Introduction to Graphical Models[1]

Dmitry Adamskiy, David Barber

University College London

[1] These slides accompany the book *Bayesian Reasoning and Machine Learning*. The book and demos can be downloaded from `www.cs.ucl.ac.uk/staff/D.Barber/brml`. Feedback and corrections are also available on the site. Feel free to adapt these slides for your own purposes, but please include a link the above website.

Graphical Models

GMs are graph based representations of various factorisation assumptions of distributions. These factorisations are typically equivalent to independence statements amongst (sets of) variables in the distribution.

Belief Network Each factor is a conditional distribution. Generative models, AI, statistics. Corresponds to a DAG.

Markov Network Each factor corresponds to a potential (non negative function). Related to the strength of relationship between variables, but not directly related to dependence. Useful for collective phenomena such as image processing. Corresponds to an undirected graph.

Chain Graph A marriage of BNs and MNs. Contains both directed and undirected links.

Factor Graph A barebones representation of the factorisation of a distribution. Often used for efficient computation and deriving message passing algorithms.

The GM zoo There are many more kinds of GMs, each useful in its own right. We'll touch on some more when we consider inference.

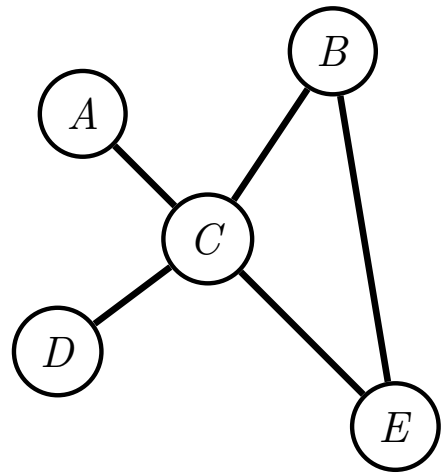
Markov Networks

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.



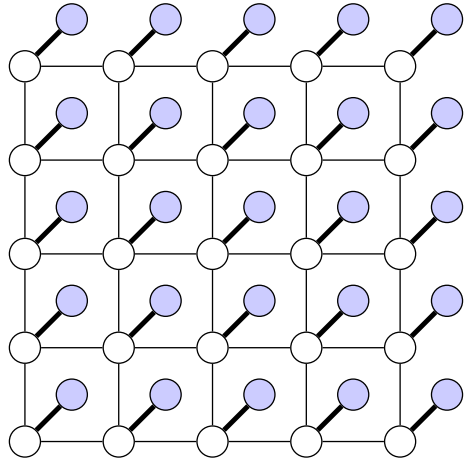
The joint distribution is proportional to the product of all clique potentials.

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

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

Example Application of Markov Network – Part I

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



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

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

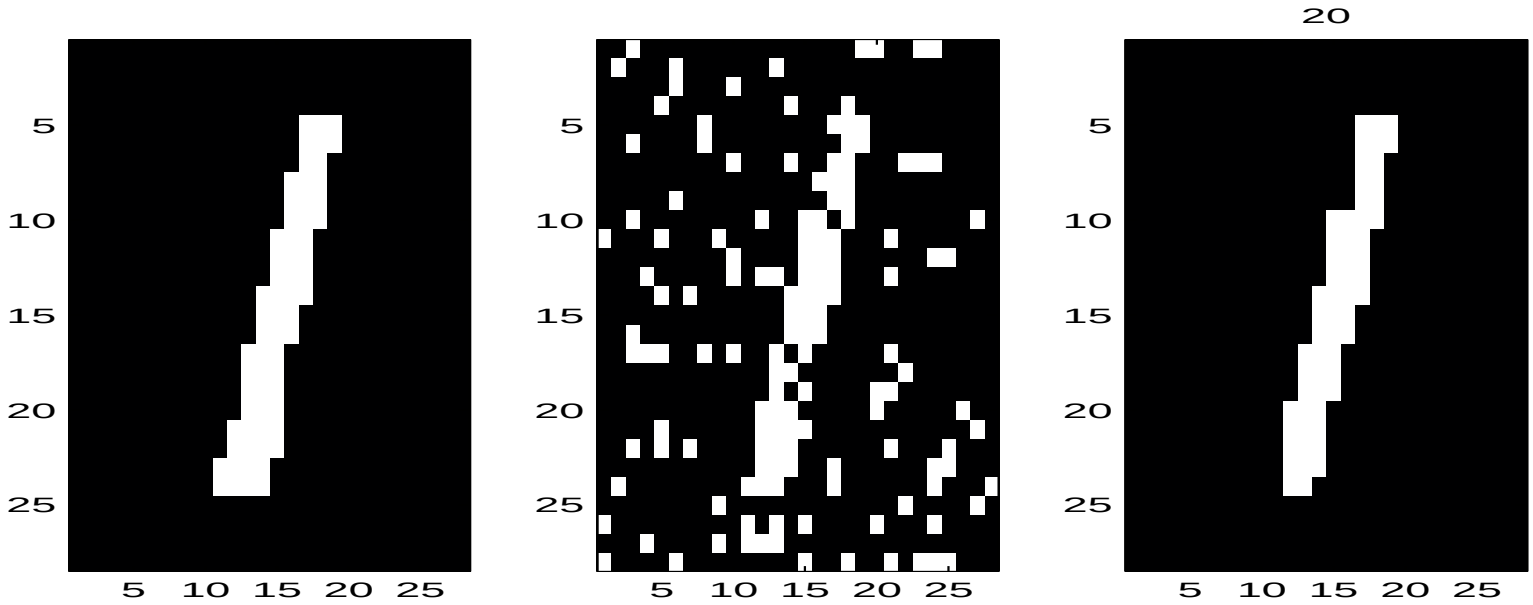
$\phi(Y_i, X_i) = e^{\gamma X_i Y_i}$ encourage Y_i and X_i to be similar

$\psi(X_i, X_j) = e^{\beta X_i X_j}$ 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 $\underset{X}{\operatorname{argmax}}\ p(X|Y)$

demoMRFclean.m In the demo the optimisation is performed by a simple algorithm that sweeps through all pixels (random ordering) and sets the state of pixel x_i according to $\underset{x_i}{\operatorname{argmax}}\ p(x_i, x_{\setminus i}|Y)$

Iterated Conditional Modes

- Given a function $f(x_1 \ \ldots, x_D)$, we want to find the minimum of this function.
- One very simple approach is to first make a guess $x_1^*, \ldots, x_D^*$.
- Then look at a single variable x_i keeping all others fixed. This forms a function

$$F(x_i) = f(x_1^*, \ldots, x_{i-1}^*, x_i, x_{i+1}^*, \ldots, x_D^*)$$

- Then set

$$x_i^* = \underset{x_i}{\operatorname{argmin}} \ F(x_i)$$

- We then repeat this, sweeping through all variables, usually in a random order.
- We then repeat the whole process, sweeping through all variables (each time with a different random ordering) until convergence.
- This is a form of 'axis aligned' optimisation that is guaranteed to converge to a minimum (though not necessarily a global minimum).
- It is most often used in the case of discrete variables x.
- This is called ICM in the Statistics community.

The Boltzmann machine

A MN on binary variables $\mathrm{dom}(x_i) = \{0, 1\}$ of the form

$$p(\mathbf{x}|\mathbf{w}, b) = \frac{1}{Z(\mathbf{w}, b)} e^{\sum_{i<j} w_{ij} x_i x_j + \sum_i b_i x_i}$$

where the interactions w_{ij} are the 'weights' and the b_i the biases.

- This model has been studied in the machine learning community as a basic model of distributed memory and computation. The $x_i = 1$ represents a neuron 'firing', and $x_i = 0$ not firing. The matrix $\mathbf{w}$ describes which neurons are connected to each other. The conditional

$$p(x_i = 1|x_{\setminus i}) = \sigma\left(b_i + \sum_{j \neq i} w_{ij} x_j\right), \qquad \sigma(x) = e^x/(1 + e^x).$$

- The graphical model of the BM is an undirected graph with a link between nodes i and j for $w_{ij} \neq 0$. For all but specially constrained $\mathbf{w}$ inference will be typically intractable.
- Given a set of data $\mathbf{x}^1, \ldots, \mathbf{x}^N$, one can set the parameters $\mathbf{w}, b$ by maximum likelihood (though this is computationally difficult).

Training a Boltzmann Machine

- First note that since $x_i^2 = x_i$ (for 0/1 variables) we can write more simply

$$p(\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T} W \mathbf{x}}}{Z}$$

- Given a set of data we would like to set the weight matrix W to maximise the log likelihood

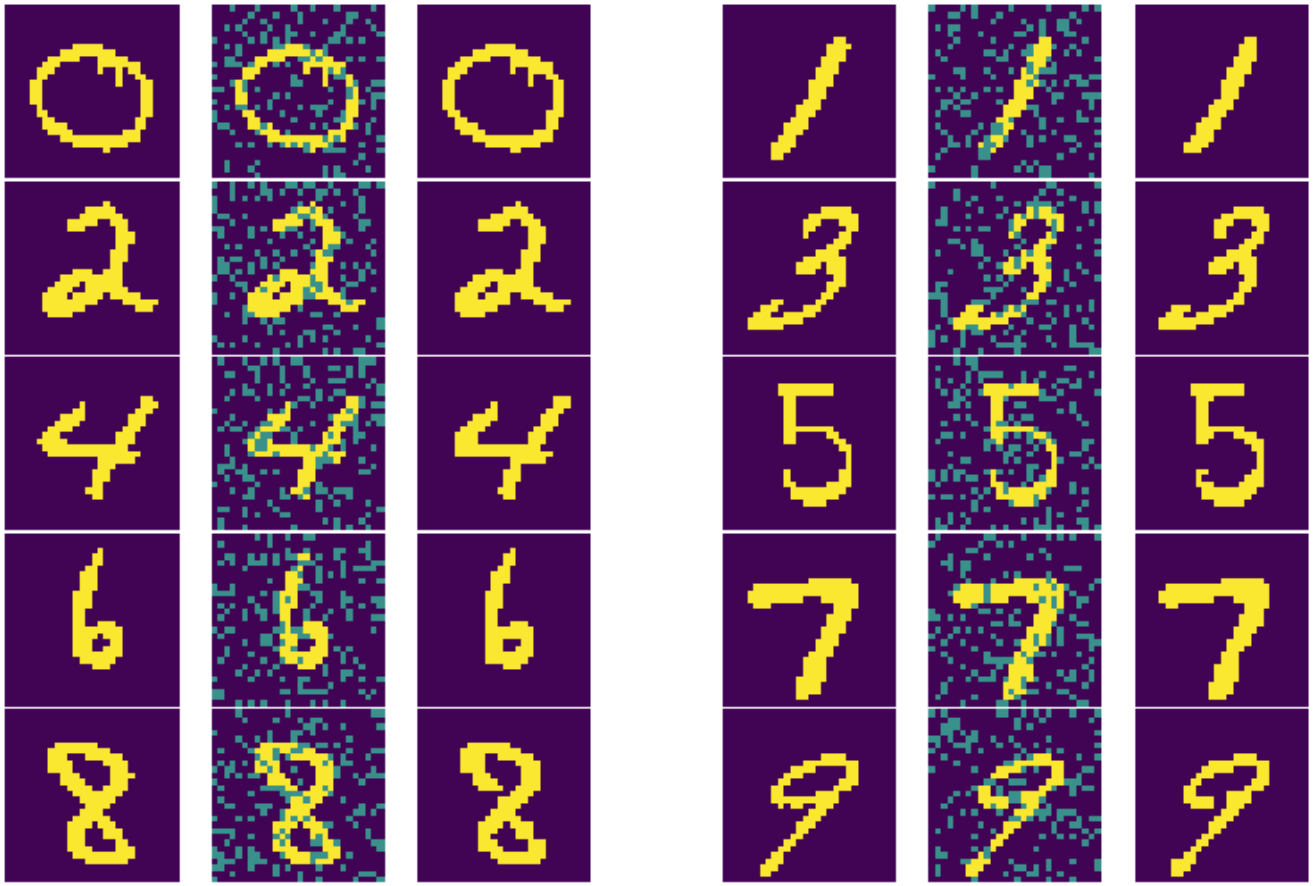
$$\sum_{n=1}^{N} \log p(\mathbf{x}^n) = \sum_n (\mathbf{x}^n)^\mathsf{T} W \mathbf{x}^n - N \log Z$$

- The gradient wrt the matrix W is (exercise)

$$\sum_n \mathbf{x}^n (\mathbf{x}^n)^\mathsf{T} - N \left\langle \mathbf{x}\mathbf{x}^\mathsf{T} \right\rangle$$

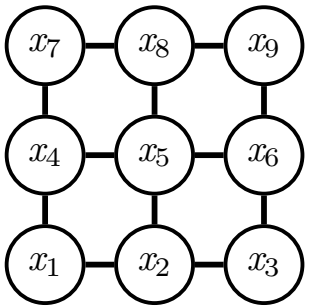
  where the expectation $\langle \ldots \rangle$ is wrt the BM $p(\mathbf{x})$.
- This expectation is computationally intractable and approximations are required (for example Contrastive Divergence). See tutorial.

Training a Boltzmann Machine



- Left: original. Centre: image with missing pixels. Right: completed.
- The MNIST dataset consists of $28 \times 28 = 784$ pixel images. There are 60,000 training images.
- We use a training method (similar to Contrastive Divergence) to learn W.
- We then assume some pixel values are missing and try to find the most likely pixel value to complete the image.

The Ising model



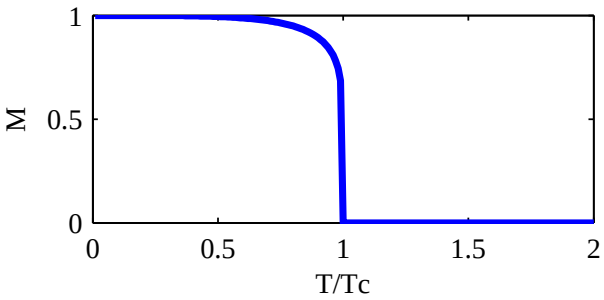
$x_i \in \{+1, -1\}$:

$$p(x_1, \ldots, x_9) = \frac{1}{Z} \prod_{i \sim j} \phi_{ij}(x_i, x_j)$$

$$\phi_{ij}(x_i, x_j) = e^{-\frac{1}{2T}(x_i - x_j)^2}$$

$i \sim j$ denotes the set of indices where i and j are neighbours in the graph. The potential encourages neighbours to be in the same state.

Spontaneous global behaviour



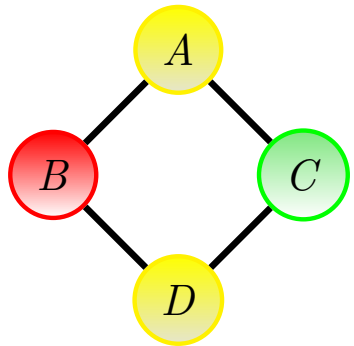
$M = |\sum_{i=1}^{N} x_i|/N$. As the temperature T decreases towards the critical temperature T_c a phase transition occurs in which a large fraction of the variables become aligned in the same state. Even though we only 'softly' encourage neighbours to be in the same state, for a low but finite T, the variables are all in the same state. Paradigm for 'emergent behaviour'.

An amazing result

- There is a nice demo at `http://physics.weber.edu/schroeder/software/demos/IsingModel.html` which shows how the pixels start to cluster into the same state as we decrease the temperature.
- Indeed, it was thought (incorrectly) in the early 20th Century that in order for all variables to be in the same state we can only be sure this will happen provided the temperature goes to zero.
- It was a major result in science that this is not the case. For a finite temperature all variables becomes aligned with extremely high probability.
- This tendency for all 'spins' to align in the same direction (even though they are not 'forced' to) is a central result and provides a canonical example of how macroscopic 'flocking' behaviour can arise from only weak local preferences to be aligned. This is a very deep result in science.
- As we increase the temperature, above the Curie-temperature (which is finite) then the spins become randomly aligned (also very surprising).

Independence Properties of Markov Networks

Independence in Markov Networks

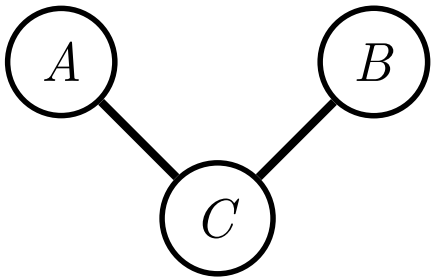


$B \perp\!\!\!\perp C \mid A, D$?

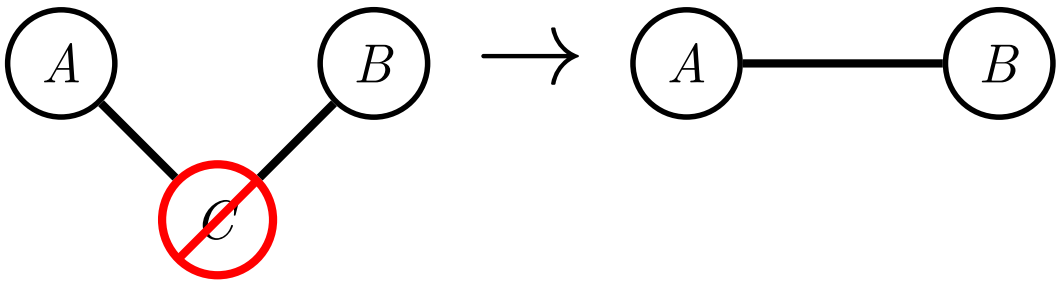
$p(B|A, D, C) = p(B|A, D)$?

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

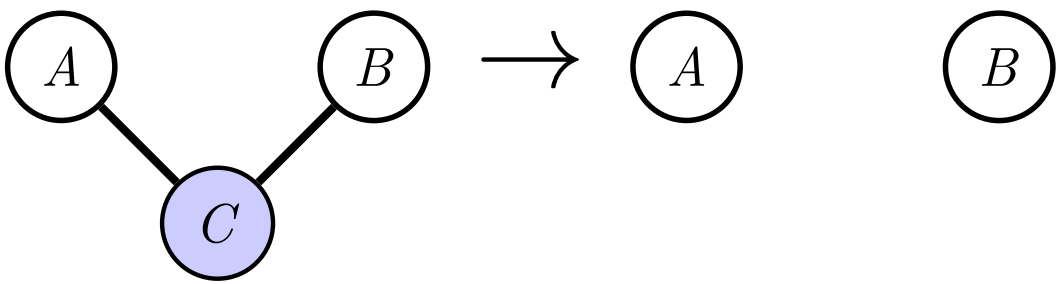
Properties of Markov Networks



$$p(A, B, C) = \phi_{AC}(A, C)\phi_{BC}(B, C)/Z$$

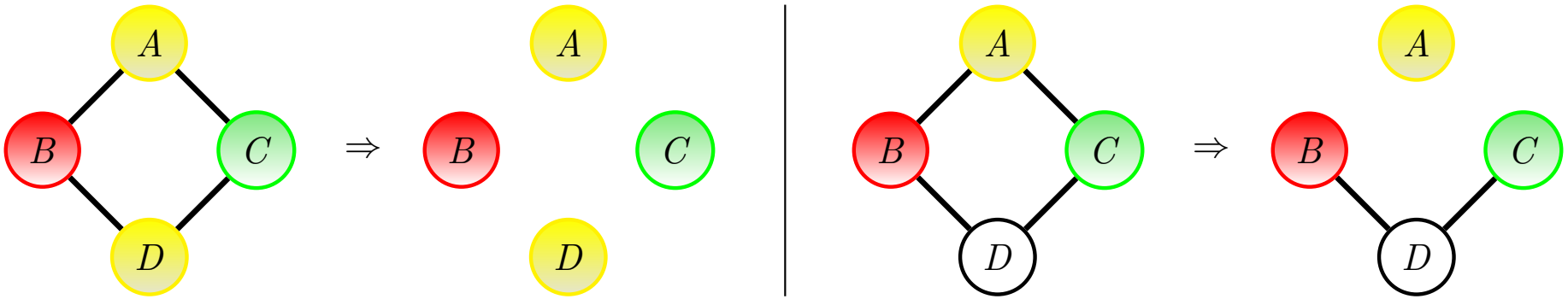


Marginalising over C makes A and B (graphically) dependent. In general $p(A, B) \neq p(A)p(B)$.



Conditioning on C makes A and B independent: $p(A, B|C) = p(A|C)p(B|C)$.

General Rule for Independence in Markov Networks

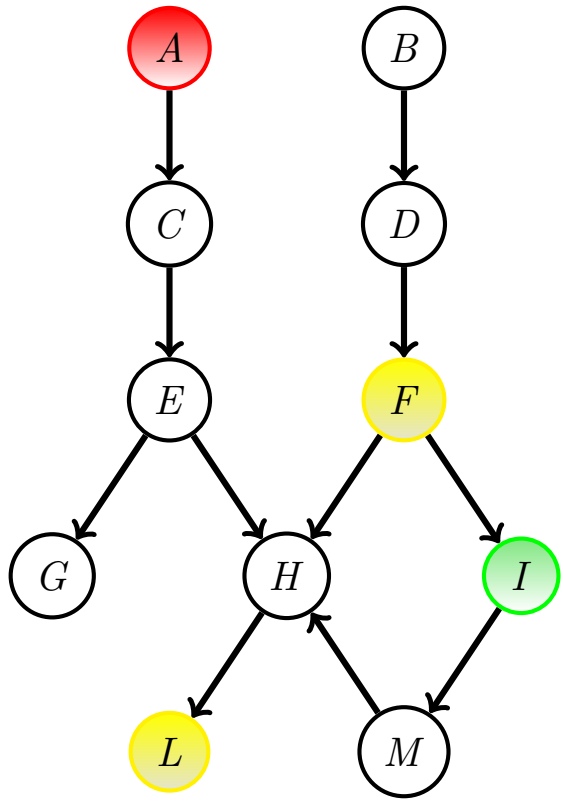


- Remove all links neighbouring the variables in the conditioning set $\mathcal{Z}$.
- If there is no path from any member of $\mathcal{X}$ to any member of $\mathcal{Y}$, then $\mathcal{X}$ and $\mathcal{Y}$ are conditionally independent given $\mathcal{Z}$.

Rule for Independence in Belief/Markov Networks

$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$?

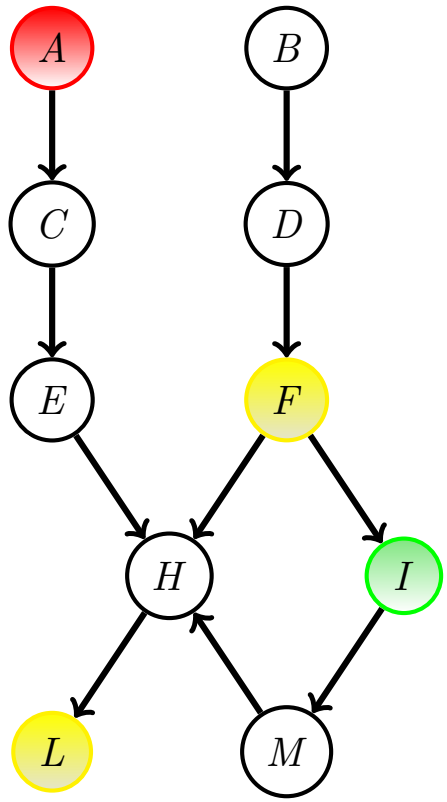
- Ancestral Graph: Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ nor an ancestor of a node in this set, together with any edges in or out of such nodes.
- Moralisation: Add a line between any two nodes which have a common child. Remove arrowheads.
- Separation: Remove all links from $\mathcal{Z}$.
- Independence: If there are no paths from any node in $\mathcal{X}$ to one in $\mathcal{Y}$ then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{Z}$.



Rule for Independence in Belief/Markov Networks

$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$?

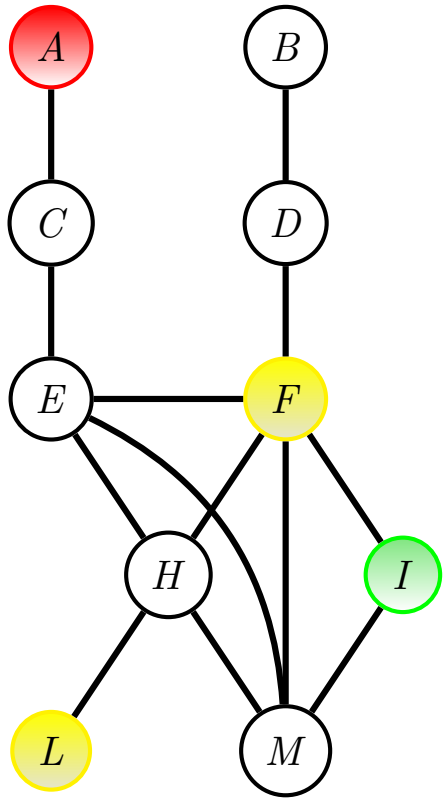
- Ancestral Graph: Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ nor an ancestor of a node in this set, together with any edges in or out of such nodes.
- Moralisation: Add a line between any two nodes which have a common child. Remove arrowheads.
- Separation: Remove all links from $\mathcal{Z}$.
- Independence: If there are no paths from any node in $\mathcal{X}$ to one in $\mathcal{Y}$ then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{Z}$.



Rule for Independence in Belief/Markov Networks

$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$?

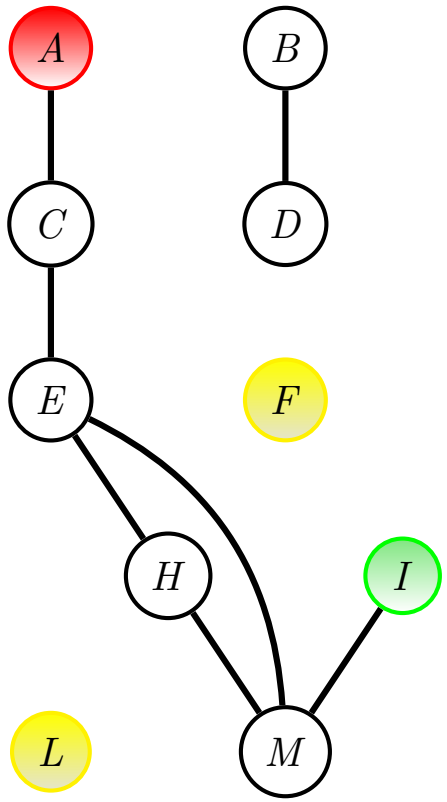
- Ancestral Graph: Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ nor an ancestor of a node in this set, together with any edges in or out of such nodes.
- Moralisation: Add a line between any two nodes which have a common child. Remove arrowheads.
- Separation: Remove all links from $\mathcal{Z}$.
- Independence: If there are no paths from any node in $\mathcal{X}$ to one in $\mathcal{Y}$ then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{Z}$.



Rule for Independence in Belief/Markov Networks

$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{Z}$?

- Ancestral Graph: Remove any node which is neither in $\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ 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.
- Separation: Remove all links from $\mathcal{Z}$.
- Independence: If there are no paths from any node in $\mathcal{X}$ to one in $\mathcal{Y}$ then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} \,|\, \mathcal{Z}$.



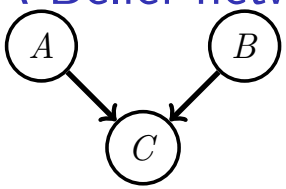
$A \top\!\!\!\top I | F, L$

Expressiveness of Markov and Belief Networks

Expressiveness of Belief and Markov Networks

Cannot represent independence information in certain belief networks with a Markov network.

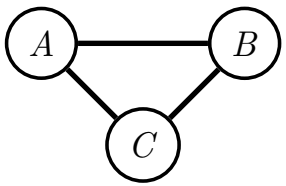
A Belief network



$$A \perp\!\!\!\perp B$$

Markov representation?

Since we have a term $p(C|A,B)$, the MN must have the clique A, B, C:

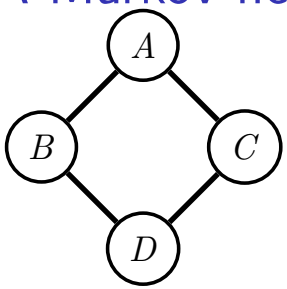


$$A \not\perp\!\!\!\perp B$$

Expressiveness of Belief and Markov Networks

Cannot represent independence information in certain Markov networks with a Belief network.

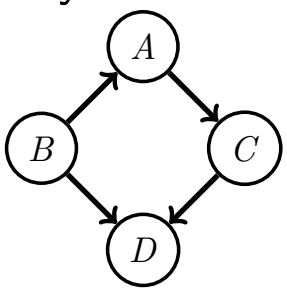
A Markov network



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

Belief Network representation?

Any DAG on A, B, C, D must have a collider.



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

Representations of distributions

- For a distribution P form the list $\mathcal{L}_P$ of all the independence statements.
- For a graph G, form the list of all independence statements $\mathcal{L}_G$.

Then we define:

$\mathcal{L}_P \subseteq \mathcal{L}_G$	Dependence Map (D-map)
$\mathcal{L}_P \supseteq \mathcal{L}_G$	Independence Map (I-map)
$\mathcal{L}_P = \mathcal{L}_G$	Perfect Map

In the above we assume the statement l is contained in $\mathcal{L}$ if it is consistent with (can be derived from) the independence statements in $\mathcal{L}$.

Representations of distributions

$$p(t_1, t_2, y_1, y_2) = p(t_1)p(t_2)\sum_h p(y_1|t_1, h)p(y_2|t_2, h)p(h)$$

$$\mathcal{L}_P = \{y_1 \perp\!\!\!\perp t_2 |\, t_1, \quad y_2 \perp\!\!\!\perp t_1 |\, t_2, \quad t_2 \perp\!\!\!\perp t_1\}$$

Consider the graph of the BN

$$p(y_2|y_1, t_1, t_2)p(y_1|t_1)p(t_1)p(t_2)$$

For this we have $\mathcal{L}_G = \{y_1 \perp\!\!\!\perp t_2 |\, t_1, \quad t_2 \perp\!\!\!\perp t_1\}$

- $\mathcal{L}_G \subset \mathcal{L}_P$ so that the BN is an I-MAP for p since every independence statement in the BN is true for the corresponding graph.
- Since $\mathcal{L}_P \not\subseteq \mathcal{L}_G$ the BN is not a D-MAP for p.
- In this case no perfect MAP (a BN or a MN) can represent p.

Representing dependence?

GMs are generally most suited to representing independence. The reason is that local dependence doesn't imply global dependencies. For example

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

$$p(a) = \begin{pmatrix} 3/5 \\ 2/5 \end{pmatrix}, p(b|a) = \begin{pmatrix} 1/4 & 15/40 \\ 1/12 & 1/8 \\ 2/3 & 1/2 \end{pmatrix}, p(c|b) = \begin{pmatrix} 1/3 & 1/2 & 15/40 \\ 2/3 & 1/2 & 5/8 \end{pmatrix}$$

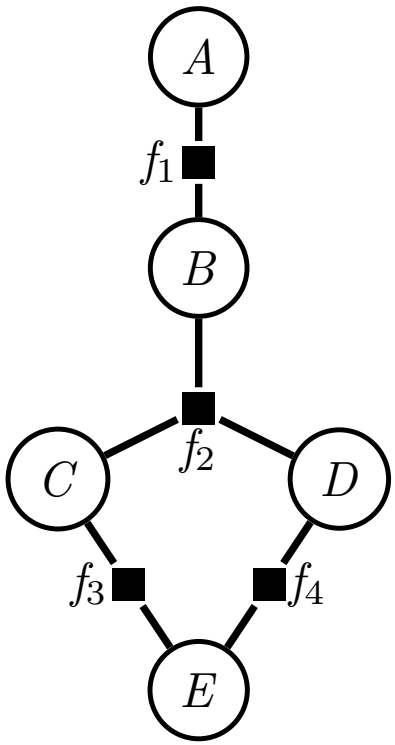
For these tables, $a \not\perp\!\!\!\perp b$, $b \not\perp\!\!\!\perp c$, but $a \perp\!\!\!\perp c$.

- Local dependence does not guarantee dependence of path-connected variables.
- Graphical independence $\rightarrow$ distribution independence.
- Graphical dependence $\not\rightarrow$ distribution dependence.
- The moral of the story is that graphical models cannot generally enforce distributions to obey the dependencies implied by the graph.

Factor Graphs

Factor Graphs

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

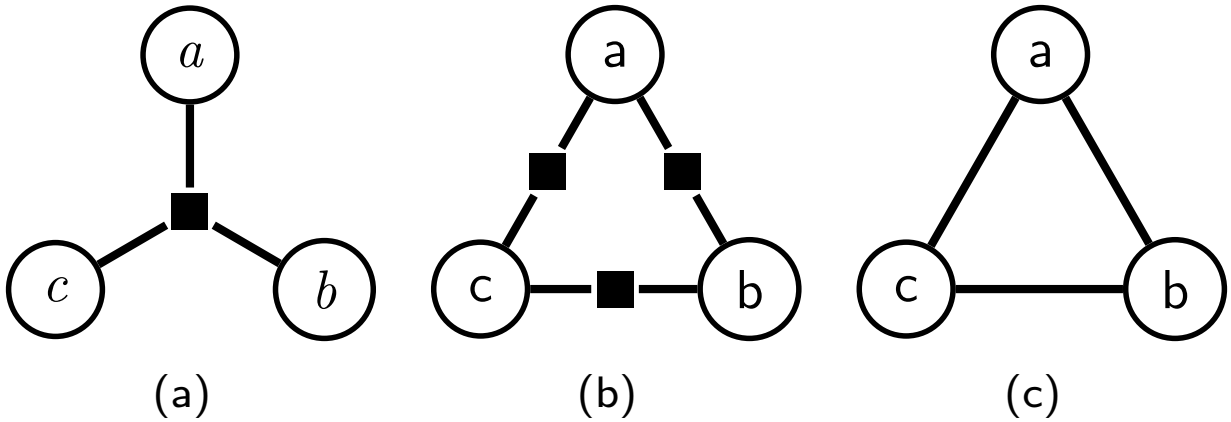


The joint function is the product of all factors:

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

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

Factor Graphs versus Markov Networks



a $\phi(a, b, c)$

b $\phi(a, b)\phi(b, c)\phi(c, a)$

c $\phi(a, b, c)$

- Both (a) and (b) have the same Markov network (c).
- Whilst (b) contains the same (lack of) independence statements as (a), it expresses more constraints on the form of the potential.

Summary

- There are many different types of Graphical Models and each corresponds to a certain factorisation assumption for the distribution.
- Belief Networks are commonly used to model situations in which there is a natural 'temporal' ordering of the variables. They are represented by DAGs.
- Markov Networks can represent dependencies between unordered variables. They have widespread application in the physical sciences (for example in lattice models) and computer science (particularly machine vision).
- A profound result is that, despite weak preferences for neighbouring variables in the MN to be in the same state, this can almost guarantee that all variables can be in the same state (for example in the Ising Model).
- There is an algorithm for ascertaining independence of variables that works for both BNs and MNs.
- The Boltzmann Machine is a classical model in Machine Learning and has been studied extensively.
- Factor Graphs can represent more refined statements about the form of the factorisation than Markov Networks can. FGs are most commonly used for inference (rather than modelling). Typically graphs are 'compiled' into a FG form, on which an inference algorithm is then executed, as we will later discuss.