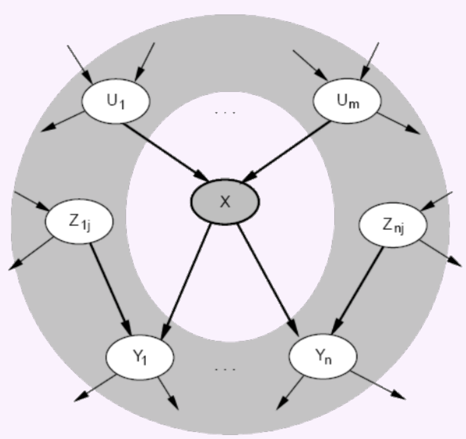
```
Function mcmc_ask(X,e,belief_network, num_samples) → estimate of P(X|e):
    count_X = [] // number of times each X occurs, initially 0 for all
    Z = [non-evidence variables] // list of non-evidence variables
    x = e // current state of the network, initially e
    initialize non-evidence values in x with random values;
    // Gibbs sampling
    For j=1 to num_samples do
        ForEach Z_i ∈ Z do
            x[Z_i] = sample from P(Z_i|markov_blanket(Z_i))
        count_X[x] += 1 // x is the value of X in x
    return normalize(count_X)
```

More samples result in better approximates.

Markov Blanket

A Markov Blanket is a set of variables that are conditionally independent of a variable given all other variables in the network. It consists of **parents (direct causes), children (direct effects) and childrens parents (co-causes)**. Alternatively: A markov blanket includes all variables that **directly influence** or **are influenced** by a variable X. Everything outside of the markov blanket is irrelevant to X. This makes it easier to compute probabilities.

$$P(X|U_1, \ldots, U_m, Y_1, \ldots, Y_n, Z_{1j}, \ldots, Z_{nj}) = P(X|\text{all variables})$$



Gibbs Sampling

Basic Idea:
1. **Initialize** all variables with random values
2. **Iterate through each variable**, updating it based on Markov Blanket
3. **Repeat** until samples converge to the true distribution

Gibbs Sampling utilized Markov Blankets by reducing the number of variables that need to be considered at each step.

Example:
Estimate $P(\text{Rain}|\text{Sprinkler} = \text{true}, \text{WetGrass} = \text{True})$

1. Sample Cloudy or Rain given its Markov Blanket, repeat n times
2. Count number of times Rain is true and false in the samples

E.g. sample 100 states and count 31 times Rain and 69 times not Rain.

$$\boxed{P(\text{Rain}|\text{Sprinkler} = \text{true}, \text{WetGrass} = \text{True}) = \texttt{Normalize}< 31, 69 > = < 0.31, 0.69 >}$$

Theorem

Chain approaches stationary distribution:
Long-run fraction of time spent in each sate is exactly proprotional to posterior probability.