

Bayesian Models in Machine Learning

Graphical Models and Inference

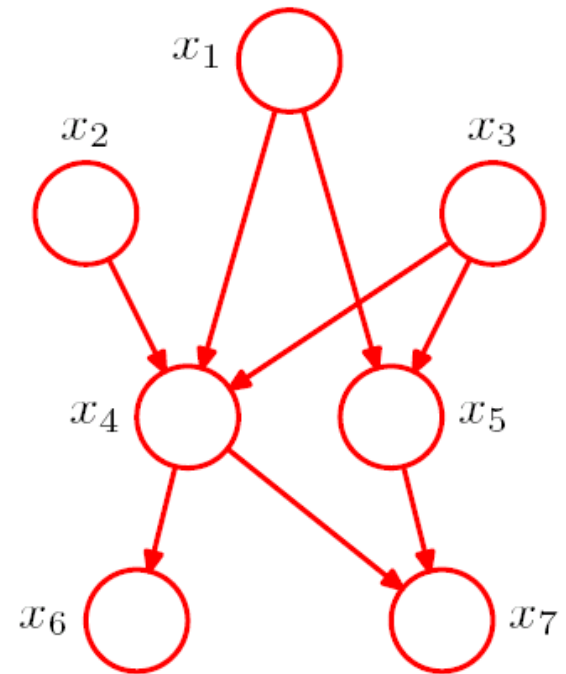
Lukáš Burget



BAYa lectures, October 2023

Bayesian Networks (BN)

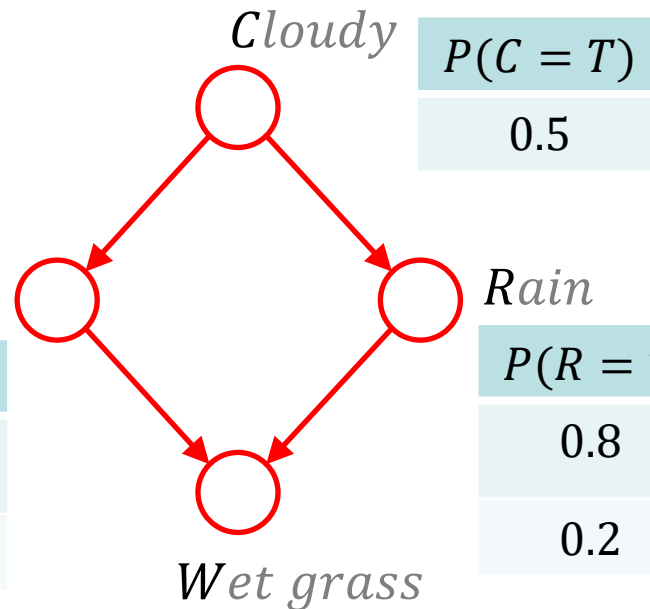
- The graph corresponds to a particular factorization of a joint probability distribution over a set of random variables
- Nodes are **random variables**, but the graph does not say what are the distributions of the variables
- The graph represents a set of distributions that conform to the factorization
- It is recipe for building more complex models out of simpler probability distributions
- **Describes the generative process**
 - To generate a sample from the joint distribution, sample variables for the nodes with no incoming arcs first and then continue sampling variables conditioned on already sampled values.
- **Generally no closed form solutions for inferences in such models** (see later)



$$p(x_1, \dots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

Simple BN example

$$P(C, S, R, W) = P(C)P(S|C)P(R|C)P(W|S, R)$$



$P(C = T)$
0.5

$P(S = T \mid C)$	C
0.1	T
0.5	F

$P(R = T \mid C)$	C
0.8	T
0.2	F

\Rightarrow

$P(R = F \mid C)$	C
0.2	T
0.8	F

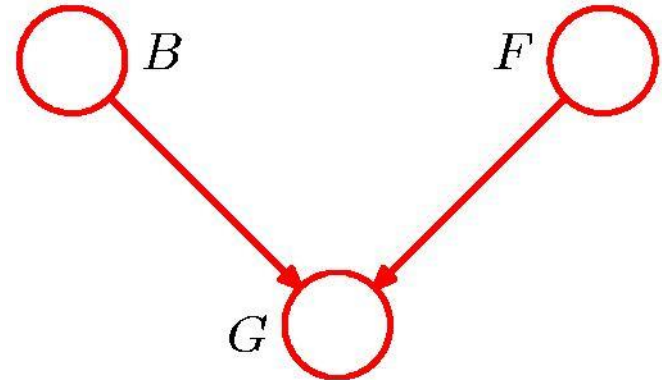
$P(W = T \mid S, R)$	S	R
0.99	T	T
0.9	T	F
0.9	F	T
0.0	F	F

- Simple example with discrete binary random variable
- Distributions can be described by tables with (conditional) probabilities
- More practical examples will come later

Example from: *Russel, Norvig: AI – A modern approach*

Example 2: Am I out of fuel?

$$\begin{aligned} p(G = 1|B = 1, F = 1) &= 0.8 \\ p(G = 1|B = 1, F = 0) &= 0.2 \\ p(G = 1|B = 0, F = 1) &= 0.2 \\ p(G = 1|B = 0, F = 0) &= 0.1 \end{aligned}$$



$$p(B = 1) = 0.9$$

$$p(F = 1) = 0.9$$

and hence

$$p(F = 0) = 0.1$$

B - Battery (0=flat, 1=fully charged)

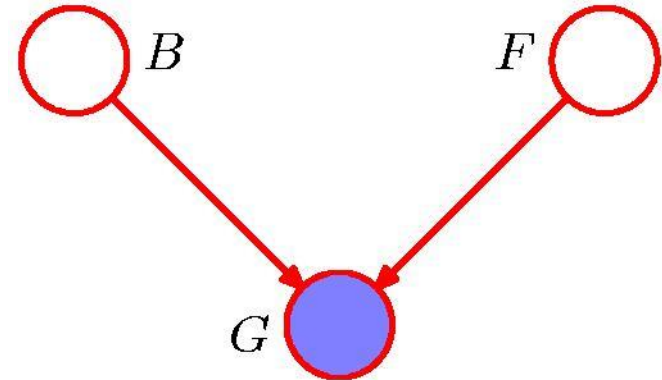
F - Fuel Tank (0=empty, 1=full)

G - Fuel Gauge Reading (0=empty, 1=full)

Note that $p(B)$ and $p(F)$ are independent $\Rightarrow p(B, F) = p(B)p(F)$

Example 2: Am I out of fuel?

Let us make some simple inference with this probabilistic model:



Bayes rule

$$p(F = 0 | G = 0) = \frac{p(G = 0 | F = 0)p(F = 0)}{p(G = 0)} \simeq 0.257$$

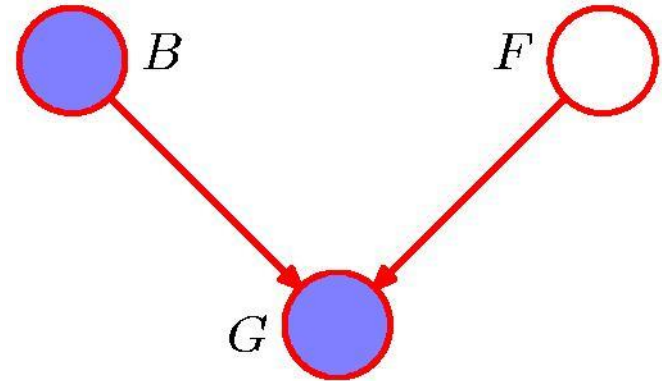
where

$$p(G | F) = \sum_B p(G, B | F) = \sum_B p(G | B, F)p(B)$$

Probability of an empty tank increased by observing $G = 0$.

Example 2: Am I out of fuel?

Probability of an empty tank reduced by observing $B = 0$. This referred to as “explaining away”.



$$p(F = 0|G = 0, B = 0) = \frac{p(G = 0|B = 0, F = 0)p(F = 0)}{\sum_{F \in \{0,1\}} p(G = 0|B = 0, F)p(F)}$$
$$\simeq 0.111$$

F is not **conditionally independent** of B given G

$$P(F|G, B) \neq P(F|G)$$

$$P(F, B|G) \neq P(F|G)P(B|G)$$

Conditional independence

- a is statistically independent of $b \Rightarrow$

$$P(a, b) = P(a)P(b)$$

- a is (conditionally) independent of b given $c \Rightarrow$

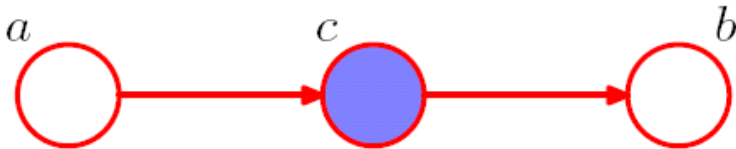
$$P(a|b, c) = P(a|c)$$

or equivalently

$$\begin{aligned} P(a, b|c) &= P(a|b, c)P(b|c) \\ &= P(a|c)P(b|c) \end{aligned}$$

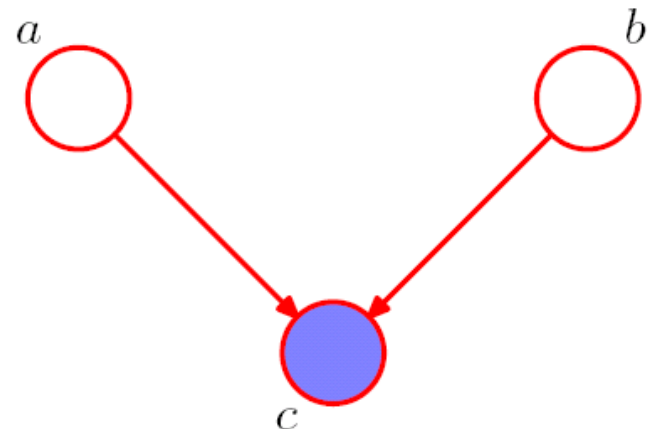
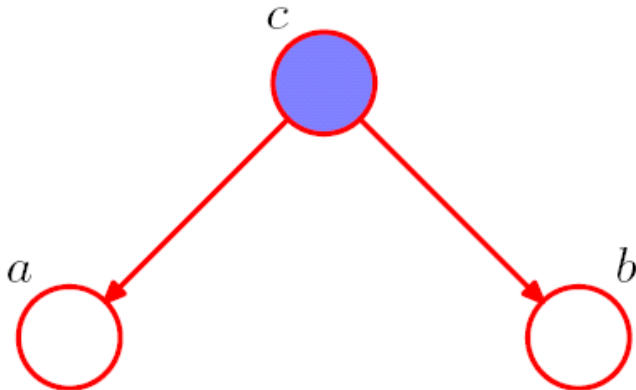
Conditional independence

- Bayesian Networks allow us to see conditional independence properties.
- Blue nodes corresponds to **observed random variables** and empty nodes to **latent (or hidden) random variables**



$$P(a, b) \neq P(a)P(b)$$
$$P(a, b|c) = P(a|c)P(b|c)$$

But the opposite is true for:

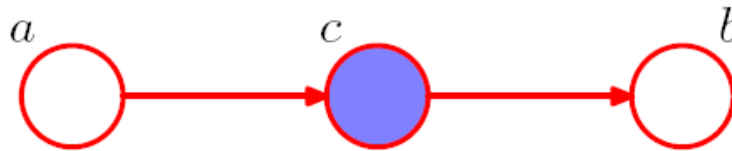


Naturally also

$$P(a|b, c) = P(a|c)$$
$$P(b|a, c) = P(b|c)$$

Conditional independence - proof

For example, Bayesian Network:



corresponds to factorization:

$$P(a, b, c) = P(b|c)P(c|a)P(a)$$

Using product rule

$$P(a, b, c) = P(a, b|c)P(c)$$

$P(a|c)$ by
Bayes rule

therefore

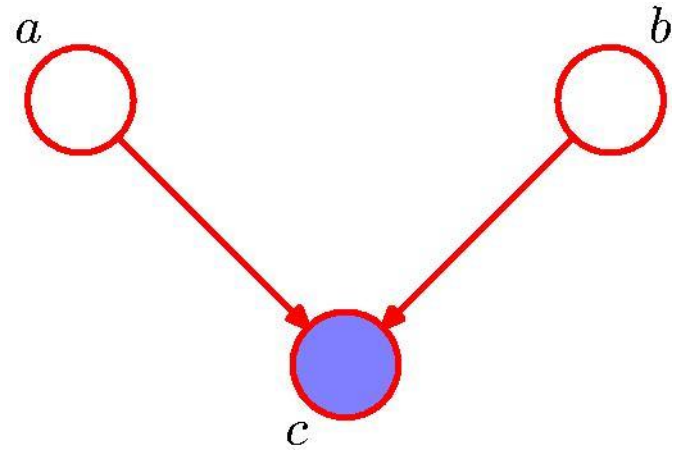
$$\begin{aligned} P(a, b|c) &= \frac{P(a, b, c)}{P(c)} = \frac{P(c|a)P(b|c)P(a)}{P(c)} = \frac{P(c|a)P(a)}{P(c)} P(b|c) \\ &= P(a|c)P(b|c) \end{aligned}$$

But

$$P(a, b) = P(a) \sum_c P(c|a)P(b|c) = P(a)P(b|a) \neq P(a)P(b)$$

“Explaining away” effect

But the opposite is true in this case:



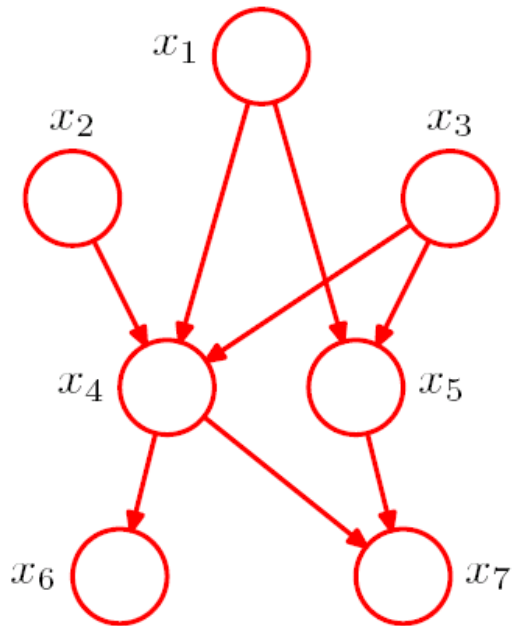
$$P(a, b, c) = P(a)P(b)P(c|a, b)$$

$$P(a, b) = \sum_c P(a, b, c) = P(a)P(b) \underbrace{\sum_c P(c|a, b)}_1 = P(a)P(b)$$

but

$$P(a, b|c) = \frac{P(a, b, c)}{P(c)} = \frac{P(b)P(a)P(c|a, b)}{P(c)} \neq P(a|c)P(b|c)$$

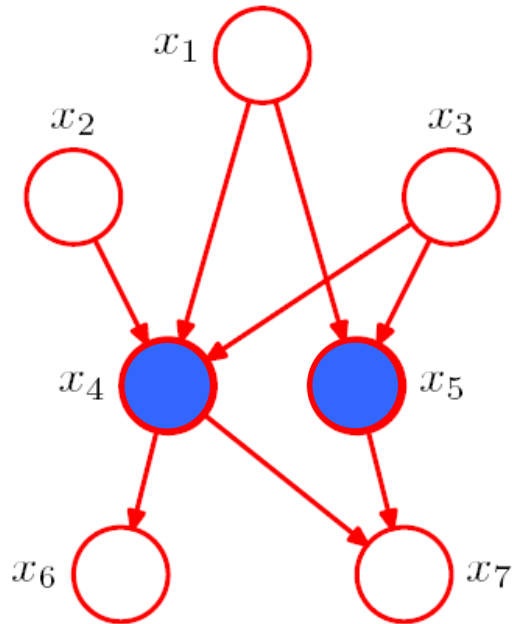
d-separation



- Let A , B and C be disjoint subsets of nodes
- Any path from a node in A to a node in B is *blocked* by C if there are arrows on the path meeting
 - head-to-tail or tail-to-tail at a node from C , or
 - head-to-head at a node **not** from C that also does not have any descendants from C
- If all the path are blocked this way, A and B are said to be d-separated by C which implies conditional independence:

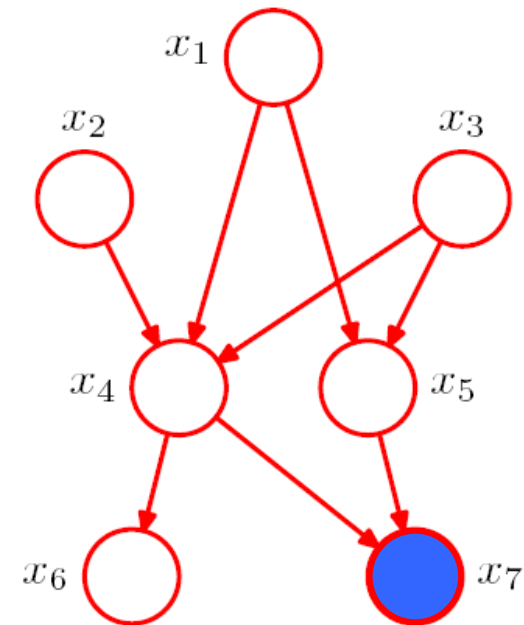
$$p(A, B|C) = p(A|C) p(B|C)$$

d-separation: Examples



All path between nodes x_1, x_7 are blocked by observed nodes $x_4, x_5 \Rightarrow$

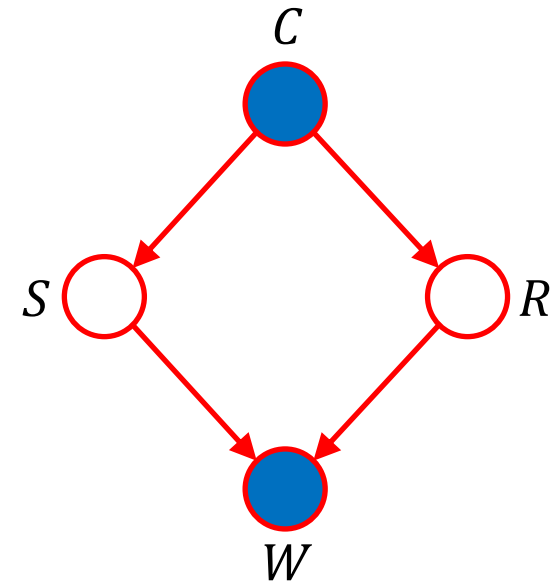
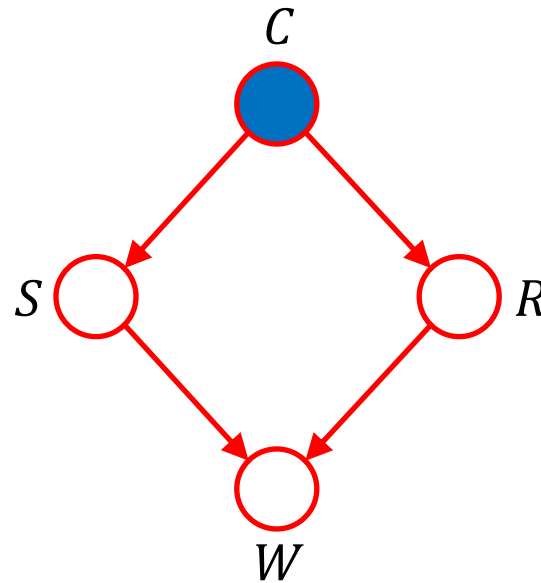
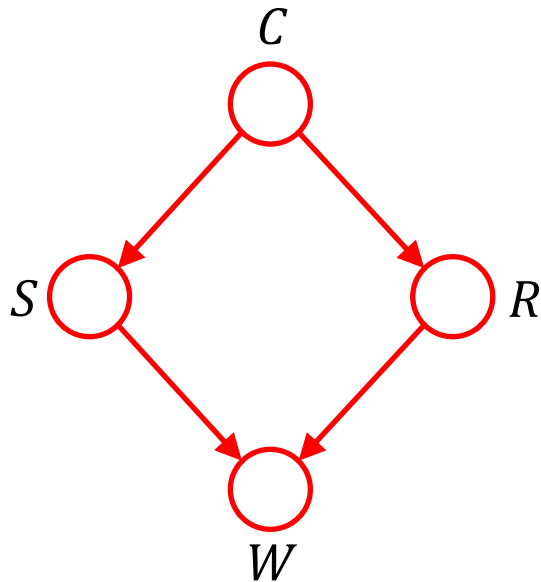
$$P(x_1, x_7 | x_4, x_5) = P(x_1 | x_4, x_5) P(x_7 | x_4, x_5)$$



Path between nodes x_1, x_3 is unblocked by observing node $x_7 \Rightarrow$

$$P(x_1, x_3 | x_7) \neq P(x_1 | x_7) P(x_3 | x_7)$$

d-separation: Examples II.



- S and R are not independent
 - $P(S, R) \neq P(S) P(R)$
- ... but are conditionally independent given (observed) C
 - $P(S, R|C) = P(S|C) P(R|C)$
 - Therefore, also $P(R|S, C) = \frac{P(S, R|C)}{P(S|C)} = P(R|C)$
- ... but become again dependent when observing also W
 - $P(S, R|C, W) \neq P(S|C, W) P(R|C, W)$

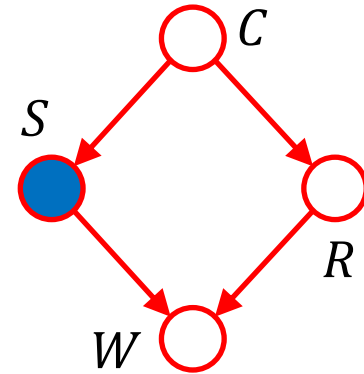
Example of inference in BN

- What is the probability that it rains given that sprinkler is on $P(R = T|S = T)$?
- We know how to evaluate joint distribution

$$P(C, S, R, W) = P(C)P(S|C)P(R|C)P(W|S, R)$$

- Using Bayes rule:

$$P(R = T|S = T) = \frac{P(R = T, S = T)}{P(S = T)} = \frac{P(R = T, S = T)}{P(R = T, S = T) + P(R = F, S = T)}$$



- To calculate $P(R, S)$ we use sum rule to marginalize out (sum over all values of) variables W and $C \rightarrow$ *brute force marginalization*.

$$\begin{aligned} P(R = T, S = T) &= P(C = T) P(S = T|C = T) P(R = T|C = T) P(W = T|R = T, S = T) \\ &\quad + P(C = T) P(S = T|C = T) P(R = T|C = T) P(W = F|R = T, S = T) \\ &\quad + P(C = F) P(S = T|C = F) P(R = T|C = F) P(W = T|R = T, S = T) \\ &\quad + P(C = F) P(S = T|C = F) P(R = T|C = F) P(W = F|R = T, S = T) \end{aligned}$$

$P(R = F, S = T)$ can be calculated similarly

Brute force inference in BN

- What is the probability that it rains given that sprinkler is on $P(R = T | S = T)$?

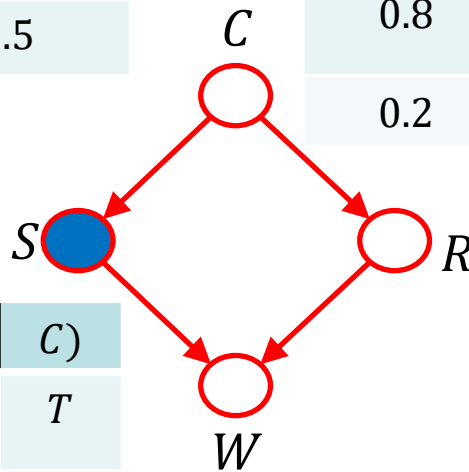
$$\begin{aligned}
 P(R = T, S = T) &= P(C = T) P(S = T | C = T) P(R = T | C = T) P(W = T | R = T, S = T) \\
 &+ P(C = T) P(S = T | C = T) P(R = T | C = T) P(W = F | R = T, S = T) \\
 &+ P(C = F) P(S = T | C = F) P(R = T | C = F) P(W = T | R = T, S = T) \\
 &+ P(C = F) P(S = T | C = F) P(R = T | C = F) P(W = F | R = T, S = T) \\
 &= 0.5 \cdot 0.1 \cdot 0.8 \cdot 0.99 \\
 &+ 0.5 \cdot 0.1 \cdot 0.8 \cdot 0.01 \\
 &+ 0.5 \cdot 0.5 \cdot 0.2 \cdot 0.99 \\
 &+ 0.5 \cdot 0.5 \cdot 0.2 \cdot 0.01 = 0.09
 \end{aligned}$$

$$\begin{aligned}
 P(R = F, S = T) &= P(C = T) P(S = T | C = T) P(R = F | C = T) P(W = T | R = F, S = T) \\
 &+ P(C = T) P(S = T | C = T) P(R = F | C = T) P(W = F | R = F, S = T) \\
 &+ P(C = F) P(S = T | C = F) P(R = F | C = F) P(W = T | R = F, S = T) \\
 &+ P(C = F) P(S = T | C = F) P(R = F | C = F) P(W = F | R = F, S = T) \\
 &= 0.5 \cdot 0.1 \cdot 0.2 \cdot 0.9 \\
 &+ 0.5 \cdot 0.1 \cdot 0.2 \cdot 0.1 \\
 &+ 0.5 \cdot 0.5 \cdot 0.8 \cdot 0.9 \\
 &+ 0.5 \cdot 0.5 \cdot 0.8 \cdot 0.1 = 0.21
 \end{aligned}$$

$$P(R = T | S = T) = \frac{P(R = T, S = T)}{P(R = T, S = T) + P(R = F, S = T)} = \frac{0.09}{0.09 + 0.21} = 0.3$$

$P(C = T)$
0.5

$P(R = T C)$	C
0.8	T
0.2	F



$P(S = T C)$	C
0.1	T
0.5	F

$P(W = T S, R)$	S	R
0.99	T	T
0.9	T	F
0.9	F	T
0.0	F	F

Optimized inference in BN

- What is the probability of rain given a known state of the sprinkler $P(R|S)$?
- Using a more general and compact notation in terms of random variables:

$$P(R, S) = \sum_C \sum_W P(C) P(S|C) P(R|C) P(W|R, S)$$

- This can be simplified using distributive property of multiplication:

$$P(R, S) = \sum_C P(C) P(S|C) P(R|C) \underbrace{\sum_W P(W|R, S)}_1$$

Summing over all possible values of C (i.e. True, False)

$$P(R|S) = \frac{P(R, S)}{\sum_R P(R, S)} = \frac{\sum_C P(C) P(S|C) P(R|C)}{\sum_R \sum_C P(C) P(S|C) P(R|C)} = \frac{\sum_C P(C) P(S|C) P(R|C)}{\sum_C P(C) P(S|C) \underbrace{\sum_R P(R|C)}_1}$$

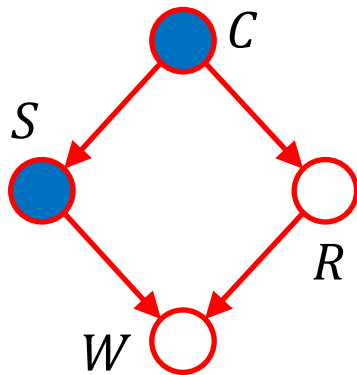
- To evaluate $P(R|S)$, we need only 6 multiplication, 2 additions and 1 division as compared to brute force 24 multiplications, 7 additions and 1 division
- We do not need table $P(W|R, S)$ at all to infer $P(R|S)$

Example II.

- Probability of rain given that it is cloudy, and sprinkler is on $P(R|C, S)$?

$$\begin{aligned} P(R, S, C) &= \sum_W P(C) P(S|C) P(R|C) P(W|R, S) \\ &= P(C) P(S|C) P(R|C) \underbrace{\sum_W P(W|R, S)}_1 \end{aligned}$$

$$P(R|S, C) = \frac{P(R, S, C)}{P(S, C)} = \frac{P(R, S, C)}{\sum_R P(R, S, C)} = \frac{P(C) P(S|C) P(R|C)}{P(C) P(S|C) \underbrace{\sum_R P(R|C)}_1} = P(R|C)$$

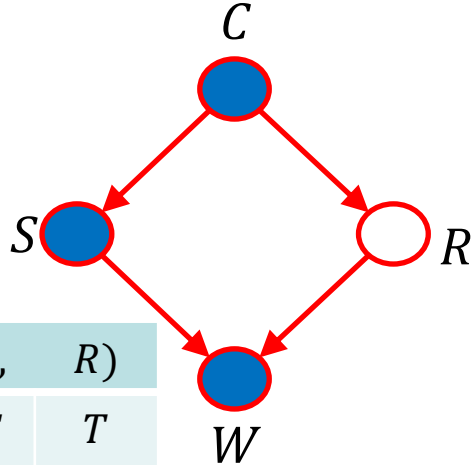


$P(R = T \mid C)$	C
0.8	T
0.2	F

R and S are conditionally independent given C and therefore $P(R|S, C) = P(R|C)$ as analyzed on slide ***d-separation: Examples II.***

Example III.

Probability of rain given being cloudy, sprinkler on and wet grass $P(R|C, S, W)$?



$P(R = T \mid C)$	C
0.8	T
0.2	F

$P(W \mid S, R)$	$S,$	$R)$
0.99	T	T
0.9	T	F
0.9	F	T
0.0	F	F

$$P(C, S, R, W) = P(C)P(S|C)P(R|C)P(W|S, R)$$

$$\begin{aligned}
 P(R|S, C, W) &= \frac{P(R, S, C, W)}{P(S, C, W)} = \frac{P(R, S, C, W)}{\sum_R P(R, S, C, W)} \\
 &= \frac{P(C)P(S|C)P(R|C)P(W|S, R)}{P(C)P(S|C)\sum_R P(R|C)P(W|S, R)} = \frac{P(R|C)P(W|S, R)}{\sum_R P(R|C)P(W|S, R)}
 \end{aligned}$$

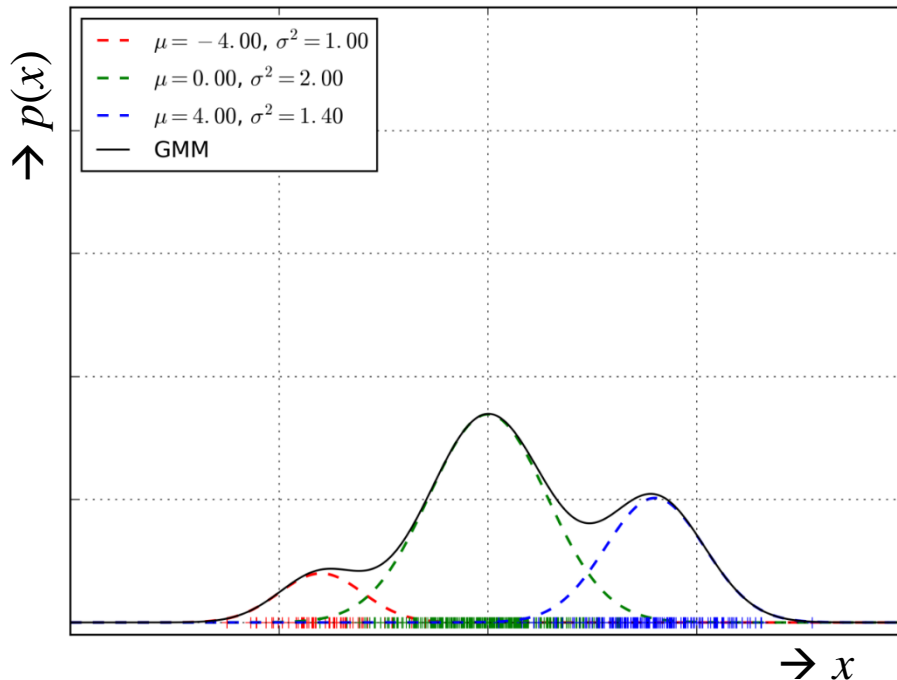
- As analyzed before, variables R and S are conditionally dependent given C and W , and therefore $P(R|S, C, W)$ depends on all the other variables

Examples of Bayesian Networks

- Some practical examples of BN (see the following slides)
 - Gaussian Mixture Model
 - Simple probability density model with discrete latent variable
 - Hidden Markov Model
 - Dynamic BN (DBN) modeling distribution of sequences
 - Probabilistic Linear Discriminant Analysis
 - Example of BN with continuous latent variables

Gaussian Mixture Model (GMM)

$$p(x|\boldsymbol{\eta}) = \sum_c \mathcal{N}(x; \mu_c, \sigma_c^2) \pi_c$$



where

$$\boldsymbol{\eta} = \{\pi_c, \mu_c, \sigma_c^2\}$$

$$\sum_c \pi_c = 1$$

- We can see the sum above just as a function defining the shape of the probability density function
- or ...

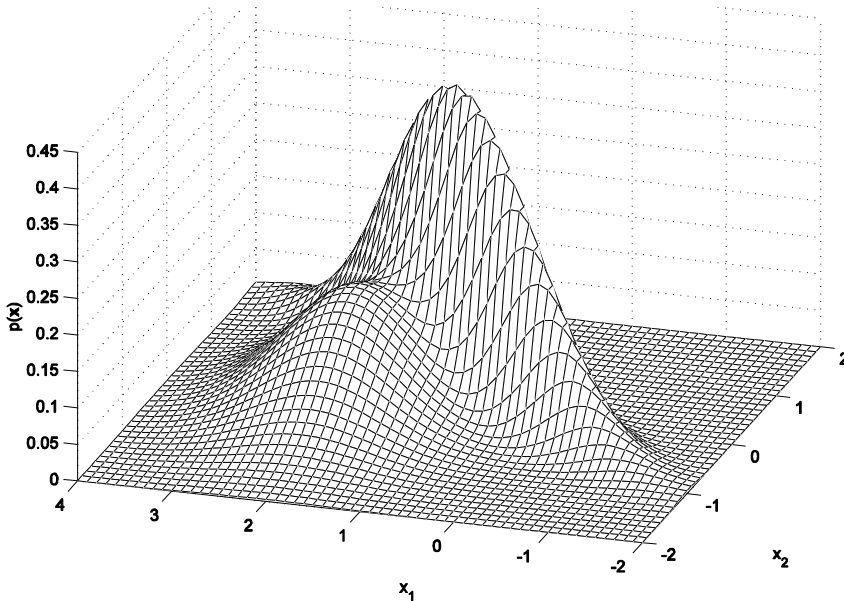
Multivariate GMM

$$p(\mathbf{x}|\boldsymbol{\eta}) = \sum_c \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \pi_c$$

where

$$\boldsymbol{\eta} = \{\pi_c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\}$$

$$\sum_c \pi_c = 1$$

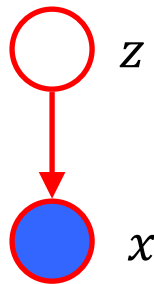


- We can see the sum above just as a function defining the shape of the probability density function
- or ...

Bayesian Networks for GMM

$$p(x) = \sum_z p(x|z)P(z) = \sum_c \mathcal{N}(x; \mu_c, \sigma_c^2) \text{Cat}(z = c | \boldsymbol{\pi})$$

- or we can see it as a generative probabilistic model described by Bayesian network with **Categorical** latent random variable z identifying **Gaussian** distribution generating the observation x

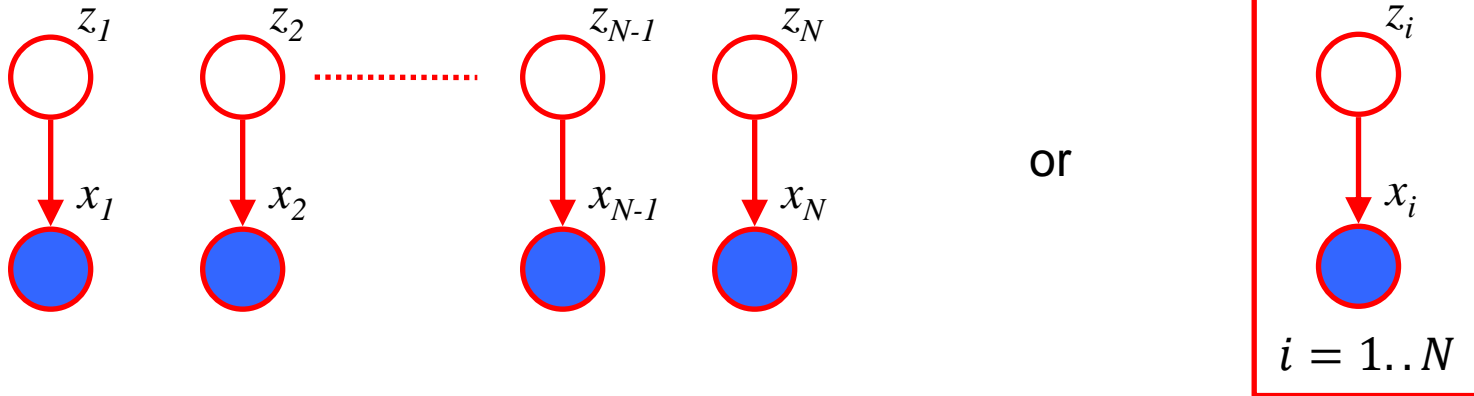


$$p(x, z) = p(x|z)P(z)$$

- Observations are assumed to be generated as follows:
 - randomly select Gaussian component according probabilities $P(z)$
 - generate observation x form the selected Gaussian distribution
- To evaluate $p(x)$, we marginalize out z
- No close form solution for training parameters $\mu_c, \sigma_c^2, \boldsymbol{\pi}$

Bayesian Networks for GMM - II

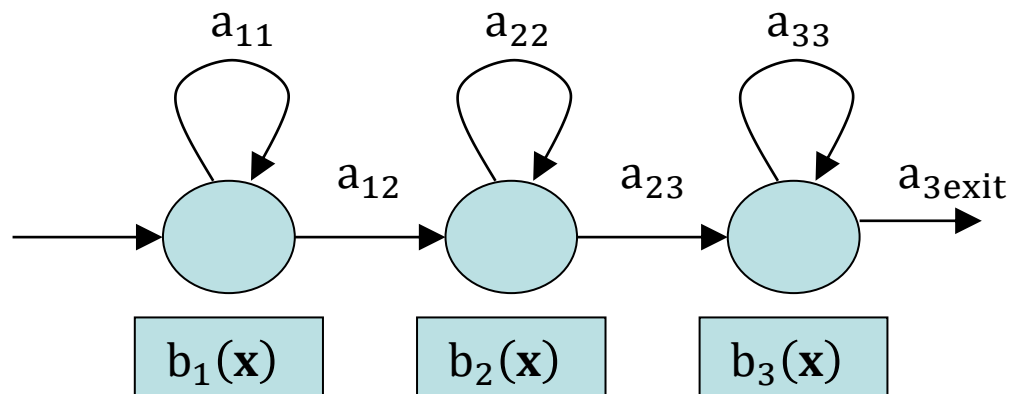
- Multiple observations:



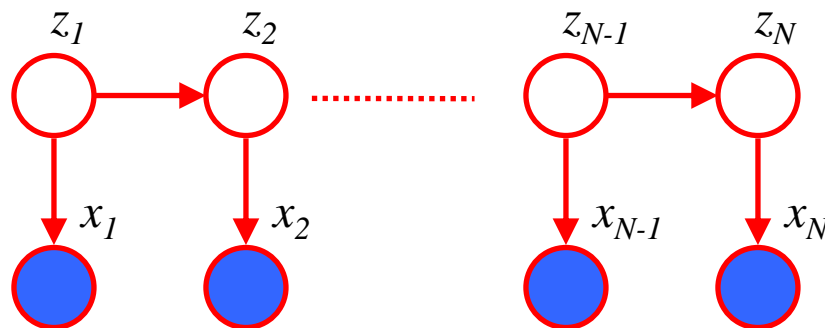
$$p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N) = \prod_{i=1}^N p(x_i | z_i) P(z_i)$$

(Dynamic) BN for HMM

- For each time frame, Hidden Markov Model moves from state j to state k according to a transition probability $a_{jk} = p(k|j)$ and generates observation \mathbf{x} from probability distribution $b_k(\mathbf{x}) = p(\mathbf{x}|k)$ associated with the entered state. More details on this model for *modeling sequences* are in SUR class.



- In BN, z_i nodes are not “HMM states”, these are random variables (one for each frame) with values saying which state we are in for a particular frame i



$$p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N) = P(z_1) \prod_{i=2}^N p(z_i|z_{i-1}) \prod_{i=1}^N p(x_i|z_i)$$

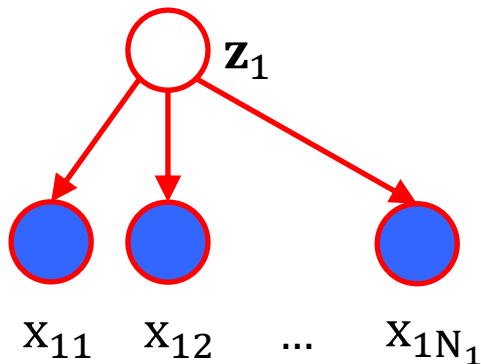
PLDA model for speaker verification

- Let each speech utterance be represented by *speaker embedding vector* \mathbf{x}
 - e.g. 512 dim. output of hidden layer of neural network trained for speaker classification
- We assume, that the distribution of the embeddings can be modeled as follows:
- We assume the same factorization as for GMM, but with continuous laten variable \mathbf{z}

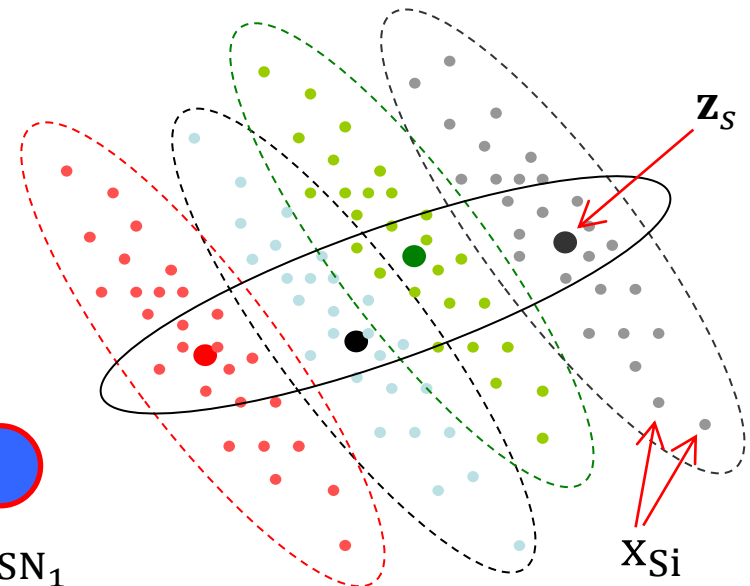
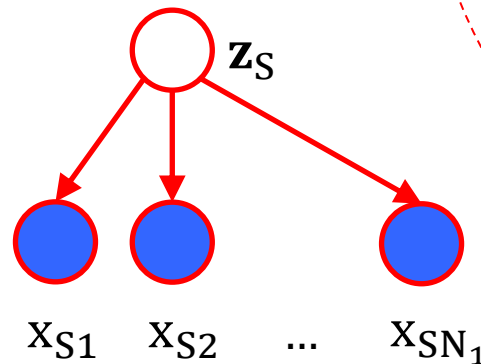
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\Sigma}_{ac}) \quad - \text{distribution of speaker means}$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{z}, \boldsymbol{\Sigma}_{wc}) \quad - \text{within class (channel) variability}$$

- Observations (embeddings) are assumed to be generated as follows:
 - Latent (speaker mean) vector \mathbf{z}_s is generated for each speaker s from gaussian distribution $p(\mathbf{z})$
 - All embeddings of speaker s are generated from Gaussian distribution $p(\mathbf{x}_{si}|\mathbf{z}_s)$



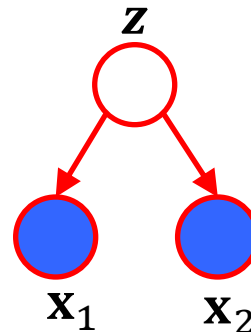
...



PLDA model for speaker verification II

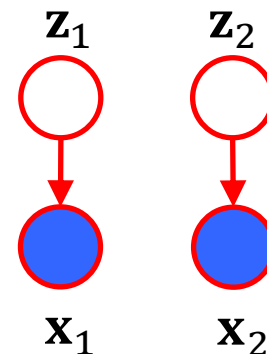
Same speaker hypothesis model:

$$p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_s) = \int p(\mathbf{x}_1 | \mathbf{z}) p(\mathbf{x}_2 | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}$$



Different speaker hypothesis model:

$$\begin{aligned} p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_d) &= p(\mathbf{x}_1) p(\mathbf{x}_2) \\ &= \int p(\mathbf{x}_1 | \mathbf{z}_1) p(\mathbf{z}_1) d\mathbf{z}_1 \int p(\mathbf{x}_2 | \mathbf{z}_2) p(\mathbf{z}_2) d\mathbf{z}_2 \end{aligned}$$



Probability that $\mathbf{x}_1, \mathbf{x}_2$ comes from the same speaker:

$$p(\mathcal{H}_s | \mathbf{x}_1, \mathbf{x}_2) = \frac{p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_s) P(\mathcal{H}_s)}{p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_s) P(\mathcal{H}_s) + p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_d) P(\mathcal{H}_d)}$$

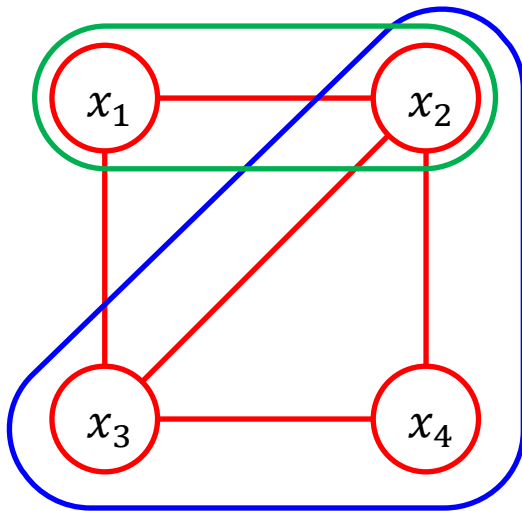
where $P(\mathcal{H}_s) = 1 - P(\mathcal{H}_d)$ is prior probability same speaker hypothesis

Usually, log likelihood ratio verification score is used $s = \log \frac{p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_s)}{p(\mathbf{x}_1, \mathbf{x}_2 | \mathcal{H}_d)}$

- More positive more likely $\mathbf{x}_1, \mathbf{x}_2$ are from the same speaker

Markov Random Fields

- Undirected graphical model
- Directly describe the conditional independence property
 - On the example: $P(x_1, x_4 | x_2, x_3) = P(x_1 | x_2, x_3) P(x_4 | x_2, x_3)$
 - x_1 and x_4 are independent given x_2 and x_3 as there is no path from x_1 to x_4 not leading through either x_2 or x_3 .



- Subsets of nodes where all nodes are connected with each other are called cliques (see green and blue examples)
- The outline in blue is Maximal clique, where no more nodes can be added
- When factorizing distribution described by MRF, variables not connected by link must not appear in the same factor \Rightarrow let's make factors corresponding to (Maximal) cliques.

MRF - factorization

- Joint probability distribution over all random variables \mathbf{x} can be expressed as normalized product of potential functions $\psi_C(\mathbf{x}_C)$, which are positive valued functions of subsets of variables \mathbf{x}_C corresponding to maximal cliques C
- It is useful to express the potential functions in terms of energy functions $E(\mathbf{x}_C) \rightarrow$ sum of $E(\mathbf{x}_C)$ terms instead of product of $\psi_C(\mathbf{x}_C)$ terms.

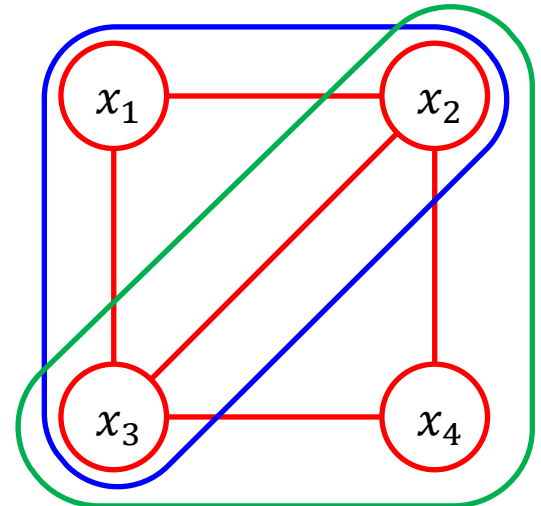
$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C)$$

$$Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

$$\psi_C(\mathbf{x}_C) = \exp\{-E(\mathbf{x}_C)\}$$

For our example:

$$\begin{aligned} P(x_1, x_2, x_3, x_4) &= \frac{1}{Z} \psi_{1,2,3}(x_1, x_2, x_3) \psi_{2,3,4}(x_2, x_3, x_4) \\ &= \frac{1}{Z} \exp\{-E(x_1, x_2, x_3) - E(x_2, x_3, x_4)\} \end{aligned}$$

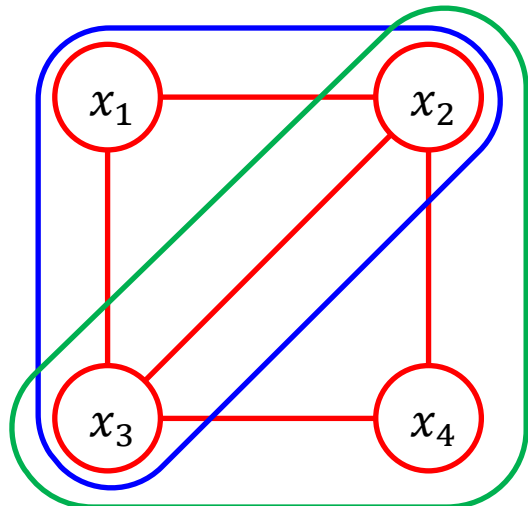


Checking the conditional independence

$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z} \psi_{1,2,3}(x_1, x_2, x_3) \psi_{2,3,4}(x_2, x_3, x_4)$$

$$P(x_2, x_3) = \sum_{x_1, x_4} \frac{1}{Z} \psi_{1,2,3}(x_1, x_2, x_3) \psi_{2,3,4}(x_2, x_3, x_4)$$

$$P(x_1, x_4 | x_2, x_3) = \frac{P(x_1, x_2, x_3, x_4)}{P(x_2, x_3)}$$



$$= \frac{\frac{1}{Z} \psi_{1,2,3}(x_1, x_2, x_3) \psi_{2,3,4}(x_2, x_3, x_4)}{\sum_{x_1, x_4} \frac{1}{Z} \psi_{1,2,3}(x_1, x_2, x_3) \psi_{2,3,4}(x_2, x_3, x_4)}$$

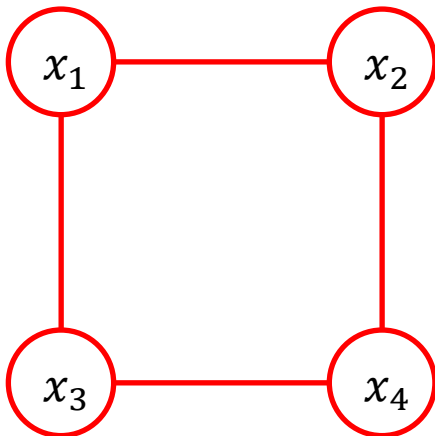
$$= \frac{\psi_{1,2,3}(x_1, x_2, x_3)}{\sum_{x_1} \psi_{1,2,3}(x_1, x_2, x_3)} \frac{\psi_{2,3,4}(x_2, x_3, x_4)}{\sum_{x_4} \psi_{2,3,4}(x_2, x_3, x_4)}$$

$$= P(x_1 | x_2, x_3) P(x_4 | x_2, x_3)$$

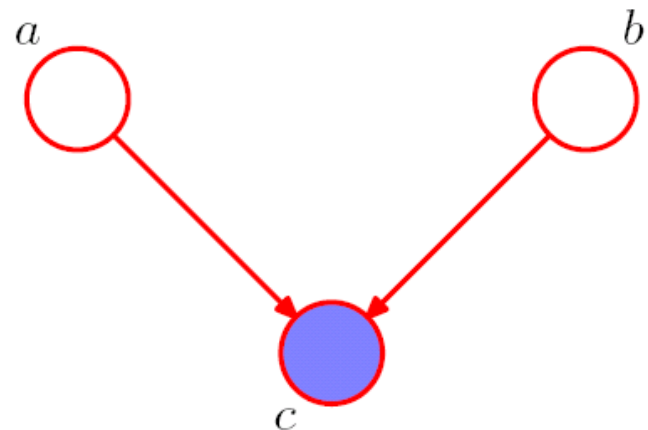
BN vs MRF

- Some of the probability distributions that can be represented using Bayesian Network cannot be fully represented as Markov Random Field and vice versa
- But often we can convert one to the other – see next slide
- We mainly introduce MRF (and later Factor Graph) to present more general class of inference algorithm (see later)

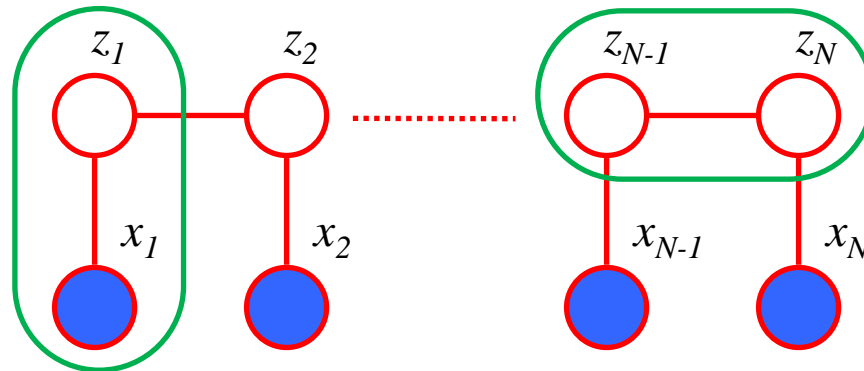
MRF whose statistical independence property cannot be represented by BN



BN whose (explain away) statistical independence property cannot be represented by MRF



Example: HMM as MRF



$$p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N) = \frac{1}{Z} \prod_{i=2}^N \tilde{\psi}(z_i, z_{i-1}) \prod_{i=1}^N \psi(x_i, z_i)$$

For

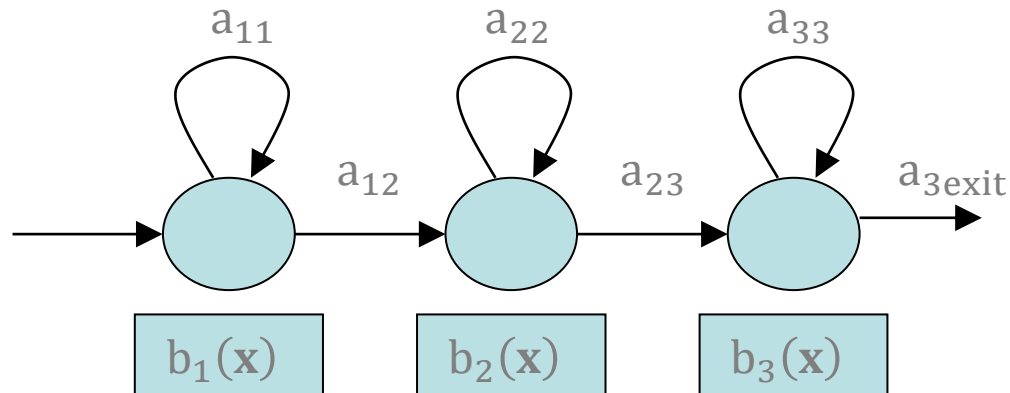
$$\begin{aligned} Z &= 1 \\ \tilde{\psi}(z_2, z_1) &= p(z_2|z_1)p(z_1) \\ \tilde{\psi}(z_i, z_{i-1}) &= p(z_i|z_{i-1}) \\ \psi(x_i, z_i) &= p(x_i|z_i) \end{aligned}$$

We recover the factorization for HMM:

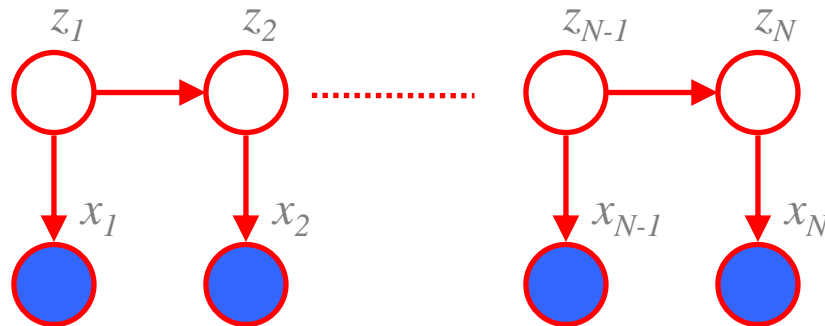
$$p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N) = P(z_1) \prod_{i=2}^N p(z_i|z_{i-1}) \prod_{i=1}^N p(x_i|z_i)$$

Flashback: (Dynamic) BN for HMM

- For each time frame, Hidden Markov Model moves from state j to state k according to a transition probability $a_{jk} = p(k|j)$ and generates observation \mathbf{x} from probability distribution $b_k(\mathbf{x}) = p(\mathbf{x}|k)$ associated with the entered state. More details on this model for *modeling sequences* are in SUR class.



- In BN, z_i nodes are not “HMM states”, these are random variables (one for each frame) with values saying which state we are in for a particular frame i



$$p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N) = P(z_1) \prod_{i=2}^N p(z_i|z_{i-1}) \prod_{i=1}^N p(x_i|z_i)$$

Inference on a chain

- Now, we will be interested in carrying out efficient inference in MRFs
- As a first example, we consider simple MRF with chain topology



$$P(\mathbf{x}) = P(x_1, x_2, \dots, x_N) = \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \dots \psi_{N-1,N}(x_{N-1}, x_N)$$

- Let all N variables x_i are discrete with K states
- Each $\psi_{i,j}(x_i, x_j)$ is represented by $K \times K$ table $\Rightarrow (N - 1)K^2$ parameters
- We would like to find marginal probability of variable x_n

$$P(x_n) = \sum_{x_1} \dots \sum_{x_{n-1}} \sum_{x_{n+1}} \dots \sum_{x_N} P(\mathbf{x})$$

Brute force marginalization has complexity $O(K^N)$

Inference on a chain II.

$$P(x_n) = \sum_{x_1} \dots \sum_{x_{n-1}} \sum_{x_{n+1}} \dots \sum_{x_N} \frac{1}{Z} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \dots \psi_{N-1,N}(x_{N-1}, x_N)$$

$$P(x_n) = \frac{1}{Z} \left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \dots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \dots \right] \left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \dots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \dots \right]$$

Complexity can be reduced to $O(NK^2)$ by rearranging the sums using distributive property of multiplication.

Inference on a chain III.

$$P(x_n) = \frac{1}{Z} \left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \dots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \dots \right] \dots \right] \left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \dots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \dots \right]$$

$\alpha(x_n)$ (bracketed above the top row)
 $\alpha(x_3)$ (bracketed above the middle row)
 $\alpha(x_2)$ (bracketed above the innermost row)
 $\beta(x_{N-1})$ (bracketed below the bottom row)
 $\beta(x_n)$ (bracketed below the entire expression)

Recursive formulas: $P(x_n) = \frac{1}{Z} \alpha(x_n) \beta(x_n)$

$$\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \alpha(x_{n-1})$$

$$\beta(x_n) = \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \beta(x_{n+1})$$

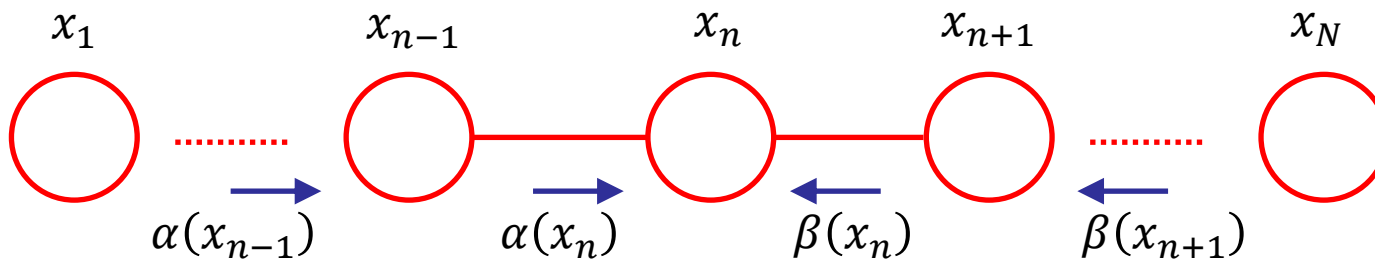
$$\alpha(x_1) = \beta(x_N) = 1$$

Inference on a chain IV.

- Marginals for ALL variables can be obtained at once with the same $O(NK^2)$ complexity, by recursively calculating all $\alpha(x_n)$ and $\beta(x_n)$ for all the nodes (i.e. for all $n = 1..N$) and then calculating

$$P(x_n) = \frac{1}{Z} \alpha(x_n) \beta(x_n)$$

- Vectors $\alpha(x_n)$ and $\beta(x_n)$ can be thought as messages passed from node to node. We will use this abstraction later

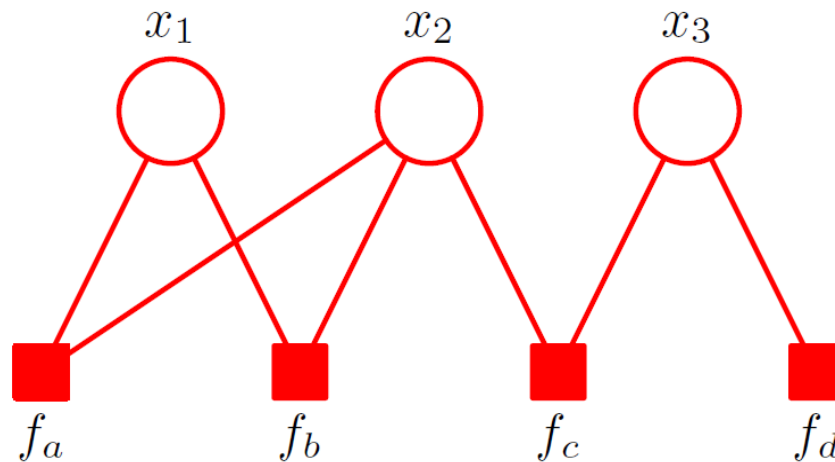


- Same inference could be done for continuous random variables just by replacing sums with integrals (if the integrals are tractable).

Factor graphs (FG)

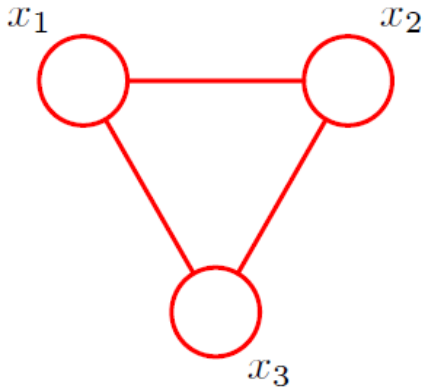
- Bipartite graph, with variable nodes x_i connected only to factor nodes f_j and vice versa.
- Like MRF, but with explicitly specified factors (or potential functions)

$$P(\mathbf{x}) = \frac{1}{Z} \prod_s f_s(\mathbf{x}_s)$$

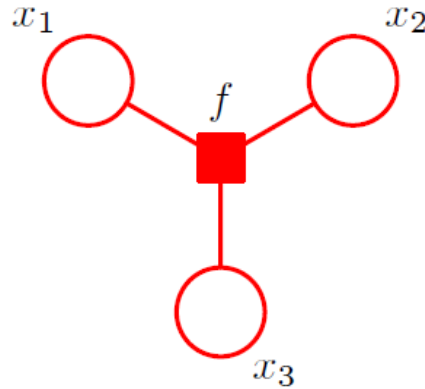


$$P(\mathbf{x}) = P(x_1, x_2, x_3) = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

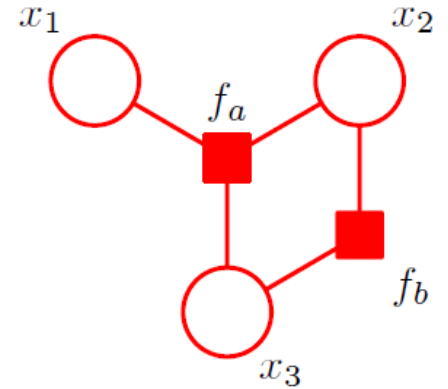
MRF to FG



$$p(\mathbf{x}) = \frac{1}{Z} \psi(x_1, x_2, x_3)$$

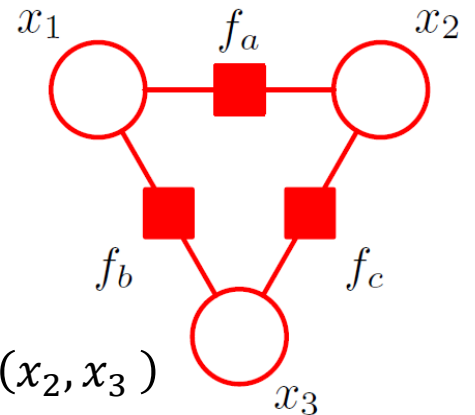


$$p(\mathbf{x}) = \frac{1}{Z} f(x_1, x_2, x_3)$$



$$p(\mathbf{x}) = \frac{1}{Z} f_a(x_1, x_2, x_3) f_b(x_1, x_2)$$

- MRF can be converted to FG
- Since FG is more explicate, different factor graphs can correspond to the same MRF



$$p(\mathbf{x}) = \frac{1}{Z} f_a(x_1, x_2) f_b(x_1, x_3) f_c(x_2, x_3)$$

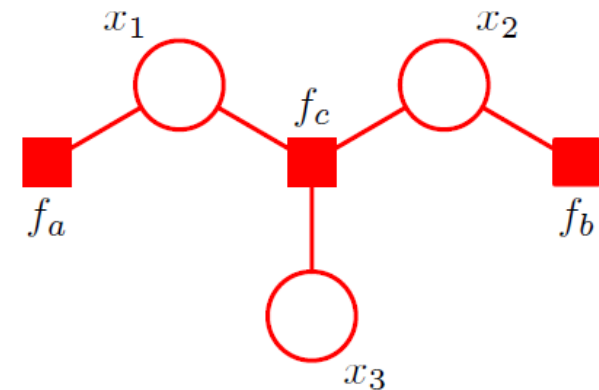
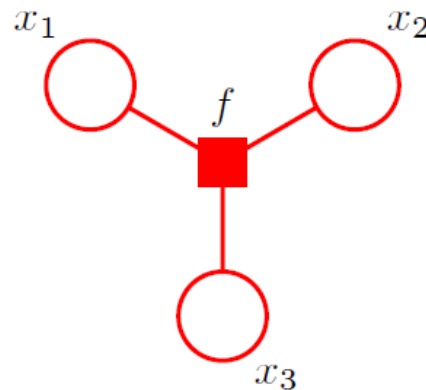
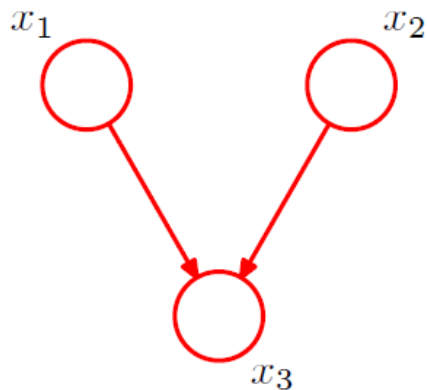
BN to FG

- BN can be also converted to FG, but such graph does not necessarily describe all the conditional independence properties
 - e.g. the explain away property of the BN from this example is not directly seen from the resulting factor graphs
- Again, different FGs can be constructed for the same BN to capturing more or less details about the original factorization.

$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3|x_1, x_2)$$

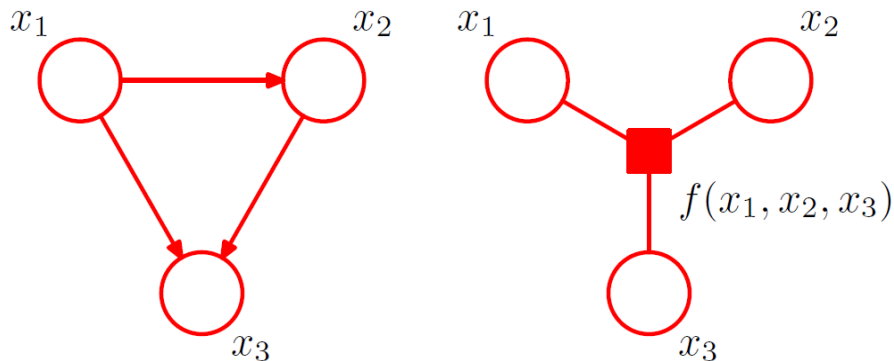
$$f(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$$

$$\begin{aligned} f_a(x_1) &= p(x_1) \\ f_b(x_2) &= p(x_2) \\ f_c(x_1, x_2, x_3) &= p(x_3|x_1, x_2) \end{aligned}$$

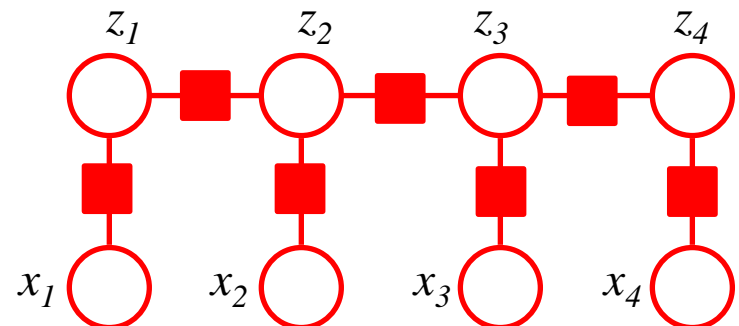


FG with Tree topology

- Efficient inference algorithms exist for FG with tree topology
- Even if the original BN or MRF is not tree, it can be often represented by FG with tree topology

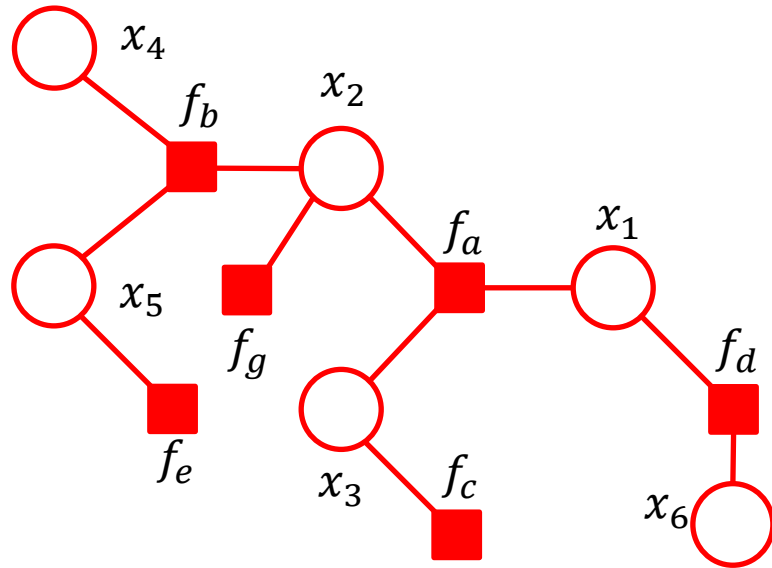


E.g. FG for HMM is tree



Belief Propagation

- Also known as **sum-product message passing** algorithm
- Algorithm for exact inference in FG (or MRF) with (poly)tree topology



- We already know that joint probability

$$P(\mathbf{x}) = P(x_1, x_2, \dots, x_N) = \frac{1}{Z} \prod_s f_s(\mathbf{x}_s)$$

\mathbf{x}_s - set of variables nodes that are neighbors of factor node f_s

- We are interested in obtaining the marginal probability of some x_n

$$P(x_n) = \sum_{\mathbf{x} \setminus x_n} P(\mathbf{x})$$

We will assume only discrete variables. For continuous, sum would be replaced by integral

$\mathbf{x} \setminus x_n$ - set of all variables excluding variable x_n

Belief Propagation - algorithm

The marginal probability for x_n can be efficiently calculated as

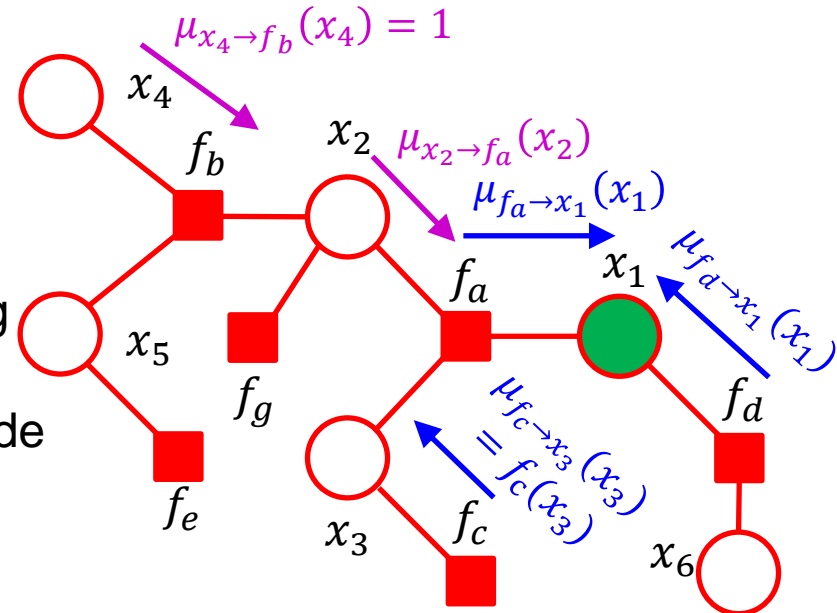
$$P(x_n) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

where $\text{ne}(x_n)$ is a set of factor nodes neighboring with variable node x_n and $\mu_{f_s \rightarrow x_n}(x_n)$ is so-called message send from factor node f_s to variable node x_n . Like $P(x_n)$, each $\mu_{f_s \rightarrow x_n}(x_n)$ is function of variable x_n and can be recursively evaluated as

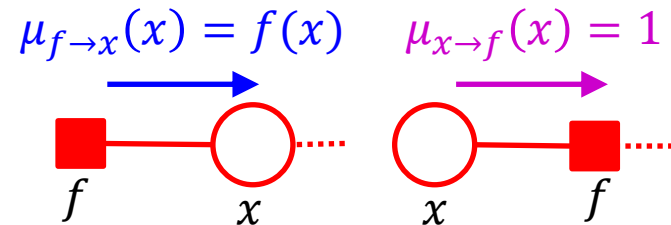
$$\mu_{f_s \rightarrow x_n}(x_n) = \sum_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

where \mathbf{x}_s is set of variables that are neighbors of factor node f_s , $\text{ne}(f_s) \setminus x_n$ is set of variable nodes neighboring with factor f_s node excluding x_n and $\mu_{x_m \rightarrow f_s}(x_m)$ is message send from variable node x_m to factor node f_s .

$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)$$



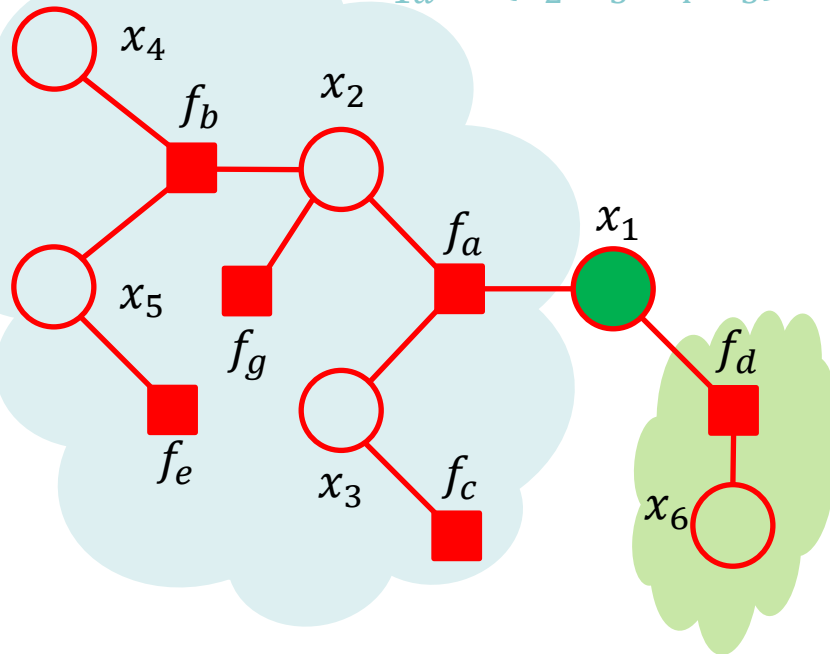
- We choose x_n as a **tree root**
- We start from the leaves



- Once a node obtains messages from all its children, it sends message to its parent.
- When root obtains all the messages, $P(x_n)$ is evaluated

Belief Propagation – derivation I

Factors in $F_a(x_1, X_{1a})$
 $X_{1a} = \{x_2, x_3, x_4, x_5\}$



Factors in $F_d(x_1, X_{1d})$
 $X_{1d} = \{x_6\}$

Let's consider node x_n as a **root** node

$$P(\mathbf{x}) = \frac{1}{Z} \prod_s f_s(\mathbf{x}_s) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} F_s(x_n, X_{ns})$$

$\text{ne}(x_n)$ - set of factor nodes that are neighbors of variable node x_n

X_{ns} - variables in the subtree connected to x_n via the factor node f_s

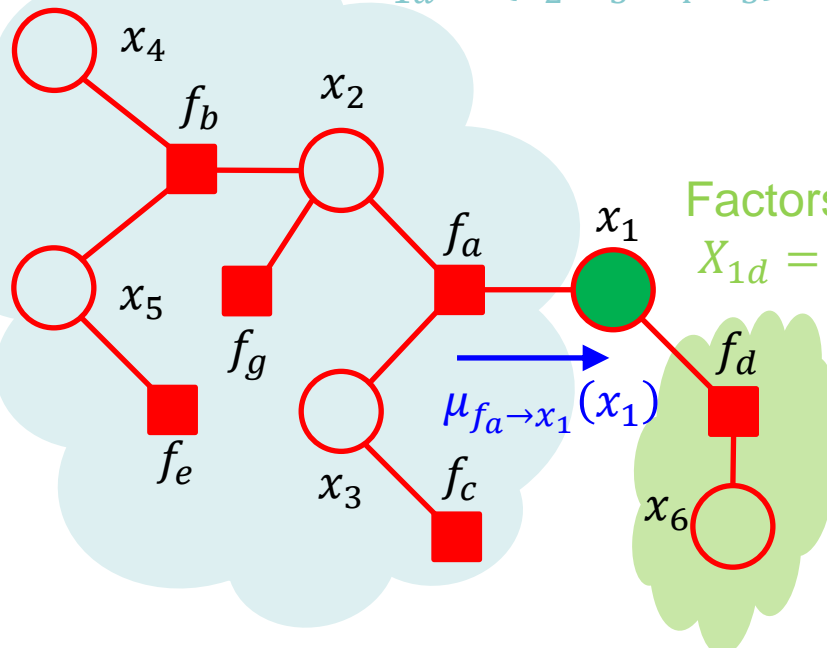
$F_s(x_n, X_s)$ - product of all the factors in the group associated with factor f_s

For our example:

$$P(\mathbf{x}) = \frac{1}{Z} \underbrace{f_a(x_1, x_2, x_3) f_b(x_2, x_4, x_5) f_e(x_5) f_g(x_2) f_c(x_3)}_{F_a(x_1, X_{1a})} \underbrace{f_d(x_1, x_6)}_{F_d(x_1, X_{1d})}$$

Belief Propagation – derivation II

Factors in $F_a(x_1, X_{1a})$
 $X_{1a} = \{x_2, x_3, x_4, x_5\}$



Factors in $F_d(x_1, X_{1d})$
 $X_{1d} = \{x_6\}$

$$\begin{aligned}
 P(x_n) &= \sum_{\mathbf{x} \setminus x_n} P(\mathbf{x}) \\
 &= \frac{1}{Z} \sum_{\mathbf{x} \setminus x_n} \prod_{s \in \text{ene}(x_n)} F_s(x_n, X_{ns}) \\
 &= \frac{1}{Z} \prod_{s \in \text{ene}(x_n)} \sum_{X_{ns}} F_s(x_n, X_{ns}) \\
 &= \frac{1}{Z} \prod_{s \in \text{ene}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)
 \end{aligned}$$

Product of all the factors in the subtree connected to x_n through factor node f_s with all the variables other than x_n marginalized (summed) out

For our example:

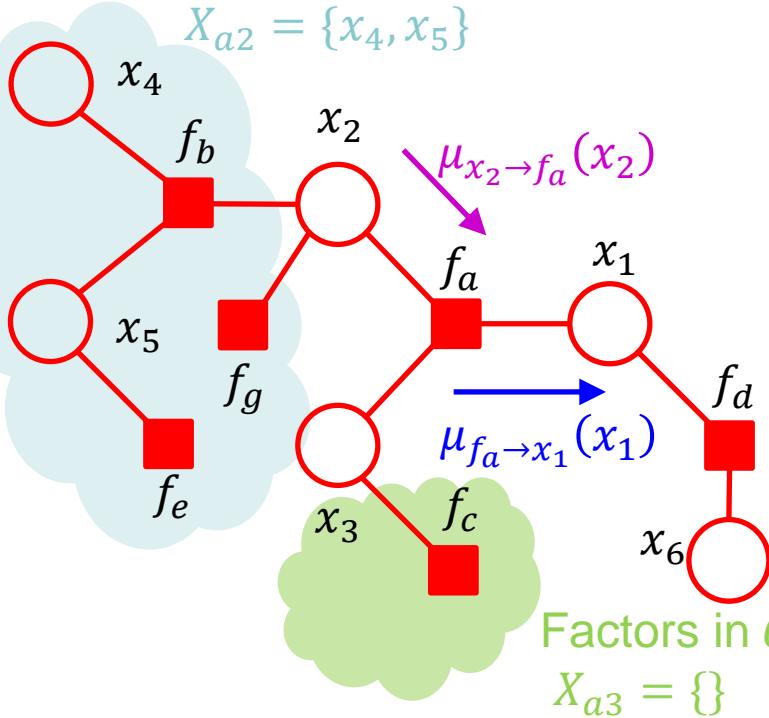
$$\begin{aligned}
 p(x_1) &= \frac{1}{Z} \sum_{x_2} \dots \sum_{x_6} F_a(x_1, X_{1a}) F_d(x_1, X_{1d}) \\
 &= \frac{1}{Z} \left[\sum_{x_2, x_3, x_4, x_5} F_a(x_1, X_{1a}) \right] \left[\sum_{x_6} F_d(x_1, X_{1d}) \right]
 \end{aligned}$$

Factors in $F_a(x_1, X_{1a})$ only depend on variables x_2, x_3, x_4, x_5

Belief Propagation – derivation III

Factors in $G_2(x_2, X_{a2})$

$X_{a2} = \{x_4, x_5\}$



Factors in $G_3(x_3, X_{a3})$

$X_{a3} = \{\}$

$$\begin{aligned}
 \mu_{f_s \rightarrow x_n}(x_n) &= \sum_{X_{ns}} F_S(x_n, X_{ns}) \\
 &= \sum_{X_{ns}} f_S(\mathbf{x}_S) \prod_{m \in \text{ne}(f_s) \setminus x_n} G_m(x_m, X_{sm}) \\
 &= \sum_{\mathbf{x}_S \setminus x_n} f_S(\mathbf{x}_S) \prod_{x_m \in \text{ne}(f_s) \setminus x_n} \sum_{X_{sm}} G_m(x_m, X_{sm}) \\
 &= \sum_{\mathbf{x}_S \setminus x_n} f_S(\mathbf{x}_S) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)
 \end{aligned}$$

X_{ns} - variables in the subtree connected to x_n via the factor node f_s

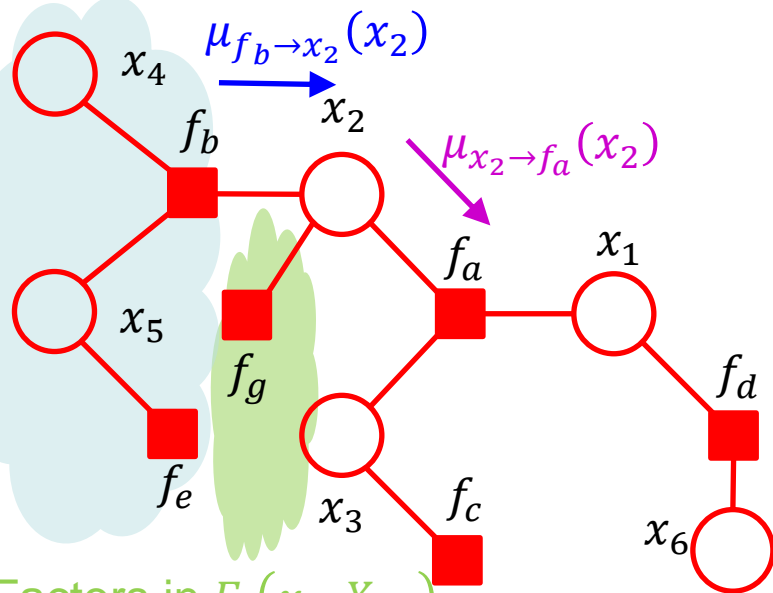
X_{sm} - variables in the subtree connected to f_s via the variable node x_m (excluding x_m)

$$\mu_{f_a \rightarrow x_1}(x_1) = \sum_{x_2, x_3} f_a(x_1, x_2, x_3) \left[\underbrace{\sum_{x_4, x_5} \underbrace{f_b(x_2, x_4, x_5) f_e(x_5) f_g(x_2)}_{G_2(x_2, X_{a2})}}_{\mu_{x_2 \rightarrow f_a}(x_2)} \right] \underbrace{\underbrace{f_c(x_3)}_{G_3(x_3, X_{a3})}}_{\mu_{x_3 \rightarrow f_a}(x_3)}$$

Belief Propagation – derivation IV

Factors in $F_b(x_2, X_{2b})$

$X_{2b} = \{x_4, x_5\}$



Factors in $F_g(x_2, X_{2g})$

$X_{2g} = \{\}$

$$\begin{aligned}
 \mu_{x_m \rightarrow f_s}(x_m) &= \sum_{X_{sm}} G_m(x_m, X_{sm}) \\
 &= \sum_{X_{sm}} \prod_{l \in \text{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml}) \\
 &= \prod_{l \in \text{ne}(x_m) \setminus f_s} \sum_{X_{ml}} F_l(x_m, X_{ml}) \\
 &= \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)
 \end{aligned}$$

$$\mu_{x_2 \rightarrow f_a}(x_2) = \underbrace{\left[\sum_{x_4, x_5} \frac{f_b(x_2, x_4, x_5) f_e(x_5)}{F_b(x_2, X_{2b})} \right]}_{\mu_{f_b \rightarrow x_2}(x_2)} \underbrace{\frac{f_g(x_2)}{F_g(x_2, X_{2g})}}_{\mu_{f_g \rightarrow x_2}(x_2)}$$

Belief Propagation - algorithm

The marginal probability for x_n can be efficiently calculated as

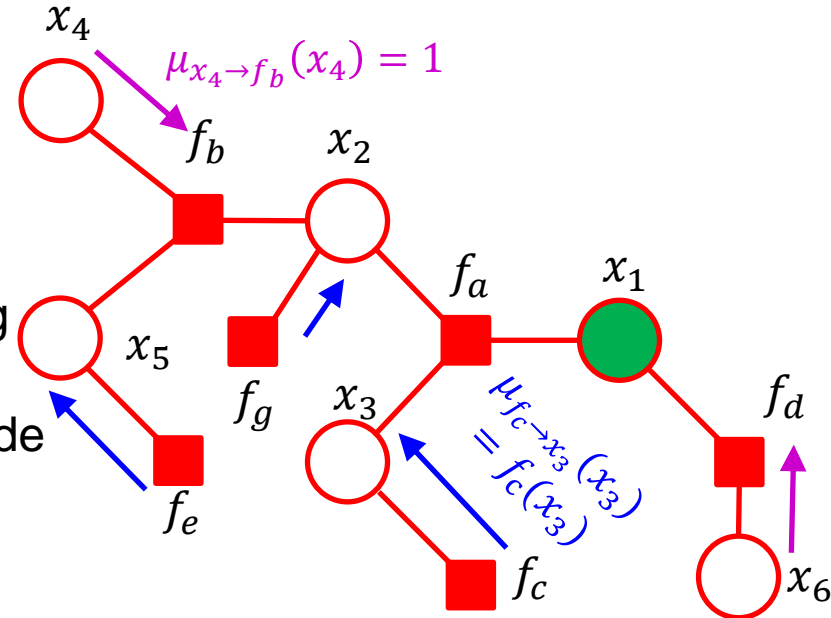
$$P(x_n) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

where $\text{ne}(x_n)$ is a set of factor nodes neighboring with variable node x_n and $\mu_{f_s \rightarrow x_n}(x_n)$ is so-called message send from factor node f_s to variable node x_n . Like $P(x_n)$, each $\mu_{f_s \rightarrow x_n}(x_n)$ is function of variable x_n and can be recursively evaluated as

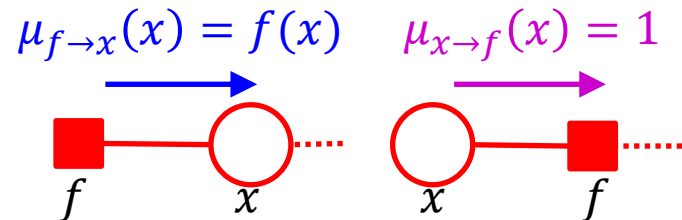
$$\mu_{f_s \rightarrow x_n}(x_n) = \sum_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

where \mathbf{x}_s is set of variables that are neighbors of factor node f_s , $\text{ne}(f_s) \setminus x_n$ is set of variable nodes neighboring with factor f_s node excluding x_n and $\mu_{x_m \rightarrow f_s}(x_m)$ is message send from variable node x_m to factor node f_s .

$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)$$



- We choose x_n as a **tree root**
- We start from the leaves



- Once a node obtains messages from all its children, it sends message to its parent.
- When root obtains all the messages, $P(x_n)$ is evaluated

Belief Propagation - algorithm

The marginal probability for x_n can be efficiently calculated as

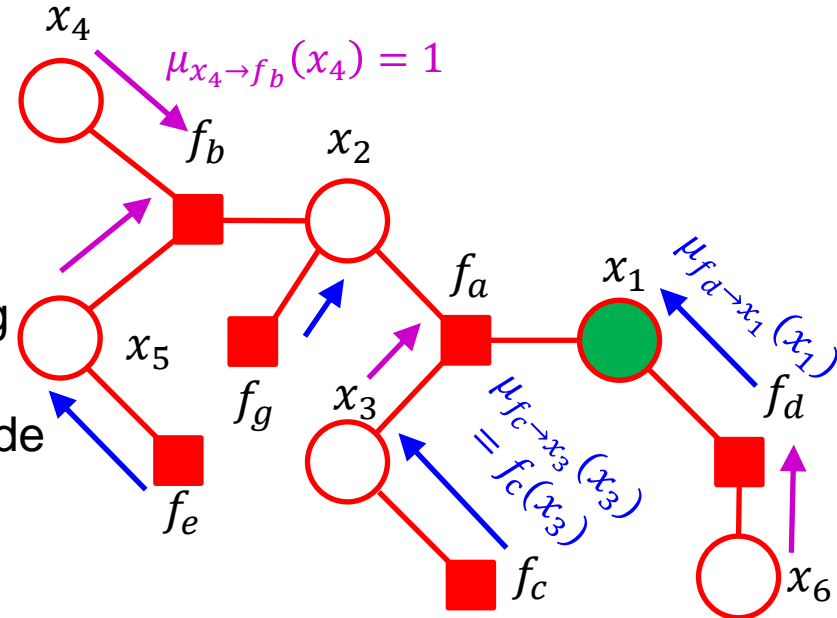
$$P(x_n) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

where $\text{ne}(x_n)$ is a set of factor nodes neighboring with variable node x_n and $\mu_{f_s \rightarrow x_n}(x_n)$ is so-called message send from factor node f_s to variable node x_n . Like $P(x_n)$, each $\mu_{f_s \rightarrow x_n}(x_n)$ is function of variable x_n and can be recursively evaluated as

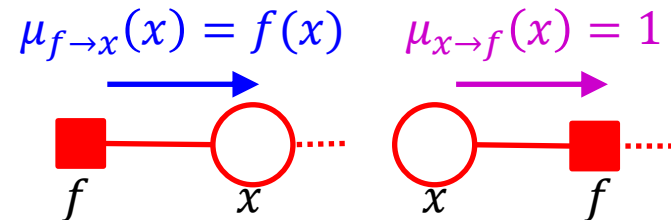
$$\mu_{f_s \rightarrow x_n}(x_n) = \sum_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

where \mathbf{x}_s is set of variables that are neighbors of factor node f_s , $\text{ne}(f_s) \setminus x_n$ is set of variable nodes neighboring with factor f_s node excluding x_n and $\mu_{x_m \rightarrow f_s}(x_m)$ is message send from variable node x_m to factor node f_s .

$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)$$



- We choose x_n as a **tree root**
- We start from the leaves



- Once a node obtains messages from all its children, it sends message to its parent.
- When root obtains all the messages, $P(x_n)$ is evaluated

Belief Propagation - algorithm

The marginal probability for x_n can be efficiently calculated as

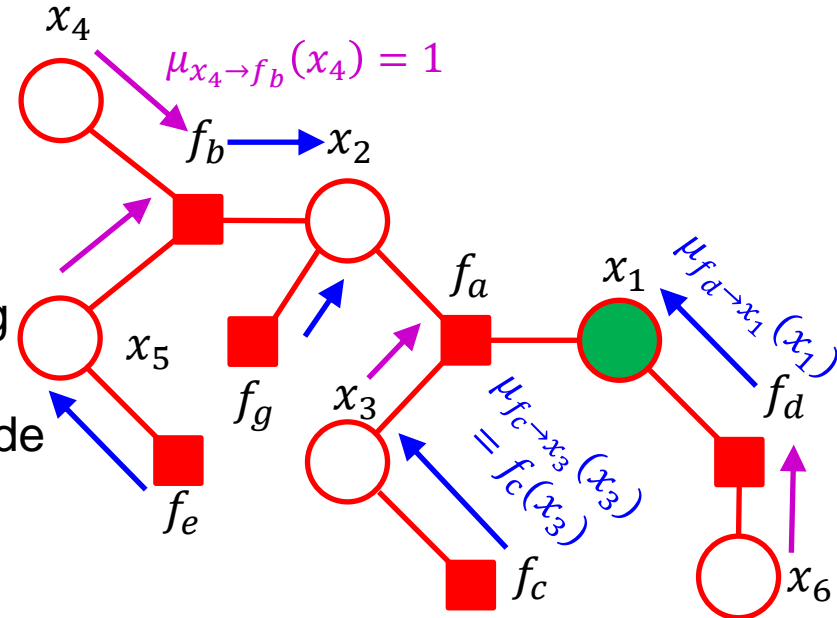
$$P(x_n) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

where $\text{ne}(x_n)$ is a set of factor nodes neighboring with variable node x_n and $\mu_{f_s \rightarrow x_n}(x_n)$ is so-called message send from factor node f_s to variable node x_n . Like $P(x_n)$, each $\mu_{f_s \rightarrow x_n}(x_n)$ is function of variable x_n and can be recursively evaluated as

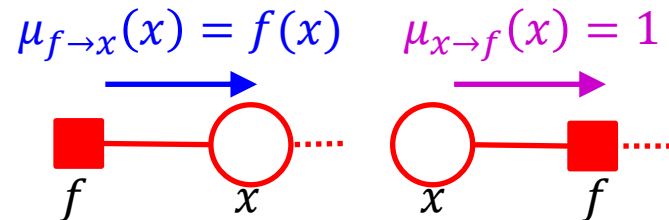
$$\mu_{f_s \rightarrow x_n}(x_n) = \sum_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

where \mathbf{x}_s is set of variables that are neighbors of factor node f_s , $\text{ne}(f_s) \setminus x_n$ is set of variable nodes neighboring with factor f_s node excluding x_n and $\mu_{x_m \rightarrow f_s}(x_m)$ is message send from variable node x_m to factor node f_s .

$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)$$



- We choose x_n as a **tree root**
- We start from the leaves



- Once a node obtains messages from all its children, it sends message to its parent.
- When root obtains all the messages, $P(x_n)$ is evaluated

Belief Propagation - algorithm

The marginal probability for x_n can be efficiently calculated as

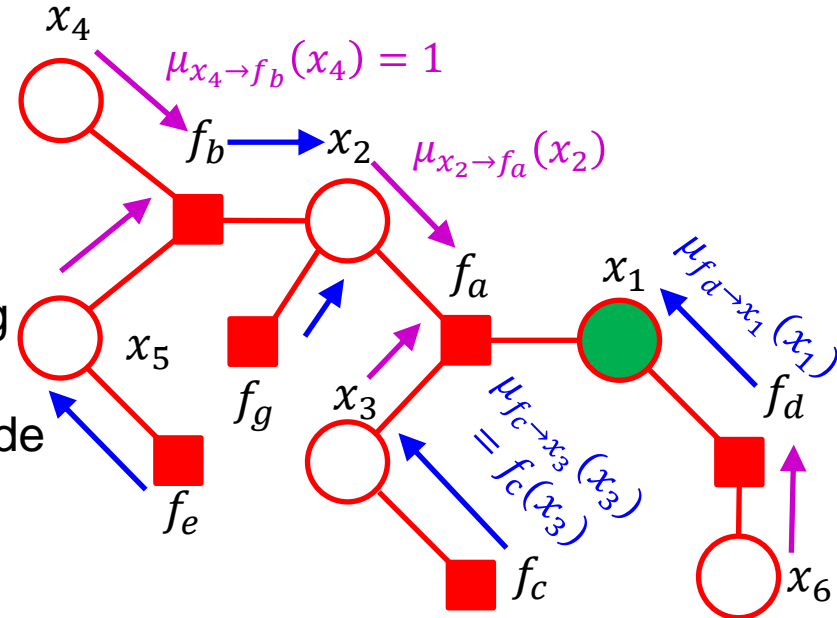
$$P(x_n) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

where $\text{ne}(x_n)$ is a set of factor nodes neighboring with variable node x_n and $\mu_{f_s \rightarrow x_n}(x_n)$ is so-called message send from factor node f_s to variable node x_n . Like $P(x_n)$, each $\mu_{f_s \rightarrow x_n}(x_n)$ is function of variable x_n and can be recursively evaluated as

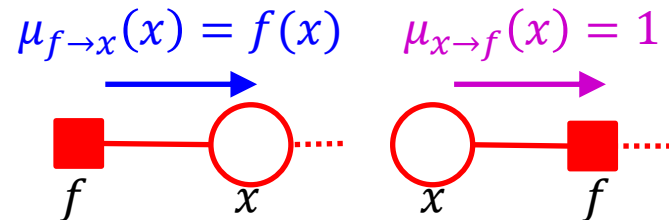
$$\mu_{f_s \rightarrow x_n}(x_n) = \sum_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

where \mathbf{x}_s is set of variables that are neighbors of factor node f_s , $\text{ne}(f_s) \setminus x_n$ is set of variable nodes neighboring with factor f_s node excluding x_n and $\mu_{x_m \rightarrow f_s}(x_m)$ is message send from variable node x_m to factor node f_s .

$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)$$



- We choose x_n as a **tree root**
- We start from the leaves



- Once a node obtains messages from all its children, it sends message to its parent.
- When root obtains all the messages, $P(x_n)$ is evaluated

Belief Propagation - algorithm

The marginal probability for x_n can be efficiently calculated as

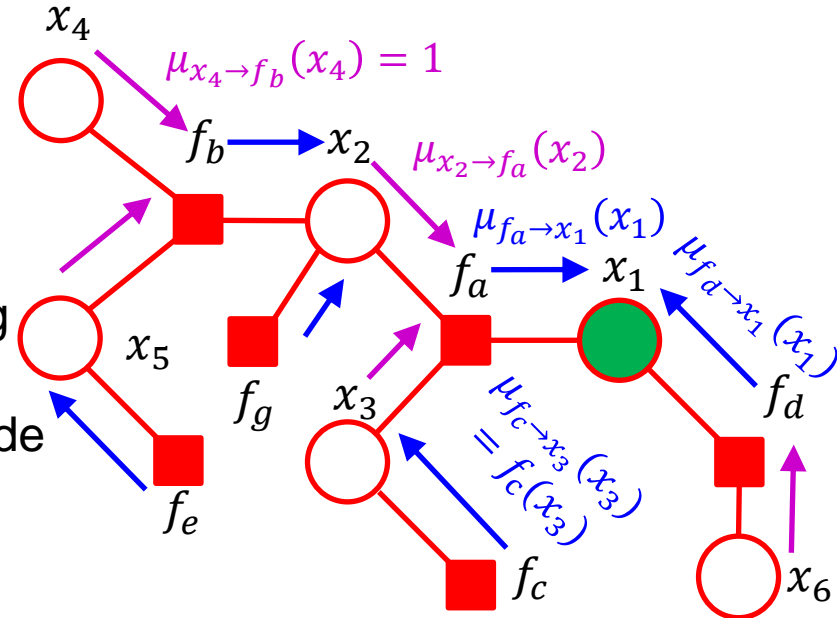
$$P(x_n) = \frac{1}{Z} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

where $\text{ne}(x_n)$ is a set of factor nodes neighboring with variable node x_n and $\mu_{f_s \rightarrow x_n}(x_n)$ is so-called message send from factor node f_s to variable node x_n . Like $P(x_n)$, each $\mu_{f_s \rightarrow x_n}(x_n)$ is function of variable x_n and can be recursively evaluated as

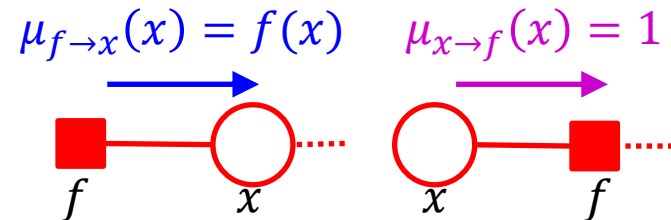
$$\mu_{f_s \rightarrow x_n}(x_n) = \sum_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

where \mathbf{x}_s is set of variables that are neighbors of factor node f_s , $\text{ne}(f_s) \setminus x_n$ is set of variable nodes neighboring with factor f_s node excluding x_n and $\mu_{x_m \rightarrow f_s}(x_m)$ is message send from variable node x_m to factor node f_s .

$$\mu_{x_m \rightarrow f_s}(x_m) = \prod_{l \in \text{ne}(x_m) \setminus f_s} \mu_{f_l \rightarrow x_m}(x_m)$$

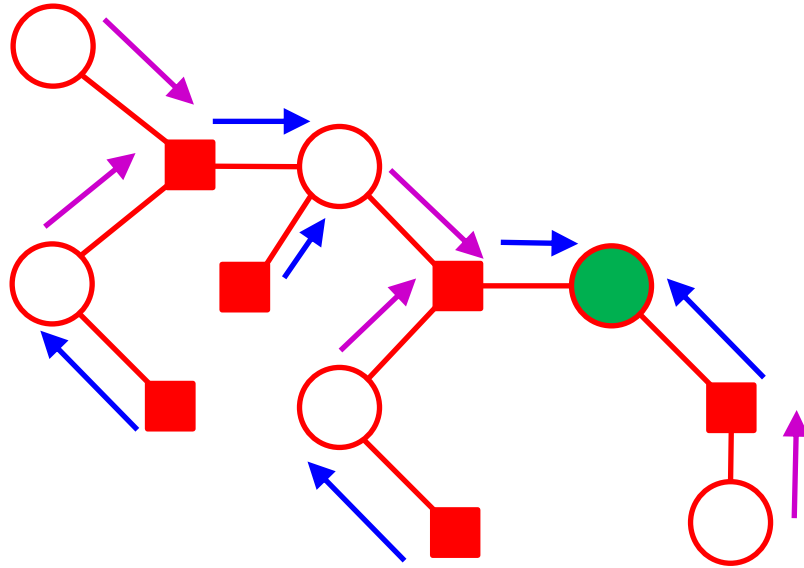


- We choose x_n as a **tree root**
- We start from the leaves



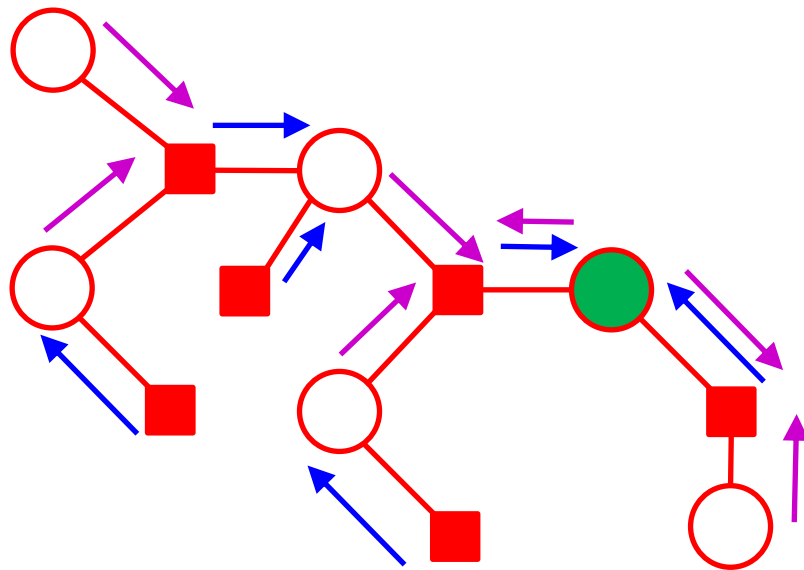
- Once a node obtains messages from all its children, it sends message to its parent.
- When root obtains all the messages, $P(x_n)$ is evaluated

BP solving all marginals at once



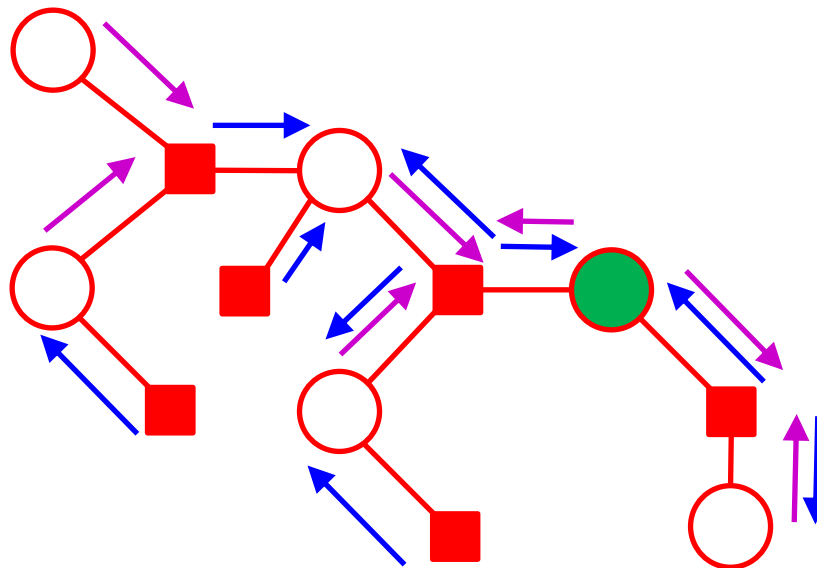
- Once root obtains all the messages, we can keep sending messages from the root towards to the leaves
- This way, all variable nodes obtain messages from the neighboring factor nodes
 - ➔ we can efficiently calculate marginals $P(x_n)$ for all the variables
 - It takes only 2x more time than calculating marginal for one node
 - ➔ the same computational complexity
 - It does not matter which node is selected as the root

BP solving all marginals at once



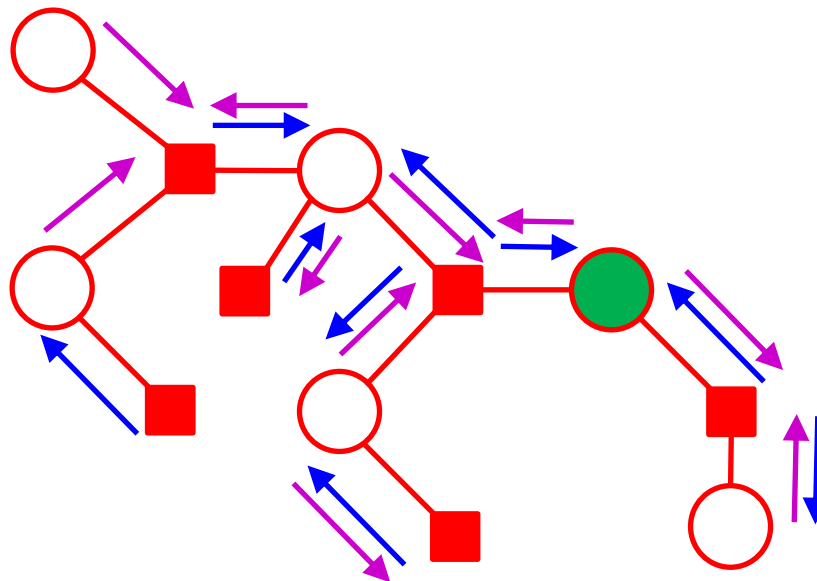
- Once root obtains all the messages, we can keep sending messages from the root towards to the leaves
- This way, all variable nodes obtain messages from the neighboring factor nodes
 - ➔ we can efficiently calculate marginals $P(x_n)$ for all the variables
 - It takes only 2x more time than calculating marginal for one node
 - ➔ the same computational complexity
 - It does not matter which node is selected as the root

BP solving all marginals at once



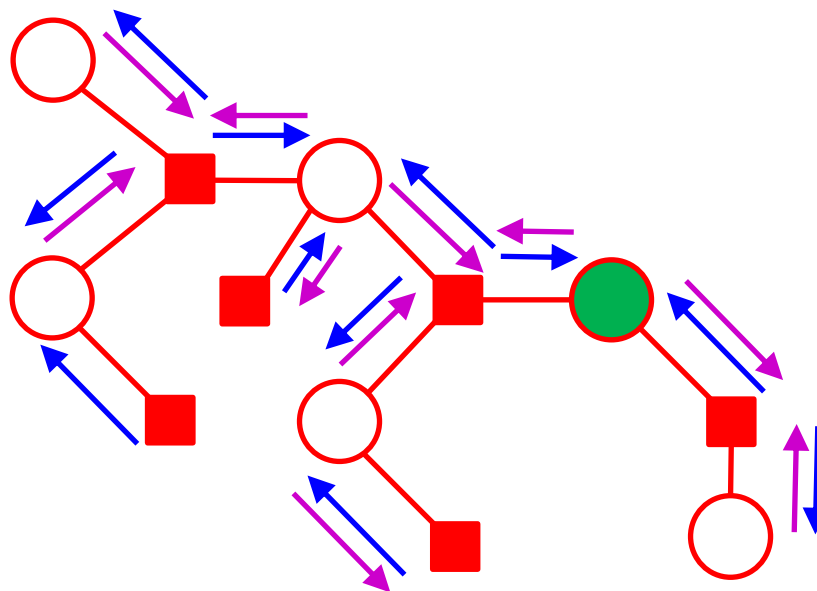
- Once root obtains all the messages, we can keep sending messages from the root towards to the leaves
- This way, all variable nodes obtain messages from the neighboring factor nodes
 - ➔ we can efficiently calculate marginals $P(x_n)$ for all the variables
 - It takes only 2x more time than calculating marginal for one node
 - ➔ the same computational complexity
 - It does not matter which node is selected as the root

BP solving all marginals at once



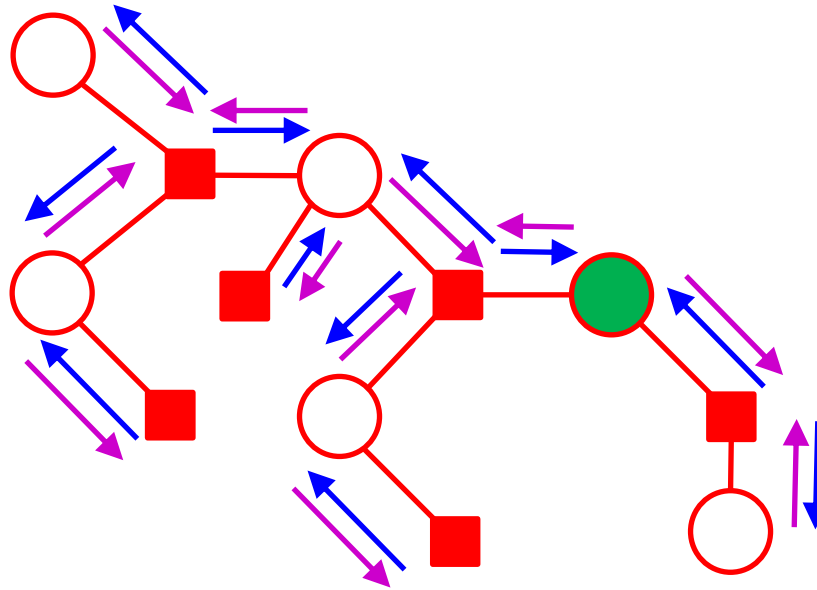
- Once root obtains all the messages, we can keep sending messages from the root towards to the leaves
- This way, all variable nodes obtain messages from the neighboring factor nodes
 - ➔ we can efficiently calculate marginals $P(x_n)$ for all the variables
 - It takes only 2x more time than calculating marginal for one node
 - ➔ the same computational complexity
 - It does not matter which node is selected as the root

BP solving all marginals at once



- Once root obtains all the messages, we can keep sending messages from the root towards to the leaves
- This way, all variable nodes obtain messages from the neighboring factor nodes
 - ➔ we can efficiently calculate marginals $P(x_n)$ for all the variables
 - It takes only 2x more time than calculating marginal for one node
 - ➔ the same computational complexity
 - It does not matter which node is selected as the root

BP solving all marginals at once



- Once root obtains all the messages, we can keep sending messages from the root towards to the leaves
- This way, all variable nodes obtain messages from the neighboring factor nodes
 - ➔ we can efficiently calculate marginals $P(x_n)$ for all the variables
 - It takes only 2x more time than calculating marginal for one node
 - ➔ the same computational complexity
 - It does not matter which node is selected as the root

BP with observed variables

- So far, we have assumed that all the variables are unobserved.
- We can also calculate **marginal probability of any x_n given a set observed of variables \mathbf{x}_C** :

$$P(x_n|\mathbf{x}_C) \propto P(x_n, \mathbf{x}_C) = \sum_{\mathbf{x} \setminus \{x_n, \mathbf{x}_C\}} P(\mathbf{x}) \propto \sum_{\mathbf{x} \setminus \{x_n, \mathbf{x}_C\}} \prod_s f_s(\mathbf{x}_s)$$

- In our example with 6 random variables, we may wish to evaluate:

$$P(x_1|x_3, x_4) \propto P(x_1, x_3, x_4) = \sum_{x_2} \sum_{x_5} \sum_{x_6} P(x_1, x_2, x_3, x_4, x_5, x_6)$$

where the observed variables x_3, x_4 have known fixed values.

- We solve the same problem as before for marginals $P(x_n)$, except that we do not sum over the observed variables → We can use the same efficient BP algorithm except that the sums are removed for the observed variables and the factors are evaluated with the given values of the observed variables
- Make sure that the resulting $P(x_1|x_3, x_4)$ is properly normalized distribution!

Marginal distribution of set of variables

- We can also easily calculate **joint marginal distribution** for subset of variables \mathbf{x}_s belonging to one factor f_s (e.g. $P(x_1, x_2, x_3)$ in our example):

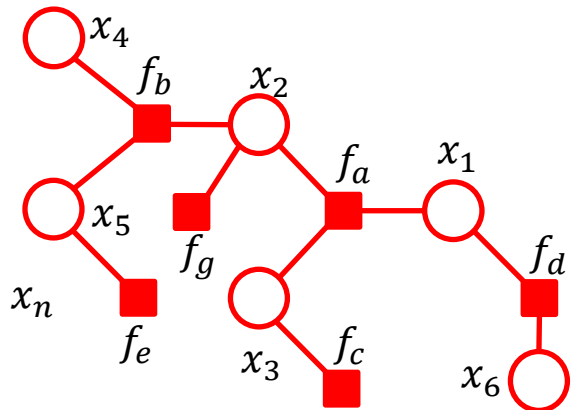
$$P(\mathbf{x}_s) = \frac{1}{Z} f_s(\mathbf{x}_s) \prod_{i \in \text{ne}(f_s)} \mu_{x_i \rightarrow f_s}(x_i)$$

- We cannot easily obtain joint marginal distribution for variables \mathbf{x}_C **not** belonging to one factor (e.g. $P(x_4, x_5, x_6)$ in our example).
- But we can use BP to efficiently **evaluate such distribution for particular (observed) values of the variables \mathbf{x}_C** (i.e. we can evaluate $P(x_4, x_5, x_6)$ for particular given values of x_4, x_5, x_6).
- Here, we require that the factor graph is created from a Bayesian Network \rightarrow the factors corresponds to well normalized distributions and therefore $Z = 1$
 - See the slide “*Example: HMM as MRF*” as an example

$$P(\mathbf{x}_C) = \sum_{\mathbf{x} \setminus \mathbf{x}_C} P(\mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{x} \setminus \mathbf{x}_C} \prod_s f_s(\mathbf{x}_s)$$

- BP algorithm can be used to efficiently evaluate the right hand side for fixed values of \mathbf{x}_C . Again, we do not sum over possible values of \mathbf{x}_C in BP. Instead, we directly use the given fixed values of \mathbf{x}_C , when evaluating the factors.
- Choosing any $x_n \in \mathbf{x}_C$ as the “root” for BP, we can calculate $P(\mathbf{x}_C)$ as the product of the incoming messages evaluated at x_n

$$P(\mathbf{x}_C) = \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$



Most likely values

- Belief Propagation allows us to calculate the marginals $p(x_n)$, so for each variable x_n , we can find the value that is *individually* the most probable

$$x_n^{\max} = \arg \max_{x_n} p(x_n)$$

- Instead, we may want to know what are values of all the variables $\mathbf{x} = [x_1, x_2, \dots, x_N]$ that have *jointly* the largest probability

$$\mathbf{x}^{\max} = \arg \max_{\mathbf{x}} p(\mathbf{x})$$

- An example where individually and jointly most probable values are different:
 - Individually, the most likely values are $x = 1$ and $y = 1$ as $P(x) = P(y) = 0.38$ are the largest marginal probabilities
 - But it is impossible that $x = 1$ and $y = 1$ at the same time as $P(x = 1, y = 1) = 0$
 - Jointly, the most likely values are $x = 4$ and $y = 4$ as $P(x = 4, y = 4) = 0.24$ is the largest joint probability

$P(x, y)$	$x = 1$	$x = 2$	$x = 3$	$x = 4$		$P(y)$
$y = 1$	0.0	0.19	0.19	0.0		0.38
$y = 2$	0.19	0.0	0.0	0.0		0.19
$y = 3$	0.19	0.0	0.0	0.0		0.19
$y = 4$	0.0	0.0	0.0	0.24		0.24

$P(x)$	0.38	0.19	0.19	0.24
--------	------	------	------	------

Max-product algorithm

- Sum-product algorithm (or Belief Propagation) allowed us to find marginals

$$P(x_n) = \sum_{\mathbf{x} \setminus x_n} P(\mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{x} \setminus x_n} \prod_s f_s(\mathbf{x}_s)$$

efficiently by re-arranging the order of sums and products using the *distributive property of multiplication over addition*: $ab + ac = a(b + c)$

- Similar distributive property holds also max operator: $\max(ab, ac) = a \max(b, c)$ which allows for the same efficient calculation of the largest joint probability

$$P(\mathbf{x}^{\max}) = \max_{\mathbf{x}} P(\mathbf{x}) = \frac{1}{Z} \max_{\mathbf{x}} \prod_s f_s(\mathbf{x}_s)$$

- We use the same BP where *sum* is replaced with *max* in the messages from factor nodes to variable nodes

$$\mu_{f_s \rightarrow x_n}(x_n) = \max_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

- The solution is the maximum of the product of the messages arriving to the root node evaluated for the possible values of the “root” variable x_n

$$ZP(\mathbf{x}^{\max}) = \max_{x_n} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

Max-product algorithm II

- The solution on the previous slide is the (unnormalized unless $Z = 1$) maximum of the joint distribution $P(\mathbf{x}^{\max})$

- However, we usually need the most likely values

$$\mathbf{x}^{\max} = \arg \max_{\mathbf{x}} P(\mathbf{x})$$

- Solution: whenever max operator is applied during the max-product algorithm, remember the variable values giving the maximum

- When calculating message from factor node to variable node

$$\mu_{f_s \rightarrow x_n}(x_n) = \max_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

for each x_n , remember the most likely values of the other factor's variables $\mathbf{x}_s \setminus x_n$

$$\phi_{f_s}(x_n) = \arg \max_{\mathbf{x}_s \setminus x_n} f_s(\mathbf{x}_s) \prod_{m \in \text{ne}(f_s) \setminus x_n} \mu_{x_m \rightarrow f_s}(x_m)$$

- When calculating

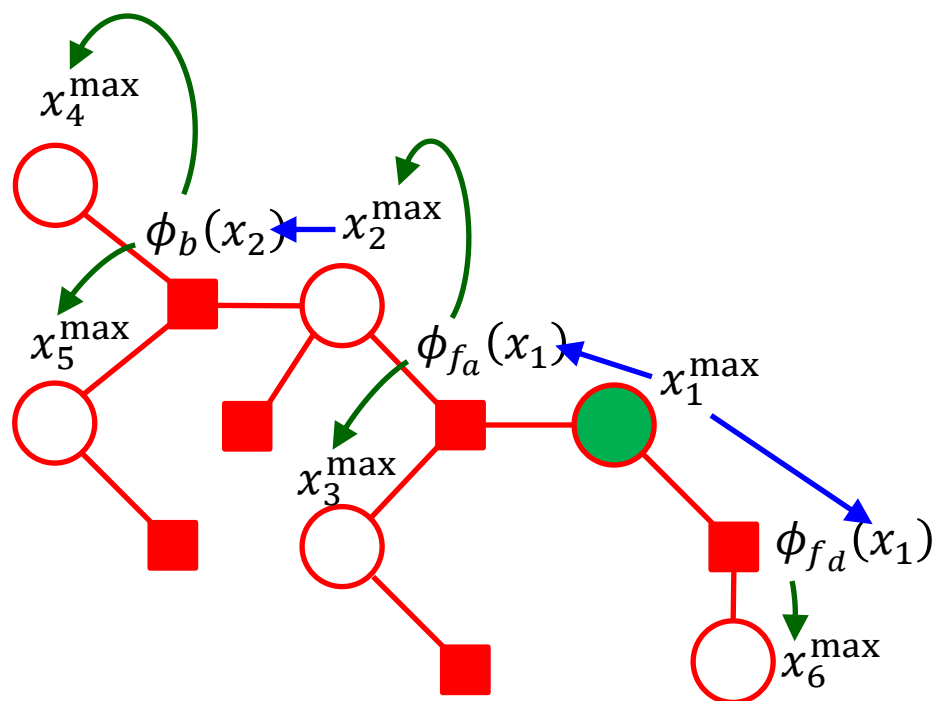
$$ZP(\mathbf{x}^{\max}) = \max_{x_n} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

obtain also the most likely setting of the “root” variable

$$x_n^{\max} = \arg \max_{x_n} \prod_{s \in \text{ne}(x_n)} \mu_{f_s \rightarrow x_n}(x_n)$$

Max-product algorithm - backtracking

- Now, we have the most likely value for the root x_n^{\max}
- We back-propagate x_n^{\max} to each neighboring factor node f_s
- Here, we evaluate the stored $\phi_{f_s}(x_n)$ for x_n^{\max} , which gives us the most likely values of the other factor's variables $\mathbf{x}_s \setminus x_n$
- We back-propagate these new “most likely values” to the next factor nodes f_s further away from the root to evaluate their functions $\phi_{f_s}(x_n)$
- This is repeated until we obtain the most likely values for all the variables \mathbf{x}^{\max}



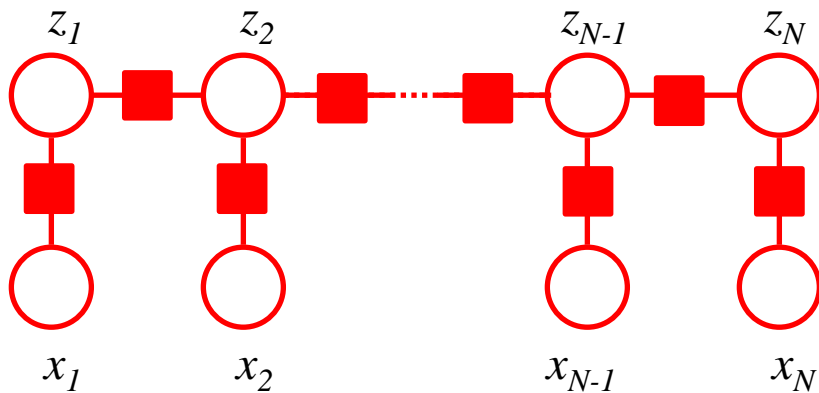
Example: BP for HMM

- To evaluate an HMM, given a sequence of observations $\mathbf{X} = [x_1, x_2, \dots, x_N]$, we need to infer

$$p(\mathbf{X}) = p(x_1, x_2, \dots, x_N) = \sum_{z_1} \sum_{z_2} \dots \sum_{z_N} p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N)$$

- To train an HMM using an EM algorithm (see next lesson), for every $t = 1..N$, we need to infer

$$p(z_t | \mathbf{X}) = \frac{p(z_t, \mathbf{X})}{p(\mathbf{X})} = \frac{\sum_{z_1} \sum_{z_2} \dots \sum_{z_{t-1}} \sum_{z_{t+1}} \dots \sum_{z_N} p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N)}{p(\mathbf{X})}$$



Forward-backward algorithm

s are state ids (i.e., possible values of z_t)

$$\alpha(t, s) = p(\mathbf{x}_t | s) \sum_{s'} \alpha(t-1, s') p(s | s')$$

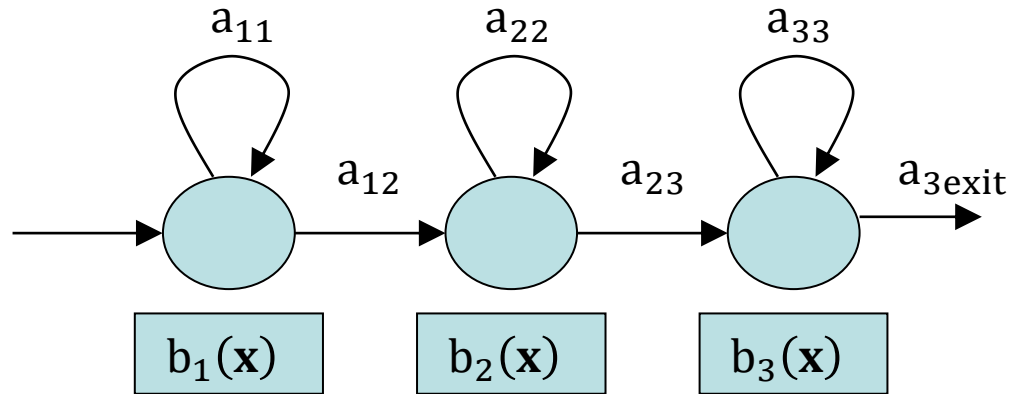
$$\beta(t, s) = \sum_{s'} \beta(t+1, s') p(\mathbf{x}_{t+1} | s') p(s' | s)$$

$$p(\mathbf{X}) = \sum_{s' \in \text{FinalStates}} \alpha(N, s')$$

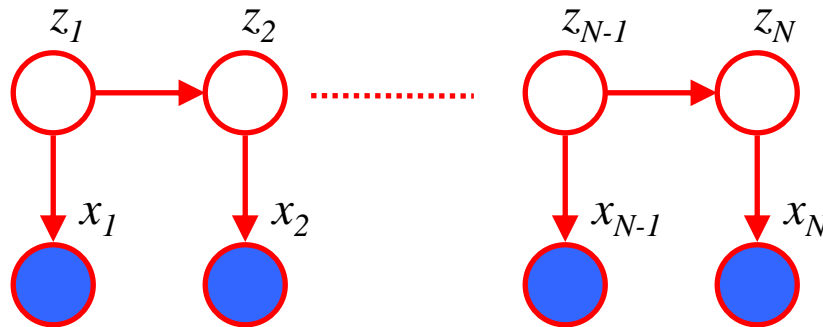
$$p(z_t = s | \mathbf{X}) = \frac{\alpha(t, s) \beta(t, s)}{P(\mathbf{X})}$$

(Dynamic) BN for HMM - flashback

- For each time frame, Hidden Markov Model moves from state j to state k according to a transition probability $a_{jk} = p(k|j)$ and generates observation \mathbf{x} from probability distribution $b_k(\mathbf{x}) = p(\mathbf{x}|k)$ associated with the entered state. More details on this model for *modeling sequences* are in SUR class.



- In BN, z_i nodes are not “HMM states”, these are random variables (one for each frame) with values saying which state we are in for a particular frame i



$$p(x_1, x_2, \dots, x_N, z_1, z_2, \dots, z_N) = P(z_1) \prod_{i=2}^N p(z_i|z_{i-1}) \prod_{i=1}^N p(x_i|z_i)$$

Inference in non-tree graphs

- Junction tree algorithm
 - Exact inference in general graphs (not only trees or polytrees)
 - Belief propagation on a modified graph where cycles are eliminated by merging nodes into single nodes
 - Usually too computationally expensive and impractical
- Loopy Belief Propagation
 - Initialize Sum-Product algorithm so that each node has already “messages from all neighbors”
 - these are only randomly initialized vectors, not real messages sent from the neighbors
 - Start sending messages like in the Sum-Product algorithm
 - Choose some *message passing schedule*
 - Send messages from *pending nodes* that received new messages and updated their states
 - Approximate inference, no guarantee to converge