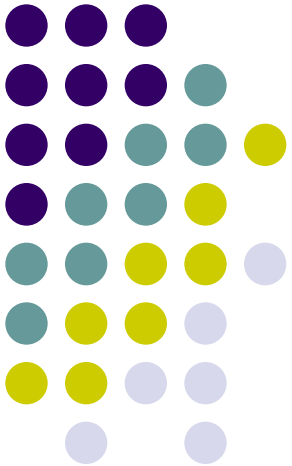
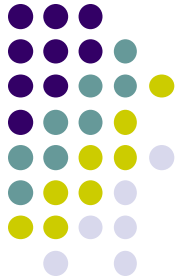
Probabilistic Graphical Models

Hongxin Zhang
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2020-03-17



Probabilistic Graphical Models



- Modeling many real-world problems => a large number of random variables
  - Dependences among variables may be used to reduce the size to encode the model (PCA ?), or
  - They may be the goal by themselves, that is, the idea is to understand the correlations among variables.

Modeling the domain

- Discrete random variables
  - Take 5 random binary variables (A, B, C, D, E)
  - i.i.d. data from a multinomial distribution

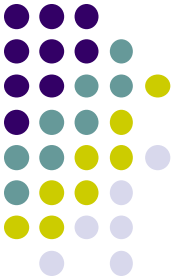
A	B	C	D	E
a	$\sim b$	$\sim c$	$\sim d$	$\sim e$
a	b	$\sim c$	d	$\sim e$
a	$\sim b$	c	d	$\sim e$

Goals

- **(Parameter) Learning:** using training data, estimate the joint distribution
  - Which are the values $P(A, B, C, D, E)$?
  - ... and if there were one hundred binary variables? (Size of model certainly greater than number of atoms on Earth!)

- **Inference:** Given the distribution $P(A, B, C, D, E)$
  - Belief updating: compute the probability of an event
    - What is the probability of $A{=}a$ given $E{=}e$?
  - Maximum a posterior: compute the states of variables that maximize their probability.
    - Which state of A maximizes $P(A \mid E{=}e)$? Is it a or $\sim a$?

The unstructured approach



- To specify the joint distribution, there is an exponential number of values:

$$p(a, b, c, d, e), p(a, b, c, d, \neg e), p(a, b, c, \neg d, e),$$

$$p(a, b, c, \neg d, \neg e), p(a, b, \neg c, d, e), p(a, b, \neg c, d, \neg e),$$

$$p(a, b, \neg c, \neg d, e), p(a, b, \neg c, \neg d, \neg e), \ldots$$

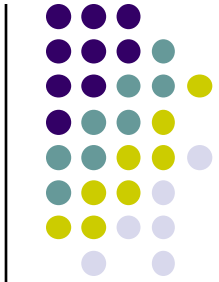
- We can compute the probability of events by:

$$p(a) = \sum_{B,C,D,E} p(a, B, C, D, E)$$

$$p(a|d, \neg e) = \frac{p(a, d, \neg e)}{p(d, \neg e)} = \frac{\sum_{B,C} p(a, B, C, d, \neg e)}{\sum_{A,B,C} p(A, B, C, d, \neg e)}$$

- There are exponentially many terms in the summations...

The naïve Bayesian approach



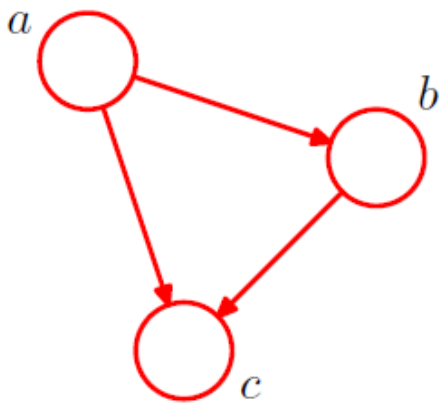
$$p(a, b) = p(a)\, p(b)$$

- Application: Email spanning

Bayesian Networks

- An arbitrary **joint distribution** $p(a, b, c)$ over three variables a, b, and c
  - the product rule of probability:

$$p(a, b, c) = p(\mathrm{c} \mid a, b)\, p(a, b)$$
$$= p(\mathrm{c} \mid a, b)\, p(b \mid a)\, p(a)$$



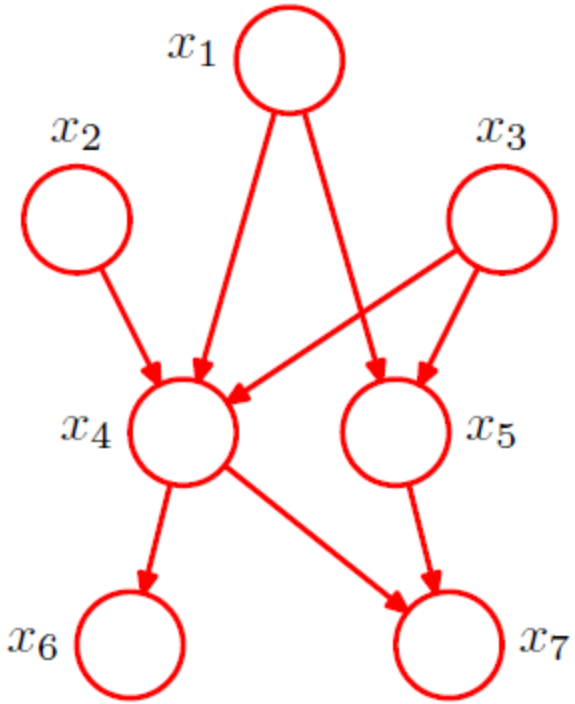
- General case: $p(x_1, x_2, \ldots, x_K)$

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

Not fully connected graph

- Joint distribution: $p(x_1, x_2, \ldots, x_7)$

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



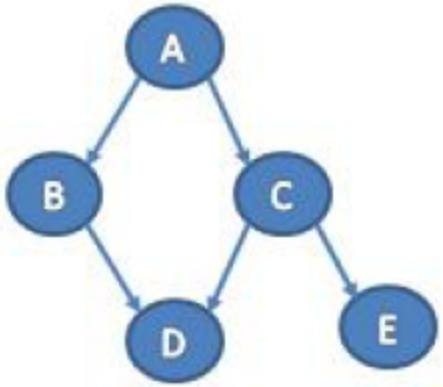
General form

- For a graph with *K* nodes, *the joint distribution* is given by:

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

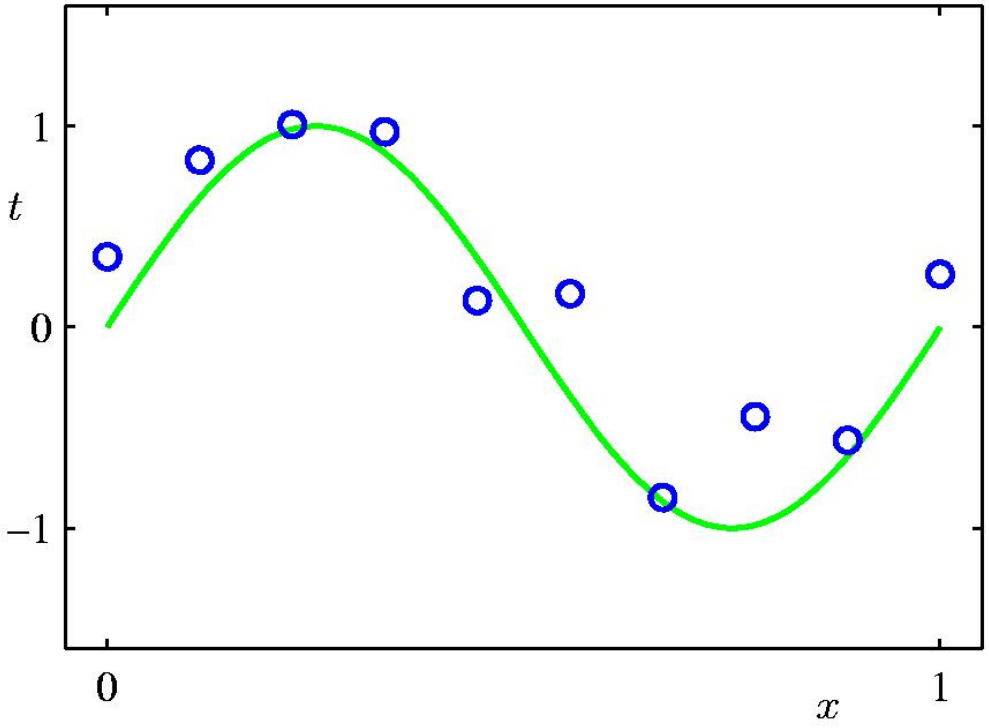
- where pa_k denotes the set of parents of x_k, *and* $\boldsymbol{x} = \{x_1, \ldots, x_K\}$

Definitions



- A set of variables associated with nodes of a Directed Acyclic Graph (DAG).
  - Markov condition (w.r.t. the DAG): each variable is independent of its non-descendants given its parents.
  - For each variable (node), local probability distributions:
    - *P(A), P(B|A=a), P(B|A=a), P(C|A=a), P(C|A=~a), P(D|b, c), P(D|~b, c), P(D|~b,c); P(D|~b,~c), P(E|c), P(E|~c),*
  - All these values are precise.

Regression revisit: Polynomial Curve Fitting



$$\mathbf{h}(x) = \begin{pmatrix} h_0(x) \\ h_1(x) \\ \vdots \\ h_M(x) \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{pmatrix} \quad \mathbf{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_n \end{pmatrix}$$

$$t = \mathbf{h}(x) \cdot \mathbf{w}$$

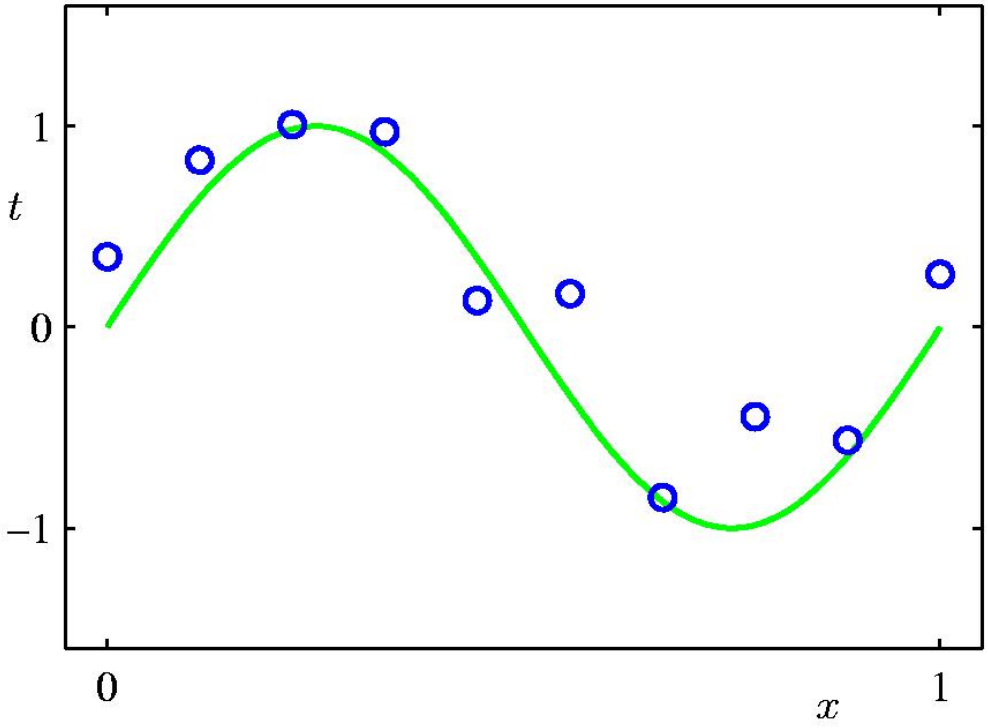
$$\mathbf{w} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{t}$$

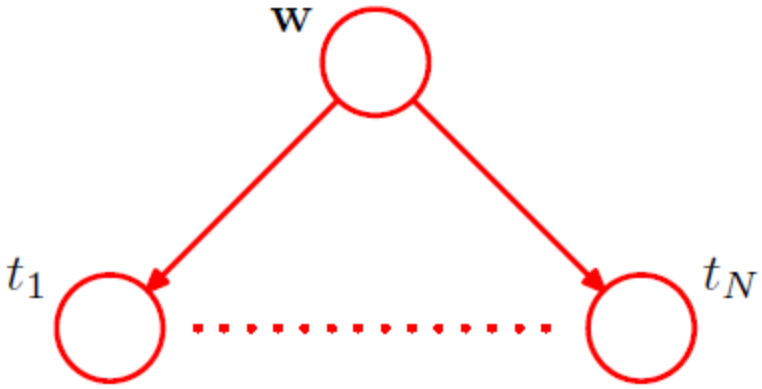
Normal equation

$$t(x, \mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_N h_N(x) = \sum_{j=0}^{N} w_j h_j(x)$$

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

Example: Polynomial regression

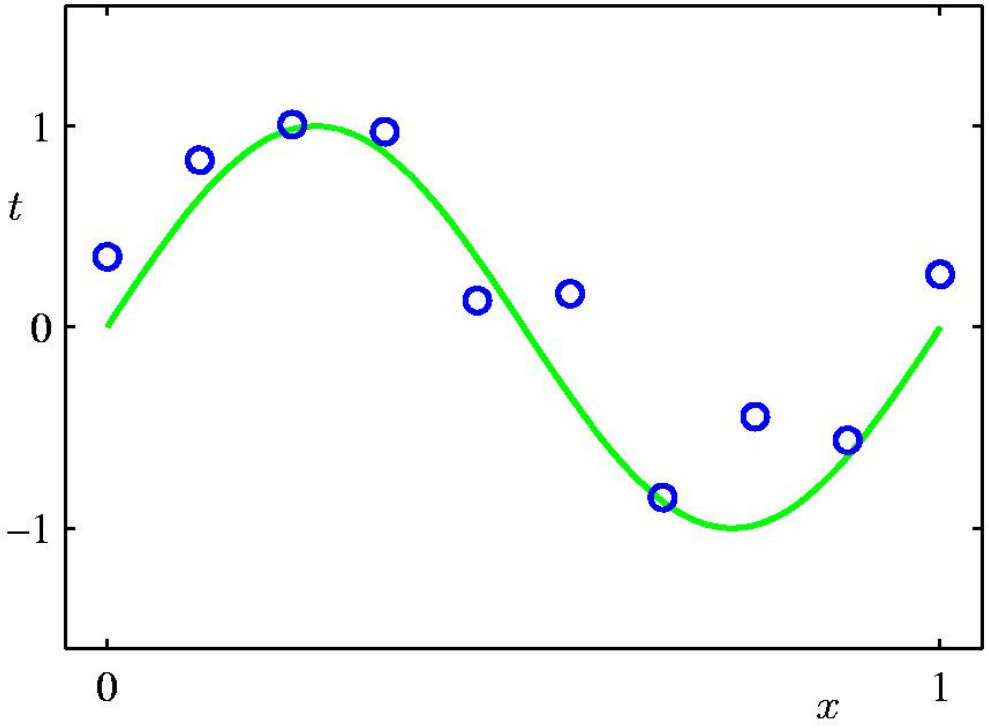


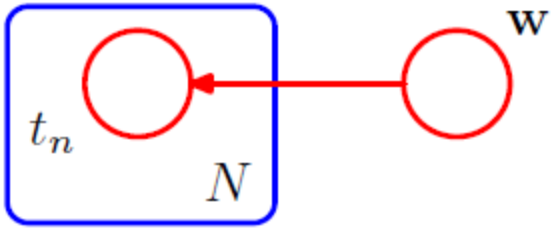


$$t(x, \mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_N h_N(x) = \sum_{j=0}^{N} w_j h_j(x)$$

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

Example: Polynomial regression

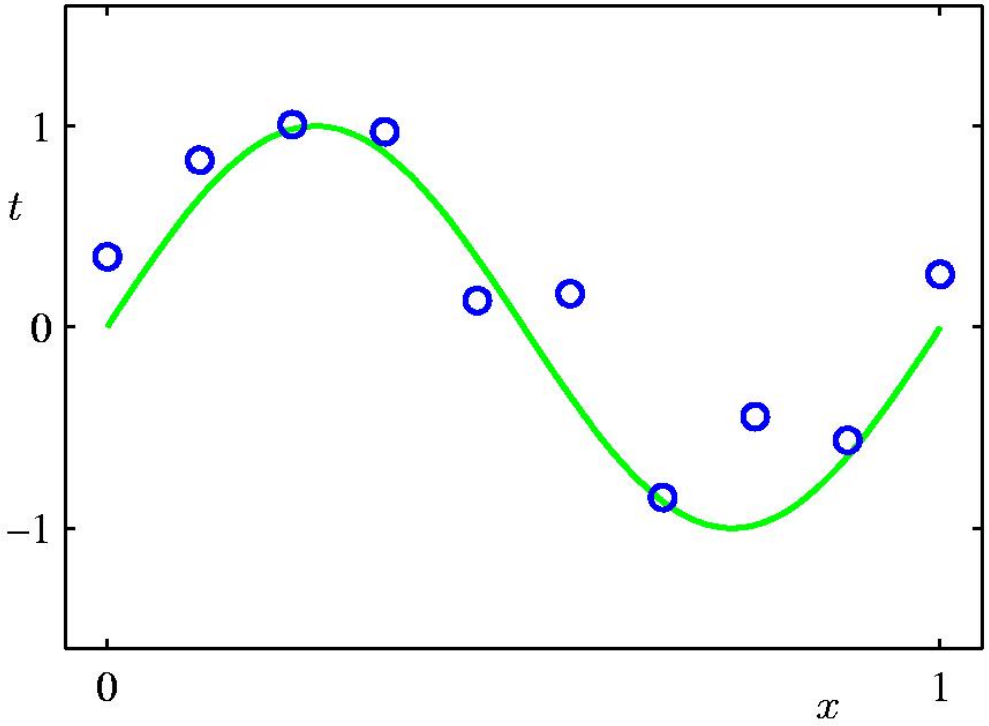


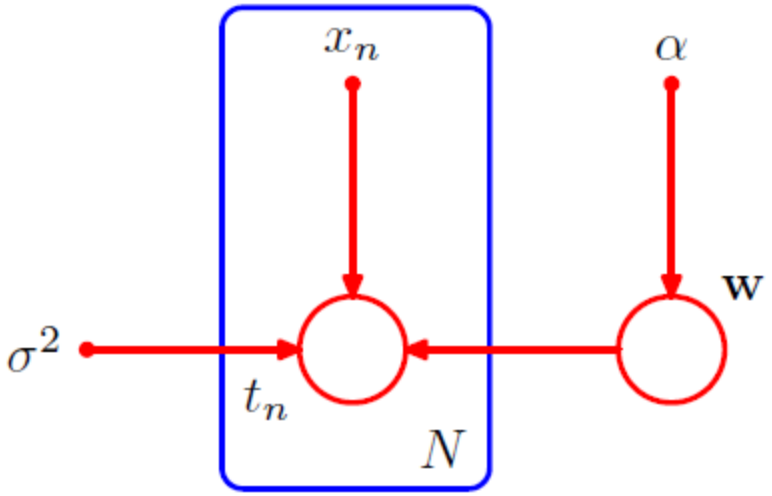


$$t(x, \mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_N h_N(x) = \sum_{j=0}^{N} w_j h_j(x)$$

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

Example: Polynomial regression





$$t(x,\mathbf{w}) = w_0 + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_N h_N(x) = \sum_{j=0}^{N} w_j h_j(x)$$

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

the noise variance *σ2, and the hyperparameter α* representing the precision of the Gaussian prior over **w**

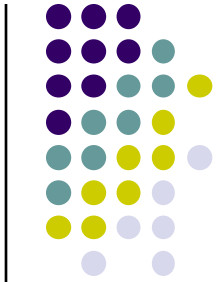
Linear-Gaussian models

- Consider an arbitrary DAG over *D* variables in which node i represents a single continuous random variable x_i having a *Gaussian distribution*

- The mean of this distribution is taken to be a linear combination of the states of its parent nodes pa_i of node i

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

Linear-Gaussian models



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

$$\begin{aligned}
\ln p(\mathbf{x}) &= \sum_{i=1}^{D} \ln p(x_i|\mathrm{pa}_i) \\
&= -\sum_{i=1}^{D} \frac{1}{2v_i}\left(x_i - \sum_{j\in\mathrm{pa}_i} w_{ij}x_j - b_i\right)^2 + \text{const}
\end{aligned}$$

Linear-Gaussian models

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

$$x_i = \sum_{j\in\mathrm{pa}_i} w_{ij}x_j + b_i + \sqrt{v_i}\epsilon_i \qquad \mathbb{E}[x_i] = \sum_{j\in\mathrm{pa}_i} w_{ij}\mathbb{E}[x_j] + b_i$$

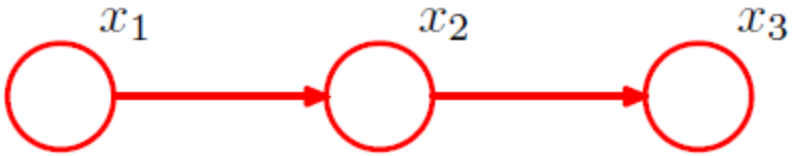
$$\begin{aligned}
\mathrm{cov}[x_i, x_j] &= \mathbb{E}\left[(x_i - \mathbb{E}[x_i])(x_j - \mathbb{E}[x_j])\right] \\
&= \mathbb{E}\left[(x_i - \mathbb{E}[x_i])\left\{\sum_{k\in\mathrm{pa}_j} w_{jk}(x_k - \mathbb{E}[x_k]) + \sqrt{v_j}\epsilon_j\right\}\right] \\
&= \sum_{k\in\mathrm{pa}_j} w_{jk}\mathrm{cov}[x_i, x_k] + I_{ij}v_j
\end{aligned}$$

Linear-Gaussian models

- Case 1: no links in the graph
  - The joint distribution:
    - 2*D* parameters and represents
    - *D* independent univariate Gaussian distributions.

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

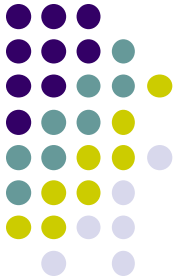
- Case 2: fully connected graph
  - $D(D\text{-}1)/2+D$ independent parameters
- Case 3:



$$\boldsymbol{\mu} = (b_1, b_2 + w_{21}b_1, b_3 + w_{32}b_2 + w_{32}w_{21}b_1)^{\mathrm{T}}$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} v_1 & w_{21}v_1 & w_{32}w_{21}v_1 \\ w_{21}v_1 & v_2 + w_{21}^2 v_1 & w_{32}(v_2 + w_{21}^2 v_1) \\ w_{32}w_{21}v_1 & w_{32}(v_2 + w_{21}^2 v_1) & v_3 + w_{32}^2(v_2 + w_{21}^2 v_1) \end{pmatrix}$$

Conditional independence



- Three random variables: a, b and c
  - a is conditionally independent of b given c iff
  - $P(a \mid b, c) = P(a \mid c)$

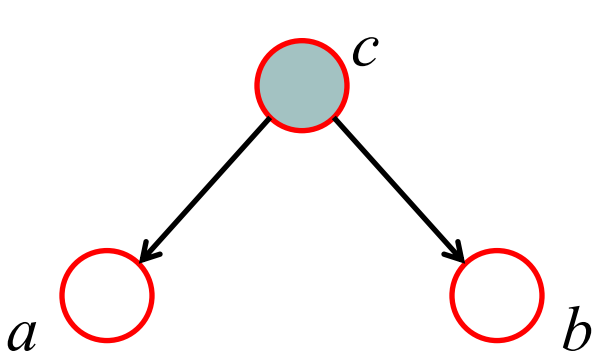
$$a \perp\!\!\!\perp b \mid c$$

- This can be re-written in following way
  - $P(a, b \mid c) = P(a \mid b, c)\, P(b \mid c)$

    $= P(a \mid c)\, P(b \mid c)$

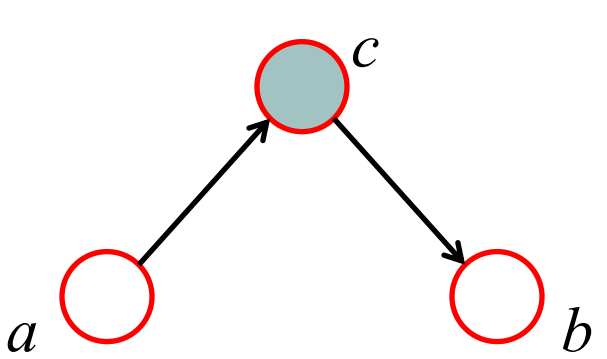
The joint distribution of a and b factorizes into the product of the marginal distribution of a and ~b.

Simple example (1)



- Joint distribution:
  - $P(a, b, c) = P(a \mid c)\, P(b \mid c)\, P(c)$
- Condition on c:
  - $P(a, b \mid c) = P(a, b, c) / P(c) = P(a \mid c)\, P(b \mid c)$
  - => $a \perp\!\!\!\perp b \mid c$

Simple example (2)



- Joint distribution:
  - $P(a, b, c) = P(a)\,P(c \mid a)\,P(b \mid c)$

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

- Factorization:

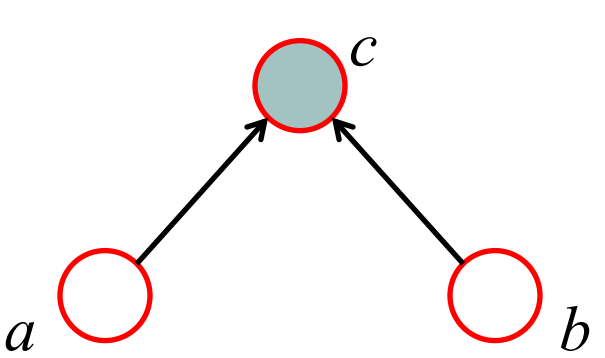
$$P(a,b) = \sum_c P(a,b,c) = P(a)\sum_c P(c \mid a)P(b \mid c)$$
$$= P(a)P(b \mid a)$$

- Condition on c:

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

→ Bayesian Theorem

Simple example (3)



- Joint distribution:
  - $P(a, b, c) = P(a)\,P(b)\,P(c \mid a, b)$

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

- Factorization:

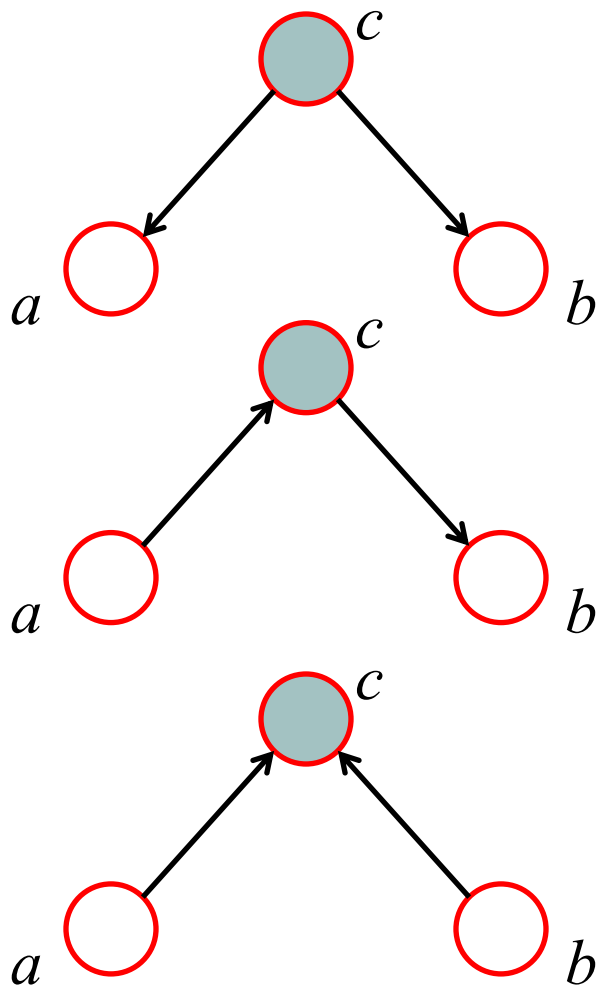
$$P(a,b) = \sum_c P(a,b,c) = P(a)P(b)\sum_c P(c \mid a,b)$$
$$= P(a)P(b)$$

- Condition on c:

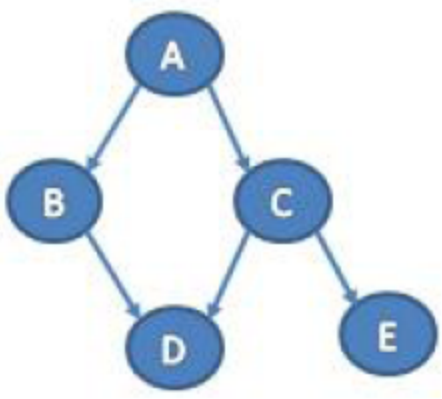
$$P(a,b \mid c) = \frac{P(a,b,c)}{P(c)} = \frac{P(a)P(b)P(c \mid a,b)}{P(c)}$$
$$\neq P(a \mid c)P(b \mid c)$$

Conditional independence

- Tail-to-Tail: yes
- Head-to-Tail: yes
- Head-to-Head: no



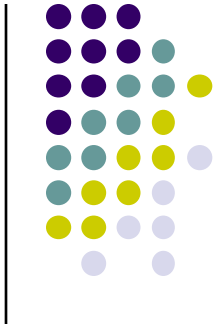
Markov condition



- We say that node *y* is a *descendant* of node *x* if there is a path from x to y in which each step of the path follows the directions of the arrows.
- If each variable is independent of its non-descendants given its parents, then:

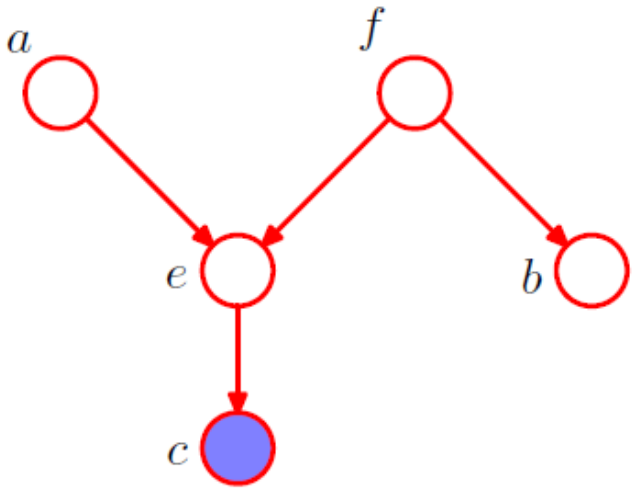
$$B \perp\!\!\!\perp (C, E) | A,$$
$$D \perp\!\!\!\perp (A, E) | (B, C),$$
$$E \perp\!\!\!\perp (A, B, D) | C.$$

D-separation

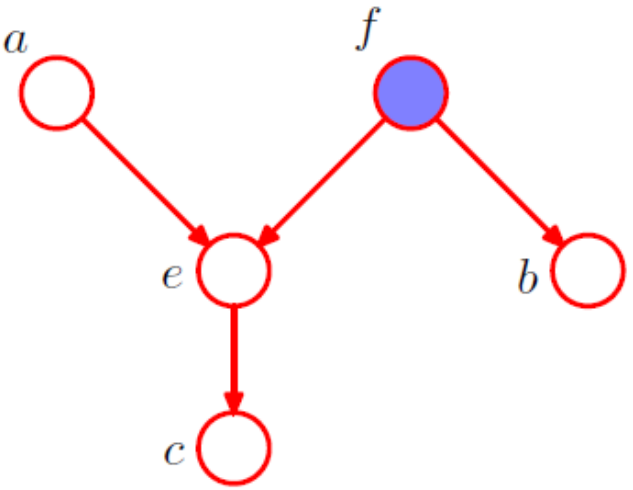


- All possible paths from any node in *A* to any node in *B.* Any such path is said to be *blocked* if it includes a node such that either
  - the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set *C,* or
  - the arrows meet head-to-head at the node, and neither the node, nor any of its descendants, is in the set *C*
- If all paths are blocked, then *A* is said to be *d-separated* from *B* by *C.*

D-separation



(a) ~~$a \perp\!\!\!\perp b \mid c$~~

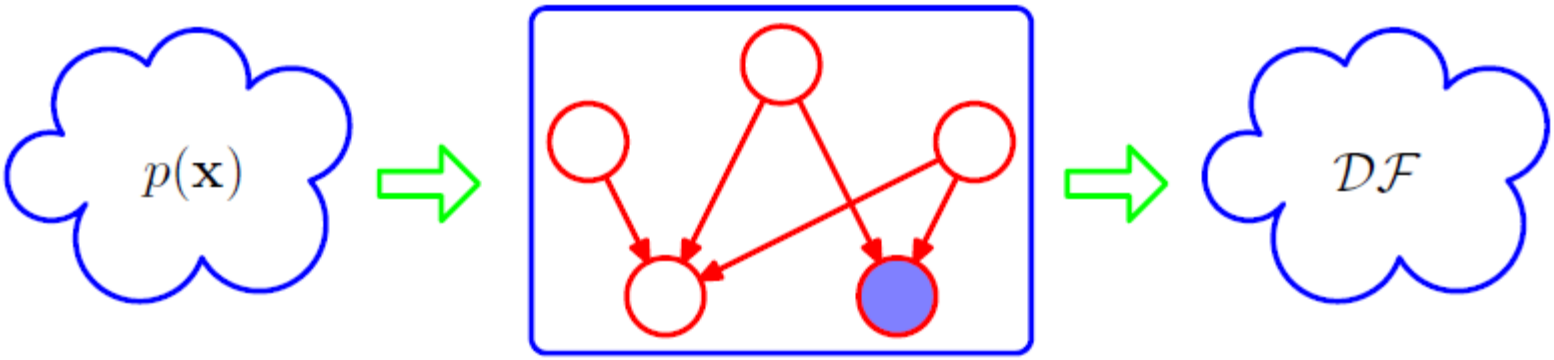


(b) $a \perp\!\!\!\perp b \mid f$

- In graph (a), the path from a *to* b is *not blocked* by node c
- In graph (b), the path from a *to* b is *blocked* by node f and e

D-separation

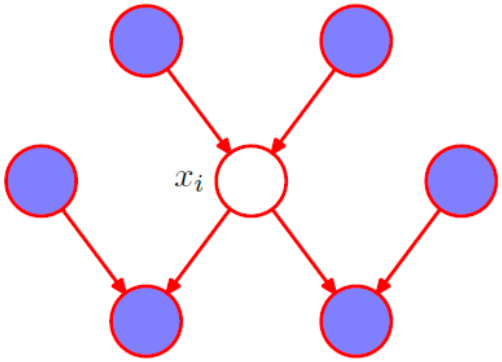
- A particular directed graph represents a specific decomposition of a joint probability distribution into a product of conditional probabilities
- A directed graph is a filter



Markov blanket

- Joint distribution $p(x_1, \ldots, x_D)$ represented by a directed graph having *D* nodes
  - conditional distribution

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



- The set of nodes comprising the parents, the children and the co-parents is called the Markov blanket
- The Markov blanket is the minimal set of nodes which d-separates node x_i from all other nodes

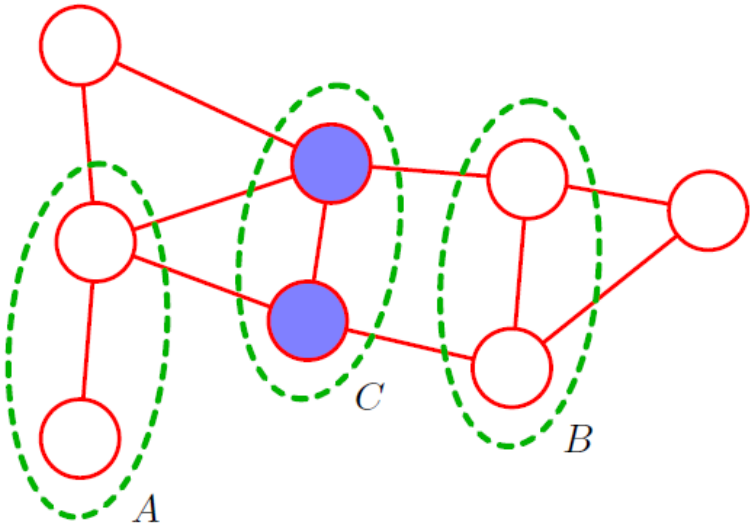
Markov Random Fields

- Also known as a *Markov network or an undirected graphical model*

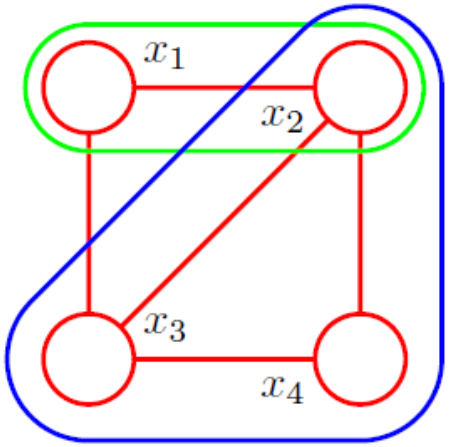
- Conditional independence properties:

Conditional dependence exists if there exists a path that connects any node in *A to any* node in *B.*

If there are no such paths, then the conditional independence property must hold.

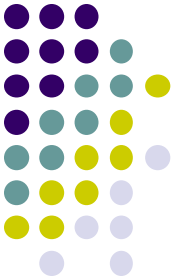


Clique



- A subset of the nodes in a graph such that there exists a link between all pairs of nodes in the subset
  - In other words, the set of nodes in a clique is fully connected
  - Maximal clique …
  - A four-node undirected graph showing a clique (outlined in green) and a maximal clique (outlined in blue)

Potential function



- $\boldsymbol{x}_C$: the set of variables in that clique C
- The joint distribution is written as a product of *potential functions* $\psi_C(\boldsymbol{x}_C)$ over the maximal cliques of the graph

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

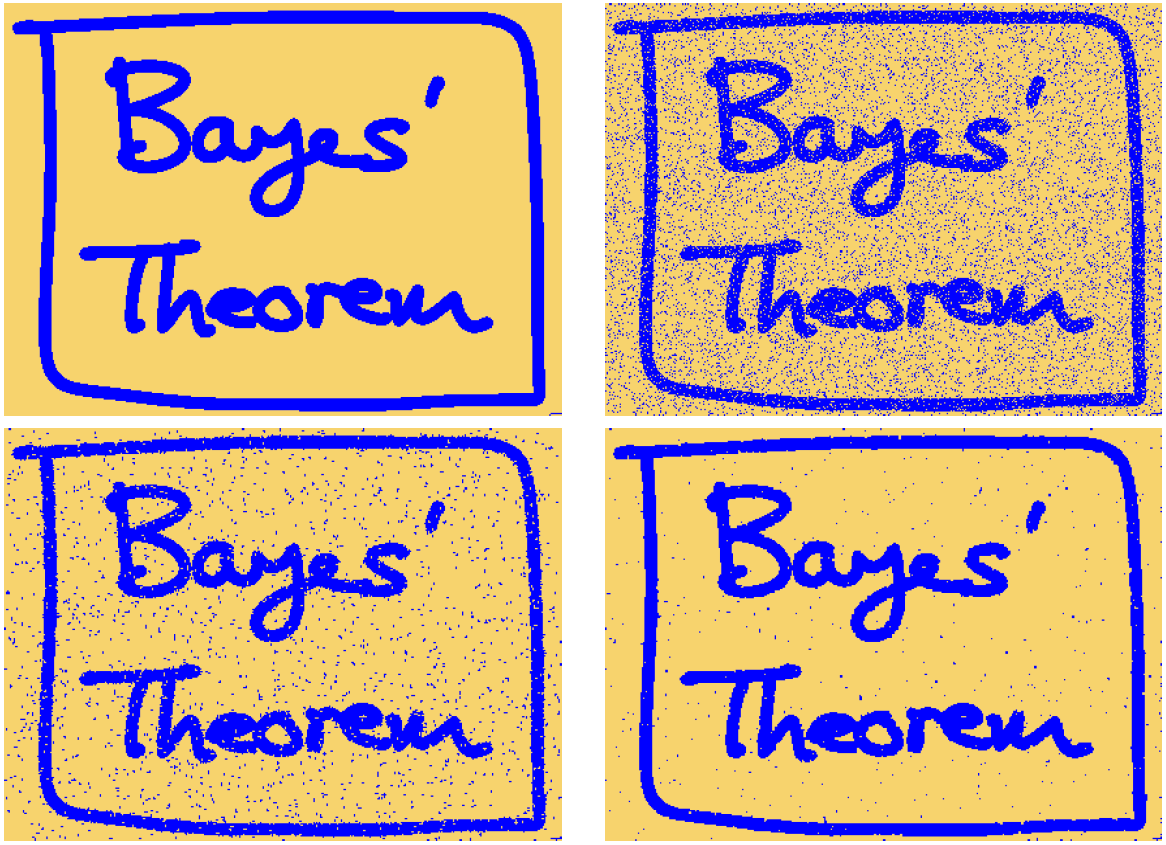
- The quantity *Z,* called the *partition function,* is a normalization constant

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

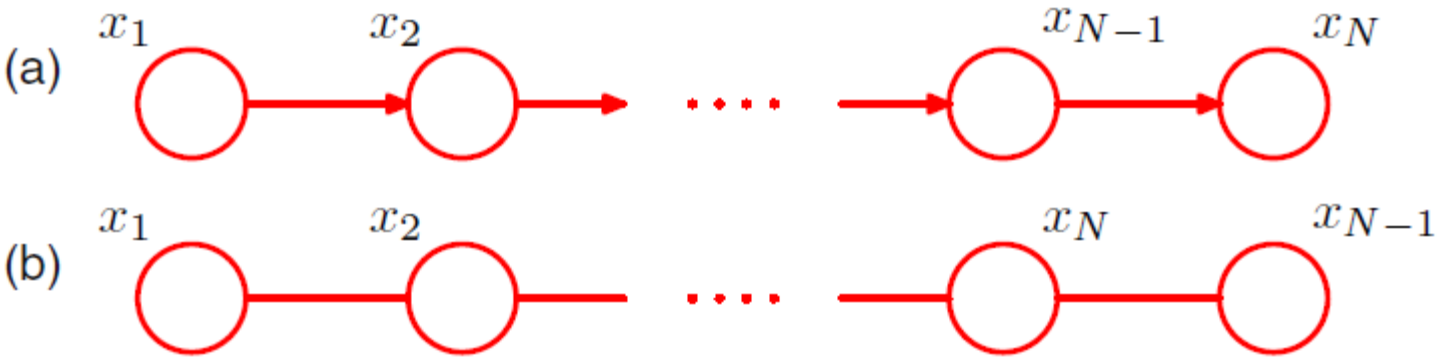
- Potential functions $\psi_C(\boldsymbol{x}_C)$ are **strictly positive.** Possible choice

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

Image de-noising



Relation to directed graphs



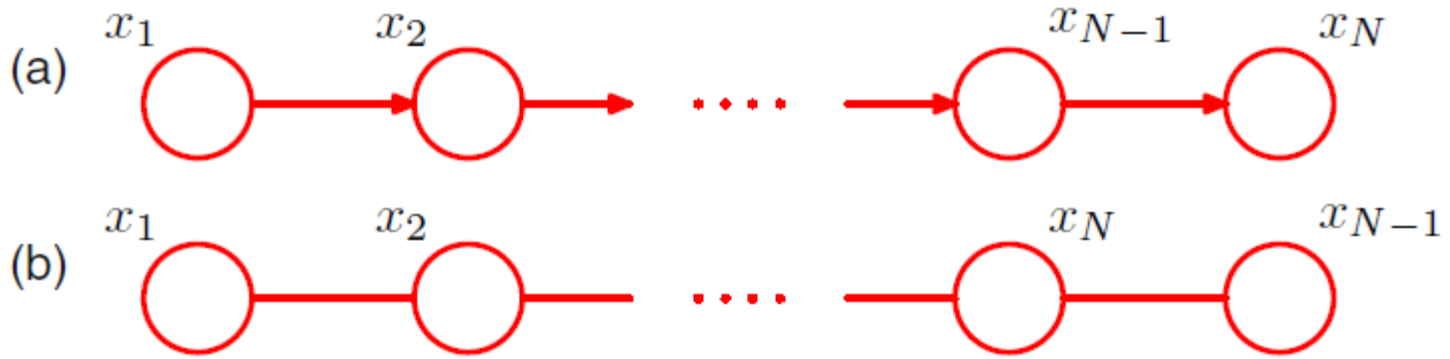
- Joint distribution:
  - Directed:

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

  - Undirected:

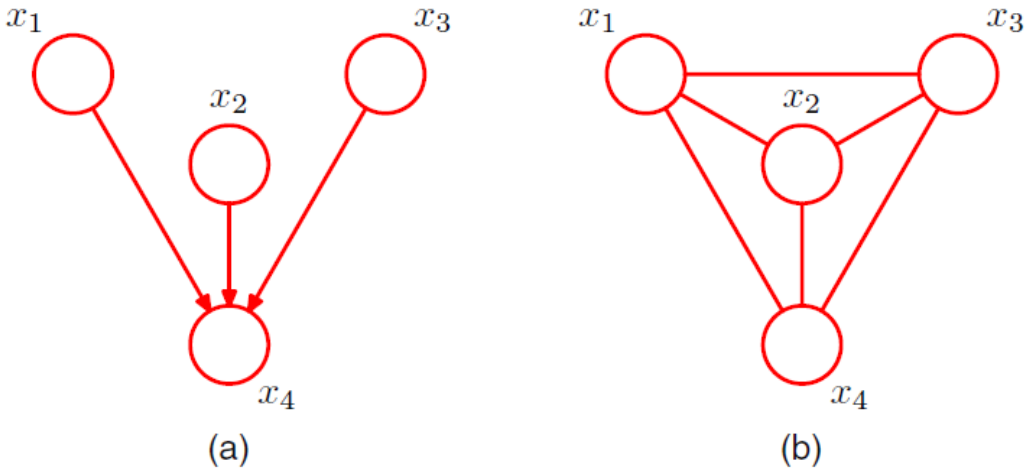
$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

Relation to directed graphs



$$\begin{aligned}
\psi_{1,2}(x_1, x_2) &= p(x_1)p(x_2|x_1) \\
\psi_{2,3}(x_2, x_3) &= p(x_3|x_2) \\
&\vdots \\
\psi_{N-1,N}(x_{N-1}, x_N) &= p(x_N|x_{N-1})
\end{aligned}$$

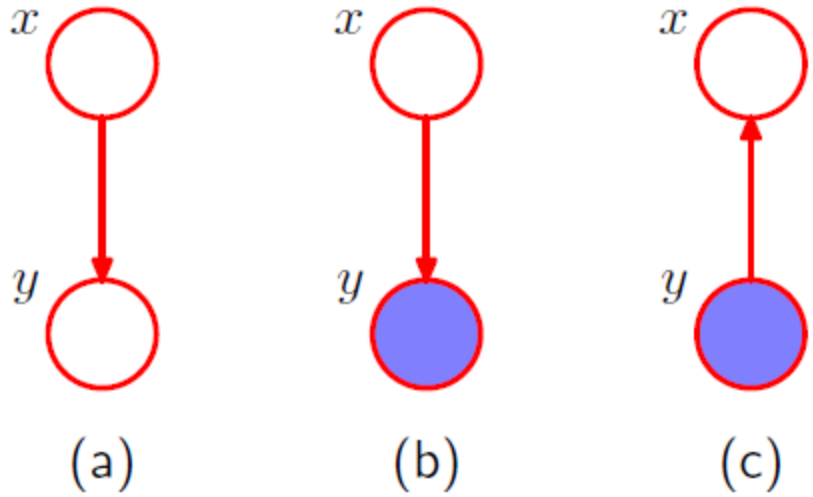
Relation to directed graphs



$$p(\mathbf{x}) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)$$

- this process of 'marrying the parents' has become known as *moralization*, and the resulting undirected graph, after dropping the arrows, is called the *moral graph*.

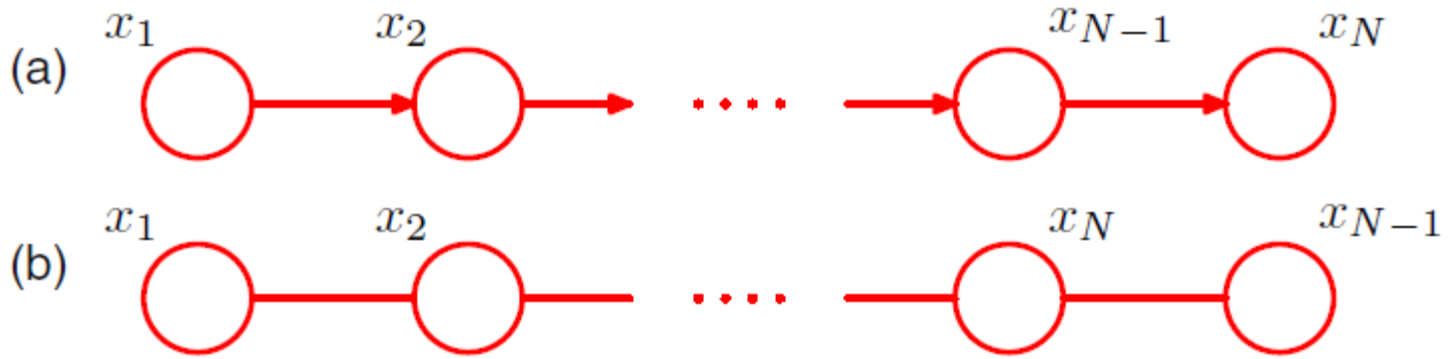
Inference in Graphical Models



$$p(x, y) = p(x)p(y|x)$$

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

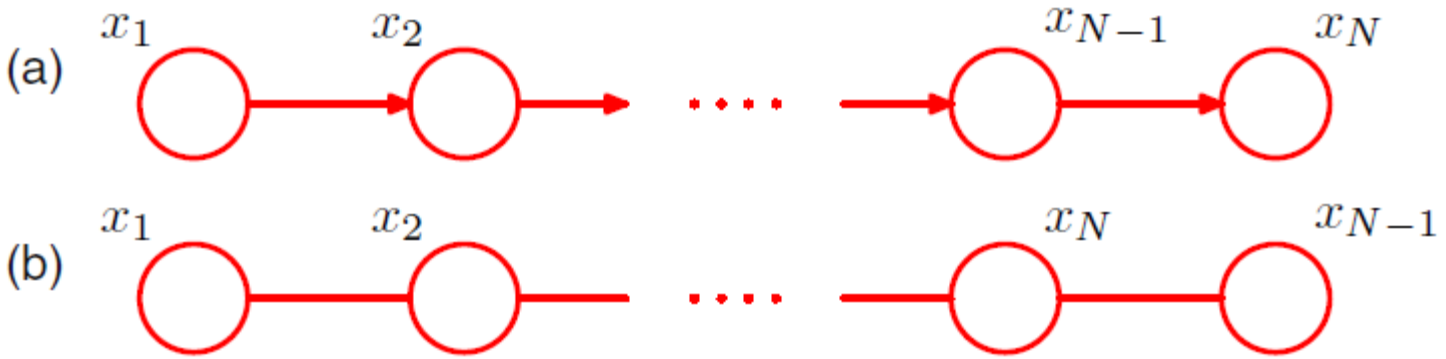
Inference on a chain



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

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

Inference on a chain

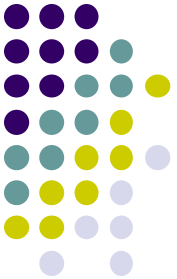


$$p(x_n) = \frac{1}{Z}$$

$$\underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2)\right]\right] \cdots\right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N)\right] \cdots\right]}_{\mu_\beta(x_n)}. \qquad (8.52)$$

Inference on a chain



Passing of local *messages* around on the graph

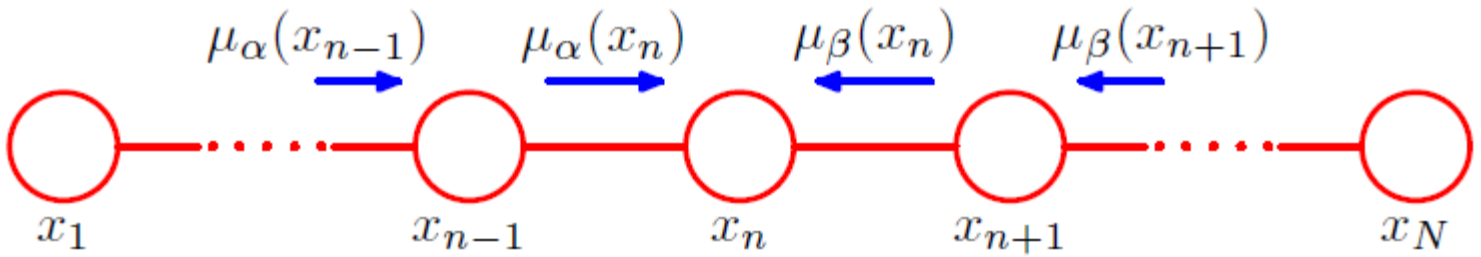
$$p(x_n) = \frac{1}{Z} \underbrace{\left[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \cdots \left[\sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \right] \cdots \right]}_{\mu_\alpha(x_n)}$$

$$\underbrace{\left[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \cdots \left[\sum_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \cdots \right]}_{\mu_\beta(x_n)}. \qquad (8.52)$$

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

Inference on a chain

Passing of local *messages* around on the graph



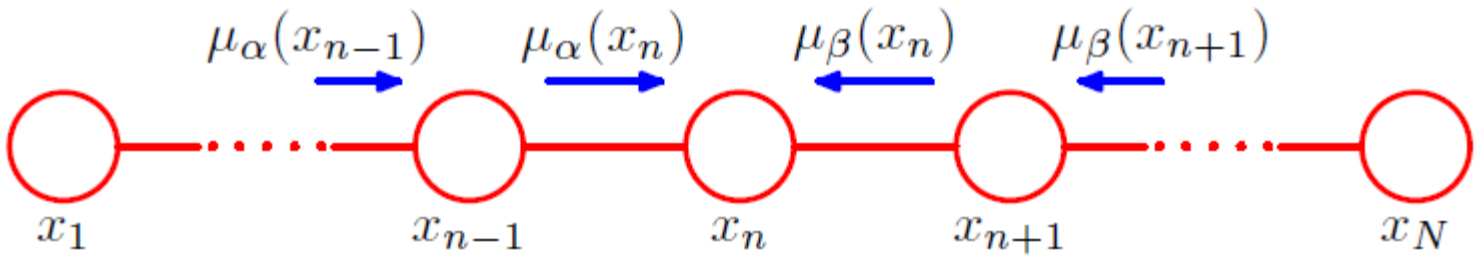
$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \left[\sum_{x_{n-2}} \cdots \right]$$

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)\mu_\alpha(x_{n-1}).$$

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$$

Inference on a chain

Passing of local *messages* around on the graph

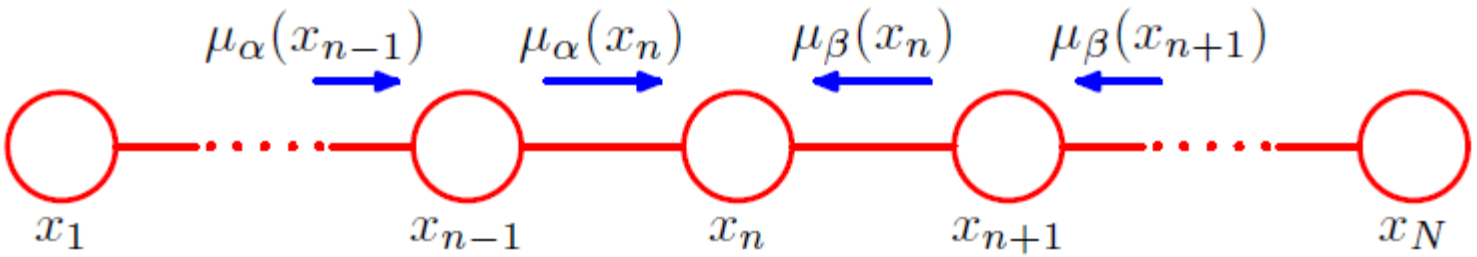


$$\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n) \left[\sum_{x_{n+2}} \cdots \right]$$

$$p(x_n) = \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n) = \sum_{x_{n+1}} \psi_{n+1,n}(x_{n+1}, x_n)\mu_\beta(x_{n+1}).$$

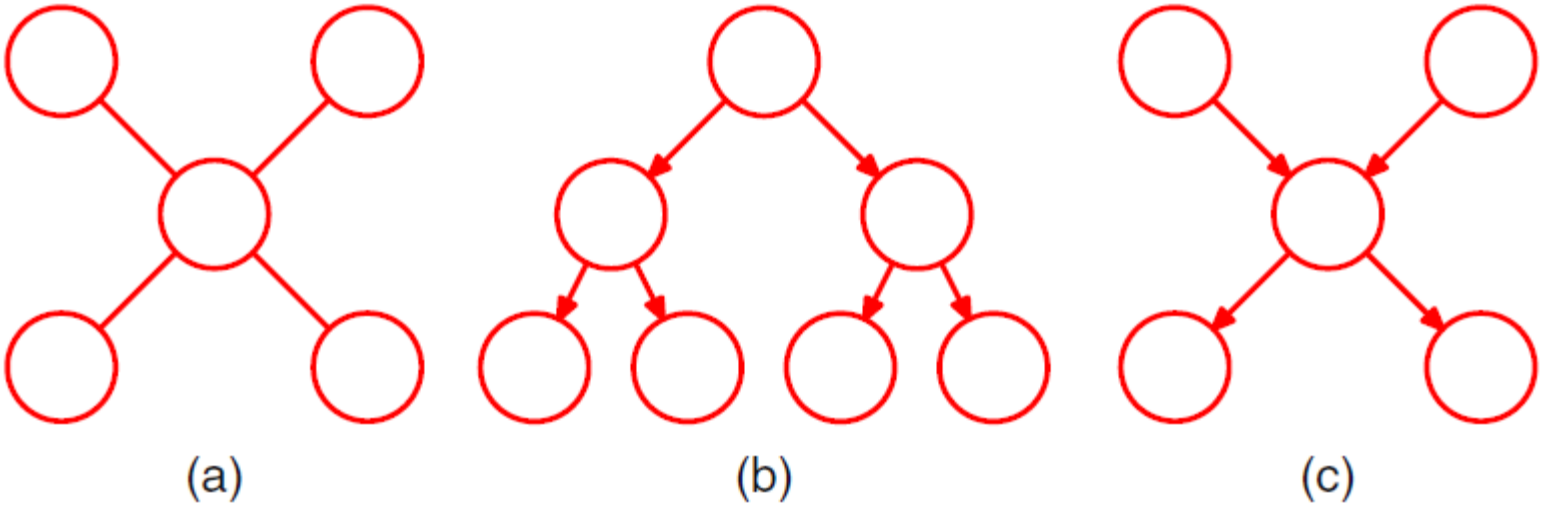
Inference on a chain

Passing of local *messages* around on the graph



$$p(x_{n-1}, x_n) = \frac{1}{Z}\mu_\alpha(x_{n-1})\psi_{n-1,n}(x_{n-1}, x_n)\mu_\beta(x_n)$$

Tree



Factor graph

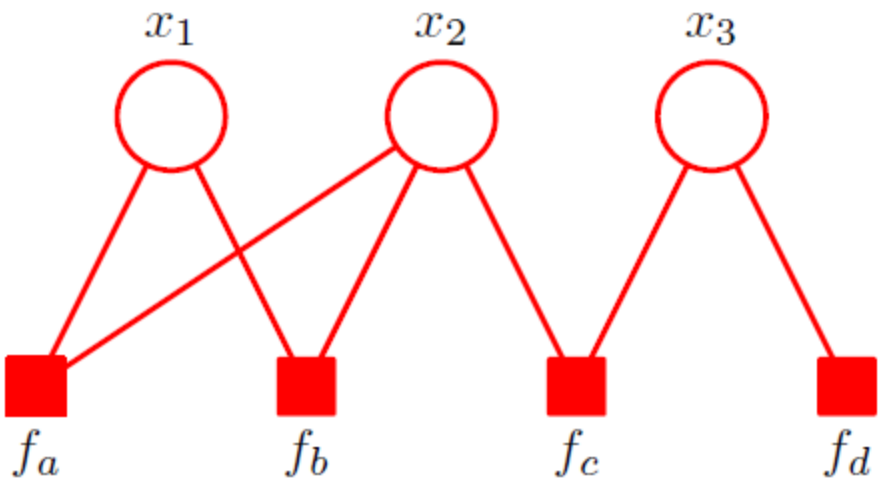
- the joint distribution over a set of variables in the form of a product of factors

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

- where $\mathbf{x}_s$ denotes a subset of the variables

Factor graph

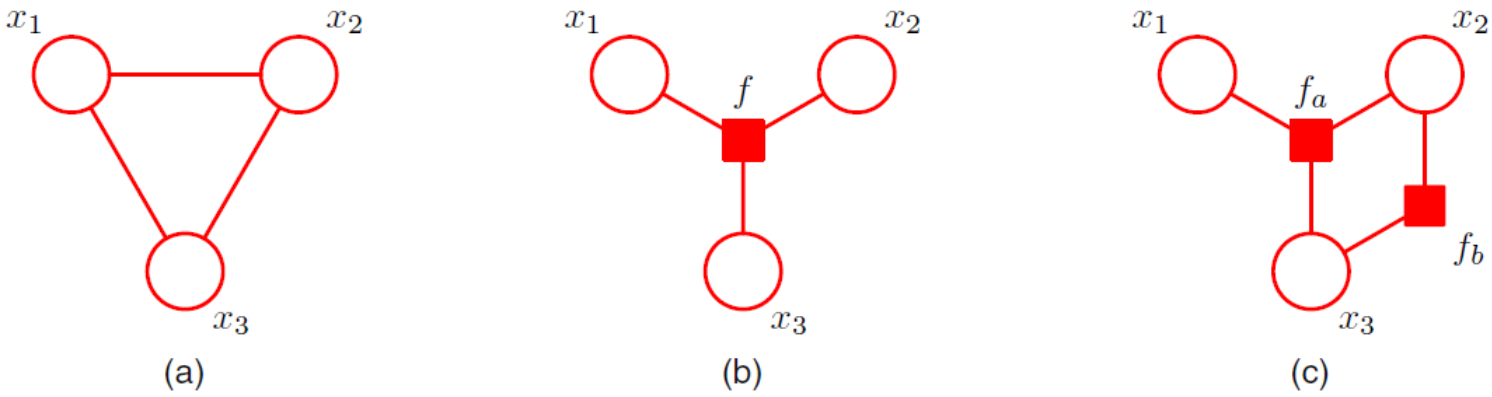
$$p(\mathbf{x}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$



Factor graph

- an undirected graph => a factor graph
  - create variable nodes corresponding to the nodes in the original undirected graph
  - create additional factor nodes corresponding to the maximal cliques $\mathbf{x}_s$
  - Multiple choices of f_{g}

Factor graph



(a) An undirected graph with a single clique potential *ψ(x1, x2, x3).*

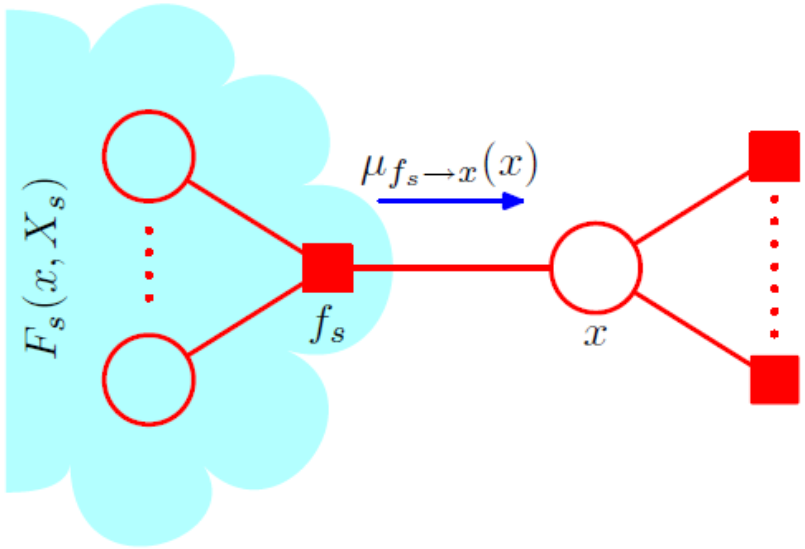
(b) A factor graph with factor *f(x1, x2, x3) = ψ(x1, x2, x3)* representing the same distribution as the undirected graph.

(c) A *different factor* graph representing the same distribution, whose factors satisfy *fa(x1, x2, x3)fb(x1, x2) = ψ(x1, x2, x3).*

The sum-product algorithm

- The problem of finding the marginal $p(x)$ for particular variable node x

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

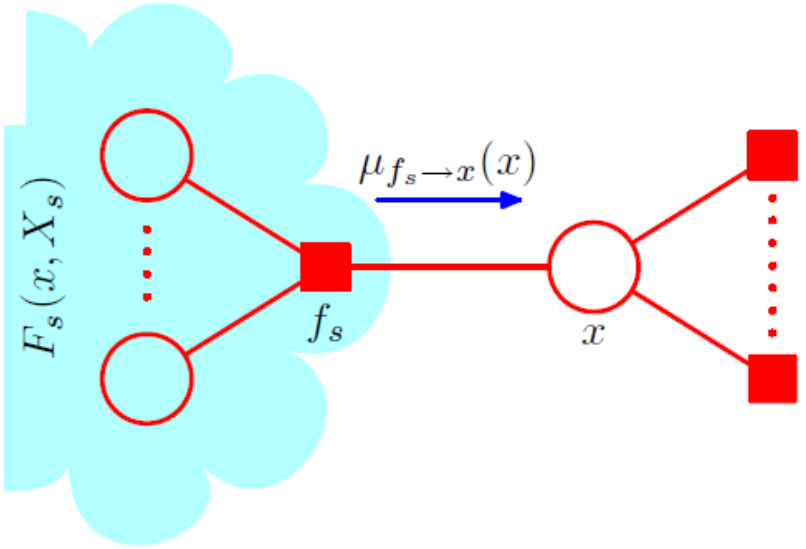


The sum-product algorithm

- The problem of finding the marginal $p(x)$ for particular variable node x

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

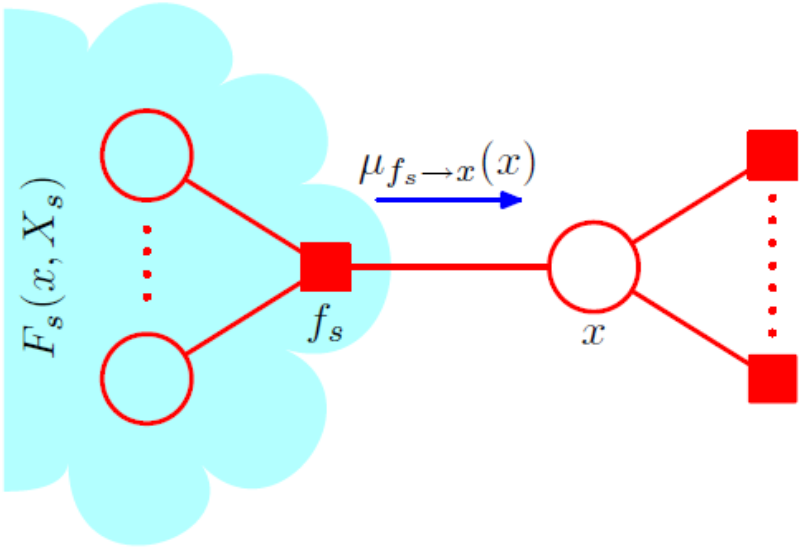
$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$



The sum-product algorithm

- The problem of finding the marginal $p(x)$ for particular variable node x

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

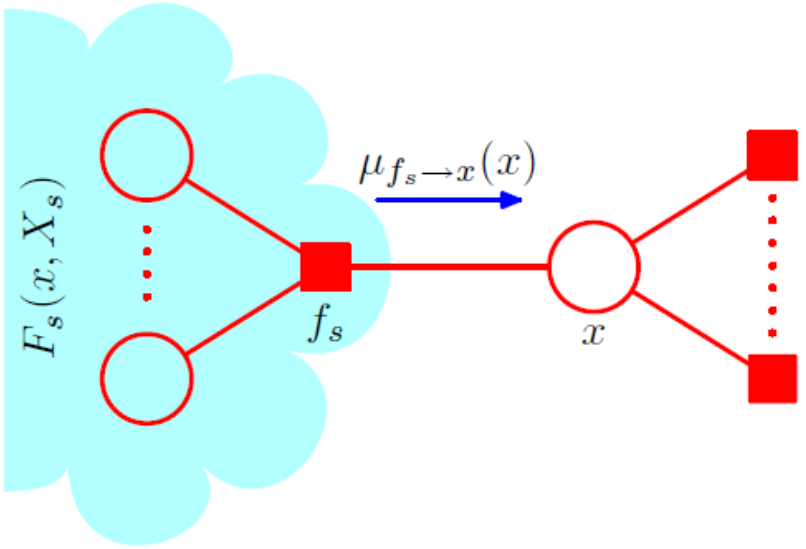


The sum-product algorithm

- The problem of finding the marginal $p(x)$ for particular variable node x

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

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



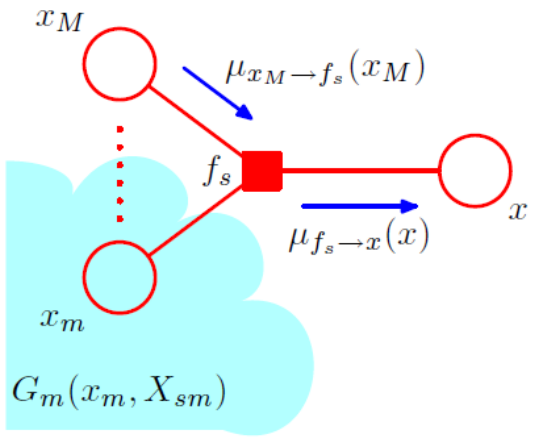
The sum-product algorithm

- The problem of finding the marginal $p(x)$ for particular variable node x

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

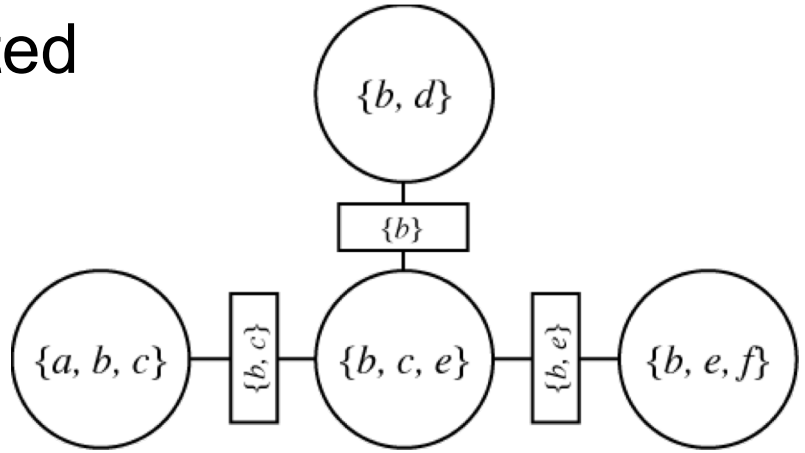
$$= \sum_{x_1} \cdots \sum_{x_M} f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} \mu_{x_m \to f_s}(x_m) \qquad (8.66)$$

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm})$$



Junction tree algorithm

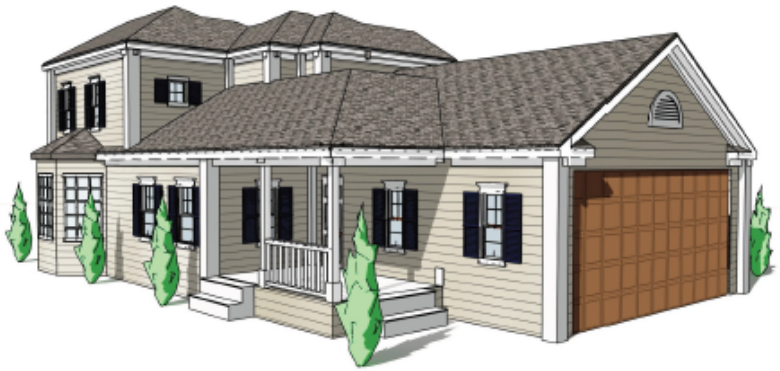
- Deal with graphs having loops
- Algorithm:
  - directed graph => undirected graph (moralization)
  - The graph is triangulated
  - join tree
  - Junction tree
  - a two-stage message passing algorithm, essentially equivalent to the sum-product algorithm

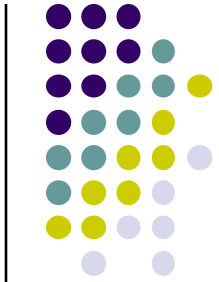


http://ai.stanford.edu/~paskin/gm-short-course/lec3.pdf
https://www.cs.helsinki.fi/u/bmmalone/probabilistic-models-spring-

Graph inference example

- Computer-Generated Residential Building Layouts [SIG ASIA 2010]





MARKOV CHAINS AND HMM

Example: Video Textures

- Problem statement

video clip video texture

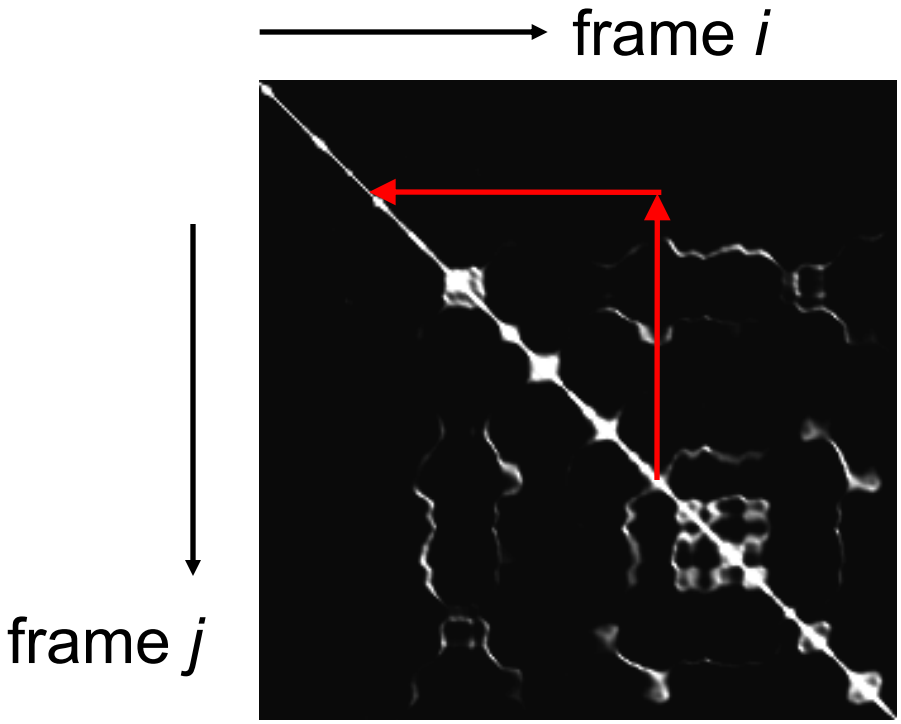
SIGGRAPH 2000. Schoedl et. al.

The approach

How do we find good transitions?

Finding good transitions

Compute L_2 distance $D_{i,j}$ between all frames



Similar frames make good transitions

Demo: Fish Tank

Mathematic model of Video Texture

A sequence of random variables

{ADEABEDADBCAD}

A sequence of random variables

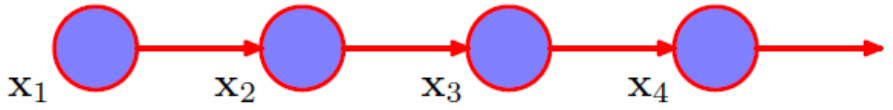
{BDACBDCACDBCADCBADCA}

Markov Model

The future is independent of the past and given by the present.

Markov Property

- Formal definition



  - Let $X=\{X_n\}_{n=0\ldots N}$ be a sequence of random variables taking values $s_k \in N$ if and only if

$$P(X_m=s_m|X_0=s_0,\ldots,X_{m-1}=s_{m-1}) = P(X_m=s_m|\, X_{m-1}=s_{m-1})$$

    then the X fulfills Markov property

- Informal definition
  - The future is independent of the past given the present.

History of MC

- Markov chain theory developed around 1900.
- Hidden Markov Models developed in late 1960's.
- Used extensively in speech recognition in 1960-70.
- Introduced to computer science in 1989.

Andrei Andreyevich Markov

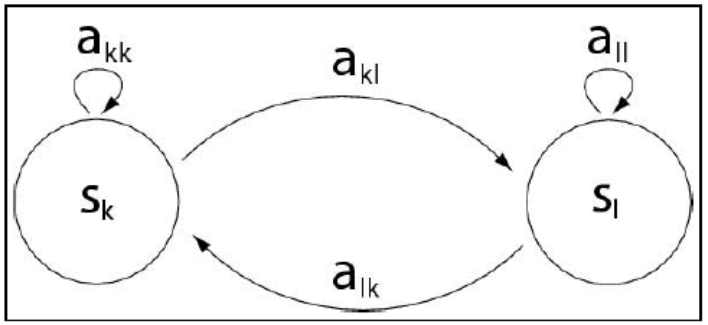
Applications

- Bioinformatics.
- Signal Processing
- Data analysis and Pattern recognition

Markov Chain

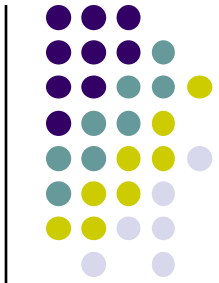
- A Markov chain is specified by
  - **state space** $S = \{ s_1, s_2..., s_n \}$
  - **initial distribution** a_0
  - **transition matrix** A

Where $A(n)_{ij} = a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$



- Graphical Representation
  as a directed graph where
  - Vertices represent states
  - Edges represent transitions with positive probability

Probability Axioms



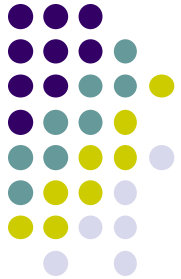
- Marginal Probability – sum the joint probability

$$P(x = a_i) \equiv \sum_{y \in A_Y} P(x = a_i, y)$$

- Conditional Probability

$$P(x = a_i \mid y = b_j) \equiv \frac{P(x = a_i, y = b_j)}{P(y = b_j)} \text{ if } P(y = b_j) \neq 0.$$

Calculating with Markov chains



- Probability of an observation sequence:
  - Let $X=\{x_t\}^L_{t=0}$ be an observation sequence from the Markov chain $\{S, a_0, A\}$

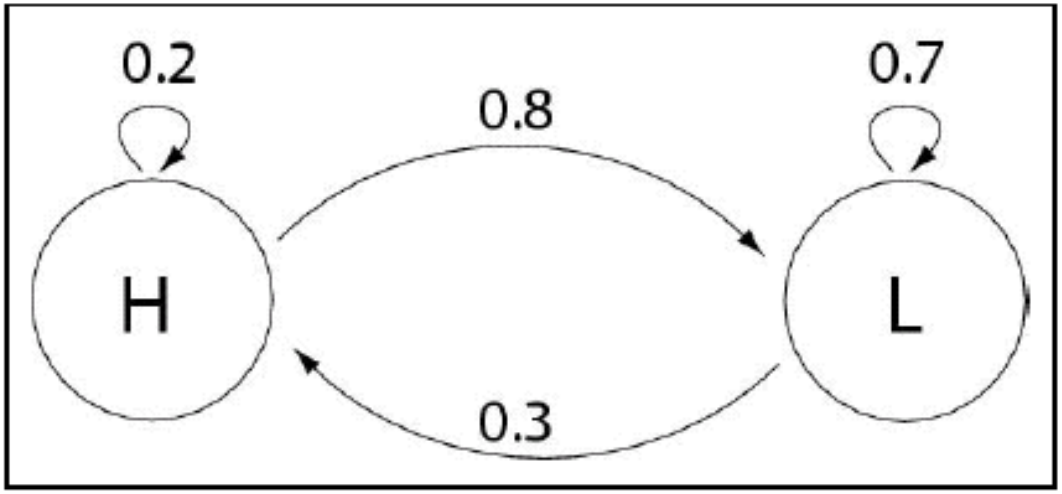
$$
\begin{aligned}
P(x) &= P(x_L, \ldots, x_1, x_0) \\
&= P(x_L \mid x_{L-1}, \ldots, x_0) P(x_{L-1} \mid x_{L-2}, \ldots, x_0) \cdots P(x_0) \\
&= P(x_L \mid x_{L-1}) P(x_{L-1} \mid x_{L-2}) \cdots P(x_0) \\
&= \mathbf{b}_{x_0} \prod_{i=1}^{L} a_{x_{i-1} x_i}
\end{aligned}
$$

Example

Assume we are modeling a time series of high and low pressures during the Danish autumn.

Let $S = \{H, L\}$, $\mathbf{b} = \pi = \left[\frac{3}{11}, \frac{8}{11}\right]$, and $A = \begin{bmatrix} 0.2 & 0.8 \\ 0.3 & 0.7 \end{bmatrix}$.

Graphical representation of A

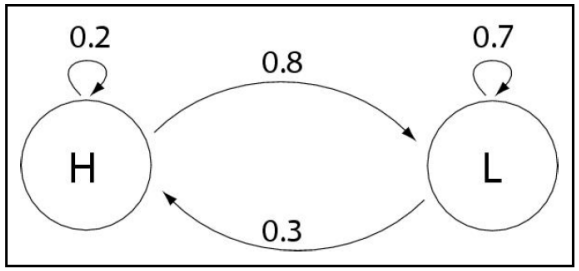


Example

Comparing likelihoods

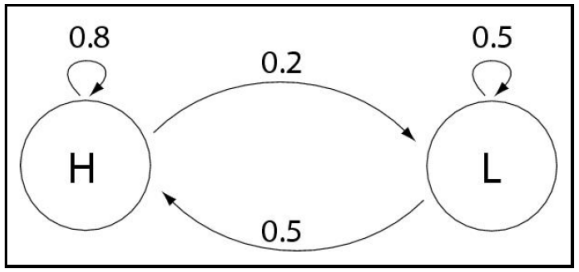
We want to know the likelihood of one week of high pressure in Denmark (DK) versus California (Cal).

x=HHHHHHH



$$P(x \mid DK) = \mathbf{b}_H a_{HH} a_{HH} a_{HH} a_{HH} a_{HH} a_{HH} a_{HH}$$

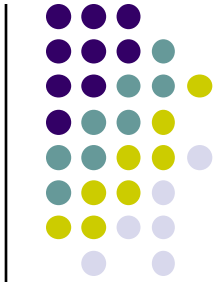
$$= \frac{3}{11}\left(\frac{1}{5}\right)^6 \approx 0.0017\%$$



$$P(x \mid Cal) = \mathbf{b}_H a_{HH} a_{HH} a_{HH} a_{HH} a_{HH} a_{HH} a_{HH}$$

$$= \frac{5}{7}\left(\frac{4}{5}\right)^6 \approx 0.19\%$$

Motivation of Hidden Markov Models

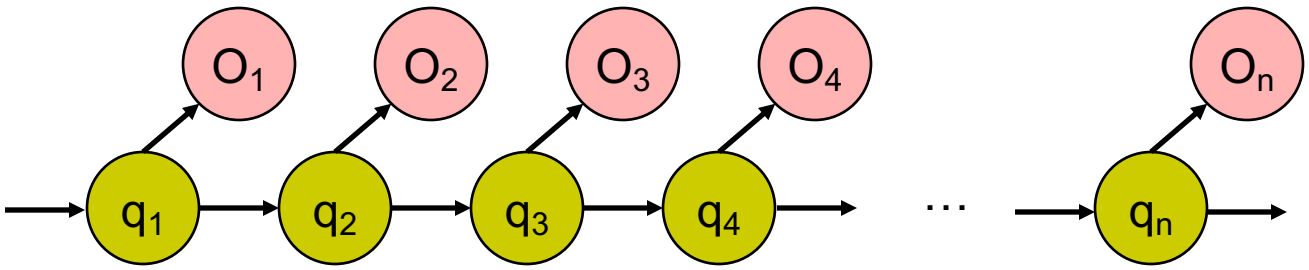


- **Hidden states**
  - The state of the entity we want to model is often not observable:
    - The state is then said to be hidden.
- **Observables**
  - Sometimes we can instead observe the state of entities influenced by the hidden state.
- **A system can be modeled by an HMM if:**
  - The sequence of hidden states is Markov
  - The sequence of observations are independent (or Markov) given the hidden

Hidden Markov Model

- Definition $M=\{S, V, A, B, \pi\}$
  - **Set of states** $S = \{ s_1, s_2, \ldots, s_N\}$
  - **Observation symbols** $V = \{ v_1, v_2, \ldots, v_M\}$
  - **Transition probabilities**
    - A between any two states $a_{ij} = P(q_t=s_j|q_{t-1}=s_i)$
  - **Emission probabilities**
    - B within each state $b_j(O_t) = P(O_t=v_j| q_t = s_j)$
  - **Start probabilities** $\pi = \{a_0\}$

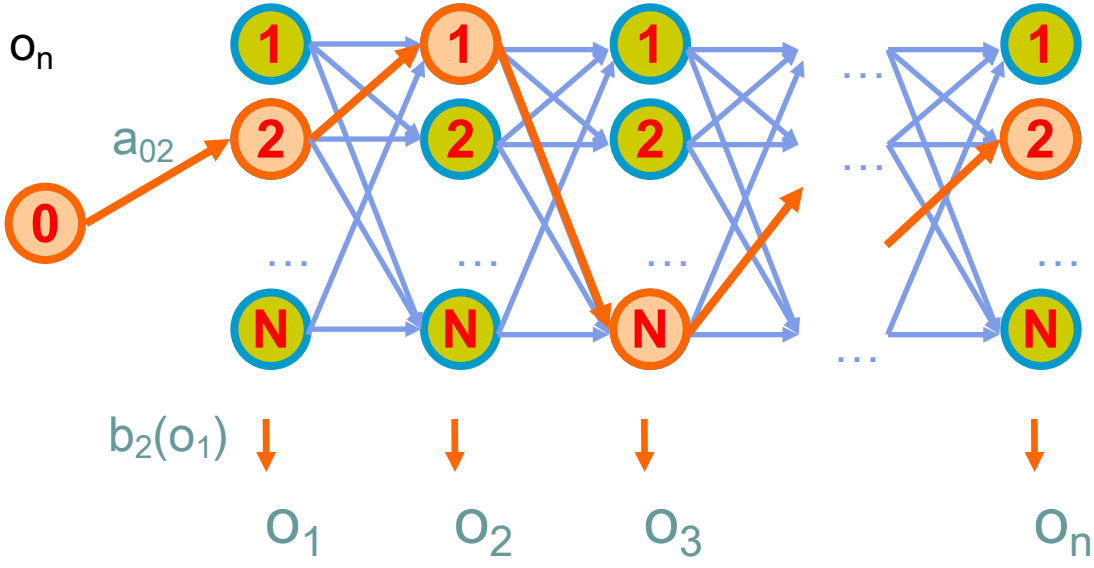
Use $\theta = (A, B, \pi)$ to indicate the parameter set of the model.



Generating a sequence by the model

Given a HMM, we can generate a sequence of length n as follows:

1. Start at state q_1 according to prob a_{0t1}
2. Emit letter o_1 according to prob $e_{t1}(o_1)$
3. Go to state q_2 according to prob a_{t1t2}
4. … until emitting o_n



Example

Model of high and low pressures

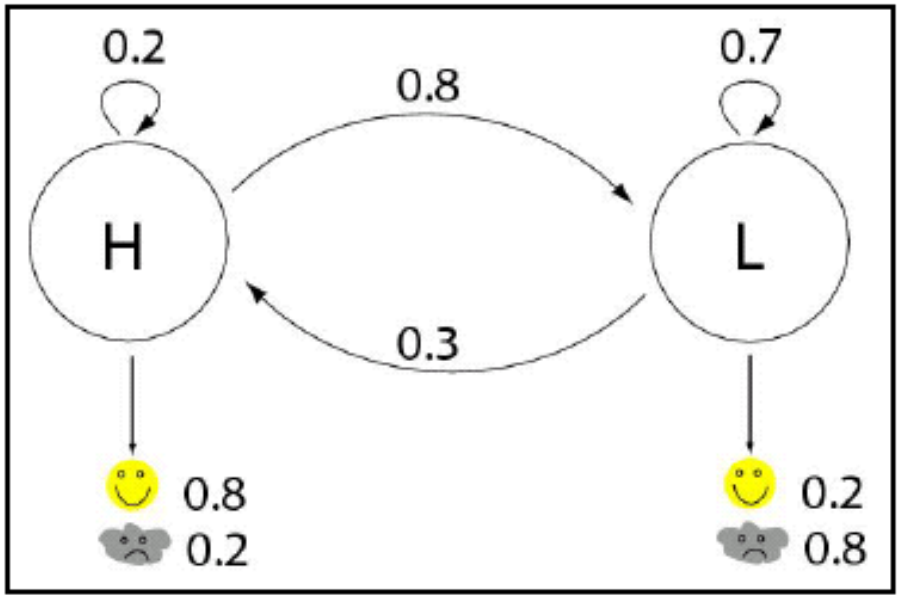
Assume we can not measure high and low pressures.

The state of the weather is influenced by the air pressure.

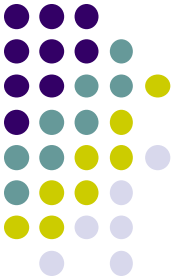
We make an HMM with hidden states representing high and low pressure and observations representing the weather:

Hidden states: L L L L H H L

Observations: 



Calculating with Hidden Markov Model



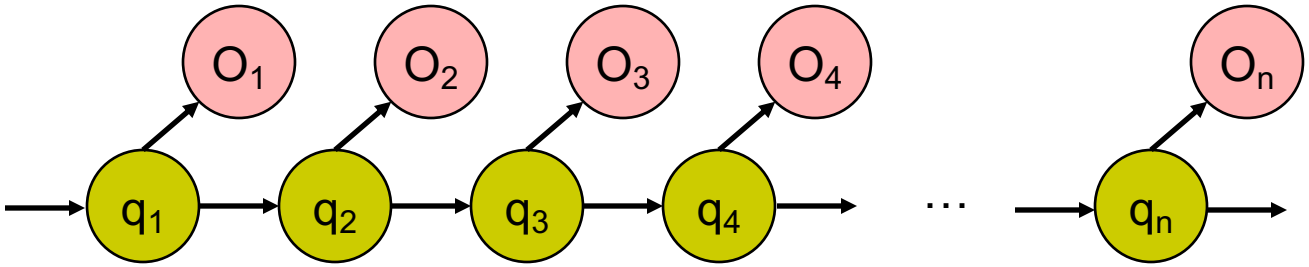
Consider one such fixed state sequence

$$Q = q_1 q_2 \cdots q_T$$

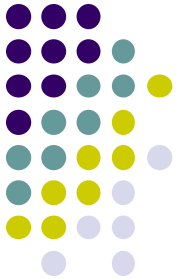
The observation sequence *O* for the *Q* is

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda)$$

$$= b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T)$$



Calculating with Hidden Markov Model (cont.)



The probability of such a state sequence Q

$$P(Q \mid \lambda) = a_{0q_1} a_{q_1q_2} \cdot a_{q_2q_3} \cdots a_{q_{T-1}q_T}$$

The probability that O and Q occur simultaneously, is simply the product of the above two terms, i.e.,

$$P(O, Q \mid \lambda) = P(O \mid Q, \lambda) P(Q \mid \lambda)$$

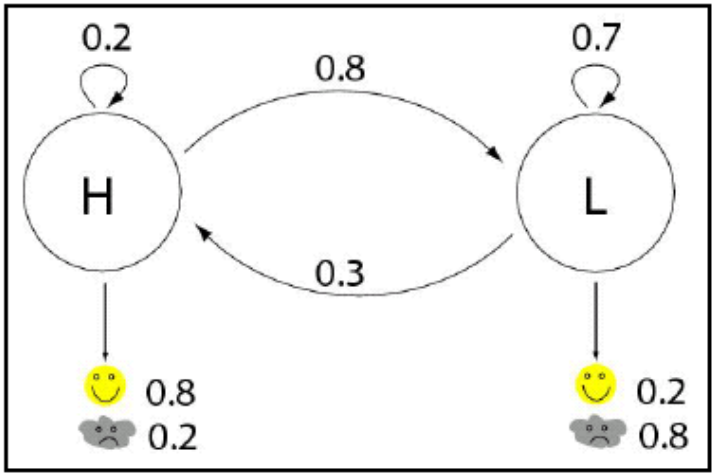
$$P(O, Q \mid \lambda) = a_{0q_1} b_{q_1}(O_1) a_{q_1q_2} b_{q_2}(O_2) a_{q_2q_3} \cdots a_{q_{T-1}q_T} b_{q_T}(O_T)$$

Example

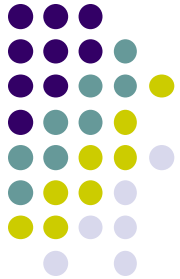
$$
\begin{aligned}
&P(x,\pi)\\
&= (a_{0L}e_L(R))(a_{LL}e_L(R))(a_{LL}e_L(S))(a_{LL}e_L(R))(a_{LH}e_H(S))(a_{HH}e_H(S))(a_{HL}e_L(R))\\
&= \left(\frac{8}{11}\frac{8}{10}\right)\left(\frac{7}{10}\frac{8}{10}\right)\left(\frac{7}{10}\frac{2}{10}\right)\left(\frac{7}{10}\frac{8}{10}\right)\left(\frac{3}{10}\frac{8}{10}\right)\left(\frac{2}{10}\frac{8}{10}\right)\left(\frac{8}{10}\frac{8}{10}\right)\\
&= 0.0006278
\end{aligned}
$$

Hidden states: L L L L H H L

Observations: ☹ ☹ ☺ ☹ ☺ ☺ ☹



The three main questions on HMMs



1. **Evaluation**

   GIVEN a HMM $M=(S, V, A, B, \pi)$, and a sequence O,
   FIND $P[O|M]$

2. **Decoding**

   GIVEN a HMM $M=(S, V, A, B, \pi)$, and a sequence O,
   FIND the sequence Q of states that maximizes $P(O, Q|\lambda)$

3. **Learning**

   GIVEN a HMM $M=(S, V, A, B, \pi)$, with unspecified transition/emission probabilities and a sequence Q,
   FIND parameters $\theta = (e_i(.), a_{ij})$ that maximize $P[X|\theta]$

Evaluation

- Find the likelihood a sequence is generated by the model

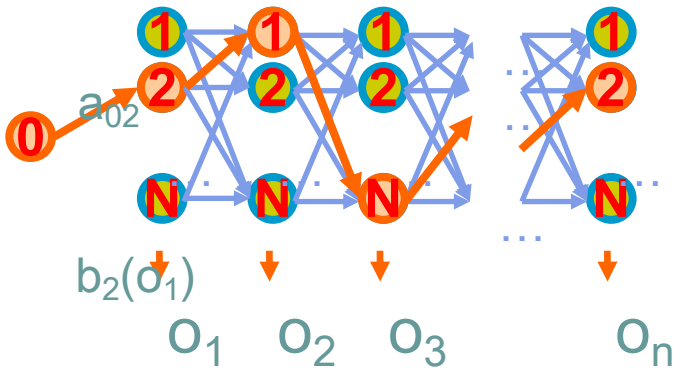
- A straightforward way （穷举法）
  - The probability of *O* is obtained by summing all possible state sequences *q* giving

$$P(O|\lambda) = \sum_{all\ Q} P(O|Q,\lambda)P(Q|\lambda)$$

$$= \sum_{q_1,q_2,\ldots q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) a_{q_2 q_3} \cdots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

Complexity is *O(N^T)*

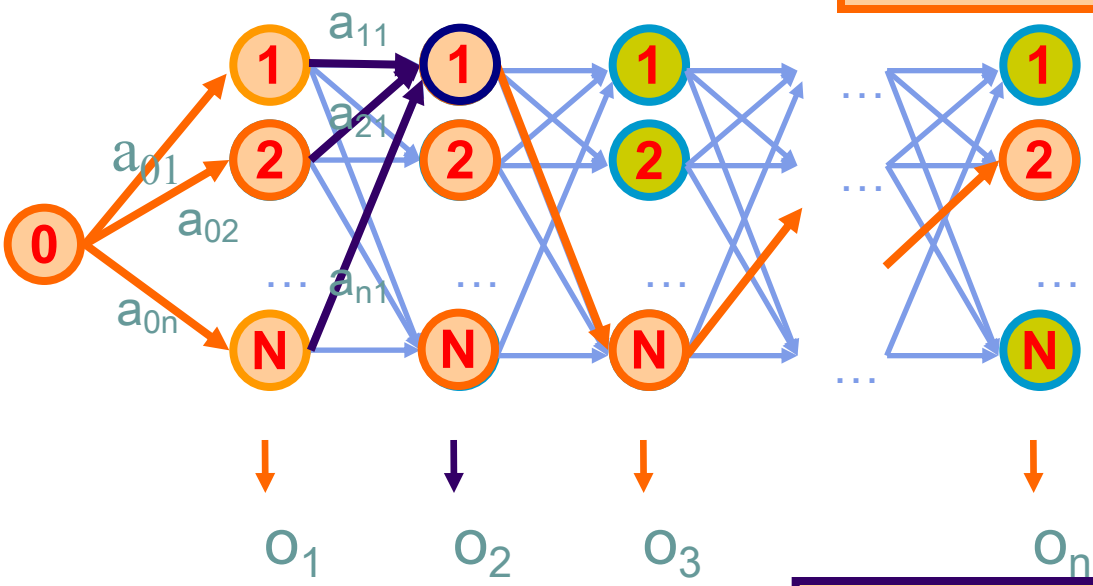
Calculations is unfeasible



The Forward Algorithm

- A more elaborate algorithm
  - The Forward Algorithm

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$



$$\alpha_2(1) = [\sum_{i=1}^{N} \alpha_1(i) a_{i1}] b_1(O_2)$$

$$P(O_1 O_2 \mid \lambda) = \sum_{i=1}^{N} \alpha_2(i)$$

The Forward Algorithm

The Forward variable

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i \mid \lambda)$$

We can compute $\alpha(i)$ for all N, i,

Initialization:

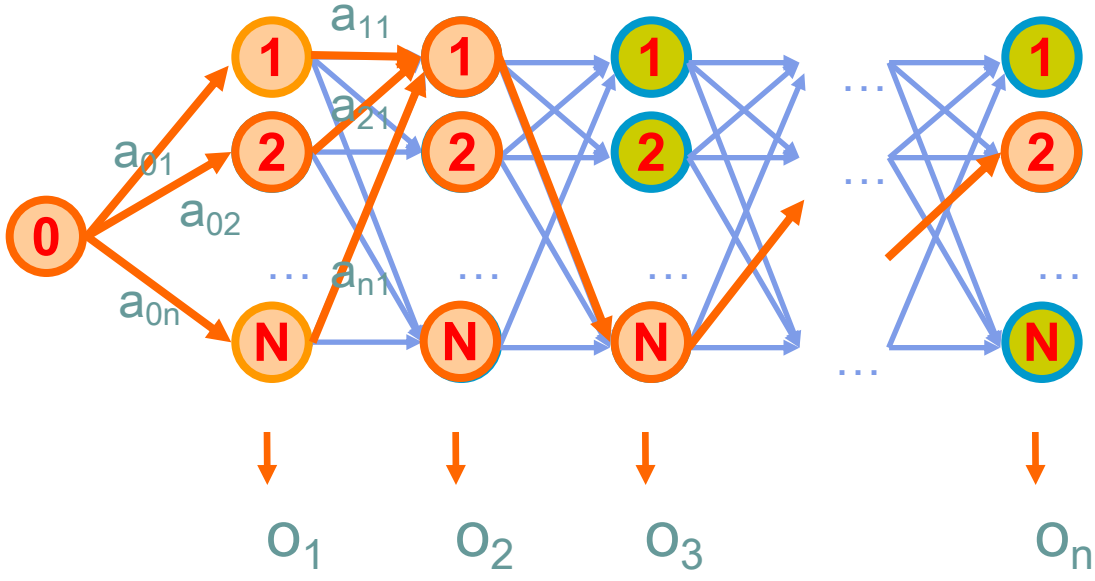
$$\alpha_1(i) = a_{0i} b_{0i}(O_1) \qquad i = 1 \ldots N$$

Iteration:

$$\alpha_{t+1}(i) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1}) \qquad t = 1 \ldots T-1$$

Termination:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$



The Backward Algorithm

The backward variable

$$\beta_t(i) = P(O_{t+1}O_{t+2}\cdots O_T \mid q_t = S_i, \lambda)$$

Similar, we can compute backward variable for all *N, i,*

Initialization:

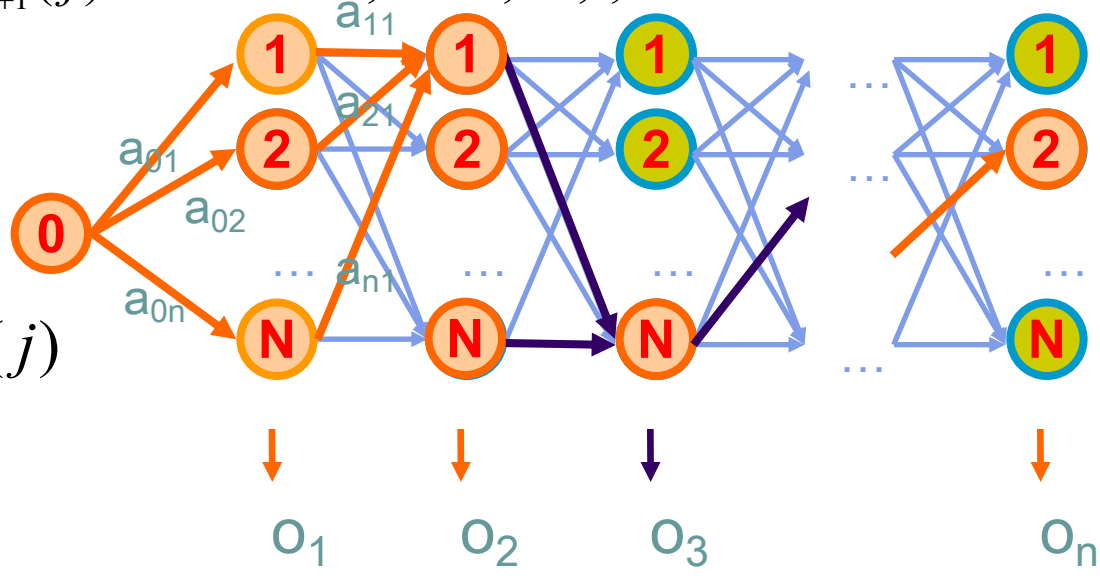
$$\beta_T(i) = 1,\ i = 1,\ldots,N$$

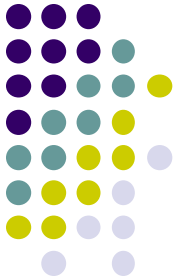
Iteration:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1})\beta_{t+1}(j) \qquad t = T-1, T-2, \cdots, 1, 1 \le i \le N$$

Termination:

$$P(O \mid \lambda) = \sum_{j=1}^{N} a_{0j} b_1(O_1)\beta_1(j)$$





Consider $\alpha_T(i) = P(O_1 O_2 \ldots O_T, q_T = S_i | \lambda)$

Thus $P(q_T = S_i | O) = \dfrac{P(O, q_T = S_i)}{P(O)} = \dfrac{\alpha_T(i_T)}{\sum_i \alpha_T(i_T)}$

Also $P(q_t = S_i | O) = \dfrac{P(\mathrm{O}, q_t = S_i)}{P(\mathrm{O})}$

$= \dfrac{P(O_1 O_2 \cdots O_t, q_t = S_{i_t}, O_{t+1} O_{t+2} \cdots O_T)}{P(\mathrm{O})}$

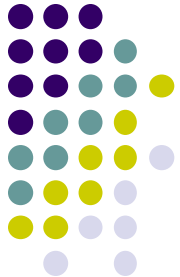
$= \dfrac{P(O_1 O_2 \cdots O_t, q_t = S_i) P(O_{t+1} O_{t+2} \cdots O_T | O_1 O_2 \cdots O_t, q_t = S_i)}{P(\mathrm{O})}$

Forward, $\alpha_k(i)$ Backward, $\beta_k(i)$

$= \dfrac{P(O_1 O_2 \cdots O_t, q_t = S_i) P(O_{t+1} O_{t+2} \cdots O_T | q_t = S_i)}{P(\mathrm{O})}$

$= \dfrac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_T(i)} = \gamma(i)$

Decoding



GIVEN a HMM, and a sequence O.

Suppose that we know the parameters of the Hidden Markov Model and the observed sequence of observations $O_1, O_2, \ldots, O_T$.

FIND the sequence Q of states that maximizes

$$P(Q|O,\lambda)$$

Determining the sequence of States $q_1, q_2, \ldots, q_T$, which is optimal in some meaningful sense. (i.e. best "explain" the observations)

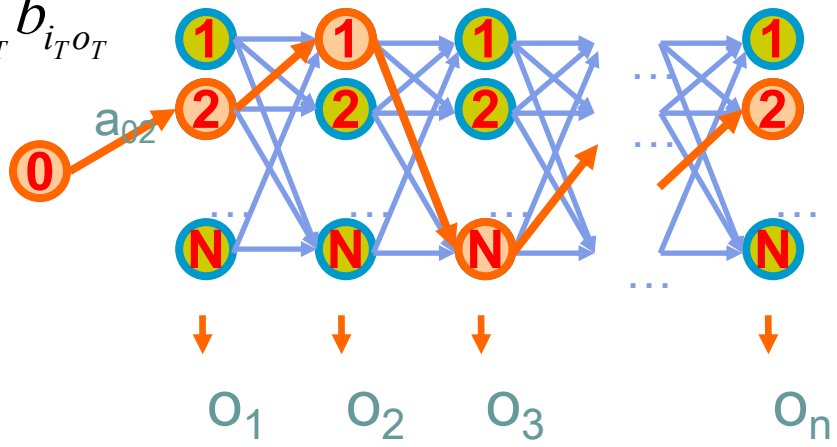
Decoding

Consider $P(Q|O,\lambda) = \dfrac{P(\mathrm{O},Q|\lambda)}{P(\mathrm{O}|\lambda)}$

To maximize the above probability is equivalent to maximizing

$$P(\mathrm{O},Q|\lambda)$$

$$= a_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} a_{i_2 i_3} b_{i_3 o_3} \ldots a_{i_{T-1} i_T} b_{i_T o_T}$$



A best path finding problem

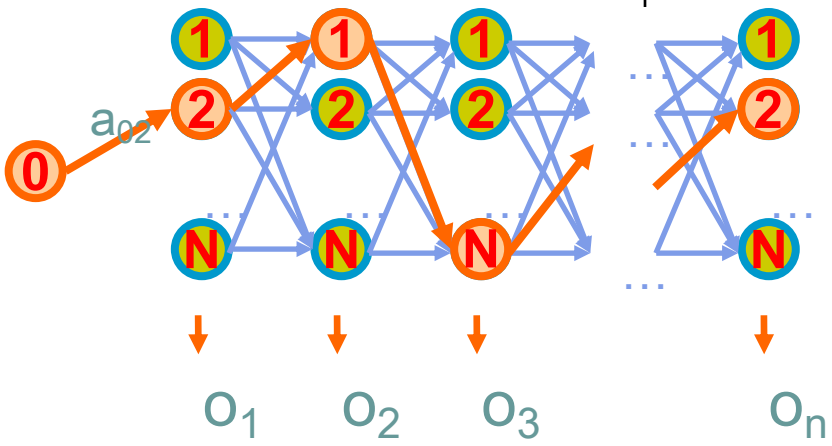
$$\max P(\mathrm{O},Q|\lambda)$$

$$= \max \ln(P(\mathrm{O},Q|\lambda))$$

$$= \max(\ln(a_{i_1} b_{i_1 o_1}) + \ln(a_{i_1 i_2} b_{i_2 o_2}) \ldots + \ln(a_{i_{T-1} i_T} b_{i_T o_T}))$$

Viterbi Algorithm

[Dynamic programming]



Initialization:

$\delta_1(i) = a_{0i} b_i(O_1)$, $i = 1 \ldots N$

$\psi_1(i) = 0.$

Recursion:

$\delta_t(j) = \max_i [\delta_{t-1}(i)\, a_{ij}] b_j(O_t)$ $t=2 \ldots T$ $j=1 \ldots N$

$\psi_1(j) = \text{argmax}_i [\delta_{t-1}(i)\, a_{ij}]$ $t=2 \ldots T$ $j=1 \ldots N$

Termination:

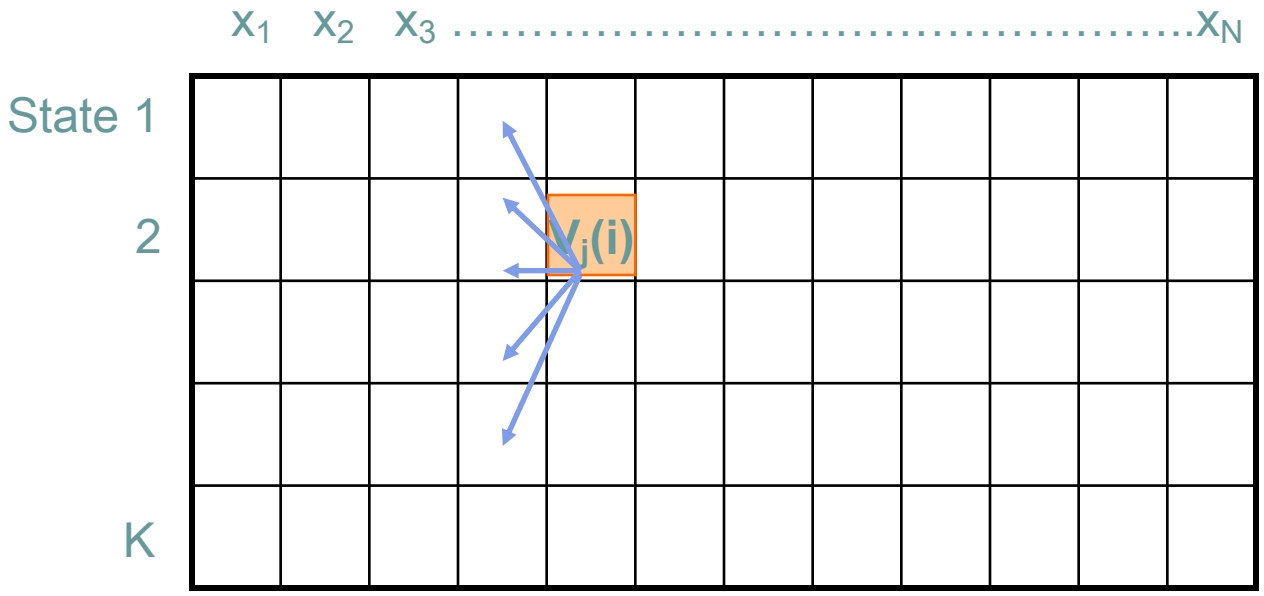
$P^* = \max_i \delta_T(i)$

$q_T^* = \text{argmax}_i [\delta_T(i)]$

Traceback:

$q_t^* = \psi_1(q^*_{t+1})$ $t=T-1, T-2, \ldots, 1.$

The Viterbi Algorithm

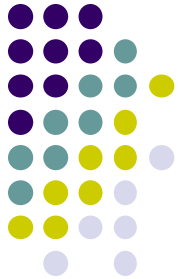


Similar to "aligning" a set of states to a sequence

Time: $O(K^2N)$

Space: $O(KN)$

Learning



- Estimation of Parameters of a Hidden Markov Model
  1. Both the sequence of observations O and the sequence of states Q is observed

     learning $\lambda = (A, B, \pi)$
  2. Only the sequence of observations O are observed

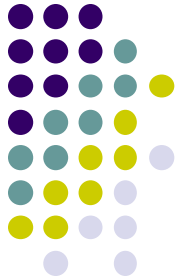
     learning Q and $\lambda = (A, B, \pi)$

Maximal Likelihood Estimation

- Given O and Q, the Likelihood is given by:

$$L(A,B,\pi) = a_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} a_{i_2 i_3} b_{i_3 o_3} \ldots a_{i_{T-1} i_T} b_{i_T o_T}$$

Maximal Likelihood Estimation



- the log-Likelihood is:

$$l(A,B,\pi) = \ln L(A,B,\pi) = \ln(a_{i_1}) + \ln(b_{i_1 o_1}) + \ln(a_{i_1 i_2}) + \ln(a_{i_2 i_3}) + \ln(b_{i_3 o_3}) \ldots + \ln(a_{i_{T-1} i_T}) + \ln(b_{i_T o_T})$$

$$= \sum_{i=1}^{M} f_{i0} \ln(a_i) + \sum_{i=1}^{M}\sum_{j=1}^{M} f_{ij} \ln(a_{ij}) + \sum_{i=1}^{M}\sum_{o(i)} \ln(b_{io})$$

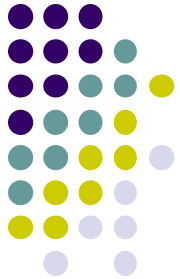
where f_{i0} = the number of times state i occurs in the first state

f_{ij} = the number of times state i changes to state j.

$\beta_{iy} = f(y|\theta_i)$ (or $p(y|\theta_i)$ in the discrete case)

$\sum_{o(i)} \bullet$ = the sum of all observations o_t where $q_t = S_i$

Maximal Likelihood Estimation

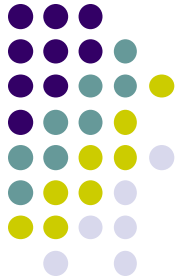


In such case these parameters computed by *Maximum Likelihood Estimation* are:

$$\hat{a}_i = \frac{f_{i0}}{1} \qquad \hat{a}_{ij} = \frac{f_{ij}}{\sum_{j=1}^{M} f_{ij}}, \text{ and}$$

$\hat{b}_i$ = the MLE of b_i computed from the observations o_t where $q_t = S_i$.

Maximal Likelihood Estimation



- Only the sequence of observations O are observed

$$L(A,B,\pi)=\sum_{i_1,i_2\ldots i_T} a_{i_1} b_{i_1 o_1} a_{i_1 i_2} b_{i_2 o_2} a_{i_2 i_3} b_{i_3 o_3} \ldots a_{i_{T-1} i_T} b_{i_T o_T}$$

  - It is difficult to find the Maximum Likelihood Estimates directly from the Likelihood function.
  - The Techniques that are used are

    1. ***The Segmental K-means Algorith***
    2. ***The Baum-Welch (E-M) Algorithm***

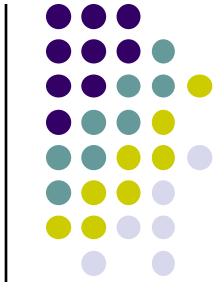
The Baum-Welch Algorithm

- The E-M algorithm was designed originally to handle "Missing observations".

- In this case the missing observations are the states $\{q_1, q_2, \ldots, q_T\}$.

- Assuming a model, the states are estimated by finding their expected values under this model. (The E part of the E-M algorithm).

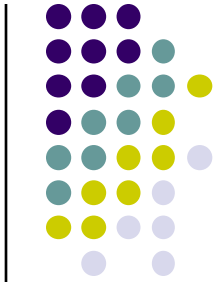
The Baum-Welch Algorithm



- With these values the model is estimated by Maximum Likelihood Estimation (The M part of the E-M algorithm).

- The process is repeated until the estimated model converges.

The Baum-Welch Algorithm



Initialization:

Pick the best-guess for model parameters (or arbitrary)

Iteration:

Forward

Backward

Calculate A_{kl}, $E_k(b)$

Calculate new model parameters a_{kl}, $e_k(b)$

Calculate new log-likelihood $P(x|\theta)$

GUARANTEED TO BE HIGHER BY EXPECTATION-MAXIMIZATION

Until $P(x|\theta)$ does not change much

The Baum-Welch Algorithm

Let $f(O,Q|\lambda) = L(O,Q,\lambda)$ denote the joint distribution of Q,O Consider the function:

$$Q(\lambda,\lambda') = E_{\mathbf{X}}\left(\ln L(O,Q,\lambda)|Q,\lambda'\right)$$

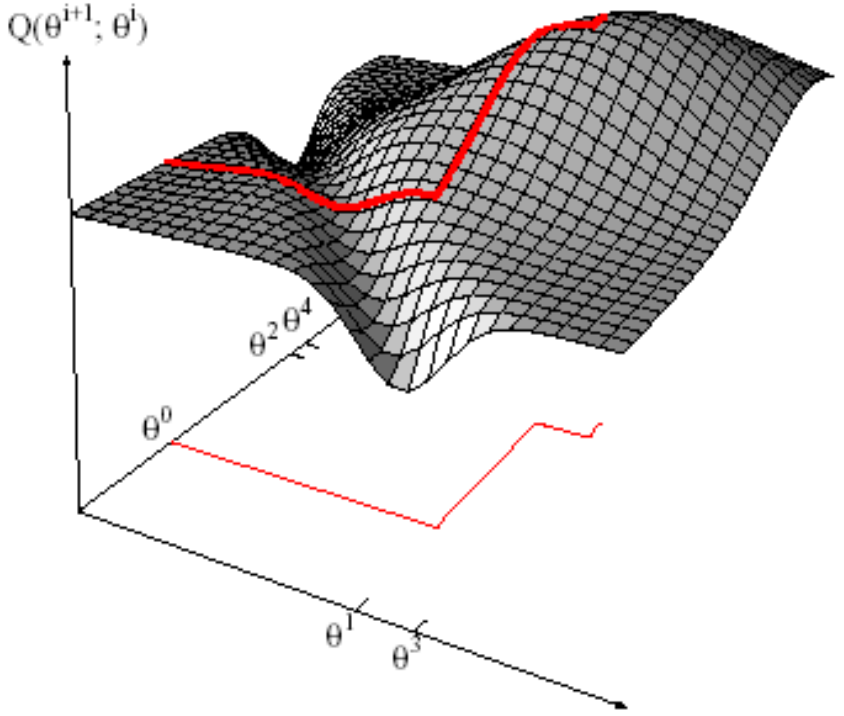
Starting with an initial estimate of λ $\left(\lambda^{(1)}\right)$.

A sequence of estimates $\left\{\lambda^{(m)}\right\}$ are formed by finding $\lambda = \lambda^{(m+1)}$ to maximize $Q\left(\lambda,\lambda^{(m)}\right)$ with respect to λ.

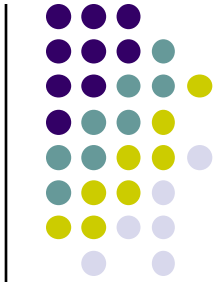
The Baum-Welch Algorithm

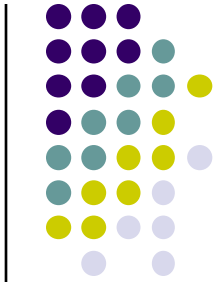
The sequence of estimates $\{\lambda^{(m)}\}$ converge to a local maximum of the likelihood

$$L(Q,\lambda) = f(Q|\lambda)$$



Particle Filters

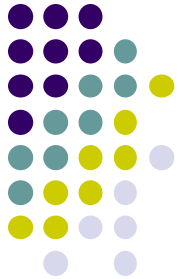




An example of HMM

SPEECH RECOGNITION

Speech Recognition



- On-line documents of Java™ Speech API
  - http://java.sun.com/products/java-media/speech/
- On-line documents of Free TTS
  - http://freetts.sourceforge.net/docs/
- On-line documents of Sphinx-II
  - http://www.speech.cs.cmu.edu/sphinx/
- A Chinese tutorial of ASR
  - http://blog.csdn.net/abcjennifer/article/details/27346787

Speech Recognition and Mobile applications

- Siri
- Cortana
- Google speech
- WeChat / Laiwang

- 科大讯飞
- ROKID（若琪）

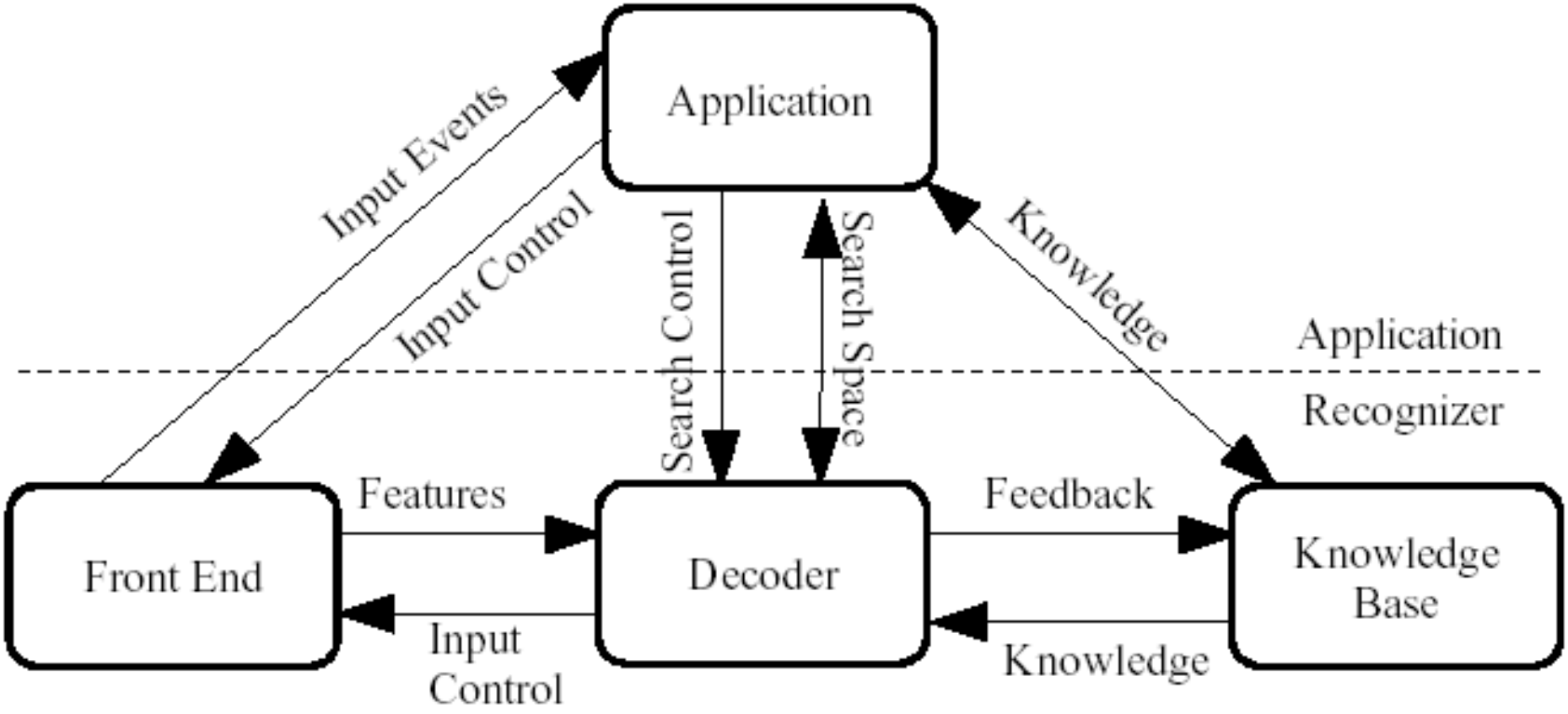
Speech Recognition and Mobile applications

- ROKID（若琪）

Brief History of CMU Sphinx

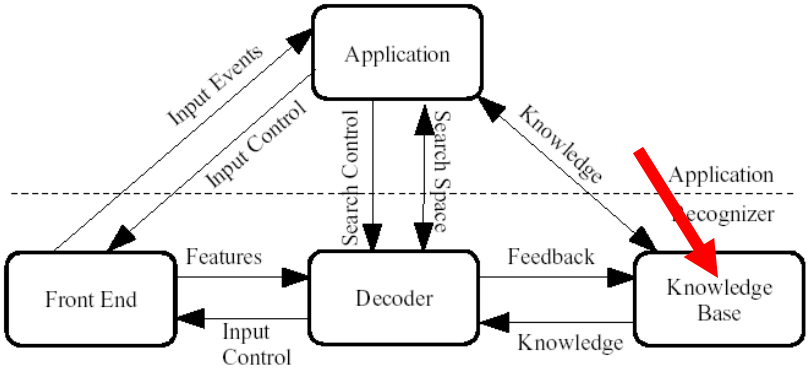
- Sphinx-I (1987)
  - The first user independent, high performance ASR of the world.
  - Written in C by Kai-Fu Lee (李開復博士，現任Google副總裁).
- Sphinx-II (1992)
  - Written by Xuedong Huang in C. (黃學東博士，現為Microsoft Speech.NET團隊領導人)
  - 5-state HMM / N-gram LM.
- Sphinx-III (1996)
  - Built by Eric Thayer and Mosur Ravishankar.
  - Slower than Sphinx-II but the design is more flexible.
- Sphinx-4 (Originally Sphinx 3j)
  - Refactored from Sphinx 3.
  - Fully implemented in Java. (Not finished yet …)

Components of CMU Sphinx



Knowledge Base

- The data that drives the decoder.
- Three sets of data
  - Acoustic Model.
  - Language Model.
  - Lexicon (Dictionary).



Speech Recognition Architecture

- Observations : $O = o_1, o_2, o_3, \cdots, o_t$
- Word Sequences : $W = w_1, w_2, w_3, \cdots, w_n$
- Probabilistic implementation can be expressed :

$$\hat{W} = \arg\max_{W \in L} P(W \mid O)$$

- Then we can use Bayes' rule to break it down :

$$\hat{W} = \arg\max_{W \in L} P(W \mid O) = \arg\max_{W \in L} \frac{P(O \mid W) P(W)}{P(O)}$$

$$\left(\because P(W \mid O) = \frac{P(WO)}{P(O)} \quad and \quad P(O \mid W) = \frac{P(WO)}{P(W)} \right)$$

$$\therefore P(W \mid O) \cdot P(O) = P(WO) = P(O \mid W) \cdot P(W)$$

Speech Recognition Architecture

- For each potential sentence we are still examining the same observations O, which must have the same probability $P(O)$.

$$\hat{W} = \arg\max_{W \in L} P(W \mid O)$$

Posterior probability

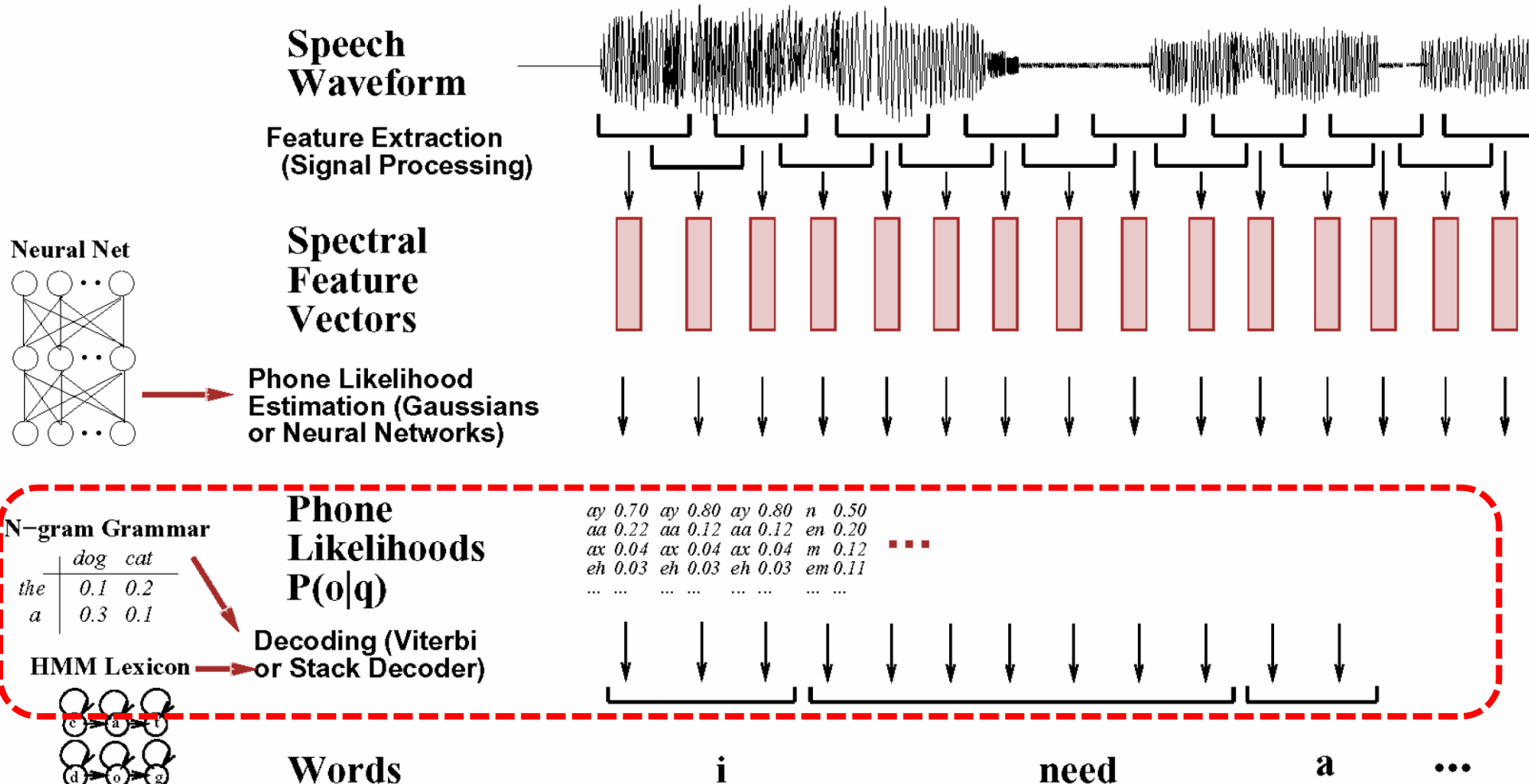
$$= \arg\max_{W \in L} \frac{P(O \mid W) P(W)}{P(O)} = \arg\max_{W \in L} P(O \mid W) P(W)$$

Observation likelihood
Acoustic model

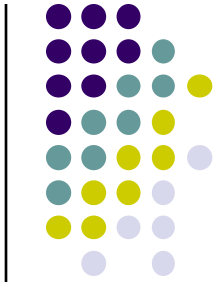
Prior probability
Language model

Speech Recognition Architecture

↓Figure 7.2 Schematic architecture for a speech recognition

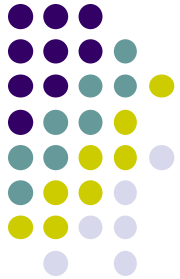


Acoustic Model



- /model/hmm/6k
- Database of statistical model
- Each statistical model represents a phoneme
- Acoustic Models are trained by analyzing large amount of speech data

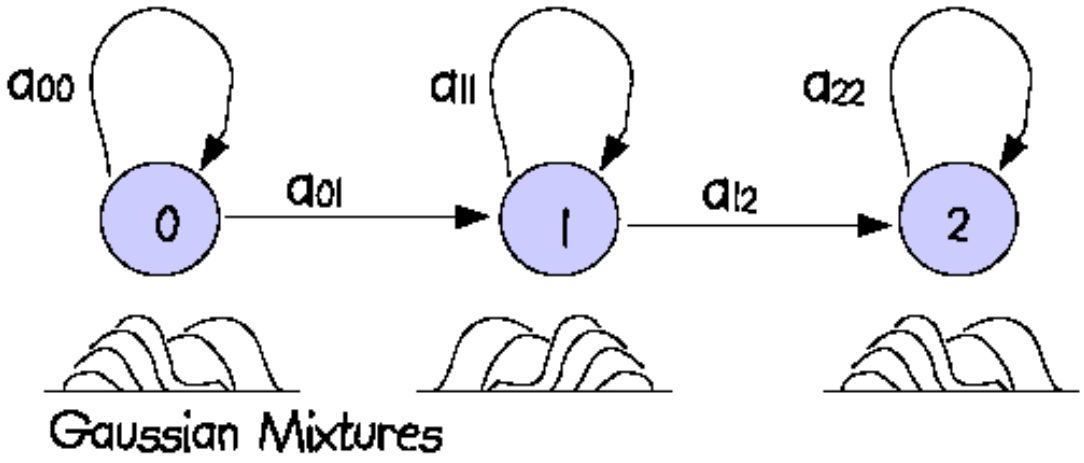
HMM in Acoustic Model



- HMM represent each unit of speech in the Acoustic Model.
- Typical HMM use 3-5 states to model a phoneme.
- Each state of HMM is represented by a set of *Gaussian mixture density functions*.
- Sphinx2 default phone set.

Mixture of Gaussians

- Represent each state in HMM.
- Each set of Gaussian Mixtures are called “senones”.
- HMM can share “senones”.



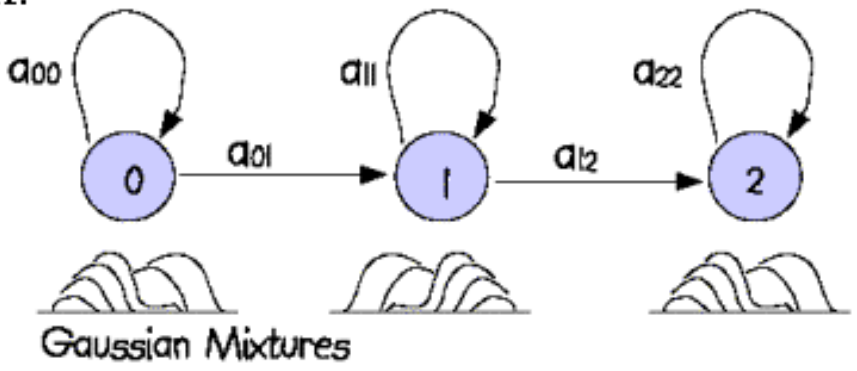
Mixture of Gaussians

$$N(x;\mu,\Sigma)=\frac{1}{(2\pi)^{n/2}\,|\Sigma|^{1/2}}\exp\left[-\frac{1}{2}(x-\mu)^{t}\Sigma^{-1}(x-\mu)\right]$$

$$f(x)=\sum_{k=1}^{K}c_k N_k(x;\mu_k,\Sigma_k) \qquad \text{其中}$$

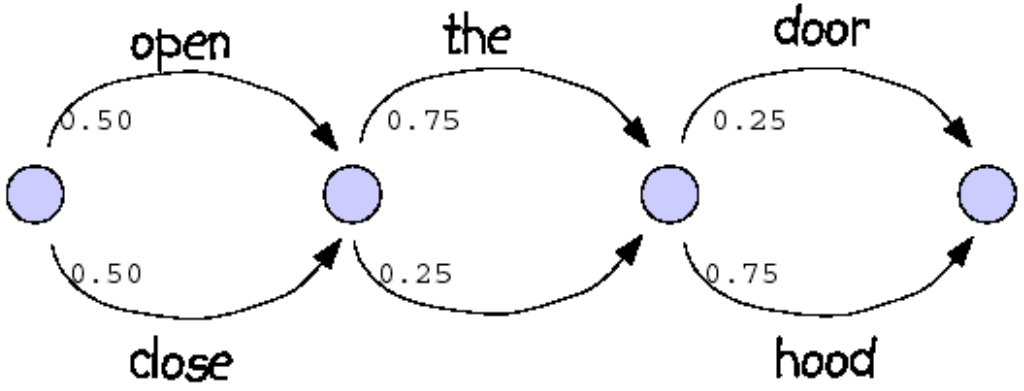
$$c_k \geq 0 \quad \text{且} \quad \sum_{k=1}^{K} c_k = 1$$

Gaussian mixtures with enough mixture components can approximate any distribution.



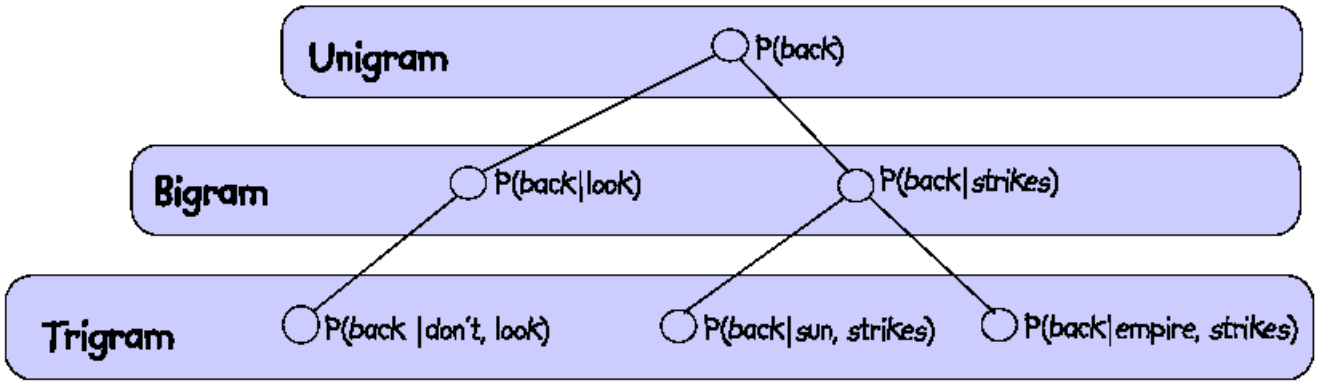
Language Model

- Describes what is likely to be spoken in a particular context
- Word transitions are defined in terms of transition probabilities
- Helps to constrain the search space



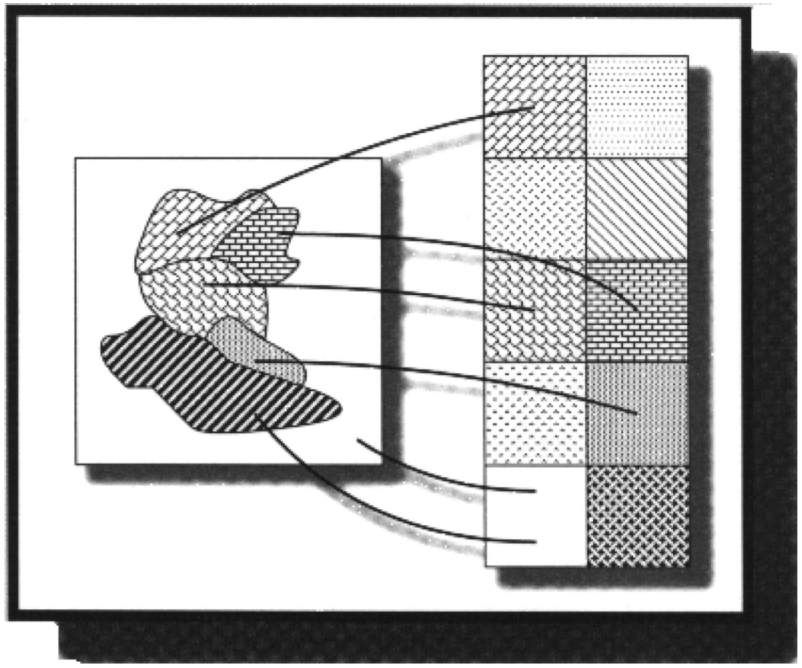
N-gram Language Model

- Probability of word N dependent on word N-1, N-2, ...
- Bigrams and trigrams most commonly used
- Used for large vocabulary applications such as dictation
- Typically trained by very large (millions of words) corpus



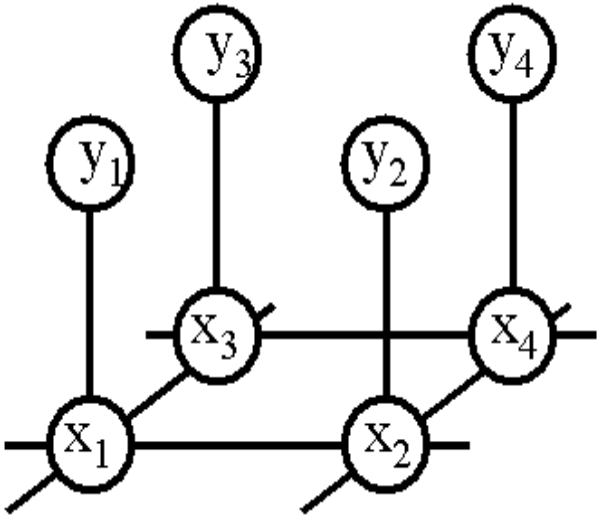
Markov Random field and CRF

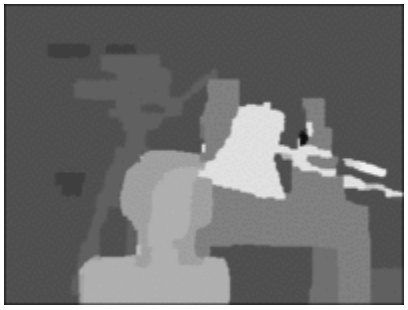
- See webpage
- http://www.nlpr.ia.ac.cn/users/szli/MRF_Book/MRF_Book.html



Belief Network (Propagation)

Y. Weiss and W. T. Freeman
Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology. in: Advances in Neural Information Processing Systems 12, edited by S. A. Solla, T. K. Leen, and K-R Muller, 2000. MERL-TR99-38.





CRF as RNN

- http://www.robots.ox.ac.uk/~szheng/crfasrnndemo

Original image (hover to highlight segmented parts)

Semantic segmentation

Objects appearing in the image:

Bicycle | Person

Objects not appearing in the image:

Aeroplane	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
Dining table	Dog	Horse	Motorbike	Potted plant	Sheep	Sofa	Train	TV/Monitor

Feedback: Please leave your feedback on our results. By uploading images, you grant the University of Oxford the rights to make use of the uploaded photographs for academic purposes.

Contact: Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Philip Torr. Department of Engineering Science, University of Oxford.

This software is built on top of the Caffe deep learning library. Full source code is available here.

The End

新浪微博：@浙大张宏鑫

微信公众号：