Bayesian networks

- Independence
- Bayesian networks
- Markov conditions
- Inference
  - by enumeration
  - rejection sampling
  - Gibbs sampler

Independence

- if P(A=a,B=a) = P(A=a)P(B=b) for all a and b, then we call A and B (marginally) independent.
- if P(A=a,B=a | C=c) = P(A=a|C=c)P(B=b|C=c) for all a and b, then we call A and B conditionally independent given C=c.
- if P(A=a,B=a | C=c) = P(A=a|C=c)P(B=b|C=c) for all a, b and c, then we call A and B conditionally independent given C.
- $P(A,B)=P(A)P(B)$ implies

$$P(A|B)=\frac{P(A,B)}{P(B)}=\frac{P(A)P(B)}{P(B)}=P(A)$$

Independence saves space

- If A and B are independent given C

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

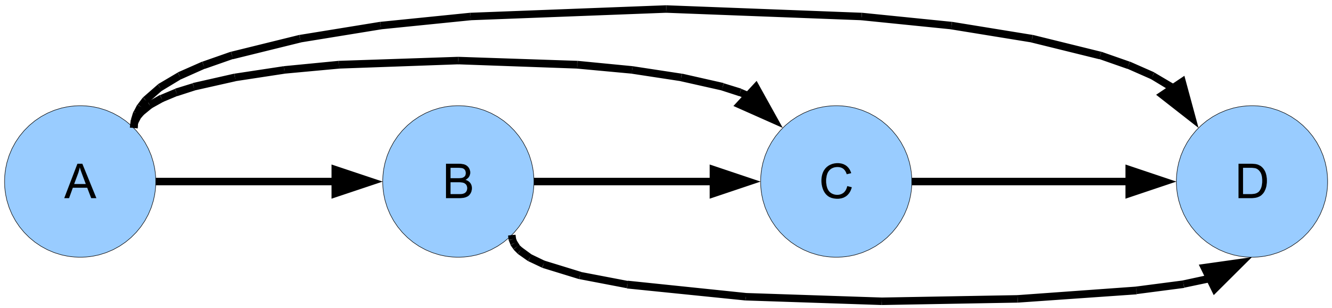
  $= P(C)P(A|C)P(B|A,C)$

  $= P(C)P(A|C)P(B|C)$

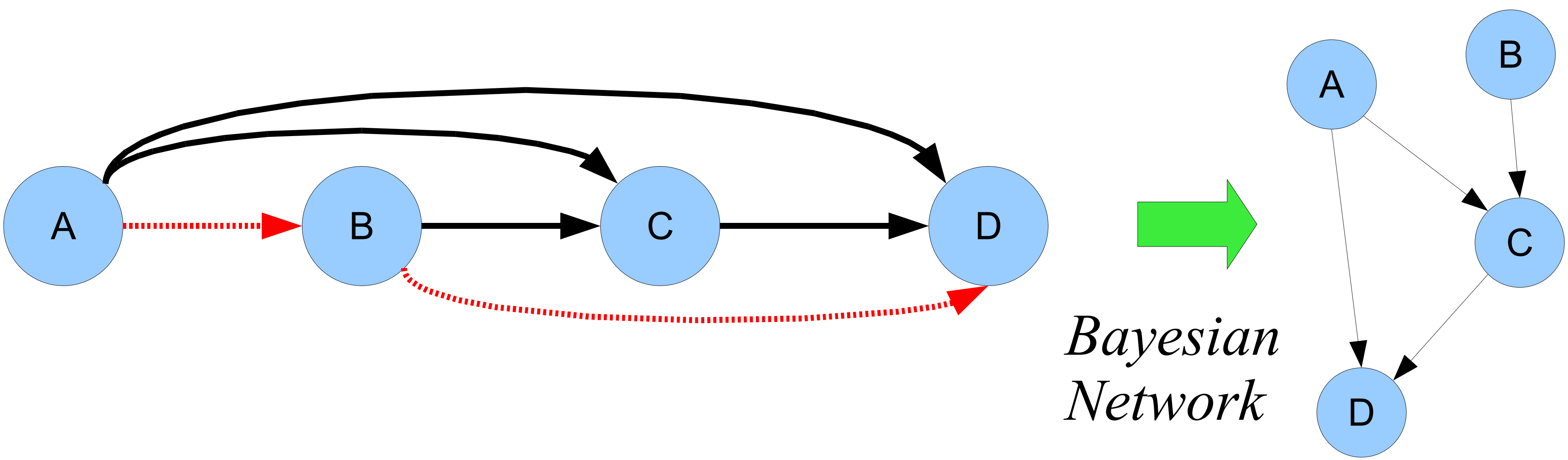
- Instead of having a full joint probability table for $P(A,B,C)$, we can have a table for $P(C)$ and tables $P(A|C=c)$ and $P(B|C=c)$ for each c.
  - Even for binary variables this saves space:
    - $2^3 = 8$ vs. $2 + 2 + 2 = 6$.
  - With many variables and many independences you save a lot.

Chain Rule – Independence - BN

$$Chain\,rule: P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$



$$Independence: P(A,B,C,D) = P(A)P(B)P(C|A,B)P(D|A,C)$$

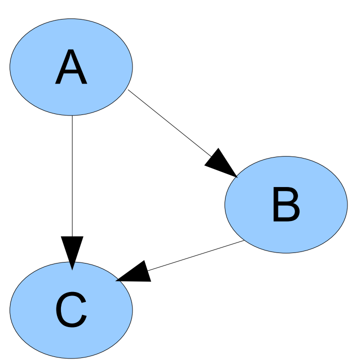


But order matters

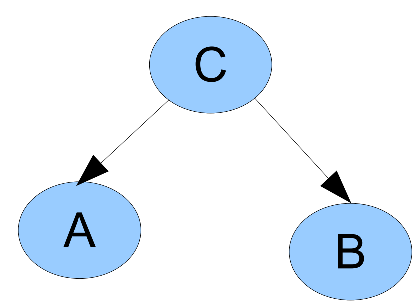
- $P(A,B,C) = P(C,A,B)$
  - $P(A)P(B|A)P(C|A,B) = P(C)P(A|C)P(B|A,C)$
  - And if A and B are conditionally independent given C:

    1. $P(A,B,C) = P(A)P(B|A)P(C|A,B)$
    2. $P(C,A,B) = P(C)P(A|C)P(B|C)$

1.



2.



With the same independence assumptions, some orders yield simpler networks.

Bayes net as a factorization

- Bayesian network structure forms a directed acyclic graph (DAG).
- If we have a DAG G, we denote the parents of the node (variable) X_i with $Pa_G(x_i)$ and a value configuration of $Pa_G(x_i)$ with $pa_G(x_i)$:

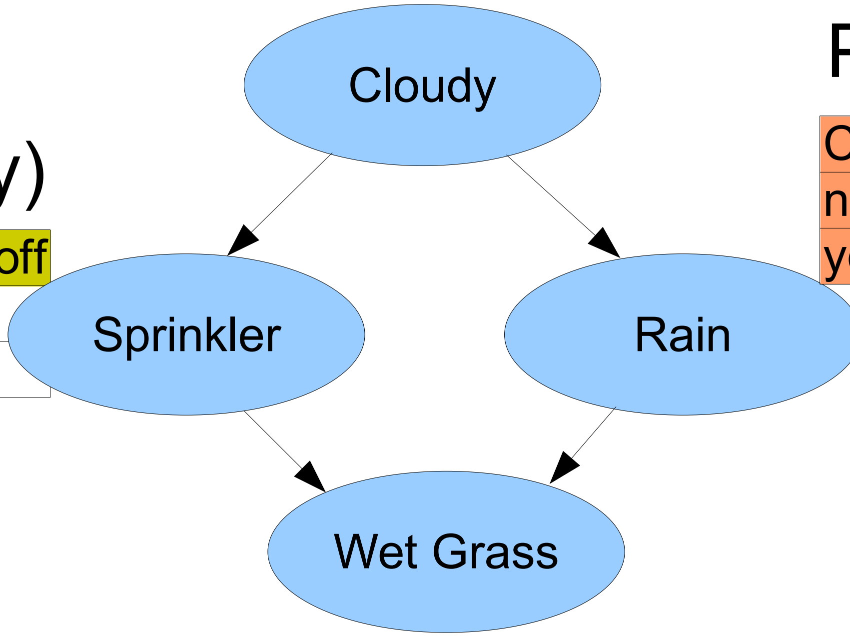
$$P(x_{1,} x_{2,} \ldots, x_n | G) = \prod_{i=1}^{n} P(x_i | pa_G(x_i)),$$

- where $P(x_i|pa_G(x_i))$ are called local probabilities.
  - Local probabilities are stored in conditional probability tables CPTs.

A Bayesian network

P(Cloudy)

	Cloudy=no	Cloudy=yes
	0.5	0.5



P(Sprinkler | Cloudy)

Cloudy	Sprinkler=on	Sprinkler=off
no	0.5	0.5
yes	0.9	0.1

P(Rain | Cloudy)

Cloudy	Rain=yes	Rain=no
no	0.2	0.8
yes	0.8	0.2

P(WetGrass | Sprinkler, Rain)

Sprinkler	Rain	WetGrass=yes	WetGrass=no
on	no	0.90	0.10
on	yes	0.99	0.01
off	no	0.01	0.99
off	yes	0.90	0.10

Causal order recommended

- Causes first, then effects.
- Since causes render direct consequences independent yielding smaller CPTs
- Causal CPTs are easier to assess by human experts
- Smaller CPT:s are easier to estimate reliably from a finite set of observations (data)
- Causal networks can be used to make causal inferences too.

Markov conditions

- Local (parental) Markov condition
  - X is independent of its ancestors given its parents.
- Global Markov Condition
  - X is independent of any set of other variables given its parents, children and parents of its children (Markov blanket)
- D-separation
  - X and Y are dependent given Z, if there is an unblocked path without colliders between X and Y.
  - or if each collider or some descendant of each collider is in Z.

Inference in Bayesian networks

- Given a Bayesian network B (i.e., DAG and CPTs) , calculate P(**X**|**e**) where **X** is a set of query variables and **e** is an instantiaton of observed variables **E** (**X** and **E** separate).
- There is always the way through marginals:
  - normalize $P(\mathbf{x},\mathbf{e}) = \Sigma_{\mathbf{y} \in dom(\mathbf{Y})} P(\mathbf{x},\mathbf{y},\mathbf{e})$, where dom(**Y**), is a set of all possible instantiations of the unobserved non-query variables **Y**.
- There are much smarter algorithms too, but in general the problem is NP hard.

Approximate inference in Bayesian networks

- How to estimate how probably it rains next day, if the previous night temperature is above the month average.
  - count rainy and non rainy days after warm nights (and count relative frequencies).
- Rejection sampling for P(**X**|**e**) :
  1. Generate random vectors ($\mathbf{x}_r$,$\mathbf{e}_r$,$\mathbf{y}_r$).
  2. Discard those those that do not match **e**.
  3. Count frequencies of different $\mathbf{x}_r$ and normalize.

How to generate random vectors from a Bayesian network

	Cloudy=no	Cloudy=yes
	0.5	0.5

Cloudy	Sprinkler=on	Sprinkler=off
no	0.5	0.5
yes	0.9	0.1

Cloudy	Rain=yes	Rain=no
no	0.2	0.8
yes	0.8	0.2

Sprinkler	Rain	WetGrass=yes	WetGrass=no
on	no	0.90	0.10
on	yes	0.99	0.01
off	no	0.01	0.99
off	yes	0.90	0.10

- Sample parents first
  - P(C)
    - (0.5, 0,5) → yes
  - P(S|C=yes)
    - (0.9, 0.1) → on
  - P(R | C=yes)
    - (0.8, 0.2) → no
  - P(W | S=on, R=no)
    - (0.9, 0.1) → yes
- P(C,S,R,W) = P(yes,on,no,yes) = 0.5 x 0.9 x 0.2 x 0.9 = 0.081

Rejection sampling, bad news

- Good news first:
  - super easy to implement
- Bad news:
  - if evidence **e** is improbable, generated random vectors seldom conform with **e**, thus it takes a long time before we get a good estimate P(**X**|**e**).
  - With long **E**, all **e** are improbable.
- So called likelihood weighting can alleviate the problem a little bit, but not enough.

Gibbs sampling

- Given a Bayesian network for n variables **X**∪**E**∪**Y**, calculate P(**X**|**e**) as follows:

```
N = (associative) array of zeros
Generate random vector x,y.
While True:
    for V in X,Y:
        generate v from P(V | MarkovBlanket(V))
        replace v in x,y.
        N[x] +=1
        print normalize(N[x])
```

P(X|mb(X))?

$$
\begin{aligned}
&P(X|mb(X))\\
&\quad = P(X|mb(x), Rest)\\
&\quad = \frac{P(X, mb(X), Rest)}{P(mb(X), Rest)}\\
&\quad \propto P(All)\\
&\quad = \prod_{X_i \in \boldsymbol{X}} P(X_i|Pa(X_i))\\
&\quad = P(X|Pa(X)) \prod_{C \in ch(X)} P(C|Pa(C)) \prod_{R \in Rest \cup Pa(V)} P(R|Pa(R))\\
&\quad \propto P(X|Pa(X)) \prod_{C \in ch(X)} P(C|Pa(C))
\end{aligned}
$$

Why does it work

- All decent Markov Chains q have a unique stationary distribution P* that can be estimated by simulation.
- Detailed balance of transition function q and state distribution P* implies stationarity of P*.
- Proposed q, P(V|mb(V)), and P(**X**|**e**) form a detailed balance, thus P(**X**|**e**) is a stationary distribution, so it can be estimated by simulation.

Markov chains stationary distribution

- Defined by transition probabilities between states $q(x \to x')$, where x and x' belong to a set of states X.
- Distribution P* over X is called stationary distribution for the Markov Chain q, if $P^*(x') = \sum_x P^*(x) q(x \to x')$.
- P*(X) can be found out by simulating Markov Chain q starting from the random state x_r.

Markov Chain detailed balance

- Distribution P over X and a state transition distribution q are said to form a detailed balance, if for any states x and x', $P(x)q(x \rightarrow x') = P(x')q(x' \rightarrow x)$, i.e. it is equally probable to witness transition from x to x' as it is to witness transition from x' to x.
- If P and q form a detailed balance, $\sum_x P(x)q(x \rightarrow x') = \sum_x P(x')q(x' \rightarrow x) = P(x')\sum_x q(x' \rightarrow x) = P(x')$, thus P is stationary.

Gibbs sampler as Markov Chain

- Consider $\mathbf{Z}=(\mathbf{X},\mathbf{Y})$ to be states of a Markov chain, and $q((v,\mathbf{z}_{-V}))\rightarrow(v',\mathbf{z}_{-V}))=P(v'|\mathbf{z}_{-V}, \mathbf{e})$, where $\mathbf{Z}_{-V} = \mathbf{Z}\text{-}\{V\}$. Now $P^*(\mathbf{Z})=P(\mathbf{Z}|e)$ and q form a detailed balance, thus P* is a stationary distribution of q and it can be found with the sampling algorithm.
  - $P^*(\mathbf{z})q(\mathbf{z}\rightarrow\mathbf{z'}) = P(\mathbf{z}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})$
    $= P(v,\mathbf{z}_{-V}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})$
    $= P(v|\mathbf{z}_{-V},\mathbf{e})P(\mathbf{z}_{-V}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})$
    $= P(v|\mathbf{z}_{-V},\mathbf{e})P(v', \mathbf{z}_{-V}|\mathbf{e}) = q(\mathbf{z'}\rightarrow\mathbf{z})P^*(\mathbf{z'})$, thus balance.