Graphical Models: Introduction

Thomas Hamelryck, April 2015
Bioinformatics center
Department of Biology
University of Copenhagen

Some history (1)

- Expert systems were developed in the 60s-70s and commercialized in the 80s.
- Expert system=knowledge base+inference engine
- Typical example application is disease diagnosis based on a list of symptoms.
- Expert systems are based on “IF ... THEN ...” rules:
  - Decision trees
  - Rule based or production systems (PROLOG, LISP)

Some history (1a)

- Example

  IF the animal has hair THEN it is a mammal

  IF the animal gives milk THEN it is a mammal

  IF the animal lays eggs and flies THEN it is a bird

Some history (2)

- Expert systems use logical deduction
  - If A is true, then B is true
    - A is true, therefore, B is true
    - B is false, therefore A is false
- A single unexpected or absent symptom can corrupt disease diagnosis completely.
- Difficult to deal with unobserved variables (for example, blood pressure that was not measured).
- The set of rules can become huge.

Some history (3)

- Expert systems do not use logical induction
  - If A is true, then B is true
    - B is true, therefore, A becomes more plausible
    - A is false, therefore, B becomes less plausible
- Rule based systems need to cope with degrees of uncertainty.
  - The AI community tried various solutions. Each rule is associated with a certainty factor (CF). These factors are combined according to a certain algebra.
    - fuzzy logic, belief functions,...

Some history (3a)

Pierre-Simon Laplace
(1749-1827)

- Example

  IF headache, fever THEN influenza (CF=0.7)

  IF influenza THEN sneezing (CF=0.9)

  IF influenza THEN weakness (CF=0.6)

- CF algebra
  - What is CF(influenza | sneezing, no headache)?
  - These *ad hoc* algebra's were proven to be inconsistent
    - Only Bayesian probability will do if we follow some elementary desiderata (Richard T. Cox, 1946,1961; Edwin T. Jaynes, 2003, see Bishop, p. 21)

Some history (4)

- Despite this, using probabilities as certainty factor was seen as problematic, mainly because
  - According to the frequentist interpretation, probabilities are essentially frequencies in a large number of trials. Often this interpretation is meaningless in expert systems.
  - Full probability distributions over many variables are intractable.
- Then came Judea Pearl's “Probabilistic inference in intelligent systems”, 1988
  - Bayesian interpretation of probability
  - Efficient local computations on graphs

Background references

- Probability theory
  - *Probability theory – the logic of science.* (2003) Edwin T. Jaynes (Cambridge university press)
  - *The algebra of probable inference.* (1961) Richard T. Cox (Johns Hopkins univ. press)
- Expert systems & BNs
  - *Probabilistic inference in intelligent systems.* (1988) Judea Pearl (Morgan Kauffman)
  - *Probabilistic networks and expert systems.* (1999) Cowell, Dawid, Lauritzen, Spiegelhalter (Springer)
  - *Bayesian networks and decision graphs*. (2007) Finn V. Jensen, Thomas D. Nielsen (Springer)

Some preliminaries (1)

- Conditional probability tables (CPTs)
  - Discrete random variables with finite number of states
  - Probability of one variable conditional on one or more variables
    - Example: P(car ownership|size of income)

	No car	Second hand car	New car
Low income	0.2	0.4	0.4
High income	0.1	0.3	0.6

Some preliminaries (2)

- Discrete random variables with finite range
  - K possible states
  - 1-of-K representation
    - **x** is a K-dimensional vector of binary indicators
    - Example
      - **x**=(0,0,1,0) indicates the third state (out of K=4)
  - If the probabilities of the K states are $(\mu_1,\mu_2,\ldots,\mu_K)$ then:

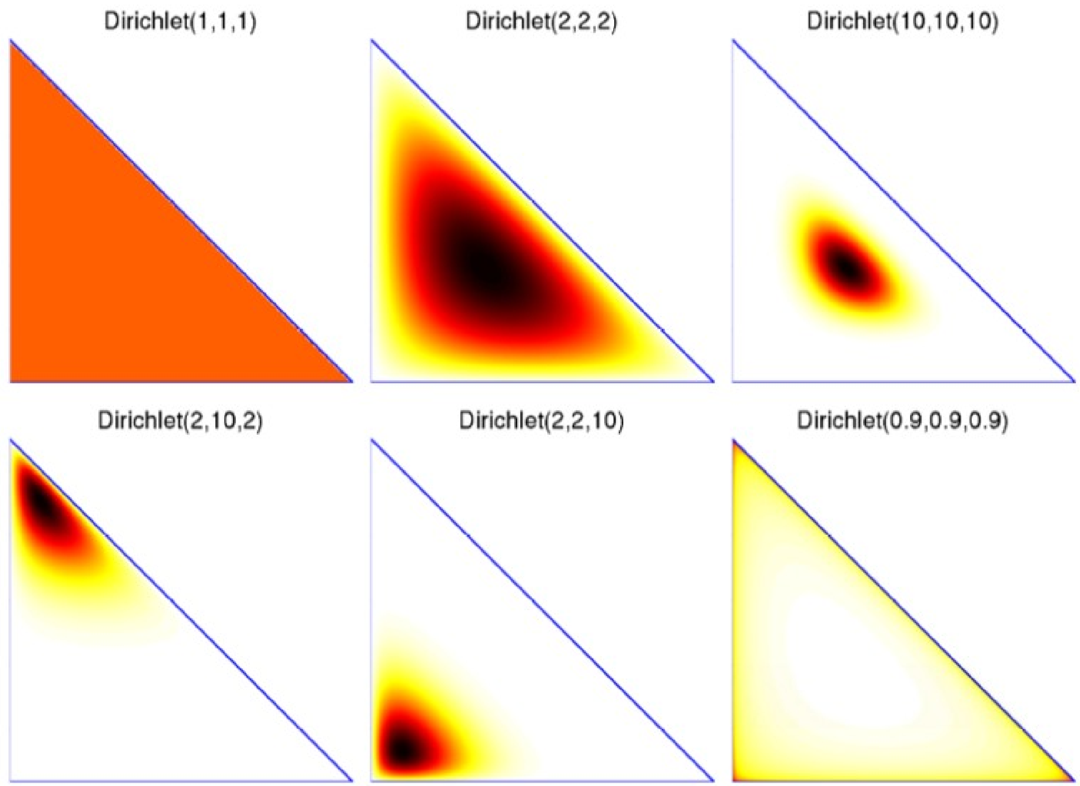
$$P(\mathbf{x}|\boldsymbol{\mu})=\prod_{k=1}^{K}\mu_k^{x_k}$$

Some preliminaries (3a)

- Dirichlet distribution $$P(\mu_1,\dots,\mu_K|\alpha_1,\dots,\alpha_K)=\frac{1}{C(\boldsymbol{\alpha})}\prod_{k=1}^{K}\mu_k^{\alpha_k-1}$$
  - Parameter $\boldsymbol{\alpha}$
    - K-dimensional vector of reals>0
  - Probability distribution over vectors $\boldsymbol{\mu}$
    - K-dimensional
    - Positive components that sum to one
    - Probability vector
  - Example of a probability distribution on a “special” manifold
    - K-dimensional simplex
      - Generalization of a triangle

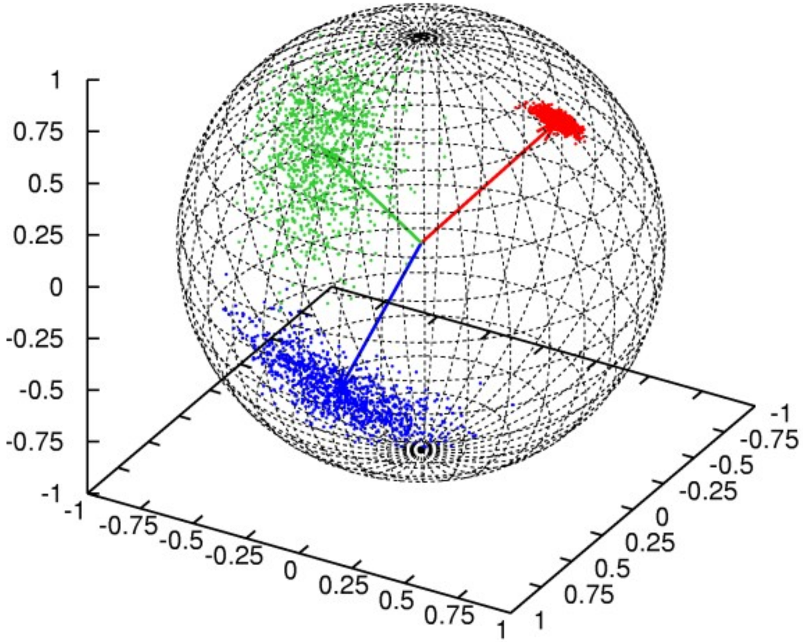
Some preliminaries (3b)

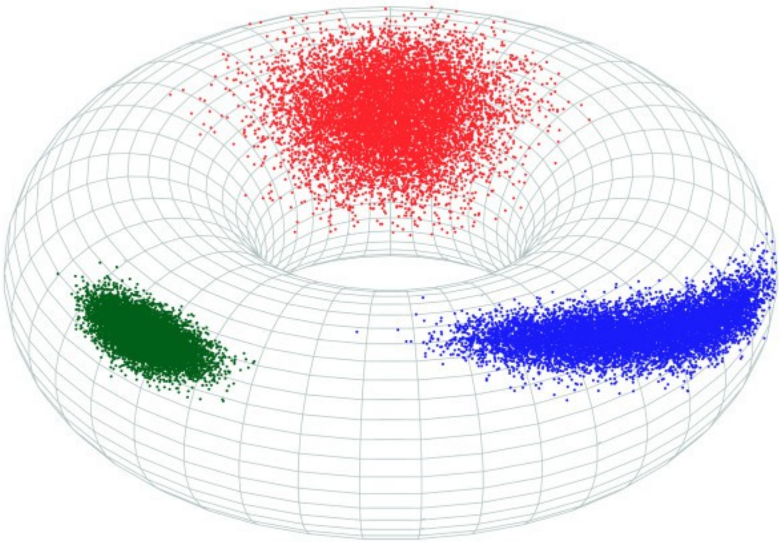
- Dirichlet distributions with K=3



Some preliminaries (4)

- Distributions on the sphere (S^2) and the torus (T^2)
  - Probabilistic models of protein structure



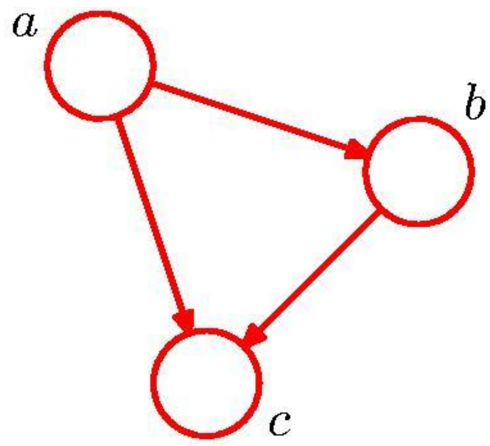


PATTERN RECOGNITION AND MACHINE LEARNING

CHAPTER 8: GRAPHICAL MODELS

Bayesian Networks

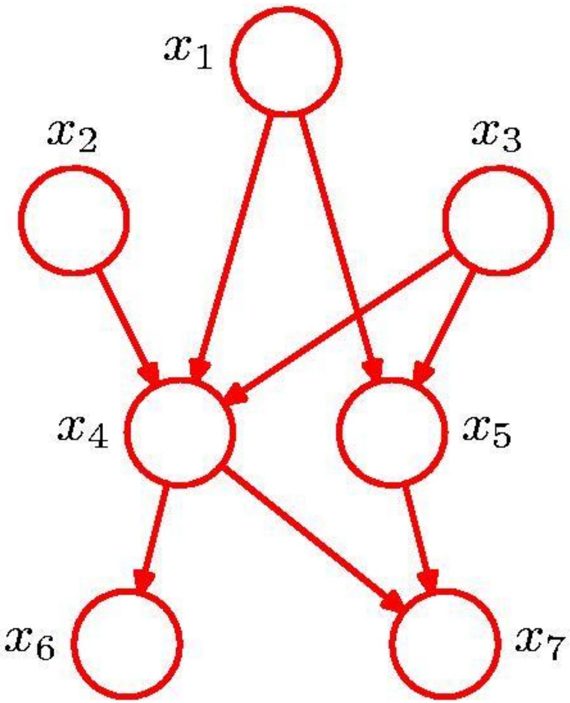
Directed Acyclic Graph (DAG)



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

Product rule: $p(x_1, \ldots, x_K) = p(x_K|x_1, \ldots, x_{K-1}) \ldots p(x_2|x_1)p(x_1)$

Bayesian Networks

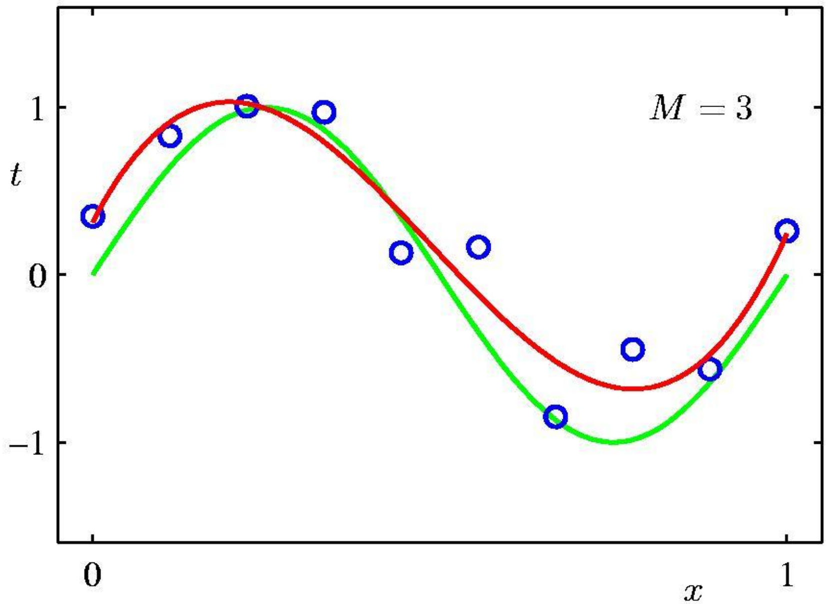


$$
\begin{aligned}
p(x_1, \ldots, x_7) \quad = \quad & p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3) \\
& p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)
\end{aligned}
$$

General Factorization

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

Bayesian Curve Fitting (1)



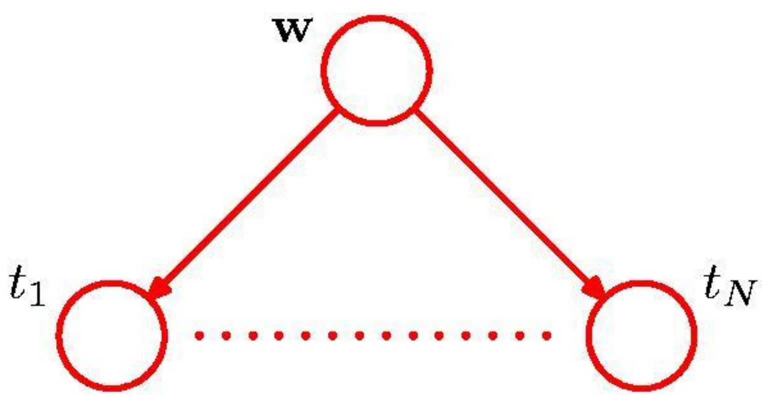
Polynomial

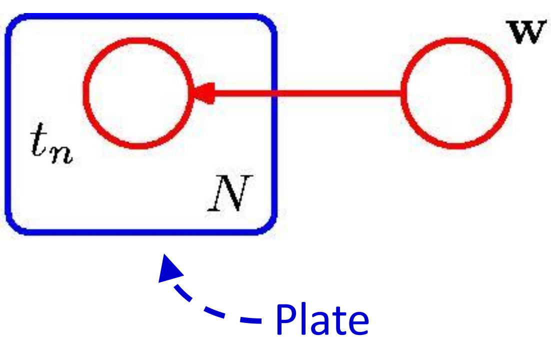
$$y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j$$

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^{N} p(t_n | y(\mathbf{w}, x_n))$$

Bayesian Curve Fitting (2)

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^{N} p(t_n | y(\mathbf{w}, x_n))$$

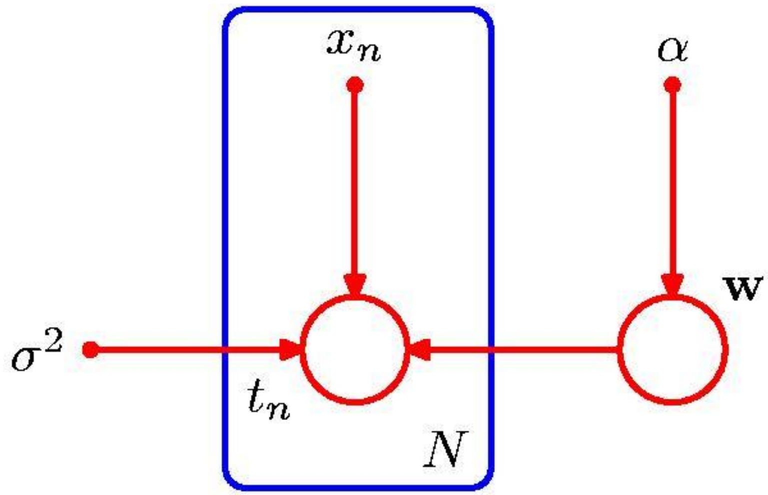




Bayesian Curve Fitting (3)

Input variables and explicit hyperparameters

$$p(\mathbf{t}, \mathbf{w}|\mathbf{x}, \alpha, \sigma^2) = p(\mathbf{w}|\alpha) \prod_{n=1}^{N} p(t_n|\mathbf{w}, x_n, \sigma^2).$$

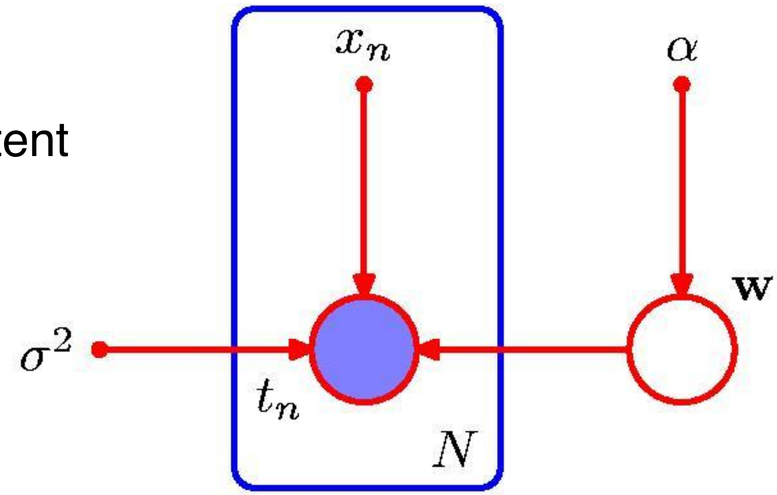


Bayesian Curve Fitting—Learning

Condition on data

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w}) \prod_{n=1}^{N} p(t_n|\mathbf{w}) \qquad \text{(Bayes)}$$

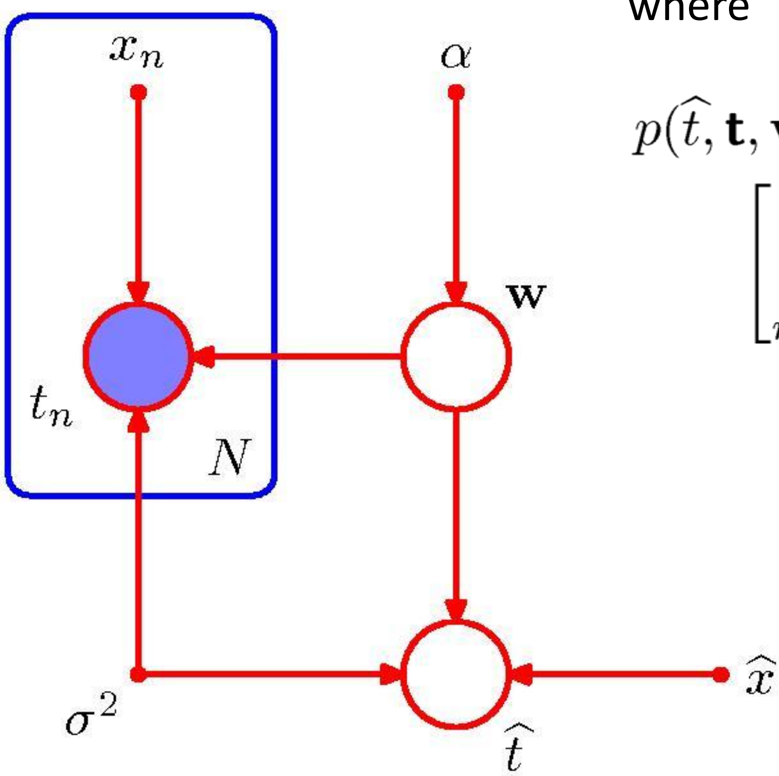
Shaded=observed
Not shaded=hidden, latent



Bayesian Curve Fitting—Prediction

(product rule)

Predictive distribution: $p(\widehat{t}|\widehat{x}, \mathbf{x}, \mathbf{t}, \alpha, \sigma^2) \propto \int p(\widehat{t}, \mathbf{t}, \mathbf{w}|\widehat{x}, \mathbf{x}, \alpha, \sigma^2)\,\mathrm{d}\mathbf{w}$

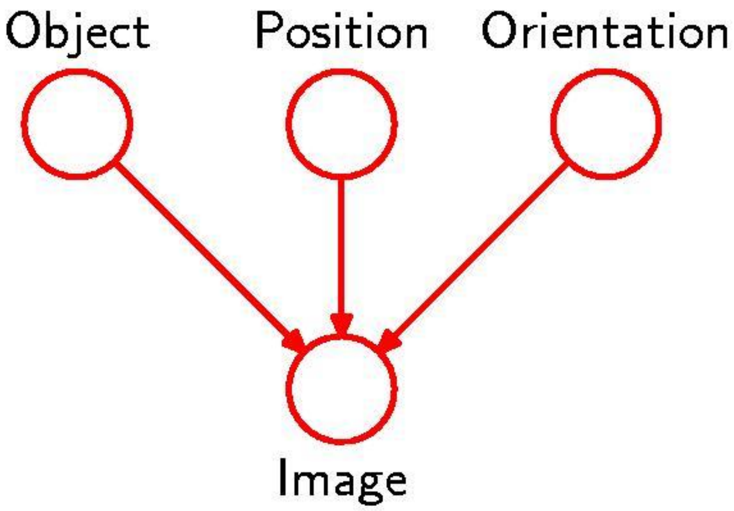


where

$$p(\widehat{t}, \mathbf{t}, \mathbf{w}|\widehat{x}, \mathbf{x}, \alpha, \sigma^2) = \left[\prod_{n=1}^{N} p(t_n|x_n, \mathbf{w}, \sigma^2)\right] p(\mathbf{w}|\alpha) p(\widehat{t}|\widehat{x}, \mathbf{w}, \sigma^2)$$

Generative Models

Causal process for generating images



Ancestral sampling:
p. 365, Bishop

Discrete Variables (1)

General joint distribution: $K^2 - 1$ parameters

$$p(\mathbf{x}_1, \mathbf{x}_2 | \boldsymbol{\mu}) = \prod_{k=1}^{K} \prod_{l=1}^{K} \mu_{kl}^{x_{1k} x_{2l}}$$

Independent joint distribution: $2(K-1)$ parameters

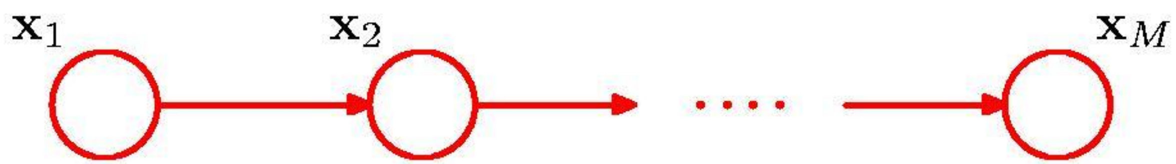
$$\hat{p}(\mathbf{x}_1, \mathbf{x}_2 | \boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_{1k}^{x_{1k}} \prod_{l=1}^{K} \mu_{2l}^{x_{2l}}$$

Independence assumptions lead to fewer parameters

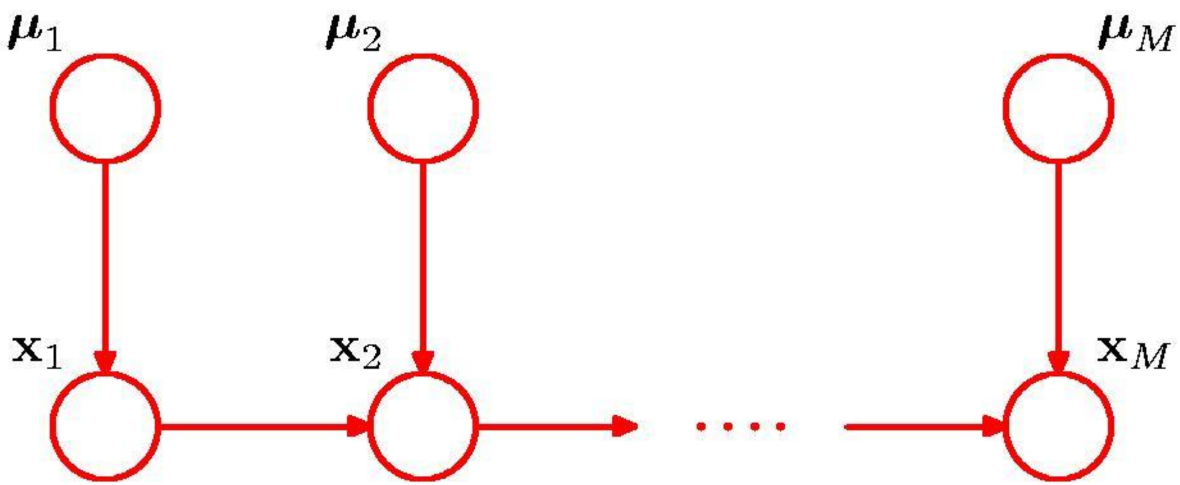
Discrete Variables (2)

General joint distribution over M variables:
$K^M - 1$ parameters

M-node Markov chain: $K - 1 + (M - 1)K(K - 1)$ parameters



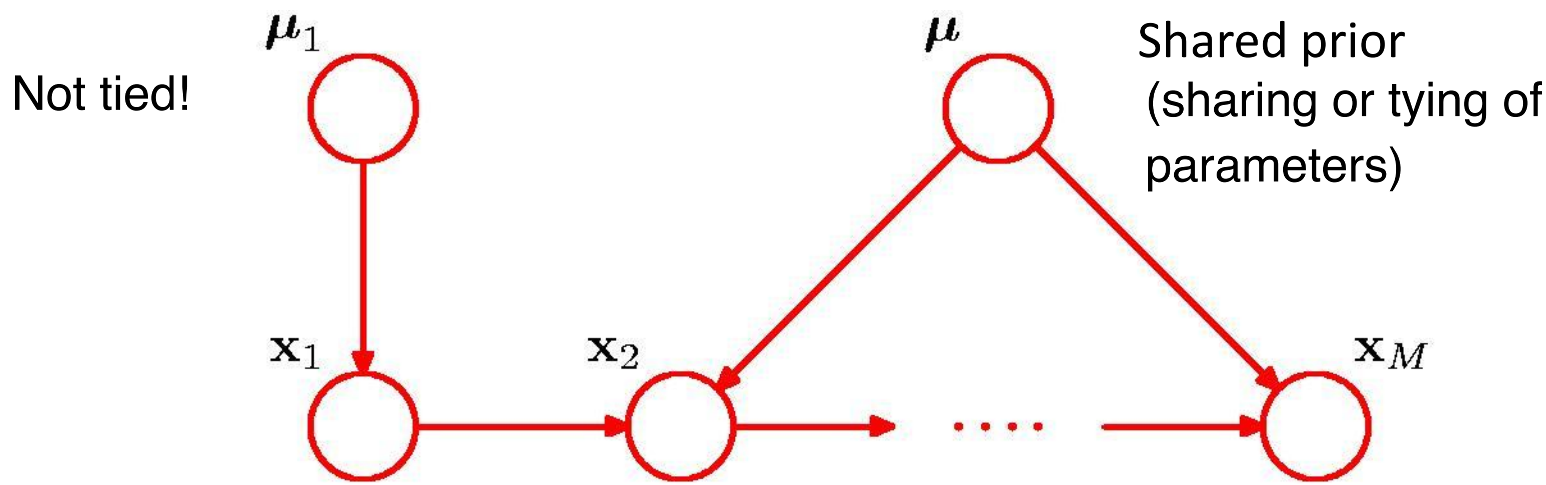
Discrete Variables: Bayesian Parameters (1)



$$p(\{\mathbf{x}_m, \boldsymbol{\mu}_m\}) = p(\mathbf{x}_1 | \boldsymbol{\mu}_1) p(\boldsymbol{\mu}_1) \prod_{m=2}^{M} p(\mathbf{x}_m | \mathbf{x}_{m-1}, \boldsymbol{\mu}_m) p(\boldsymbol{\mu}_m)$$

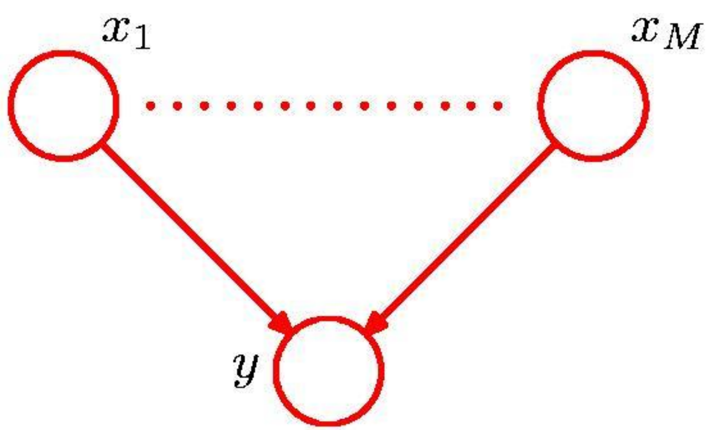
$$p(\boldsymbol{\mu}_m) = \mathrm{Dir}(\boldsymbol{\mu}_m | \boldsymbol{\alpha}_m)$$

Discrete Variables: Bayesian Parameters (2)



$$p\left(\{\mathbf{x}_m\},\boldsymbol{\mu}_1,\boldsymbol{\mu}\right)=p\left(\mathbf{x}_1\,|\boldsymbol{\mu}_1\right)p\left(\boldsymbol{\mu}_1\right)\prod_{m=2}^{M}p\left(\mathbf{x}_m|\mathbf{x}_{m-1},\boldsymbol{\mu}\right)p\left(\boldsymbol{\mu}\right)$$

Parameterized Conditional Distributions



If $x_1, \ldots, x_M$ are discrete, K-state variables, $p(y = 1 | x_1, \ldots, x_M)$ in general has $O(K^M)$ parameters.

The parameterized form

$$p(y = 1 | x_1, \ldots, x_M) = \sigma\left(w_0 + \sum_{i=1}^{M} w_i x_i\right) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x})$$

requires only $M + 1$ parameters

logistic sigmoid=1/(1+exp(-a))

Linear-Gaussian Models

Directed Graph

Mean Standard deviation

$$p(x_i|\mathrm{pa}_i) = \mathcal{N}\left(x_i \,\middle|\, \sum_{j \in \mathrm{pa}_i} w_{ij} x_j + b_i, v_i\right)$$

Each node is Gaussian, the mean is a linear function of the parents.

Vector-valued Gaussian Nodes

Covariance Matrix

$$p(\mathbf{x}_i|\mathrm{pa}_i) = \mathcal{N}\left(\mathbf{x}_i \,\middle|\, \sum_{j \in \mathrm{pa}_i} \mathbf{W}_{ij} \mathbf{x}_j + \mathbf{b}_i, \boldsymbol{\Sigma}_i\right)$$

Conditional Independence

a is independent of b given c

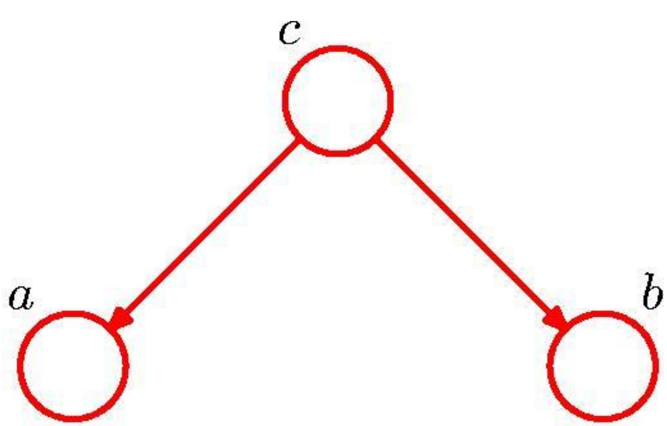
$$p(a|b,c) = p(a|c)$$

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

Notation
$$a \perp\!\!\!\perp b \mid c$$

Bayesian networks encode conditional independences

Conditional Independence: Example 1



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

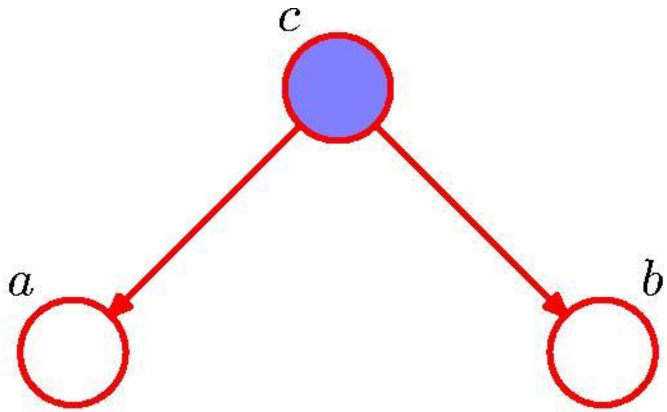
$$p(a, b) = \sum_c p(a|c)p(b|c)p(c)$$

$$a \not\perp\!\!\!\perp b \mid \emptyset$$

Path is "unblocked".

Conditional Independence: Example 1

c is tail-to-tail

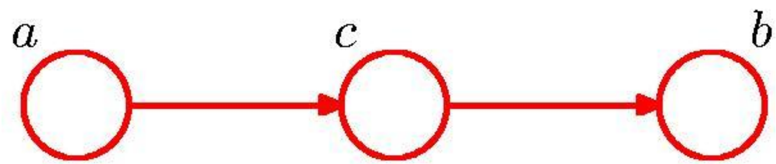


$$p(a, b|c) = \frac{p(a, b, c)}{p(c)}$$
$$= p(a|c)p(b|c)$$

$$a \perp\!\!\!\perp b \mid c$$

Observing c "blocks" the path
and makes a and b independent

Conditional Independence: Example 2



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

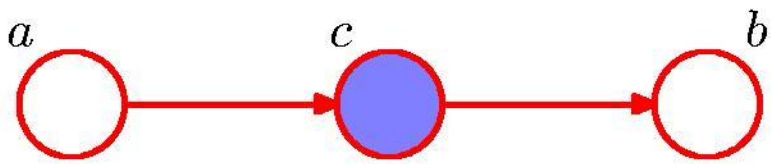
$$p(a, b) = p(a) \sum_c p(c|a)p(b|c) = p(a)p(b|a)$$

$$a \not\perp\!\!\!\perp b \mid \emptyset$$

Path is "unblocked".

Conditional Independence: Example 2

c is head-to-tail

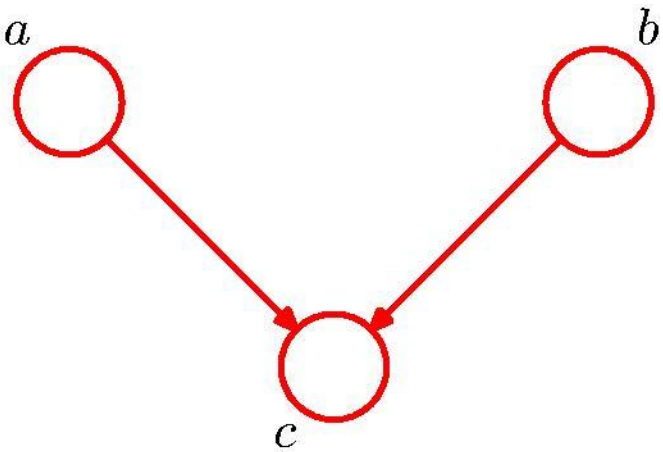


$$
\begin{aligned}
p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\
&= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\
&= p(a|c)p(b|c)
\end{aligned}
$$

Observing c "blocks" the path
and makes a and b independent

$$a \perp\!\!\!\perp b \mid c$$

Conditional Independence: Example 3



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

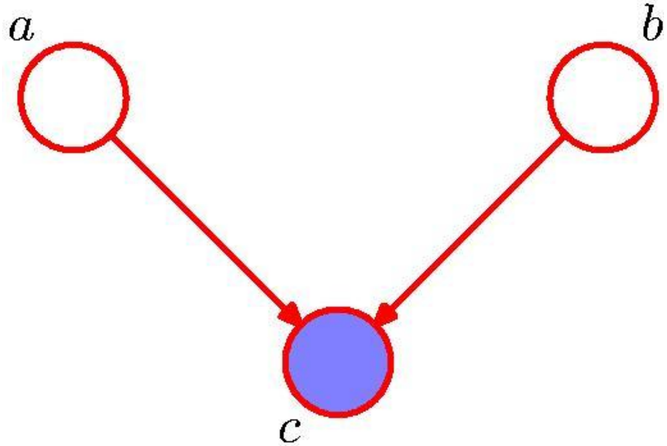
$$p(a, b) = p(a)p(b)$$

$$a \perp\!\!\!\perp b \mid \emptyset$$

Path is "blocked".

Note: this is the opposite of Example 1, with c unobserved.

Conditional Independence: Example 3



c is head-to-head

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

$$a \not\perp\!\!\!\perp b \mid c$$

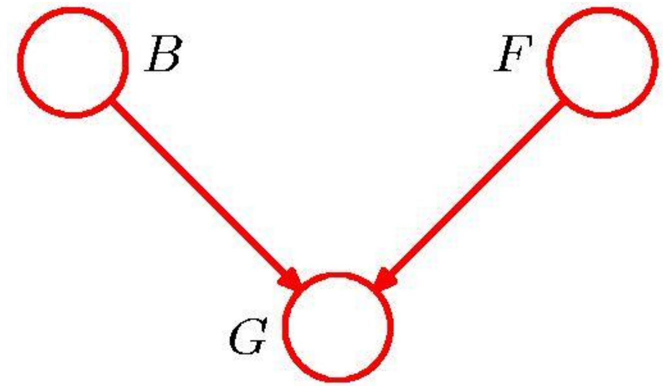
Observing c "unblocks" the path and makes a and b dependent

Note: this is the opposite of Example 1, with c observed.

"Am I out of fuel?"

A very lousy fuel gauge:

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



Priors:

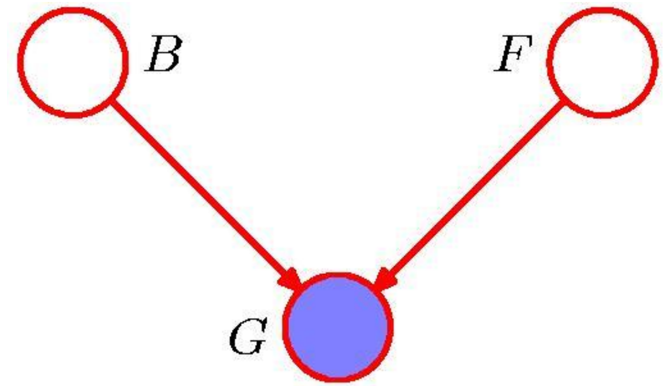
$$
\begin{aligned}
p(B=1) &= 0.9\\
p(F=1) &= 0.9
\end{aligned}
$$

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)

“Am I out of fuel?”

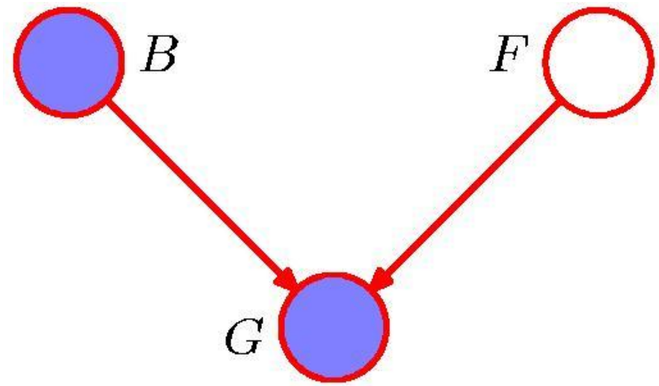


Probability of empty tank given gauge says its empty:

$$
\begin{aligned}
p(F=0|G=0) &= \frac{p(G=0|F=0)p(F=0)}{p(G=0)} \\
&\simeq 0.257
\end{aligned}
$$

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

“Am I out of fuel?”



Now also given that the battery is empty:

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

Decreased!

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

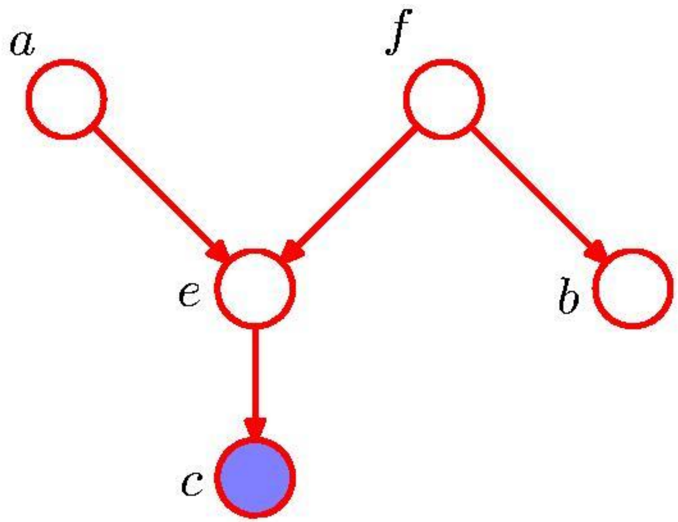
Fuel and battery are not independent because the gauge is observed.

Directed

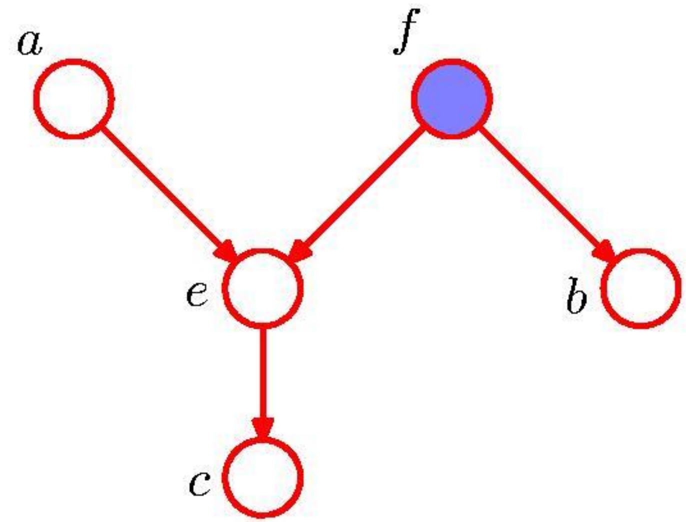
D-separation

- A, B, and C are non-intersecting subsets of nodes in a directed graph.
- A path from A to B is blocked if it contains a node such that either
  a) the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set C, or
  b) the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, are in the set C.
- If all paths from A to B are blocked, A is said to be d-separated from B by C.
- If A is d-separated from B by C, the joint distribution over all variables in the graph satisfies $A \perp\!\!\!\perp B \mid C$.

D-separation: Example



$a \not\perp\!\!\!\perp b \mid c$



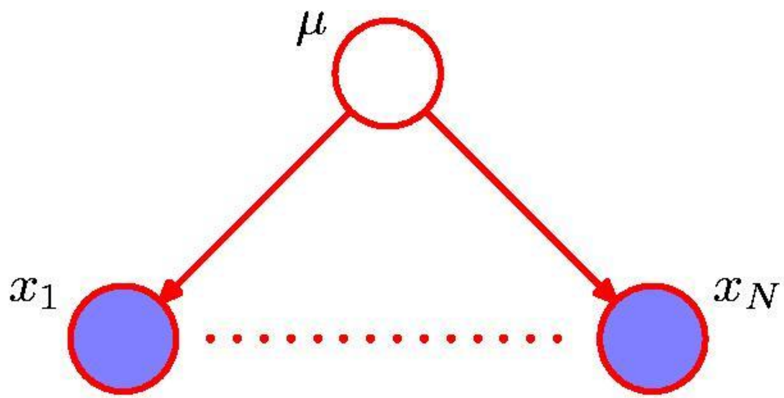
$a \perp\!\!\!\perp b \mid f$

What about I e?

What about I empty set?

D-separation: I.I.D. Data

Independent identically distributed



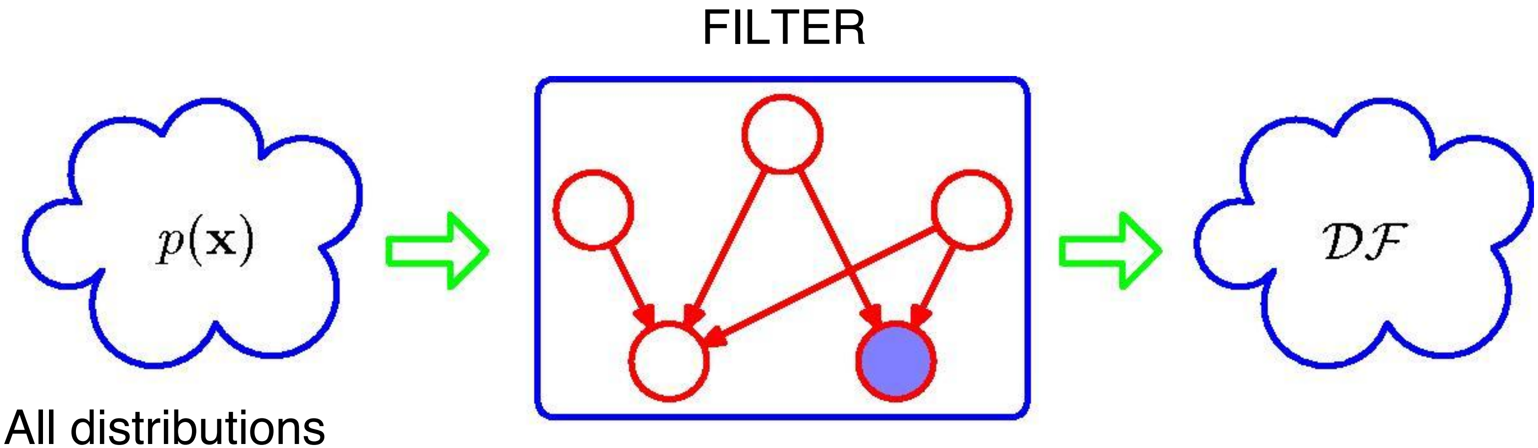
$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu)$$

d-separated by u

$$p(\mathcal{D}) = \int_{-\infty}^{\infty} p(\mathcal{D}|\mu)p(\mu)\,\mathrm{d}\mu \neq \prod_{n=1}^{N} p(x_n)$$

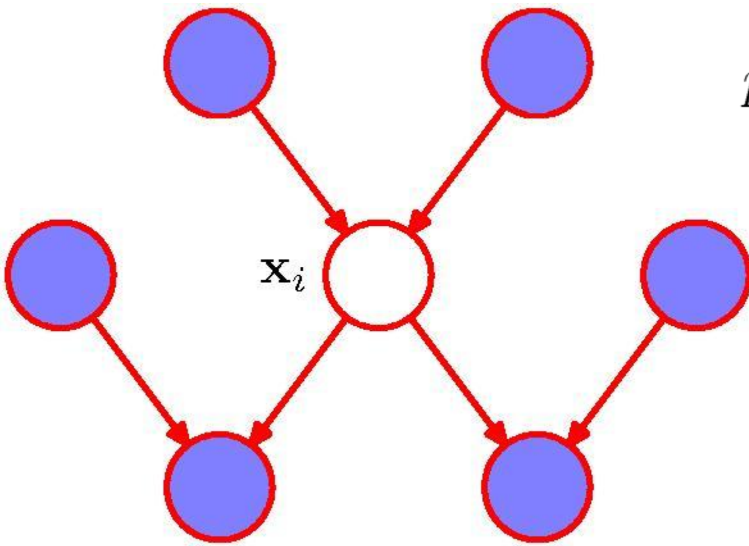
not d-separated by ø

Directed Graphs as Distribution Filters



Directed factorization (DF) properties
D-separation properties

The Markov Blanket

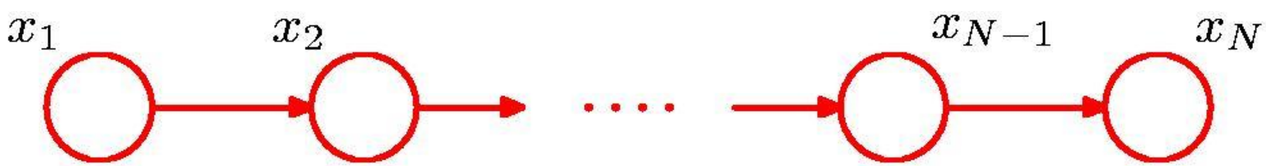


$$p(\mathbf{x}_i|\mathbf{x}_{\{j\neq i\}}) = \frac{p(\mathbf{x}_1,\ldots,\mathbf{x}_M)}{\int p(\mathbf{x}_1,\ldots,\mathbf{x}_M)\,\mathrm{d}\mathbf{x}_i} = \frac{\prod_k p(\mathbf{x}_k|\mathrm{pa}_k)}{\int \prod_k p(\mathbf{x}_k|\mathrm{pa}_k)\,\mathrm{d}\mathbf{x}_i}$$

Dependent only on:
Parents
Children
Parents of children

Factors independent of $\mathbf{x}_i$ cancel between numerator and denominator.

Converting Directed to Undirected Graphs (1)

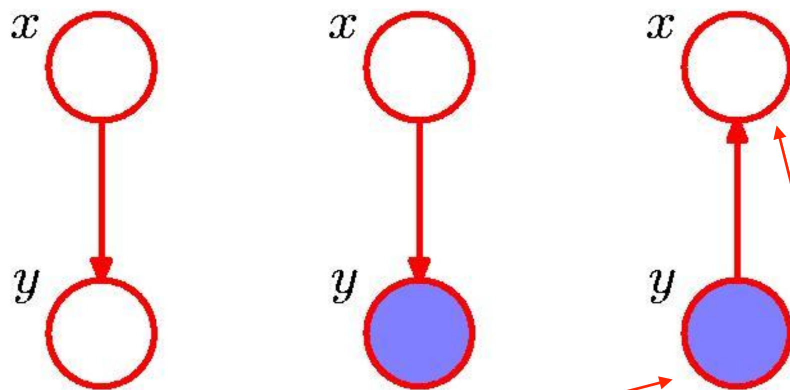


$$p(\mathbf{x}) = \underbrace{p(x_1)p(x_2|x_1)}_{}\, p(x_3|x_2) \cdots p(x_N|x_{N-1})$$

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

Inference in Graphical Models

We observe y. What is p(xly)?



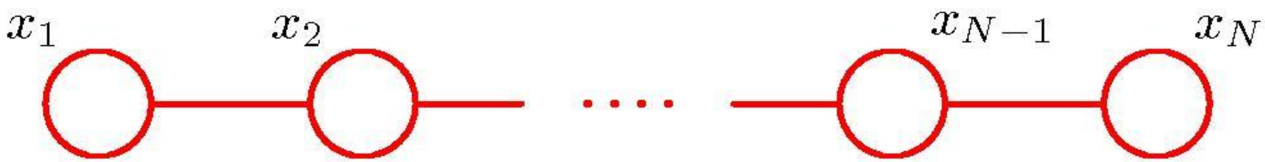
$$p(y) = \sum_{x'} p(y|x')p(x')$$

Sum rule

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Bayes' formula

Inference on a Chain



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

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

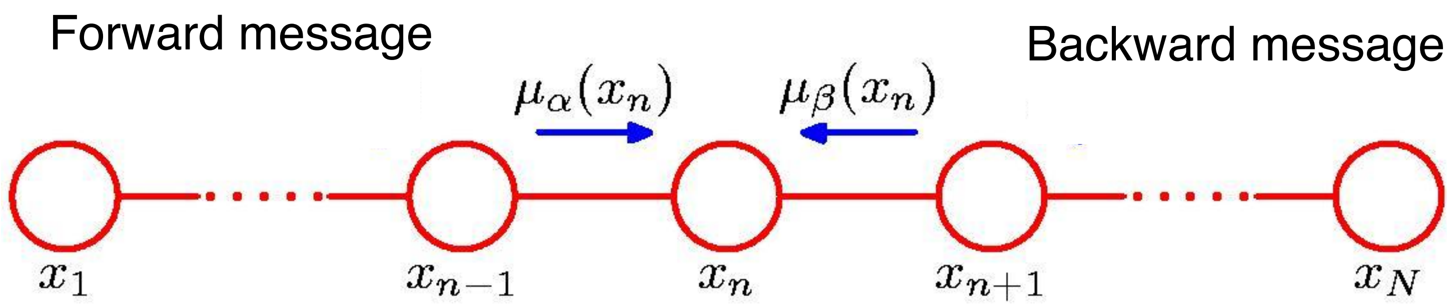
Last term is now only function of x_{N-1}

Naive summation:

For N variables with K states: K^{N-1} terms

Perform summation first over x_N, and save operations: ab+ac=a(b+c)

Inference on a Chain

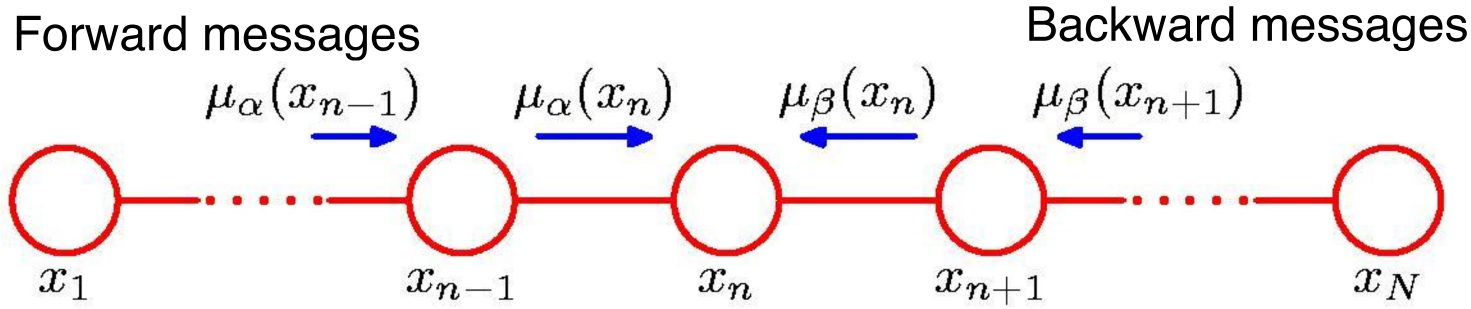


$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)}$$

This trick can be seen as passing local messages around in the graph.

Inference on a Chain

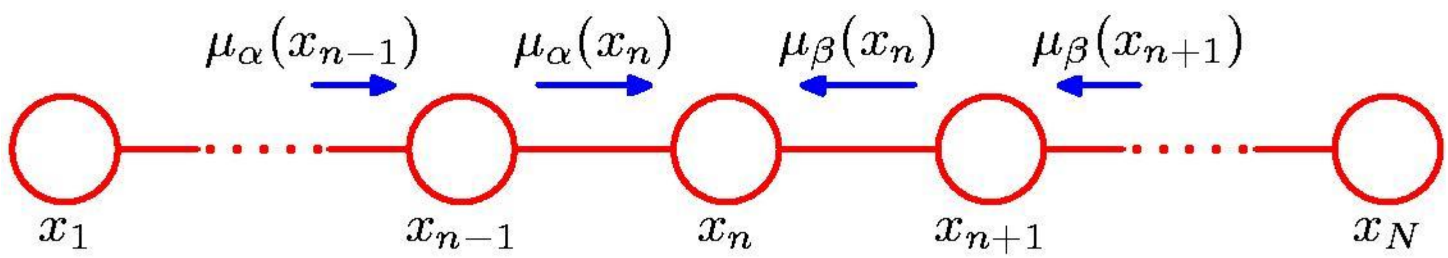


$$
\begin{aligned}
\mu_\alpha(x_n) &= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right] \\
&= \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \mu_\alpha(x_{n-1}).
\end{aligned}
$$

$$
\begin{aligned}
\mu_\beta(x_n) &= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \left[\sum_{x_{n+2}} \cdots \right] \\
&= \sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \mu_\beta(x_{n+1}).
\end{aligned}
$$

These messages can be evaluated recursively.

Inference on a Chain



$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2) \qquad\qquad \mu_\beta(x_{N-1}) = \sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)$$

$$Z = \sum_{x_n} \mu_\alpha(x_n)\mu_\beta(x_n)$$

Start of the recursion, and calculation of the normalization constant-.

Inference on a Chain

To compute local marginals:

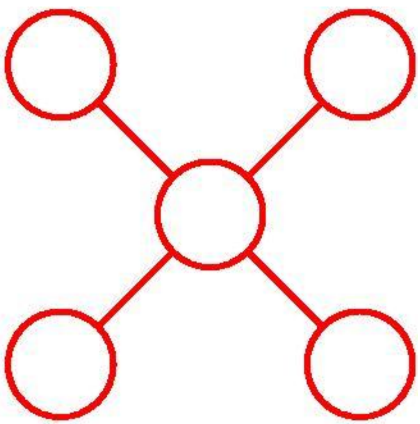
- Compute and store all forward messages, $\mu_\alpha(x_n)$.
- Compute and store all backward messages, $\mu_\beta(x_n)$.
- Compute Z at any node x_m
- Compute

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

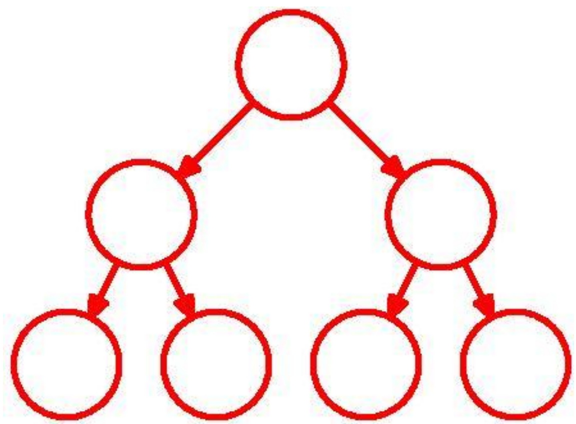
for all variables required.

Trees

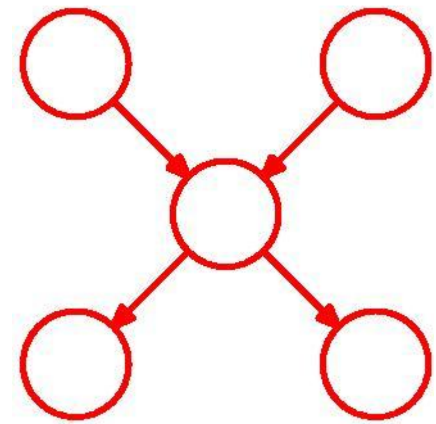
Undirected Tree



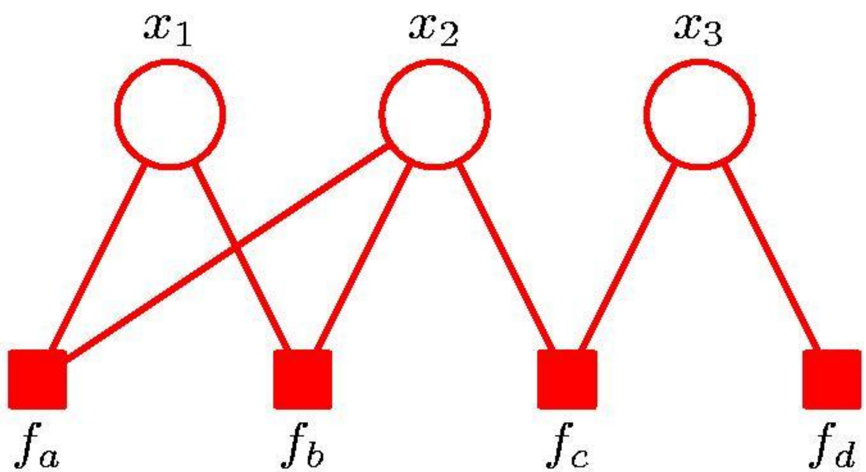
Directed Tree



Polytree



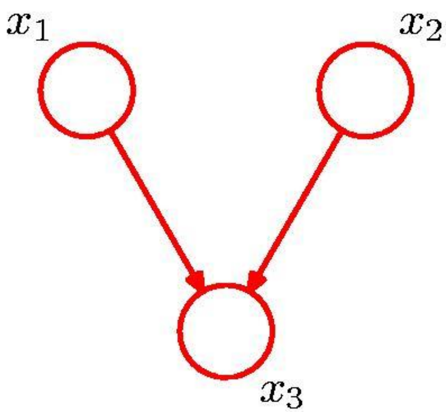
Factor Graphs

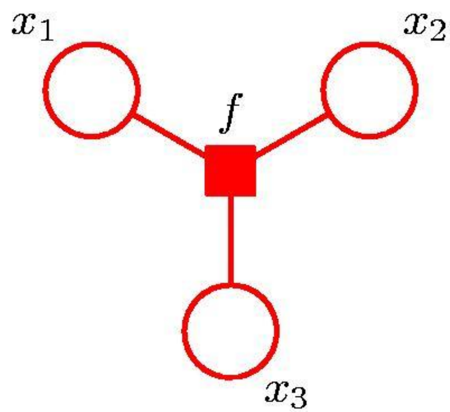


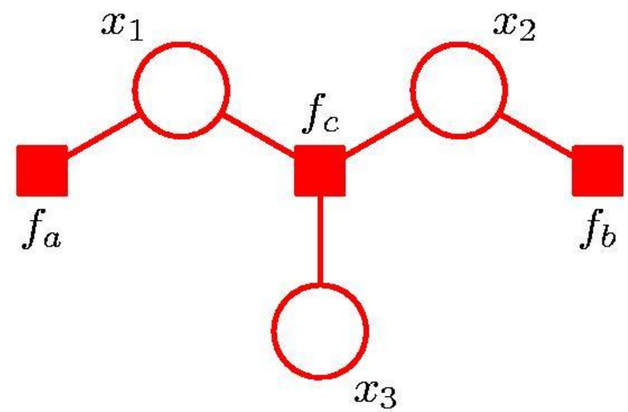
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$

$$p(\mathbf{x}) = \prod_s f_s(\mathbf{x}_s)$$

Factor Graphs from Directed Graphs







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

$$f(x_1,x_2,x_3) = p(x_1)p(x_2)p({}_3|x_1,x_2)$$

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

The Sum-Product Algorithm (1)

Objective:

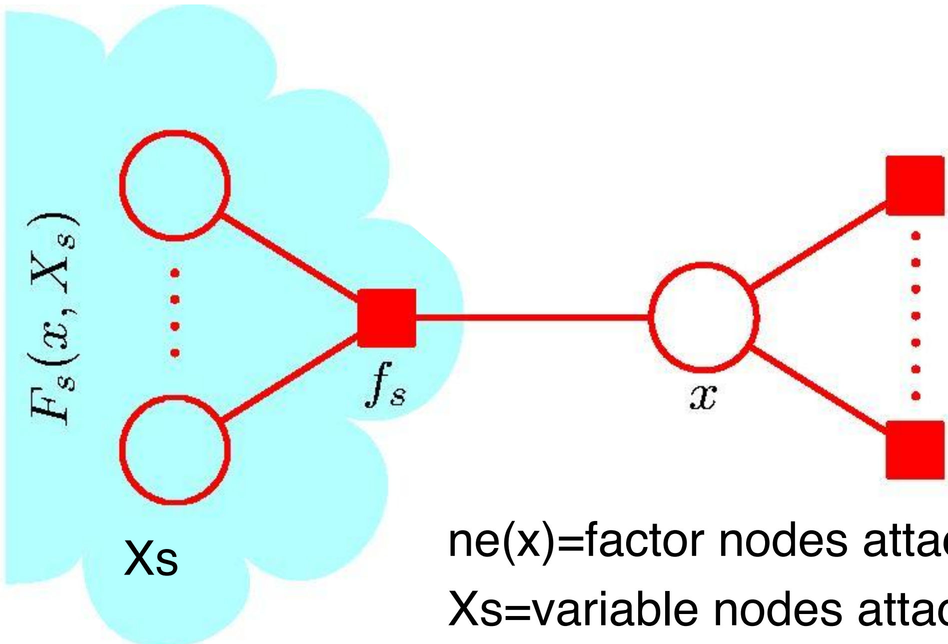
i. to obtain an efficient, exact inference algorithm for finding marginals;

ii. in situations where several marginals are required, to allow computations to be shared efficiently.

Key idea: Distributive Law

$$ab + ac = a(b + c)$$

The Sum-Product Algorithm (2)



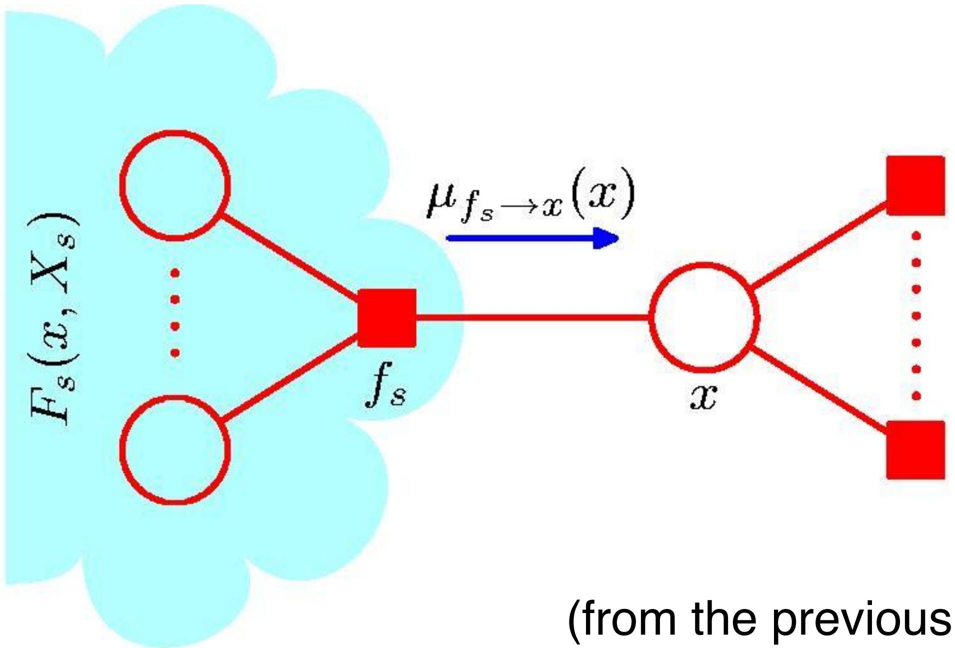
ne(x)=factor nodes attached to x

Xs=variable nodes attached to x via factor s

$$p(x) = \sum_{\mathbf{x}\setminus x} p(\mathbf{x})$$

$$p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s)$$

The Sum-Product Algorithm (3)



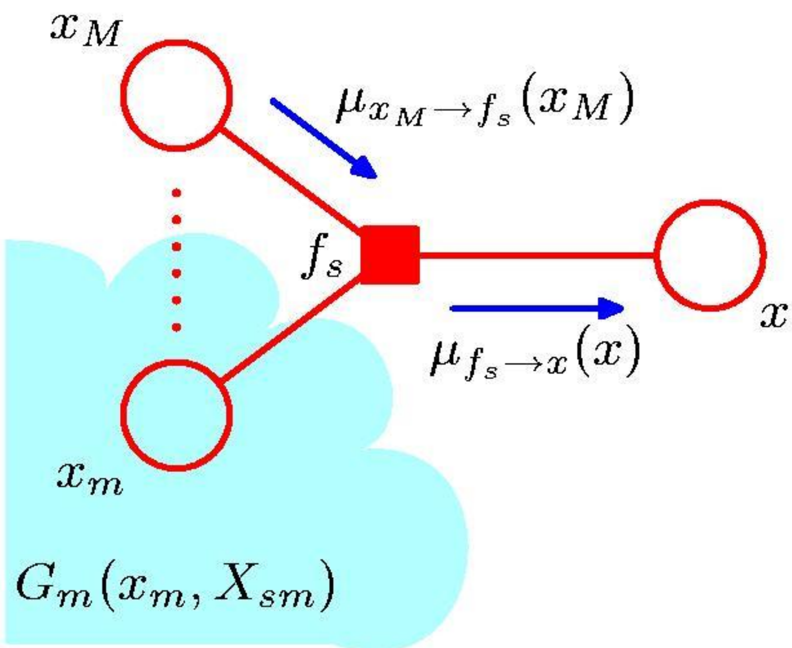
(from the previous two formula's, and interchanging sum and product)

$$
\begin{aligned}
p(x) &= \prod_{s \in \mathrm{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right] \\
&= \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x).
\end{aligned}
$$

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

Again, this can be viewed as messages, from factor nodes fs to variable node x.

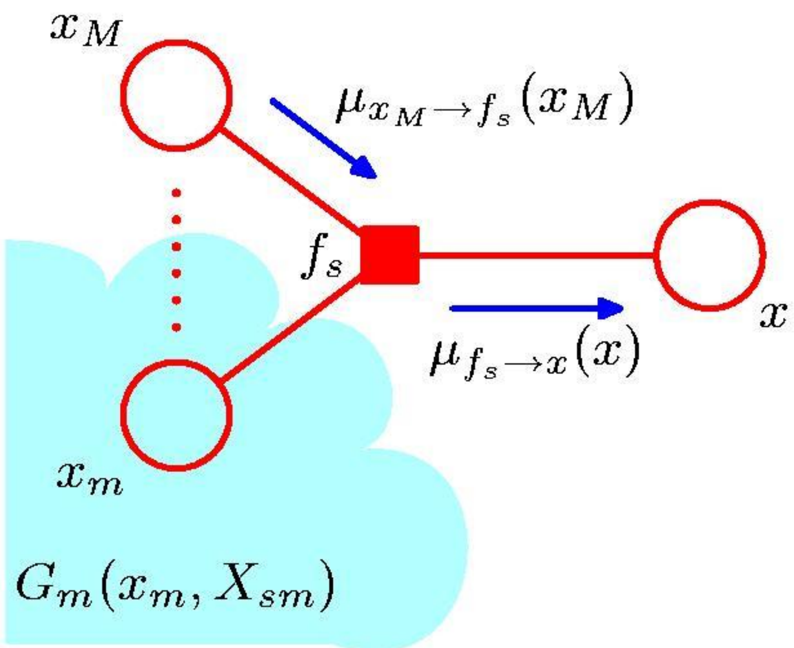
The Sum-Product Algorithm (4)



How is Fs calculated?

$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M) G_1(x_1, X_{s1}) \ldots G_M(x_M, X_{sM})$$

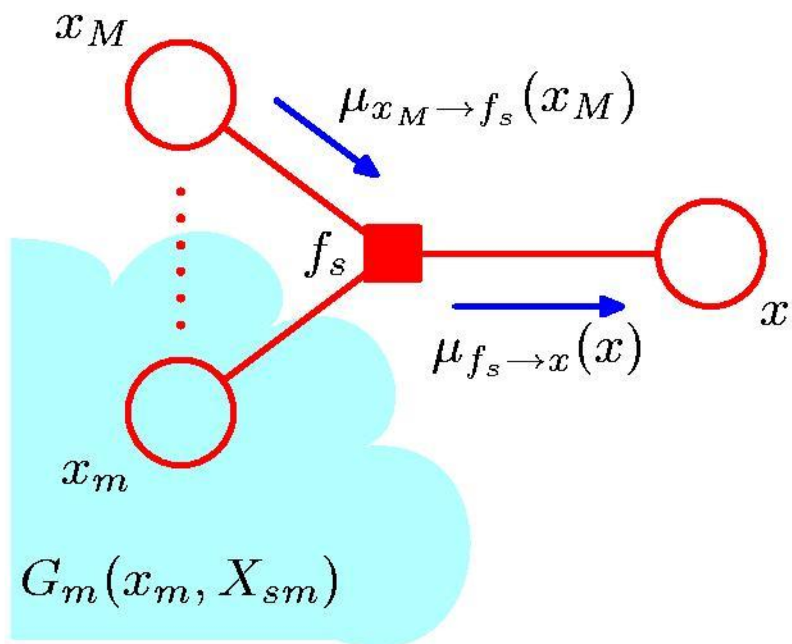
The Sum-Product Algorithm (5)



$$
\begin{aligned}
\mu_{f_s \to x}(x) &= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right] \\
&= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)
\end{aligned}
$$

Second type of message! This time from variable to factor node!

The Sum-Product Algorithm (6)



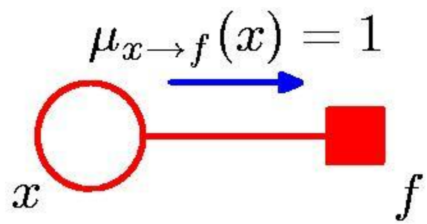
$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) = \sum_{X_{sm}} \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{ml})$$

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

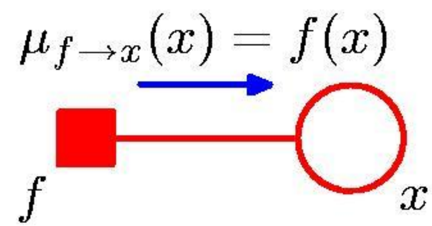
Messages from variable nodes to factor nodes.

The Sum-Product Algorithm (7)

Initialization



Messages from
variable to factor node



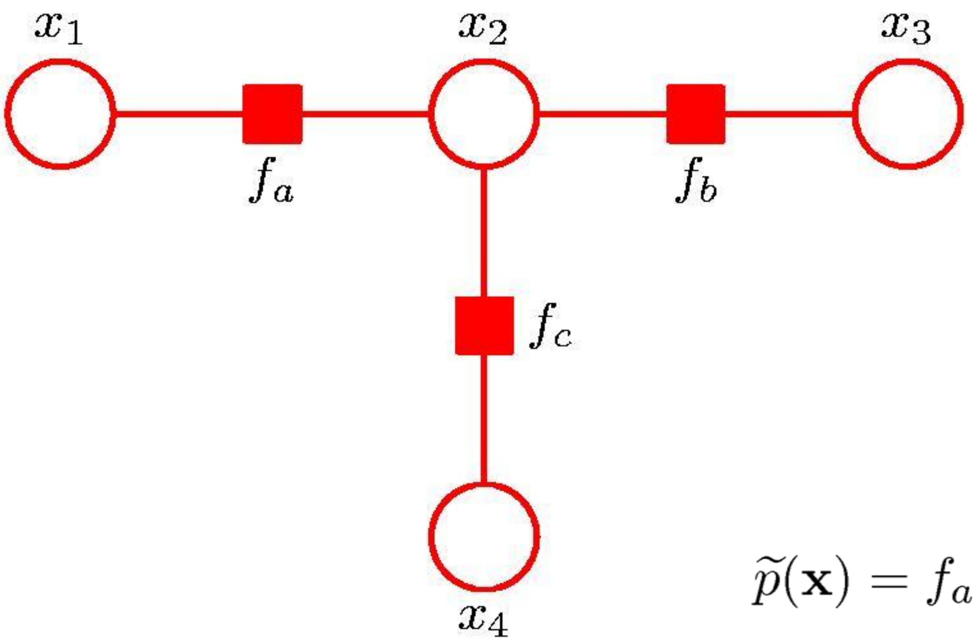
Messages from
factor to variable node

The Sum-Product Algorithm (8)

To compute local marginals:

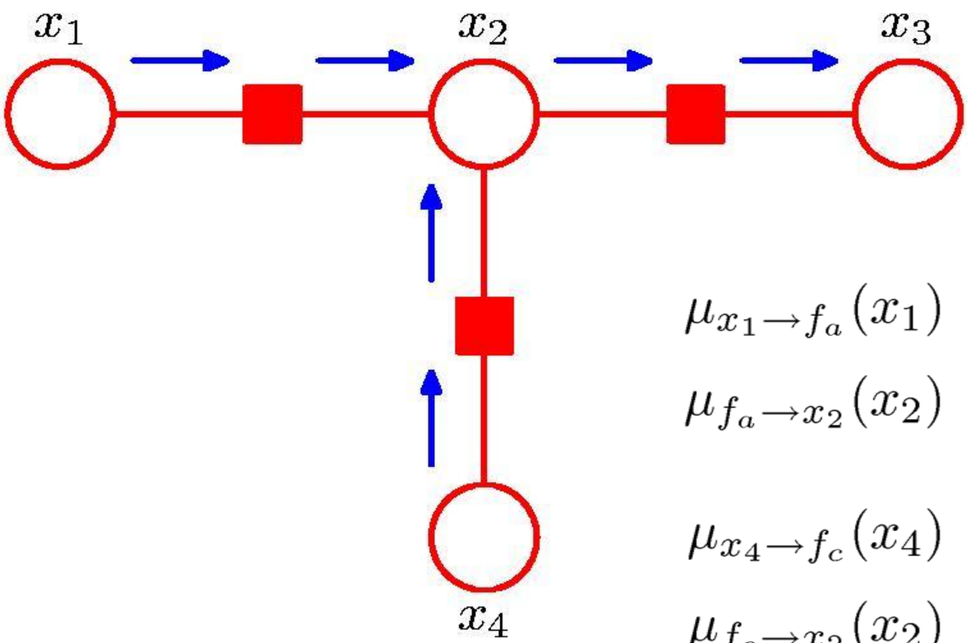
- Pick an arbitrary node as root
- Compute and propagate messages from the leaf nodes to the root, storing received messages at every node.
- Compute and propagate messages from the root to the leaf nodes, storing received messages at every node.
- Compute the product of received messages at each node for which the marginal is required, and normalize if necessary.

Sum-Product: Example (1)



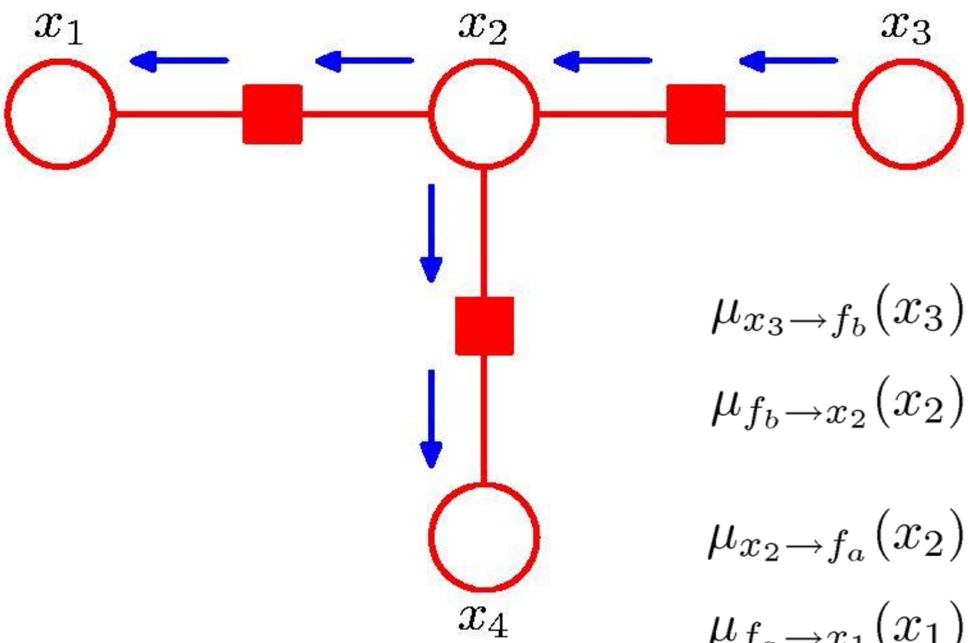
$$\widetilde{p}(\mathbf{x}) = f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4)$$

Sum-Product: Example (2)



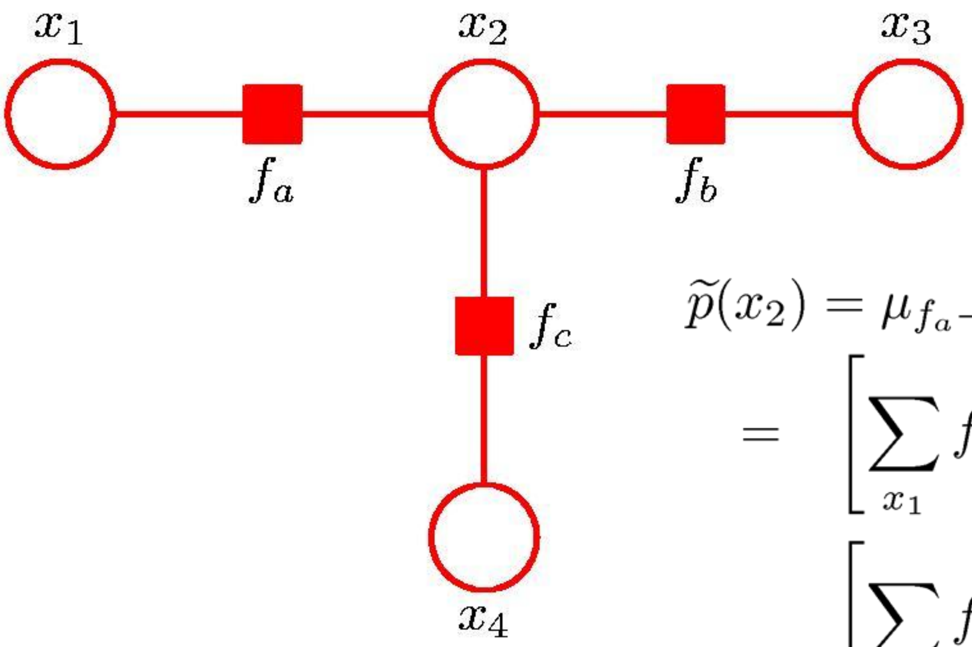
$$
\begin{aligned}
\mu_{x_1 \to f_a}(x_1) &= 1 \\
\mu_{f_a \to x_2}(x_2) &= \sum_{x_1} f_a(x_1, x_2) \\
\mu_{x_4 \to f_c}(x_4) &= 1 \\
\mu_{f_c \to x_2}(x_2) &= \sum_{x_4} f_c(x_2, x_4) \\
\mu_{x_2 \to f_b}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
\mu_{f_b \to x_3}(x_3) &= \sum_{x_2} f_b(x_2, x_3)\mu_{x_2 \to f_b}(x_2)
\end{aligned}
$$

Sum-Product: Example (3)



$$
\begin{aligned}
\mu_{x_3 \to f_b}(x_3) &= 1 \\
\mu_{f_b \to x_2}(x_2) &= \sum_{x_3} f_b(x_2, x_3) \\
\mu_{x_2 \to f_a}(x_2) &= \mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
\mu_{f_a \to x_1}(x_1) &= \sum_{x_2} f_a(x_1, x_2)\mu_{x_2 \to f_a}(x_2) \\
\mu_{x_2 \to f_c}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2) \\
\mu_{f_c \to x_4}(x_4) &= \sum_{x_2} f_c(x_2, x_4)\mu_{x_2 \to f_c}(x_2)
\end{aligned}
$$

Sum-Product: Example (4)



$$
\begin{aligned}
\widetilde{p}(x_2) &= \mu_{f_a \to x_2}(x_2)\mu_{f_b \to x_2}(x_2)\mu_{f_c \to x_2}(x_2) \\
&= \left[\sum_{x_1} f_a(x_1, x_2)\right]\left[\sum_{x_3} f_b(x_2, x_3)\right] \\
&\quad \left[\sum_{x_4} f_c(x_2, x_4)\right] \\
&= \sum_{x_1}\sum_{x_3}\sum_{x_4} f_a(x_1, x_2) f_b(x_2, x_3) f_c(x_2, x_4) \\
&= \sum_{x_1}\sum_{x_3}\sum_{x_4} \widetilde{p}(\mathbf{x})
\end{aligned}
$$

The Max-Sum Algorithm (1)

Objective: an efficient algorithm for finding

i. the value $\mathbf{x}^{\max}$ that maximises $p(\mathbf{x})$;
ii. the value of $p(\mathbf{x}^{\max})$.

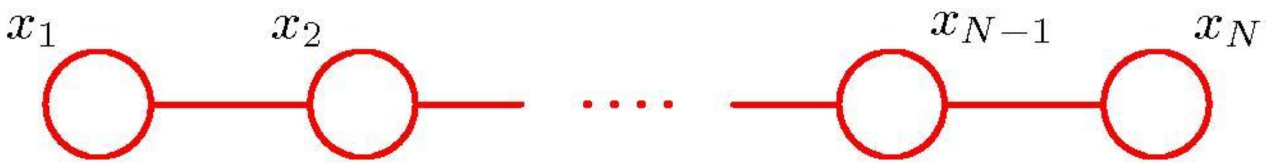
In general, maximum marginals $\neq$ joint maximum.

	$x = 0$	$x = 1$
$y = 0$	0.3	0.4
$y = 1$	0.3	0.0

$$\arg\max_x p(x, y) = 1 \qquad \arg\max_x p(x) = 0$$

The Max-Sum Algorithm (2)

Maximizing over a chain (max-product)



$$p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \ldots \max_{x_M} p(\mathbf{x})$$

$$= \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} \left[\psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N)\right]$$

$$= \frac{1}{Z} \max_{x_1} \left[\max_{x_2} \left[\psi_{1,2}(x_1, x_2) \left[\cdots \max_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots\right]\right]$$

The Max-Sum Algorithm (3)

Generalizes to tree-structured factor graph

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_n} \prod_{f_s \in \mathrm{ne}(x_n)} \max_{X_s} f_s(x_n, X_s)$$

maximizing as close to the leaf nodes as possible

The Max-Sum Algorithm (4)

Max-Product $\rightarrow$ Max-Sum

For numerical reasons, use

$$\ln\left(\max_{\mathbf{x}} p(\mathbf{x})\right) = \max_{\mathbf{x}} \ln p(\mathbf{x}).$$

Again, use distributive law

$$\max(a+b, a+c) = a + \max(b, c).$$

The Max-Sum Algorithm (5)

Initialization (leaf nodes)

$$\mu_{x\to f}(x) = 0 \qquad\qquad \mu_{f\to x}(x) = \ln f(x)$$

Recursion

$$\begin{aligned}
\mu_{f\to x}(x) &= \max_{x_1,\ldots,x_M}\left[\ln f(x, x_1, \ldots, x_M) + \sum_{m\in\mathrm{ne}(f_s)\setminus x} \mu_{x_m\to f}(x_m)\right] \\
\phi(x) &= \arg\max_{x_1,\ldots,x_M}\left[\ln f(x, x_1, \ldots, x_M) + \sum_{m\in\mathrm{ne}(f_s)\setminus x} \mu_{x_m\to f}(x_m)\right] \\
\mu_{x\to f}(x) &= \sum_{l\in\mathrm{ne}(x)\setminus f} \mu_{f_l\to x}(x)
\end{aligned}$$

The Max-Sum Algorithm (6)

Termination (root node)

$$p^{\max} = \max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

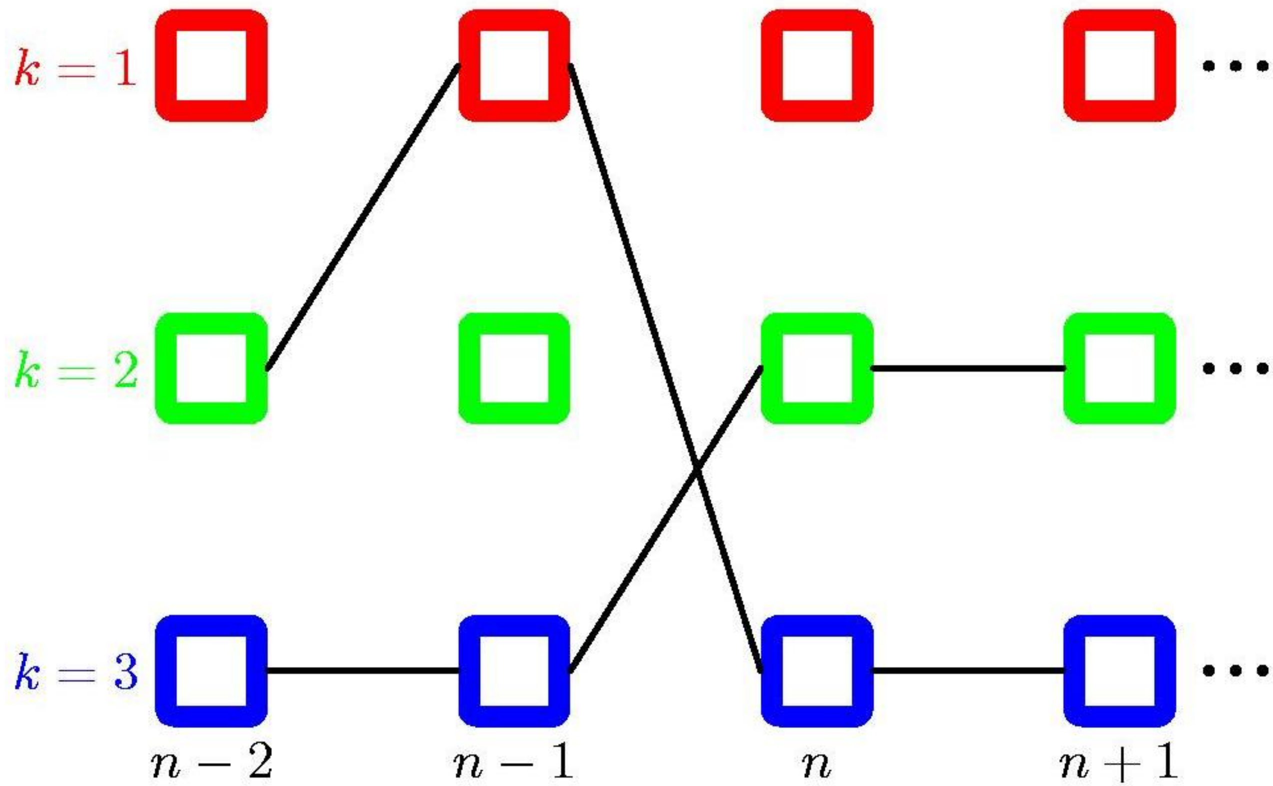
$$x^{\max} = \arg\max_x \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

Back-track, for all nodes i with l factor nodes to the root ($l=0$)

$$\mathbf{x}_l^{\max} = \phi(x_{i,l-1}^{\max})$$

The Max-Sum Algorithm (7)

Example: Markov chain



The Junction Tree Algorithm

- *Exact* inference on general graphs.
- Works by turning the initial graph into a *junction tree* and then running a sum-product-like algorithm.
- *Intractable* on graphs with large cliques.

Loopy Belief Propagation

- Sum-Product on general graphs.
- Initial unit messages passed across all links, after which messages are passed around until convergence (not guaranteed!).
- *Approximate* but *tractable* for large graphs.
- Sometime works well, sometimes not at all.

Sequential Data and Markov Models

Sargur N. Srihari

srihari@cedar.buffalo.edu

Machine Learning Course:

http://www.cedar.buffalo.edu/~srihari/CSE574/index.html

Sequential Data Examples

- Often arise through measurement of time series
  - Snowfall measurements on successive days
  - Rainfall measurements on successive days
  - Daily values of currency exchange rate
  - Acoustic features at successive time frames in speech recognition
  - Nucleotide base pairs in a strand of DNA
  - Sequence of characters in an English sentence
  - Parts of speech of successive words

Markov Model – Weather

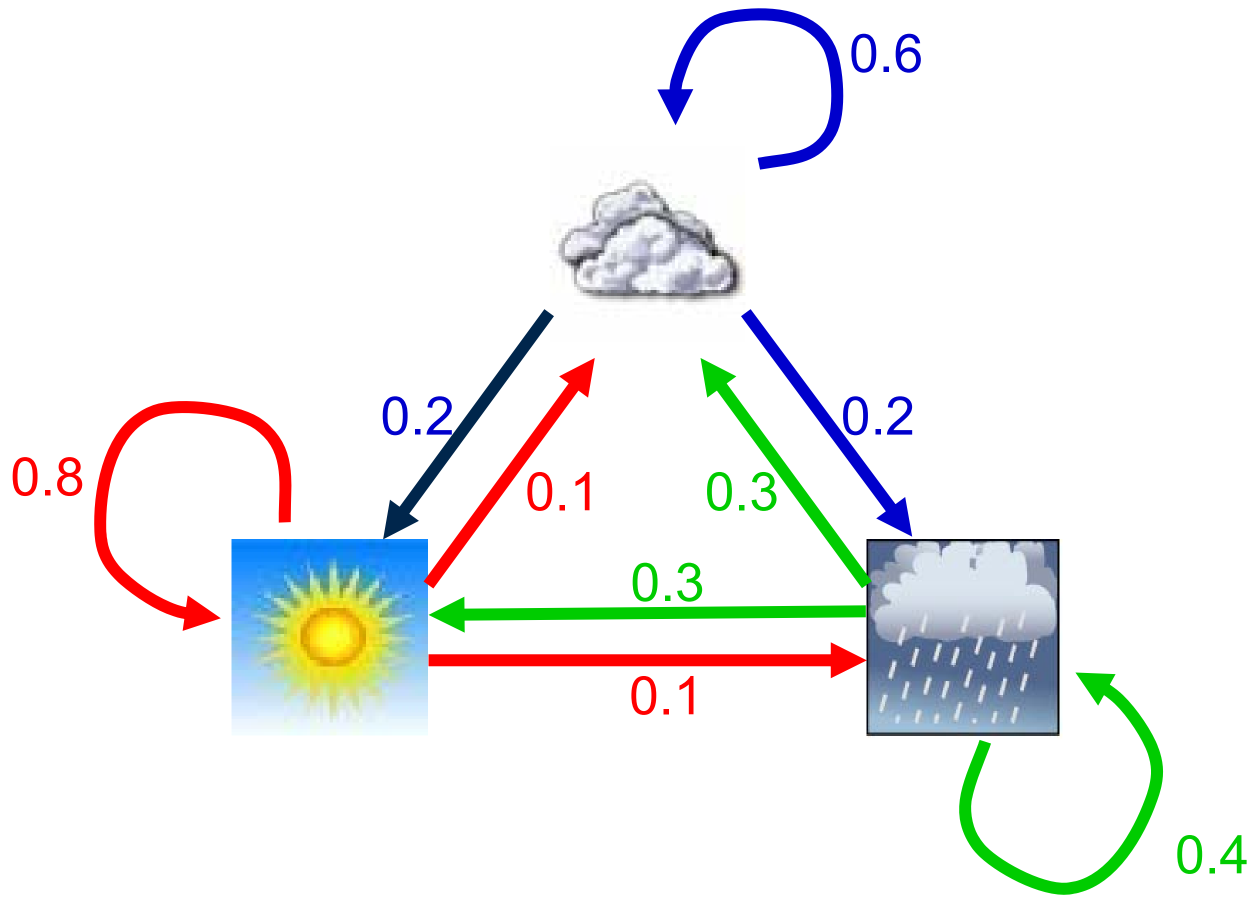
- The weather of a day is observed as being one of the following:
  - State 1: Rainy
  - State 2: Cloudy
  - State 3: Sunny



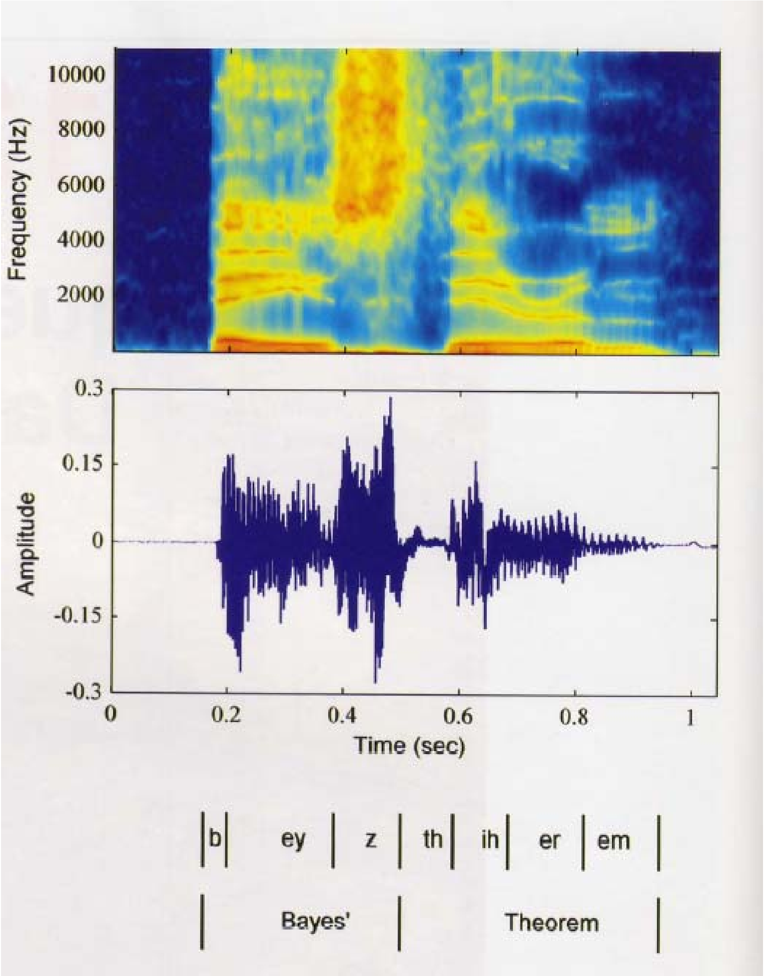
- Weather pattern of a location

		Tomorrow		
		Rain	Cloudy	Sunny
Today	Rain	0.3	0.3	0.4
	Cloudy	0.2	0.6	0.2
	Sunny	0.1	0.1	0.8

Markov Model – Weather State Diagram



Sound Spectrogram of Speech

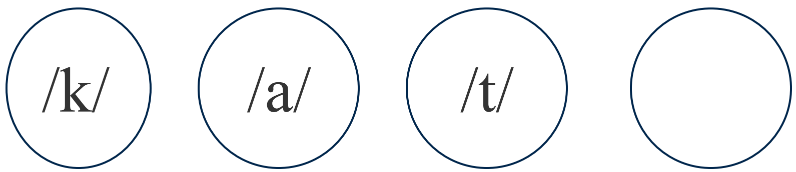


- "Bayes Theorem"
- Plot of the intensity of the spectral coefficients versus time index
- Successive observations of speech spectrum highly correlated (Markov dependency)

Markov model for the production of spoken words

- States represent phonemes
- Production of word: “cat”
- Represented by states
  /k/ /a/ /t/
- Transitions from
  - /k/ to /a/
  - /a/ to /t/
  - /t/ to a silent state
- Although only correct cat sound is represented by model, perhaps other transitions can be introduced,
  - eg, /k/ followed by /t/

Markov Model for word “cat”



Stationary *vs* Non-stationary

- Stationary: Data evolves over time but distribution remains same
  - e.g., dependence of current word over previous word remains constant
- Non-stationary: Generative distribution itself changes over time

Making a Sequence of Decisions

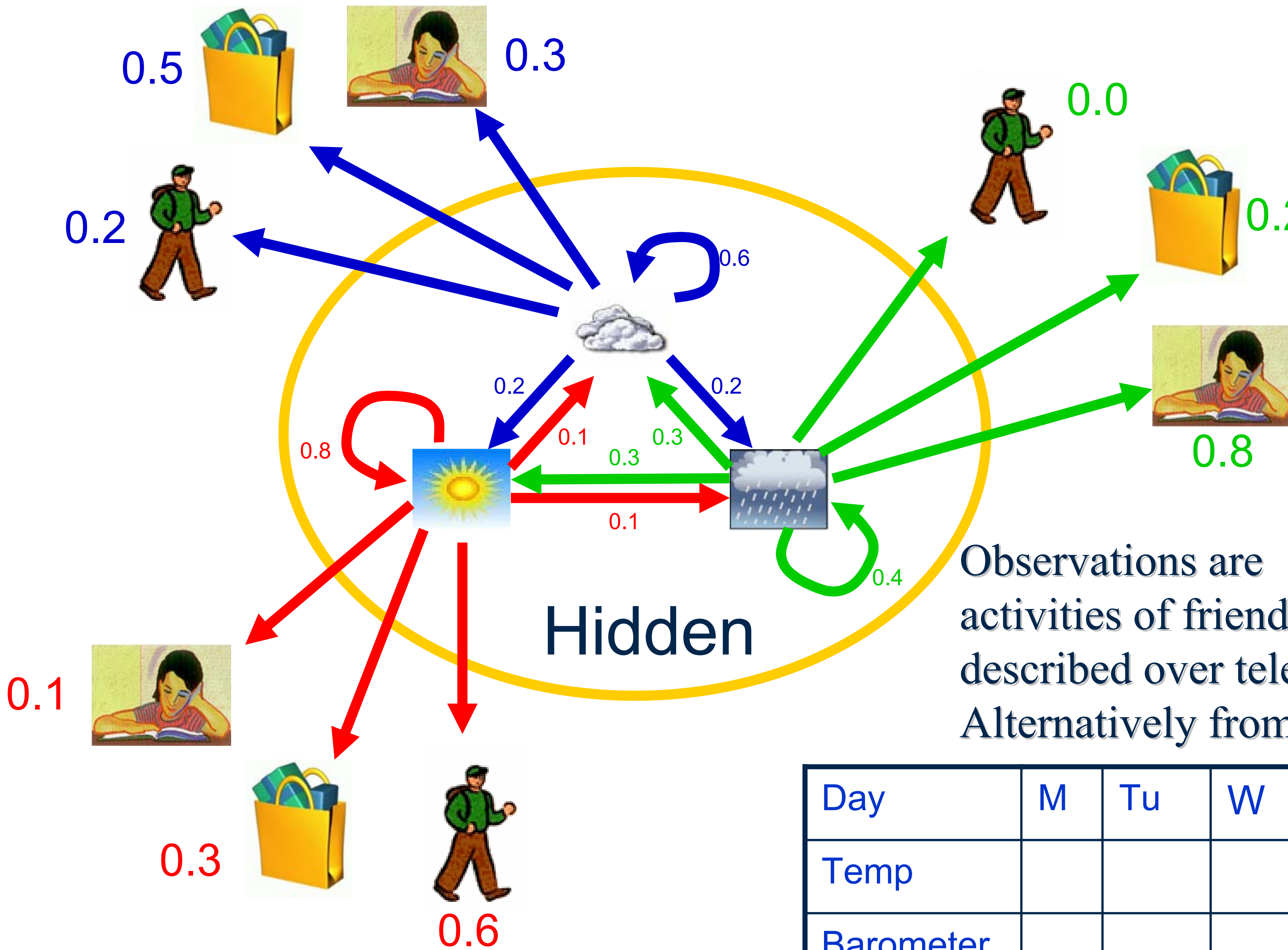
- Processes in time, states at time t are influenced by a state at time $t-1$

- Wish to predict next value from previous values, e.g., financial forecasting
- Impractical to consider general dependence of future dependence on all previous observations
  - Complexity grows without limit as number of observations increases
- Markov models assume dependence on most recent observations

Latent Variables

- While Markov models are tractable they are severely limited
- Introduction of latent variables provides a more general framework
- Lead to state-space models
- When latent variables are:
  - Discrete
    - they are called *Hidden Markov models*
  - Continuous
    - they are *linear dynamical systems*

Hidden Markov Model

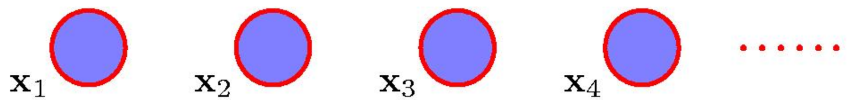


Observations are activities of friend described over telephone
Alternatively from data:

Day	M	Tu	W	Th
Temp				
Barometer				

Markov Model Assuming Independence

- Simplest model:



  - Assume observations are independent
  - Graph without links
- To predict whether it rains tomorrow is only based on relative frequency of rainy days
- Ignores influence of whether it rained the previous day

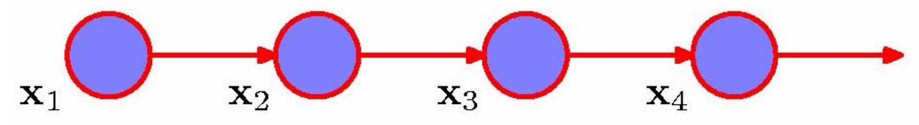
Markov Model

- Most general Markov model for observations $\{x_n\}$
- Product rule to express joint distribution of sequence of observations

$$p(x_1,\ldots x_N) = \prod_{n=1}^{N} p(x_n \mid x_1,\ldots x_{n-1})$$

First Order Markov Model

- Chain of observations $\{x_n\}$



- Joint distribution for a sequence of n variables

$$p(x_1, \ldots x_N) = p(x_1) \prod_{n=2}^{N} p(x_n \mid x_{n-1})$$

- It can be verified (using product rule from above) that

$$p(x_n \mid x_1 .. x_{n-1}) = p(x_n \mid x_{n-1})$$

- If model is used to predict next observation, distribution of prediction will only depend on preceding observation and independent of earlier observations
- Stationarity implies conditional distributions $p(x_n|x_{n-1})$ are all equal

Markov Model – Sequence probability

- What is the probability that the weather for the next 7 days will be "S-S-R-R-S-C-S"?

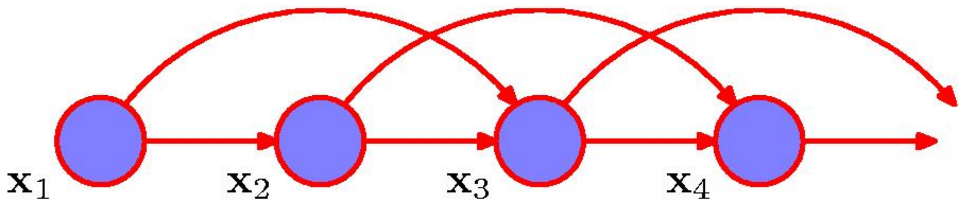
$$O = \{S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3\}$$

  - Find the probability of *O*, given the model.

$$\begin{aligned}
P(O \mid Model) &= P(S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3 \mid Model) \\
&= P(S_3) \cdot P(S_3 \mid S_3) \cdot P(S_3 \mid S_3) \cdot P(S_1 \mid S_3) \\
&\quad \cdot P(S_1 \mid S_1) \cdot P(S_3 \mid S_1) \cdot P(S_2 \mid S_3) \cdot P(S_3 \mid S_2) \\
&= \pi_3 \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\
&= 1 \cdot (0.8) \cdot (0.8) \cdot (0.1) \cdot (0.4) \cdot (0.3) \cdot (0.1) \cdot (0.2) \\
&= 1.536 \times 10^{-4}
\end{aligned}$$

Second Order Markov Model

- Conditional distribution of observation x_n depends on the values of two previous observations x_{n-1} and x_{n-2}



$$p(x_1,..x_N) = p(x_1)p(x_2 \mid x_1)\prod_{n=3}^{N} p(x_n \mid x_{n-1}, x_{n-2})$$

- Each observation is influenced by previous two observations

M^{th} Order Markov Source

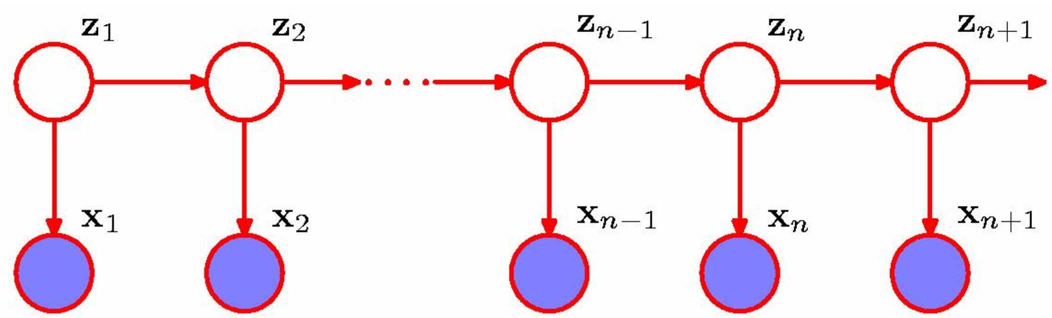
- Conditional distribution for a particular variable depends on previous M variables
- Pay a price for number of parameters
- Discrete variable with K states
  - First order: $p(x_n|x_{n-1})$ needs K-1 parameters for each value of x_{n-1} for each of K states of x_n giving $K(K$-$1)$ parameters
  - M^{th} order will need $K^{M-1}(K$-$1)$ parameters

Introducing Latent Variables

- Model for sequences not limited by Markov assumption of any order but with limited number of parameters
- For each observation x_n, introduce a latent variable z_n
- z_n may be of different type or dimensionality to the observed variable
- Latent variables form the Markov chain
- Gives the "state-space model"

Latent variables

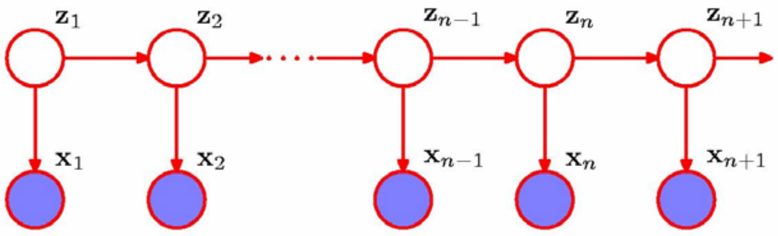
Observations



Conditional Independence with Latent Variables

- Satisfies key assumption that

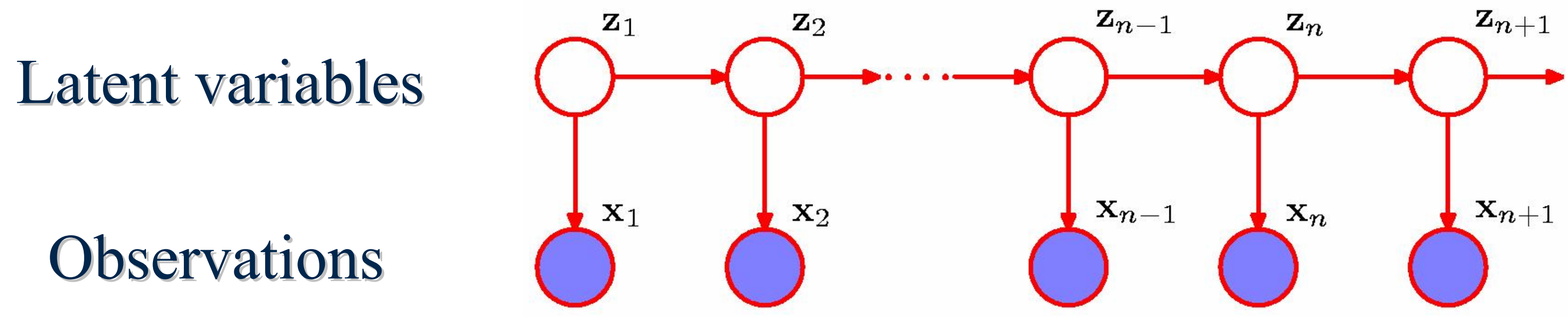
$$z_{n+1} \perp z_{n-1} \mid z_n$$



- From d-separation

  When latent node z_n is filled, the only path between z_{n-1} and z_{n+1} has a head-to-tail node that is blocked

Jt Distribution with Latent Variables

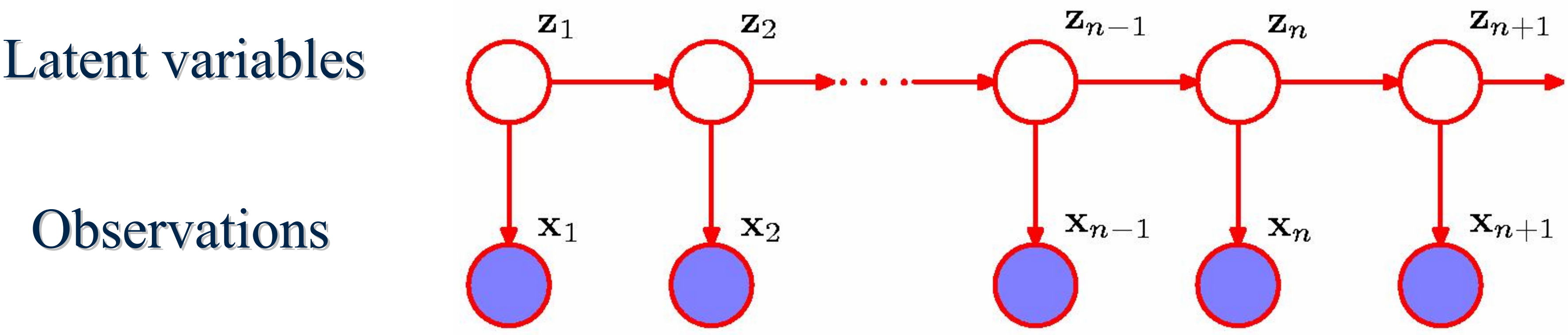


- Joint distribution for this model

$$p(x_1,..x_N, z_1,..z_n) = p(z_1)\left[\prod_{n=2}^{N} p(z_n \mid z_{n-1})\right]\prod_{n=1}^{N} p(x_n \mid z_n)$$

- There is always a path between any x_n and x_m via latent variables which is never blocked
- Thus predictive distribution $p(x_{n+1}|x_1,..,x_n)$ for observation x_{n+1} does not exhibit conditional independence properties and is hence dependent on all previous observations

Two Models Described by Graph



1. Hidden Markov Model: If latent variables are discrete:

   Observed variables in a HMM may be discrete or continuous

2. Linear Dynamical Systems: If both latent and observed variables are Gaussian

Further Topics on Sequential Data

- Hidden Markov Models:

  http://www.cedar.buffalo.edu/~srihari/CSE574/Chap11/Ch11.2-HiddenMarkovModels.pdf

- Extensions of HMMs:

  http://www.cedar.buffalo.edu/~srihari/CSE574/Chap11/Ch11.3-HMMExtensions.pdf

- Linear Dynamical Systems:

  http://www.cedar.buffalo.edu/~srihari/CSE574/Chap11/Ch11.4-LinearDynamicalSystems.pdf

- Conditional Random Fields:

  http://www.cedar.buffalo.edu/~srihari/CSE574/Chap11/Ch11.5-ConditionalRandomFields.pdf