

Introduction to Graphical Models

Silvia Chiappa

Statistical Laboratory
University of Cambridge

silvia@statslab.cam.ac.uk
www.statslab.cam.ac.uk/Dept/People/chiappa.html

Marie-Curie Intra-European Fellow (IEF-237555)

May 8, 2010

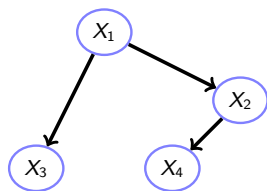
Motivation – Part I

Graphical models are **graphs** in which:

Nodes represent random variables

Links represent statistical dependencies between variables

Graphical models provide us with a visual tool for reasoning under uncertainty.



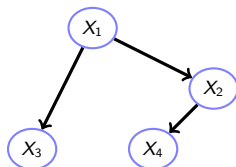
In machine learning and related disciplines uncertainty arises from

- Limited understanding of the world
- Limited amount of data

Motivation – Part II

Graphical models allow us to

- Answer questions about independence of random variables by just looking at the graph, dispensing with complex algebraic manipulations.
- Define general algorithms that perform probabilistic inference efficiently.



As a consequence

- Provided us with a common framework for representing and understanding the properties of different probabilistic models
 - ⇒ Enabled to relate models developed in different communities
- Have accelerated progress in modeling

We will focus on **Belief Networks**, **Markov Networks**, **Factor Graphs**.

Bayes Rule and Independence

Bayes Rule

$$\overbrace{p(A|B)}^{\text{posterior}} = \frac{p(B|A) \overbrace{p(A)}^{\text{prior}}}{p(B)}$$

Example: Sally throws a die. Tom tells her that she did not score 3. What is the probability that she scored 4?

$S_4 = 1$: Score = 4, $S_3 = 0$: Score $\neq 3$

$$\begin{aligned} p(S_4 = 1 | S_3 = 0) &= \frac{p(S_3 = 0 | S_4 = 1) p(S_4 = 1)}{p(S_3 = 0)} \\ &= \frac{1/6}{1 - 1/6} = 1/5 \end{aligned}$$

Marginal Independence of A and B

$$A \perp\!\!\!\perp B \iff p(A|B) = p(A) \text{ or } p(A, B) = p(A)p(B)$$

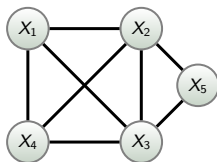
Conditional Independence of A and B given C

$$A \perp\!\!\!\perp B | C \iff p(A|B, C) = p(A|C) \text{ or } p(A, B|C) = p(A|C)p(B|C)$$

Basic Graph Definitions

- **Graph:** A graph consists of nodes and undirected or directed links between nodes.
- **Path from X_i to X_j :** Sequence of connected nodes starting at X_i and ending at X_j .

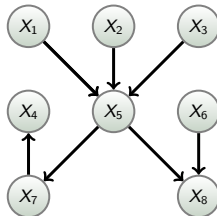
Undirected Graph



For Directed Graphs

- **Parents and Children:** X_i is a parent of X_j if there is a link from X_i to X_j . X_i is a child of X_j if there is a link from X_j to X_i .
- **Ancestors and Descendants:** The ancestors of a node X_i are the nodes with a directed path ending at X_i . The descendants of X_i are the nodes with a directed path beginning at X_i .
- **Directed Acyclic Graph:** Graph in which by following the direction of the arrows a node will never be visited more than once.

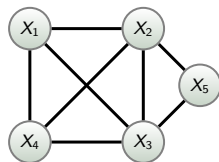
Directed Graph



Basic Graph Definitions

- **Graph:** A graph consists of nodes and undirected or directed links between nodes.
- **Path from X_i to X_j :** Sequence of connected nodes starting at X_i and ending at X_j .

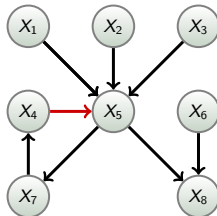
Undirected Graph



For Directed Graphs

- **Parents and Children:** X_i is a parent of X_j if there is a link from X_i to X_j . X_i is a child of X_j if there is a link from X_j to X_i .
- **Ancestors and Descendants:** The ancestors of a node X_i are the nodes with a directed path ending at X_i . The descendants of X_i are the nodes with a directed path beginning at X_i .
- **Directed Acyclic Graph:** Graph in which by following the direction of the arrows a node will never be visited more than once.

Directed Cyclic Graph

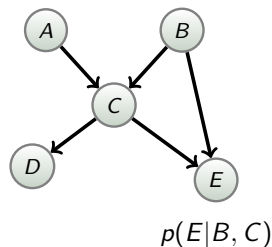


Belief Networks (Bayesian Networks)

A belief network is a directed acyclic graph in which each node has associated the conditional probability of the node given its parents.

The joint distribution is obtained by taking the product of the conditional probabilities.

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



Example – Part I

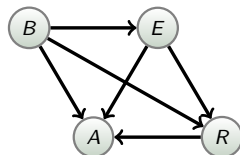
Sally's burglar **A**larm is sounding. Has she been **B**urgled, or was the alarm triggered by an **E**arthquake? She turns the car **R**adio on for news of earthquakes.

Without loss of generality, we can write

$$\begin{aligned} p(A, R, E, B) &= p(A|R, E, B)p(R, E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E|B)p(B) \end{aligned}$$

However

- The alarm is not directly influenced by any report on the radio, that is $p(A|R, E, B) = p(A|E, B)$
- $p(R|E, B) = p(R|E)$
- $p(E|B) = p(E)$



Therefore

$$p(A, R, E, B) = p(A|E, B)p(R|E)p(E)p(B)$$

Example – Part I

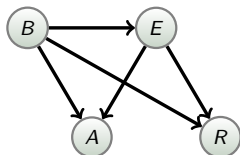
Sally's burglar **A**larm is sounding. Has she been **B**urgled, or was the alarm triggered by an **E**arthquake? She turns the car **R**adio on for news of earthquakes.

Without loss of generality, we can write

$$\begin{aligned} p(A, R, E, B) &= p(A|R, E, B)p(R, E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E|B)p(B) \end{aligned}$$

However

- The alarm is not directly influenced by any report on the radio, that is $p(A|R, E, B) = p(A|E, B)$
- $p(R|E, B) = p(R|E)$
- $p(E|B) = p(E)$



Therefore

$$p(A, R, E, B) = p(A|E, B)p(R|E)p(E)p(B)$$

Example – Part I

Sally's burglar **A**larm is sounding. Has she been **B**urgled, or was the alarm triggered by an **E**arthquake? She turns the car **R**adio on for news of earthquakes.

Without loss of generality, we can write

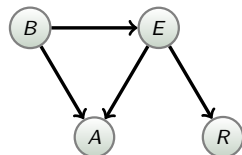
$$\begin{aligned} p(A, R, E, B) &= p(A|R, E, B)p(R, E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E|B)p(B) \end{aligned}$$

However

- The alarm is not directly influenced by any report on the radio, that is $p(A|R, E, B) = p(A|E, B)$
- $p(R|E, B) = p(R|E)$
- $p(E|B) = p(E)$

Therefore

$$p(A, R, E, B) = p(A|E, B)p(R|E)p(E)p(B)$$



Example – Part I

Sally's burglar **A**larm is sounding. Has she been **B**urgled, or was the alarm triggered by an **E**arthquake? She turns the car **R**adio on for news of earthquakes.

Without loss of generality, we can write

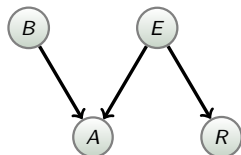
$$\begin{aligned} p(A, R, E, B) &= p(A|R, E, B)p(R, E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E, B) \\ &= p(A|R, E, B)p(R|E, B)p(E|B)p(B) \end{aligned}$$

However

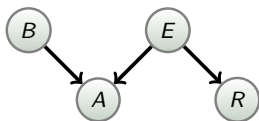
- The alarm is not directly influenced by any report on the radio, that is $p(A|R, E, B) = p(A|E, B)$
- $p(R|E, B) = p(R|E)$
- $p(E|B) = p(E)$

Therefore

$$p(A, R, E, B) = p(A|E, B)p(R|E)p(E)p(B)$$



Example – Part II: Specifying the Tables



$$p(A|B, E)$$

Alarm = 1	Burglar	Earthquake
0.9999	1	1
0.99	1	0
0.99	0	1
0.0001	0	0

$$p(R|E)$$

Radio = 1	Earthquake
1	1
0	0

The remaining tables are $p(B = 1) = 0.01$ and $p(E = 1) = 0.000001$. The tables and graphical structure fully specify the distribution.

Example Part III: Inference

Initial Evidence: The alarm is sounding

$$\begin{aligned} p(B = 1|A = 1) &= \frac{\sum_{E,R} p(B = 1, E, A = 1, R)}{\sum_{B,E,R} p(B, E, A = 1, R)} \\ &= \frac{\sum_{E,R} p(A = 1|B = 1, E)p(B = 1)p(E)p(R|E)}{\sum_{B,E,R} p(A = 1|B, E)p(B)p(E)p(R|E)} \approx 0.99 \end{aligned}$$

Additional Evidence: The radio broadcasts an earthquake warning:

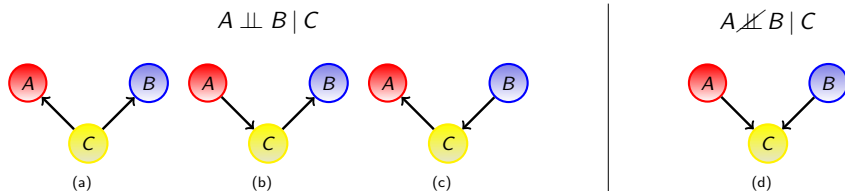
A similar calculation gives $p(B = 1|A = 1, R = 1) \approx 0.01$.

Initially, because the alarm sounds, Sally thinks that she's been burgled. However, this probability drops dramatically when she hears that there has been an earthquake.

The earthquake 'explains away' to an extent the fact that the alarm is ringing.

Independence $\perp\!\!\!\perp$ in Belief Networks – Part I

All belief networks with three nodes and two links:



- In (a), (b) and (c), A, B are conditionally independent given C .

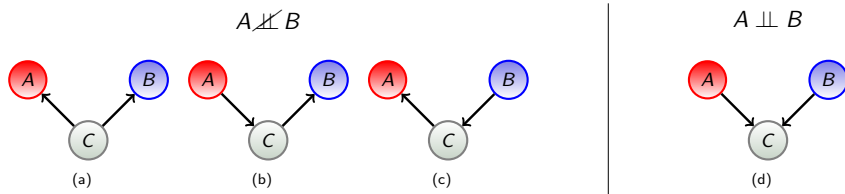
$$(a) \quad p(A, B \mid C) = \frac{p(A, B, C)}{p(C)} = \frac{p(A \mid C)p(B \mid C)p(C)}{p(C)} = p(A \mid C)p(B \mid C)$$

$$(b) \quad p(A, B \mid C) = \frac{p(A)p(C \mid A)p(B \mid C)}{p(C)} = \frac{p(A, C)p(B \mid C)}{p(C)} = p(A \mid C)p(B \mid C)$$

$$(c) \quad p(A, B \mid C) = \frac{p(A \mid C)p(C \mid B)p(B)}{p(C)} = \frac{p(A \mid C)p(B, C)}{p(C)} = p(A \mid C)p(B \mid C)$$

- In (d) the variables A, B are conditionally dependent given C .

Independence $\perp\!\!\!\perp$ in Belief Networks – Part II

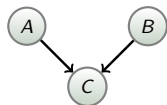


- In (a), (b) and (c), the variables A, B are marginally dependent.
- In (d) the variables A, B are marginally independent.

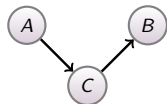
$$p(A, B) = \sum_C p(A, B, C) = \sum_C p(A)p(B)p(C|A, B) = p(A)p(B)$$

Collider

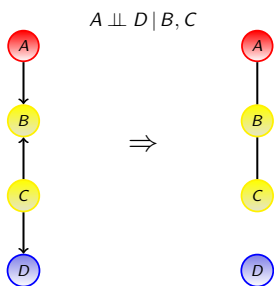
Summary of two previous slides:



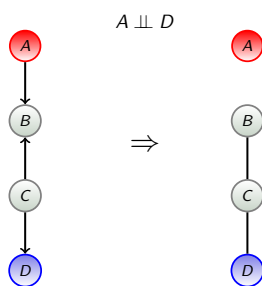
If C has more than one incoming link, then $A \perp\!\!\!\perp B$ and $A \not\perp\!\!\!\perp B \mid C$. In this case C is called **collider**.



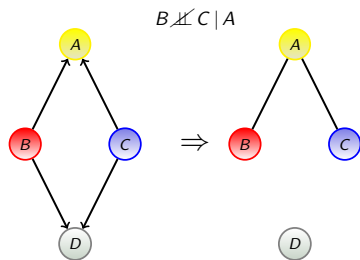
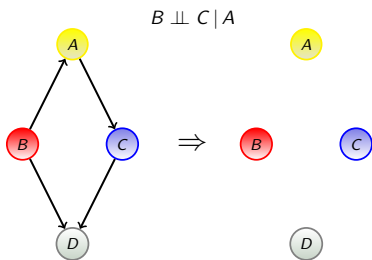
If C has at most one incoming link, then $A \perp\!\!\!\perp B \mid C$ and $A \not\perp\!\!\!\perp B$. In this case C is called **non-collider**.



non-collider in the conditioning set blocks a path



collider outside the conditioning set blocks a path




all paths need to be blocked to obtain $\perp\!\!\!\perp$

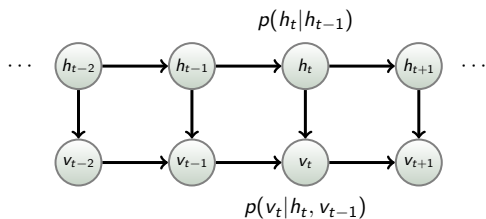
General Rule for Independence in Belief Networks

Given three sets of nodes $\mathcal{X}, \mathcal{Y}, \mathcal{C}$, if all paths from any element of \mathcal{X} to any element of \mathcal{Y} are blocked by \mathcal{C} , then \mathcal{X} and \mathcal{Y} are conditionally independent given \mathcal{C} .

A path \mathcal{P} is blocked by \mathcal{C} if one of the following conditions is satisfied:

1. there is a collider  in the path \mathcal{P} such that neither the collider nor any of its descendants is in the conditioning set \mathcal{C} .
2. there is a non-collider in the path \mathcal{P} that is in the conditioning set \mathcal{C} .

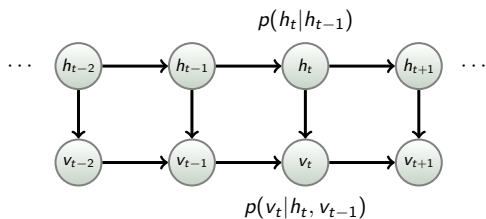
Example of using the Independence Rule for Time-Series Modeling



- Variables $v_1, \dots, v_T \equiv v_{1:T}$ represent the observed time-series.
- Discrete hidden variables $h_{1:T}$ generate the observations.

$$\begin{aligned} p(h_t, v_{1:t}) &= p(v_t|h_t, v_{1:t-1})p(h_t, v_{1:t-1}) \\ &= p(v_t|h_t, v_{1:t-1}) \sum_{h_{t-1}} p(h_{t-1:t}, v_{1:t-1}) \\ &= p(v_t|h_t, v_{t-1}) \sum_{h_{t-1}} p(h_t|h_{t-1}, v_{1:t-1})p(h_{t-1}, v_{1:t-1}) \end{aligned}$$

Example of using the Independence Rule for Time-Series Modeling

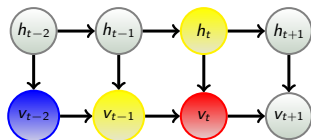


- Variables $v_1, \dots, v_T \equiv v_{1:T}$ represent the observed time-series.
- Discrete hidden variables $h_{1:T}$ generate the observations.

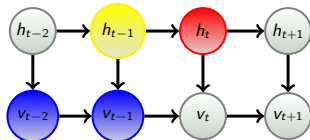
$$\begin{aligned} p(h_t, v_{1:t}) &= p(v_t | h_t, \cancel{v_{1:t-2}}, v_{t-1}) p(h_t, v_{1:t-1}) \\ &= p(v_t | h_t, v_{1:t-1}) \sum_{h_{t-1}} p(h_{t-1:t}, v_{1:t-1}) \\ &= p(v_t | h_t, v_{t-1}) \sum_{h_{t-1}} p(h_t | h_{t-1}, \cancel{v_{1:t-1}}) p(h_{t-1}, v_{1:t-1}) \end{aligned}$$

Example of using the Independence Rule for Time-Series Modeling

$$v_t \perp\!\!\!\perp v_{1:t-2} \mid \{h_t, v_{t-1}\}$$



$$h_t \perp\!\!\!\perp v_{1:t-1} \mid h_{t-1}$$



$$\begin{aligned} p(h_t, v_{1:t}) &= p(v_t | h_t, \cancel{v_{1:t-2}}, v_{t-1}) p(h_t, v_{1:t-1}) \\ &= p(v_t | h_t, v_{1:t-1}) \sum_{h_{t-1}} p(h_{t-1:t}, v_{1:t-1}) \\ &= p(v_t | h_t, v_{t-1}) \sum_{h_{t-1}} p(h_t | h_{t-1}, \cancel{v_{1:t-1}}) p(h_{t-1}, v_{1:t-1}) \end{aligned}$$

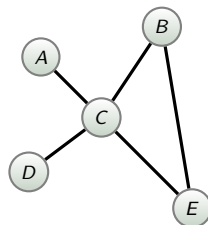
Markov Network

Clique: Fully connected subset of nodes.

Maximal Clique: Clique which is not a subset of a larger clique.

A Markov Network is an undirected graph in which there is a potential (non-negative function) ψ defined on each maximal clique.

The joint distribution is proportional to the product of all clique potentials.



$$p(A, B, C, D, E) = \frac{1}{Z} \psi(A, C) \psi(C, D) \psi(B, C, E)$$

$$Z = \sum_{A, B, C, D, E} \psi(A, C) \psi(C, D) \psi(B, C, E)$$

Example Application of Markov Network – Part I

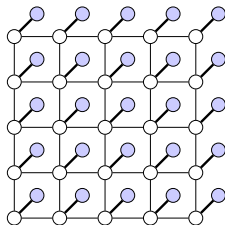
Problem: We want to recover a binary image from the observation of a corrupted version of it.

$X = \{X_i, i = 1, \dots, D\}$ $X_i \in \{-1, 1\}$: clean pixel

$Y = \{Y_i, i = 1, \dots, D\}$ $Y_i \in \{-1, 1\}$: corrupted pixel

$\phi(Y_i, X_i) = e^{\gamma X_i Y_i}$ encourage Y_i and X_i to be similar

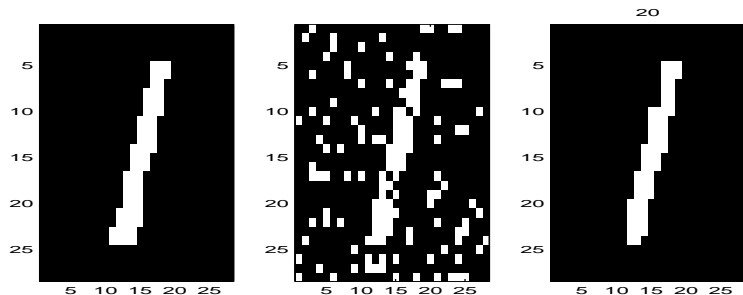
$\psi(X_i, X_j) = e^{\beta X_i X_j}$ encourage the image to be smooth



$$p(X, Y) \propto \left[\prod_{i=1}^D \phi(Y_i, X_i) \right] \left[\prod_{i \sim j} \psi(X_i, X_j) \right]$$

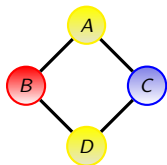
Finding the most likely X given Y is not easy (since the graph is not singly-connected), but approximate algorithms often work well.

Example Application of Markov Network – Part II



- left Original clean image
- middle Observed (corrupted) image
- right Most likely clean image $\arg \max_X p(X|Y)$

Independence $\perp\!\!\!\perp$ in Markov Networks

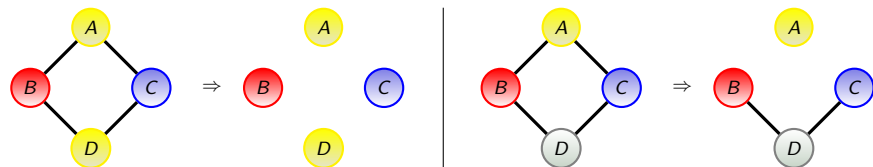


$B \perp\!\!\!\perp C \mid A, D$?

$p(B \mid A, D, C) = p(B \mid A, D)$?

$$\begin{aligned} p(B \mid A, D, C) &= \frac{p(A, B, C, D)}{p(A, C, D)} \\ &= \frac{p(A, B, C, D)}{\sum_B p(A, B, C, D)} \\ &= \frac{\psi(A, B) \cancel{\psi(A, C)} \psi(B, D) \cancel{\psi(C, D)}}{\sum_B \psi(A, B) \cancel{\psi(A, C)} \psi(B, D) \cancel{\psi(C, D)}} \\ &= \frac{p(A, B, D)}{p(A, D)} \\ &= p(B \mid A, D) \end{aligned}$$

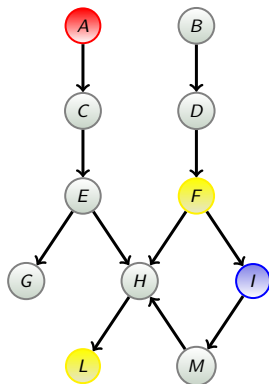
General Rule for Independence in Markov Networks



If every path from any member of \mathcal{X} to any member of \mathcal{Y} passes through any member of \mathcal{C} then \mathcal{X} and \mathcal{Y} are conditionally independent given \mathcal{C} .

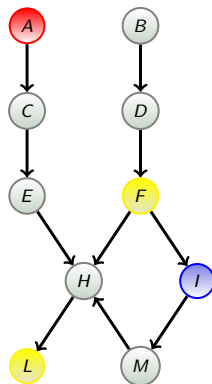
Alternative Rule for Independence in Belief Networks

- **Ancestral Graph:** Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{C}$ nor an ancestor of a node in this set, together with any edges in or out of such nodes.
- **Moralisation:** Add a line between any two nodes which have a common child. Remove arrowheads.
- **Independence:** If all paths which join a node in \mathcal{X} to one in \mathcal{Y} pass through any member of \mathcal{C} then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{C}$.



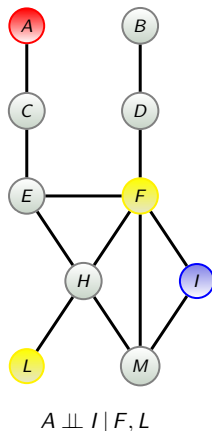
Alternative Rule for Independence in Belief Networks

- **Ancestral Graph:** Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{C}$ nor an ancestor of a node in this set, together with any edges in or out of such nodes.
- **Moralisation:** Add a line between any two nodes which have a common child. Remove arrowheads.
- **Independence:** If all paths which join a node in \mathcal{X} to one in \mathcal{Y} pass through any member of \mathcal{C} then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{C}$.

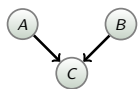


Alternative Rule for Independence in Belief Networks

- **Ancestral Graph:** Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{C}$ nor an ancestor of a node in this set, together with any edges in or out of such nodes.
- **Moralisation:** Add a line between any two nodes which have a common child. Remove arrowheads.
- **Independence:** If all paths which join a node in \mathcal{X} to one in \mathcal{Y} pass through any member of \mathcal{C} then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \mid \mathcal{C}$.

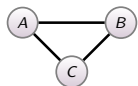


Expressiveness of Belief and Markov Networks



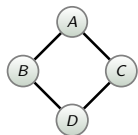
$$A \perp\!\!\!\perp B, A \not\perp\!\!\!\perp B | C$$

$$p(A, B, C) = \underbrace{p(C|A, B)}_{\psi(A, B, C)} \underbrace{p(B)}_1 \underbrace{p(C)}_1$$



$$A \not\perp\!\!\!\perp B$$

$$A \not\perp\!\!\!\perp B | C$$

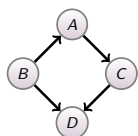


$$B \not\perp\!\!\!\perp C$$

$$B \not\perp\!\!\!\perp C | A$$

$$B \not\perp\!\!\!\perp C | D$$

$$B \perp\!\!\!\perp C | A, D$$



$B \not\perp\!\!\!\perp C$: the path B, A, C is not blocked

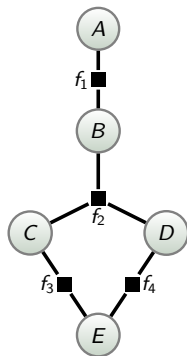
$$B \perp\!\!\!\perp C | A$$

$B \not\perp\!\!\!\perp C | D$: the path B, A, C is not blocked

$B \not\perp\!\!\!\perp C | A, D$: the path B, D, C is not blocked

Factor Graphs

A square node represents a factor (non negative function) of its neighbouring variables.



The joint function is the product of all factors:

$$f(A, B, C, D, E) = f_1(A, B)f_2(B, C, D)f_3(C, E)f_4(D, E)$$

Factor graphs are useful for performing efficient computations (not just for probability).

Inference

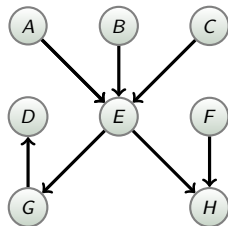
Inference corresponds to operations such as computing marginal or conditional distributions from the joint distribution.

In general inference is computationally very expensive. For singly-connected graphical models, there exist efficient algorithms based on the concept of message passing.

Singly-Connected Graph:

Graph in which there is only one path from a node to another node.

Singly-Connected Graph



Inference

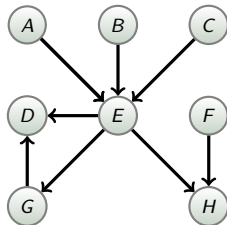
Inference corresponds to operations such as computing marginal or conditional distributions from the joint distribution.

In general inference is computationally very expensive. For singly-connected graphical models, there exist efficient algorithms based on the concept of message passing.

Singly-Connected Graph:

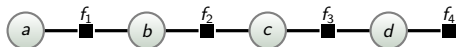
Graph in which there is only one path from a node to another node.

Multiply-Connected Graph



Sum-Product Algorithm for Factor Graphs - Non Branching Tree

$$p(a, b, c, d) \propto f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \quad a, b, c, d \text{ binary variables}$$

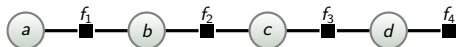


$$p(a) = \sum_{b, c, d} p(a, b, c, d)$$

$$\propto \sum_{b, c, d} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \Rightarrow 2^3 \text{ sums}$$

Sum-Product Algorithm for Factor Graphs - Non Branching Tree

$$p(a, b, c, d) \propto f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \quad a, b, c, d \text{ binary variables}$$



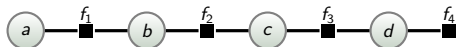
$$p(a) = \sum_{b, c, d} p(a, b, c, d)$$

$$\propto \sum_{b, c, d} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \Rightarrow 2^3 \text{ sums}$$

$$= \sum_b f_1(a, b) \sum_c f_2(b, c) \sum_d f_3(c, d) f_4(d) \Rightarrow 2 \times 3 \text{ sums}$$

Sum-Product Algorithm for Factor Graphs - Non Branching Tree

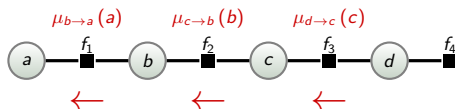
$p(a, b, c, d) \propto f_1(a, b) f_2(b, c) f_3(c, d) f_4(d)$ a, b, c, d binary variables



$$\begin{aligned} p(a) &= \sum_{b, c, d} p(a, b, c, d) \\ &\propto \sum_{b, c, d} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \\ &= \sum_b f_1(a, b) \underbrace{\sum_c f_2(b, c) \underbrace{\sum_d f_3(c, d) f_4(d)}_{\mu_{d \rightarrow c}(c)}}_{\mu_{c \rightarrow b}(b)} \\ &\quad \underbrace{\hspace{10em}}_{\mu_{b \rightarrow a}(a)} \end{aligned}$$

Sum-Product Algorithm for Factor Graphs - Non Branching Tree

$$p(a, b, c, d) \propto f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \quad a, b, c, d \text{ binary variables}$$

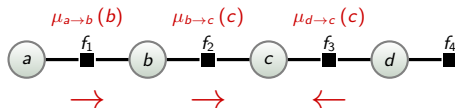


Passing variable-to-variable messages from d up to a

$$\begin{aligned} p(a) &= \sum_{b, c, d} p(a, b, c, d) \\ &\propto \sum_{b, c, d} f_1(a, b) f_2(b, c) f_3(c, d) f_4(d) \\ &= \sum_b f_1(a, b) \underbrace{\sum_c f_2(b, c) \underbrace{\sum_d f_3(c, d) f_4(d)}_{\mu_{d \rightarrow c}(c)}}_{\mu_{c \rightarrow b}(b)} \\ &\quad \underbrace{\hspace{10em}}_{\mu_{b \rightarrow a}(a)} \end{aligned}$$

Sum-Product Algorithm for Factor Graphs - Non Branching Tree

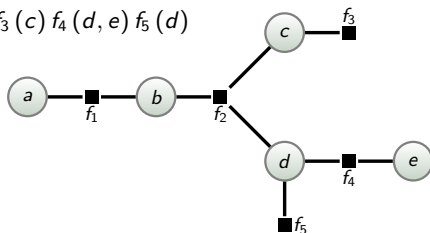
For $p(c)$ need to send messages in both directions



$$\begin{aligned} p(c) &\propto \sum_{a,b,d} f_1(a,b) f_2(b,c) f_3(c,d) f_4(d) \\ &= \sum_b \underbrace{\sum_a f_1(a,b) f_2(b,c)}_{\mu_{a \rightarrow b}(b)} \underbrace{\sum_d f_3(c,d) f_4(d)}_{\mu_{d \rightarrow c}(c)} \\ &\quad \underbrace{\hspace{10em}}_{\mu_{b \rightarrow c}(c)} \end{aligned}$$

Sum-Product Algorithm for Factor Graphs - Branching Tree

$$p(a, b, c, d, e) \propto f_1(a, b) f_2(b, c, d) f_3(c) f_4(d, e) f_5(d)$$



Need to define factor-to-variable messages and variable-to-factor messages

$$\begin{aligned}
 p(a) &\propto f_1(a, b) \sum_{c, d} f_2(b, c, d) \underbrace{f_3(c)}_{\mu_{c \rightarrow f_2}(c) = \mu_{f_3 \rightarrow c}(c)} \underbrace{f_5(d)}_{\mu_{f_5 \rightarrow d}(d)} \underbrace{\sum_e f_4(d, e)}_{\mu_{f_4 \rightarrow d}(d)} \\
 &\quad \underbrace{\hspace{10em}}_{\mu_{d \rightarrow f_2}(d)} \\
 &\quad \underbrace{\hspace{15em}}_{\mu_{b \rightarrow f_1}(b) = \mu_{f_2 \rightarrow b}(b)} \\
 &\quad \underbrace{\hspace{20em}}_{\mu_{f_1 \rightarrow a}(a)}
 \end{aligned}$$

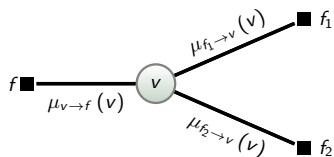
⇒ Marginal inference for a singly-connected structure is easy.

Sum-Product Algorithm for Factor Graphs

Variable to factor message

$$\mu_{v \rightarrow f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \rightarrow v}(v)$$

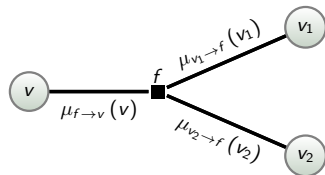
Messages from extremal variables are set to unity



Factor to variable message

$$\mu_{f \rightarrow v}(v) = \sum_{v_i} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \rightarrow f}(v_i)$$

Messages from extremal factors are set to the factor



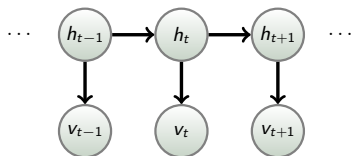
Marginal

$$p(v) \propto \prod_{f_i \sim v} \mu_{f_i \rightarrow v}(v)$$

Inference in Hidden Markov Models (HMM) – Part I

HMM: probabilistic time-series model which contains

- A set of discrete or continuous variables $v_1, \dots, v_T \equiv v_{1:T}$ which represent the observed time-series.
- A set of discrete hidden variables $h_{1:T}$ that generate the observations.



$$p(v_{1:T}, h_{1:T}) = p(v_1|h_1)p(h_1) \prod_{t=2}^T p(v_t|h_t)p(h_t|h_{t-1})$$

$$p(h_t = j|h_{t-1} = i) = \pi_{ji}, \quad \pi: \text{transition matrix}$$

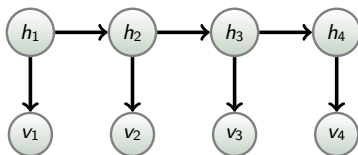
$$p(v_t = j|h_t = i) = \rho_{ji}, \quad \rho: \text{emission matrix}$$

Common inference problems:

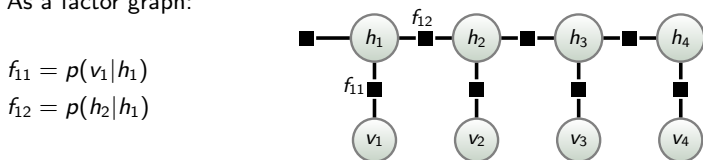
- Infer h_t from $p(h_t|v_{1:t})$ which uses the observations up to time t (filtering).
- Infer h_t from $p(h_t|v_{1:T})$ which also uses future observations (smoothing).
- Infer the most likely hidden sequence $h_{1:T}$ from $\arg \max_{h_{1:T}} p(h_{1:T}|v_{1:T})$ (Viterbi).

Inference in Hidden Markov Models – Part II

Belief network representation of a HMM:



As a factor graph:



$$f_{11} = p(v_1|h_1)$$

$$f_{12} = p(h_2|h_1)$$

- Filtering: carried out by passing messages up and to the right.
- Smoothing: combine filtering messages with messages up and to the left. Viterbi computed similarly.

Localisation example – Part I

Problem: You're asleep upstairs in your house and awoken by a burglar on the ground floor. You want to figure out where the burglar might be based on a sequence of noise information.

You mentally partition the ground floor into a 5×5 grid. For each grid position

- you know the probability that if someone is in that position the floorboard will creak
- you know the probability that if someone is in that position he will bump into something in the dark
- you assume that the burglar can move only into a neighbor grid square with uniform probability



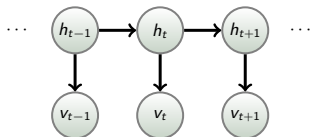
Prob. of creak



Prob. of bump

Localisation example – Part II

We can represent the scenario using a HMM where



- The hidden variable h_t represents the position of the burglar in the grid at time t

$$h_t \in \{1, \dots, 25\}$$

- The visible variable v_t represents occurrence of creak/bump at time t

$v=0$: no creak, no bump

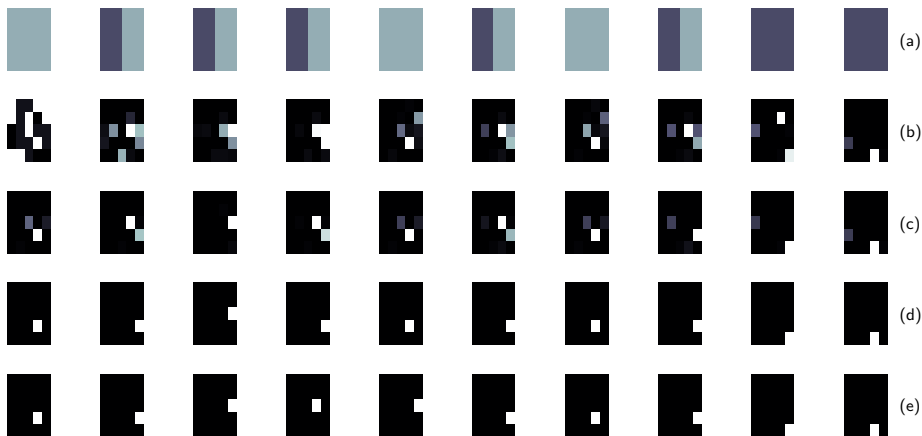
$v=1$: creak, no bump

$v=2$: no creak, bump

$v=3$: creak, bump

Localisation example – Part III

- (a) Observed cracks and bumps for 10 time-steps
- (b) Filtering $p(h_t|v_{1:t})$
- (c) Smoothing $p(h_t|v_{1:10})$
- (d) Most likely sequence $\arg \max_{h_{1:T}} p(h_{1:T}|v_{1:T})$
- (e) True Burglar position

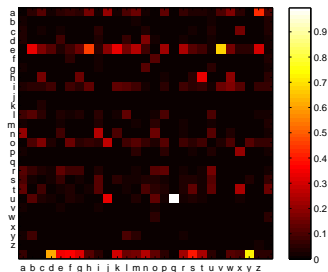


Natural Language Model Example – Part I

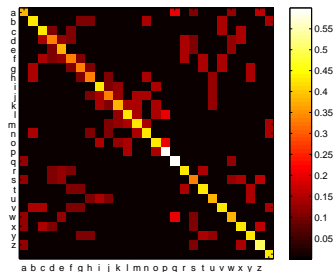
Problem: A ‘stubby finger’ typist has the tendency to hit either the correct key or a neighbouring key. Given a typed sequence you want to infer what is the most likely word that this corresponds to.

- The hidden variable h_t represents the intended letter at time t
- The visible variable v_t represents the letter that was actually typed at time t

We assume that there are 27 keys: lower case a to lower case z and the space bar.



Transition $p(h_t = j | h_{t-1} = i)$



Emission $p(v_t = j | h_t = i)$

Natural Language Model Example – Part II

Given the typed sequence `kezrninh` what is the most likely word that this corresponds to?

- Listing the 200 most likely hidden sequences (using a form of Viterbi)
- Discard those that are not in a standard English dictionary
- Take the most likely proper English word as the intended typed word

... and the answer is ...

Learning

Have a model $p(X|\theta)$ and dataset $\mathcal{D} = \{X^1, \dots, X^N\}$. What does the data say about θ ?

$$p(\mathcal{D}|\theta) = \prod_{i=1}^N p(X^i|\theta) \quad \text{identically independently distributed}$$

$p(\theta)$ prior preference on parameters

$$\underbrace{p(\theta|\mathcal{D})}_{\text{posterior}} = \frac{\underbrace{p(\mathcal{D}|\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}}}{\underbrace{p(\mathcal{D})}_{\text{marginal likelihood}}}$$

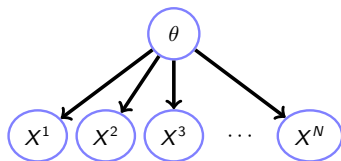
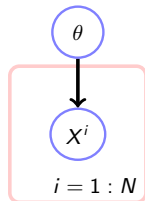


Plate Notation



Summarising the Parameter Posterior

- Maximum a Posteriori (MAP) Estimate

Mode of the posterior

$$\theta^{\text{MAP}} = \arg \max_{\theta} p(\mathcal{D}|\theta)p(\theta)$$

- Maximum Likelihood (ML)

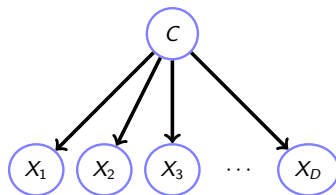
Mode of the posterior when the prior is omitted (or equivalently a 'flat prior' $p(\theta) = \text{const}$ is defined)

$$\theta^{\text{ML}} = \arg \max_{\theta} p(\mathcal{D}|\theta)$$

Naive Bayes Classifier

Popular model, e.g. for spam filtering

- C is the class label $C \in \{0, 1\}$
E.g. spam/ham
- X_i are attributes $X_i \in \{0, 1\}$
E.g. presence/absence of keyword cash



$$p(X_1, \dots, X_D, C) = p(C) \prod_{i=1}^D p(X_i | C)$$

$$p(C | X_1, \dots, X_D) \propto p(X_1, \dots, X_D | C) p(C) = p(C) \prod_i p(X_i | C)$$

Fast, simple classifier: strong independence assumption makes learning easy.

How to learn the table entries?

Naive Bayes: Learning

We are interested in learning the table entries from N observations $\mathcal{D} = \{(C^1, X_1^1, \dots, X_D^1), \dots, (C^N, X_1^N, \dots, X_D^N)\}$.

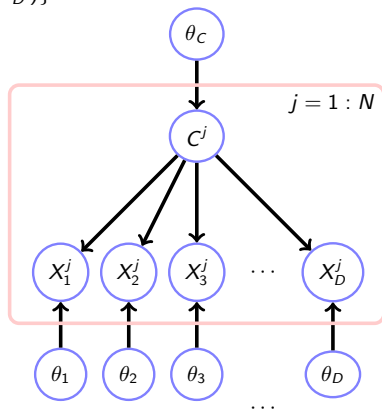
$$\theta_i = p(X_i | C, \theta_i), \quad i = 1 \dots, N$$

$$\theta_C = p(C | \theta_C)$$

$$\text{Bayes: } p(\theta | \mathcal{D})$$

$$\text{MAP/ML: } \theta = \arg \max_{\theta} p(\theta | \mathcal{D})$$

We assumed parameter prior independence.



Naive Bayes: Maximum Likelihood

The log-likelihood is given by

$$\log p(\mathcal{D}|\theta) = \log \prod_{j=1}^N p(X_1^j, \dots, X_D^j, C^j|\theta) = \log \prod_{j=1}^N p(C^j|\theta_C) \prod_{i=1}^D p(X_i^j|C^j, \theta_i)$$

We want to set $\theta_{i,0} = p(X_i = 0|C = 0, \theta_i)$. The dependence of the log-likelihood on $\theta_{i,0}$ is

$$\log \theta_{i,0}^{n_0} (1 - \theta_{i,0})^{n_1}, \quad n_{0/1} \text{ is the number of times } (C = 0, X_i = 0/1) \text{ occurs in } \mathcal{D}$$

By differentiating:

$$\frac{\partial \log p(\mathcal{D}|\theta)}{\partial \theta_{i,0}} = \frac{n_0}{\theta_{i,0}} - \frac{n_1}{1 - \theta_{i,0}}$$

and equating to zero we obtain the optimal value

$$\theta_{i,0} = \frac{n_0}{(n_0 + n_1)} \quad \text{which corresponds to calculating frequencies.}$$

Problem if there are zero data counts. Bayesian approach avoids this problem.

Naive Bayes: Bayesian Approach

Assume $D = 1$. Since $p(\theta) = p(\theta_C)p(\theta_1)$

$$\begin{aligned} p(\theta_C, \theta_1 | \mathcal{D}) &\propto p(\theta_C, \theta_1, \mathcal{D}) \\ &= p(\theta_C)p(\theta_1) \prod_{j=1}^N p(C^j | \theta_C) p(X_1^j | \theta_1) \\ &= \left\{ p(\theta_C) \prod_{j=1}^N p(C^j | \theta_C) \right\} \left\{ p(\theta_1) \prod_{j=1}^N p(X_1^j | \theta_1) \right\} \\ &\propto p(\theta_C | \mathcal{D}) p(\theta_1 | \mathcal{D}) \end{aligned}$$

If we further assume that the prior for the table factorises $p(\theta_1) = p(\theta_{1,0})p(\theta_{1,1})$ ($\theta_{1,0} = p(X_1 = 0 | C = 0, \theta_1)$, $\theta_{1,1} = p(X_1 = 1 | C = 1, \theta_1)$) and each term is beta distributed, that is

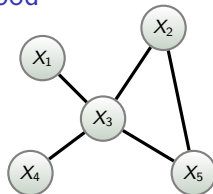
$$p(\theta_{1,0}) = \mathcal{B}(\alpha, \beta) \propto \theta_{1,0}^{\alpha-1} (1 - \theta_{1,0})^{\beta-1}$$

then

$$p(\theta_{1,0} | \mathcal{D}) \propto \theta_{1,0}^{n_0} (1 - \theta_{1,0})^{n_1} \theta_{1,0}^{\alpha-1} (1 - \theta_{1,0})^{\beta-1} = \mathcal{B}(\alpha + n_0, \beta + n_1)$$

Learning in Markov Networks: Maximum Likelihood

Consider a Markov network distribution $p(\mathcal{X})$ defined on cliques $\mathcal{X}_j \subseteq \mathcal{X}$



$$p(\mathcal{X}|\theta) = \frac{1}{Z(\theta)} \prod_{j=1}^C \psi(\mathcal{X}_j|\theta_j)$$

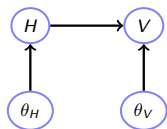
We want to learn the parameters $\theta = \{\theta_1, \dots, \theta_C\}$ from a set of observations $\mathcal{D} = \{\mathcal{X}^i, i = 1, \dots, N\}$.

The log-likelihood

$$\begin{aligned} \log p(\mathcal{D}|\theta) &= \log \prod_{i=1}^N p(\mathcal{X}^i|\theta) \\ &= \log \prod_{i=1}^N \frac{1}{Z(\theta)} \prod_{j=1}^C \psi(\mathcal{X}_j^i|\theta_j) \\ &= \sum_{i=1}^N \sum_{j=1}^C \log \psi(\mathcal{X}_j^i|\theta_j) - \underbrace{N \log Z(\theta)}_{\text{problematic}} \end{aligned}$$

does not split into a set of isolated parameter terms \Rightarrow need to use numerical methods.

Learning Parameters with Hidden Variables



- H is a hidden variable
- V is an observed or visible variable

We want to learn $\theta = \{\theta_H, \theta_V\}$ from a set of observations $\{V^i, i = 1, \dots, N\}$

Maximum Likelihood: the likelihood is no longer a product of a factor in θ_H and another factor in θ_V

$$p(\mathcal{D}|\theta) = \prod_{i=1}^N p(V^i|\theta) = \prod_{i=1}^N \sum_{H_i} p(V^i|H^i, \theta_V) p(H^i|\theta_H)$$

⇒ need numerical methods (gradient approaches, expectation maximization)

Bayesian case: the posterior of the parameter will not in general factorise

$$p(\theta|\mathcal{D}) \neq p(\theta_V|\mathcal{D})p(\theta_H|\mathcal{D})$$

⇒ need approximate Bayesian methods.

Expectation Maximisation Algorithm for Maximum Likelihood

Replace the log-likelihood with a lower bound that has a decoupled form.

Introduce distribution q and take the Kullback-Leibler divergence

$$KL(q(H|V)||p(H|V, \theta)) = \left\langle \log q(H|V) - \log \frac{p(H, V|\theta)}{p(V|\theta)} \right\rangle_{q(H|V)} \geq 0$$

which gives

$$\log p(V|\theta) \geq \underbrace{-\langle \log q(H|V) \rangle_{q(H|V)}}_{\text{Entropy}} + \langle \log p(H, V|\theta) \rangle_{q(H|V)}$$

For $q(H|V, \theta) = p(H|V, \theta^{old})$, the entropy term does not depend on θ and we obtain a bound that decouples the parameters:

$$\log p(V|\theta) \geq \text{Entropy} + \langle \log p(V|H, \theta_V) \rangle_{p(H|V, \theta^{old})} + \langle \log p(H|\theta_H) \rangle_{p(H|V, \theta^{old})}$$

Iterative algorithm:

- E-step: Compute $p(H|V, \theta^{old})$ using an inference approach
- M-step: Update $\theta^{new} = \arg \max_{\theta} \langle \log p(H, V|\theta) \rangle_{p(H|V, \theta^{old})}$

Reading

1. Bayesian Reasoning and Machine Learning. D. Barber, 2010 (examples and demos in this talk).
Can be downloaded from www.cs.ucl.ac.uk/staff/D.Barber/brml
2. Pattern Recognition and Machine Learning. C. M. Bishop, 2009.
3. Probabilistic Networks and Expert Systems. R. G. Cowell and A. P. Dawid, S. L. Lauritzen, D. Spiegelhalter, 2000.
4. Probabilistic graphical Models: Principles and Techniques. D. Koller and N. Friedman, 2009.
5. Bayesian Networks and Decision Graphs. F. V. Jensen, 2001.
6. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. J. Pearl, 1988.
7. Graphical Models. S. Lauritzen, 1996.

Appendix: Independence in Belief Networks

Yellow indicates conditioning, grey non-conditioned nodes.

Represent the structure of the graph on the reduced set of variables after conditioning/marginalisation.

