Graphical Models

Machine Learning I
Winter Semester 2014-2015

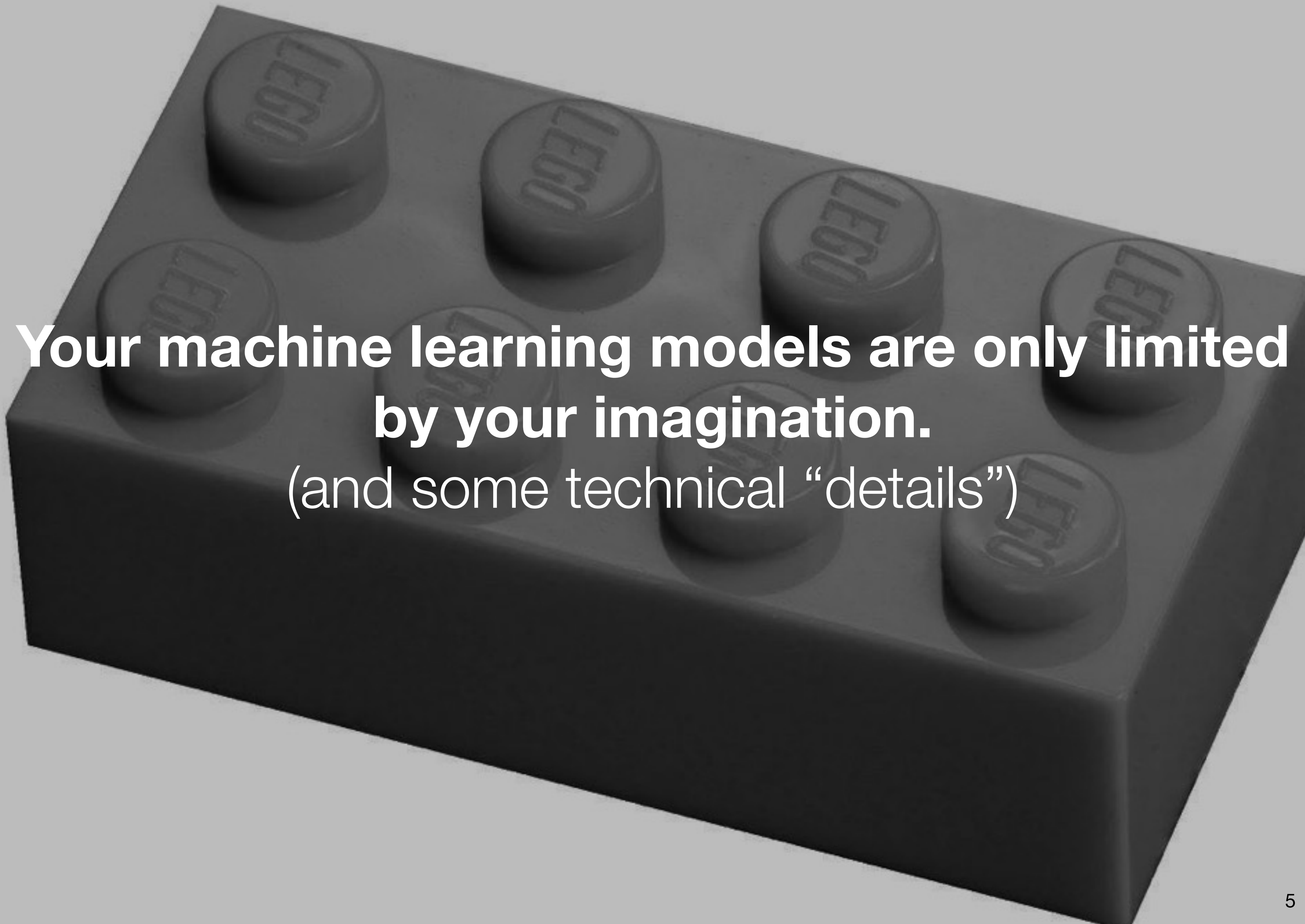
You could think of graphical models as
machine learning lego

Using graphical models you can describe

- Linear Discriminant Analysis (LDA)
- Naive Bayes (NB)
- Gaussian Mixture Models (GMM)
- Hidden Markov Models (HMM)
- Principal Component Analysis (PCA)
- Latent Dirichlet Allocation (also LDA)
- …

Anything

(that you can define in a probabilistic manner)



Your machine learning models are only limited by your imagination.

(and some technical “details”)

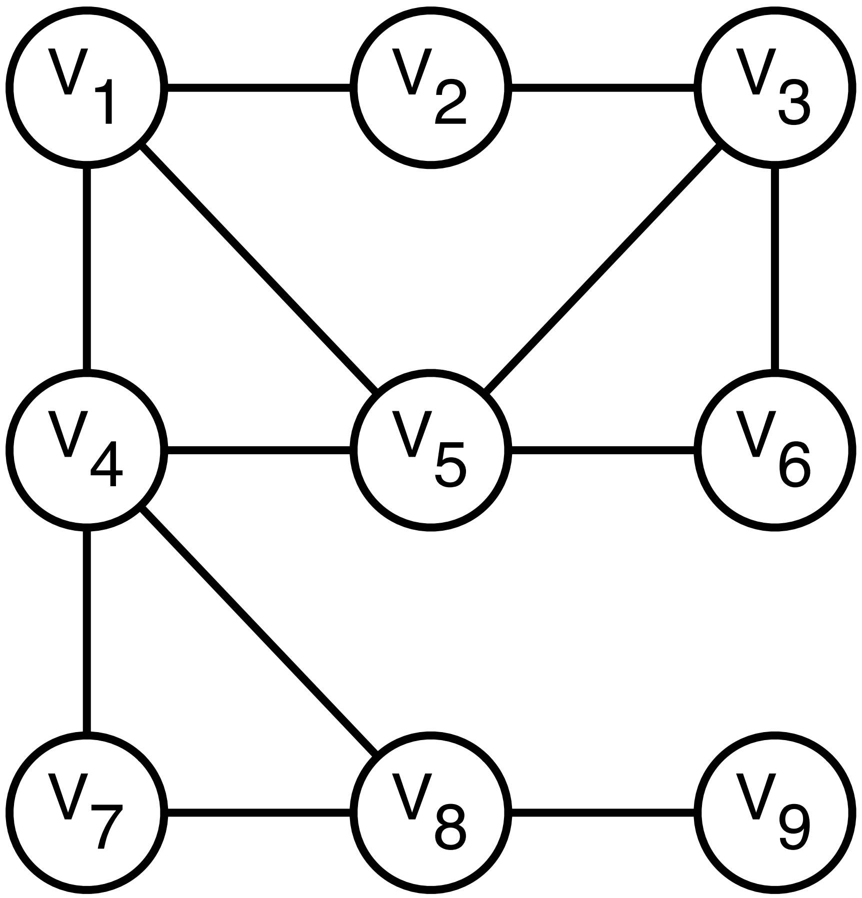
Graphical models are a framework to describe dependencies between random variables.

A language to convey concepts or ideas about probabilistic machine learning models

A graph

$G = (V, E)$

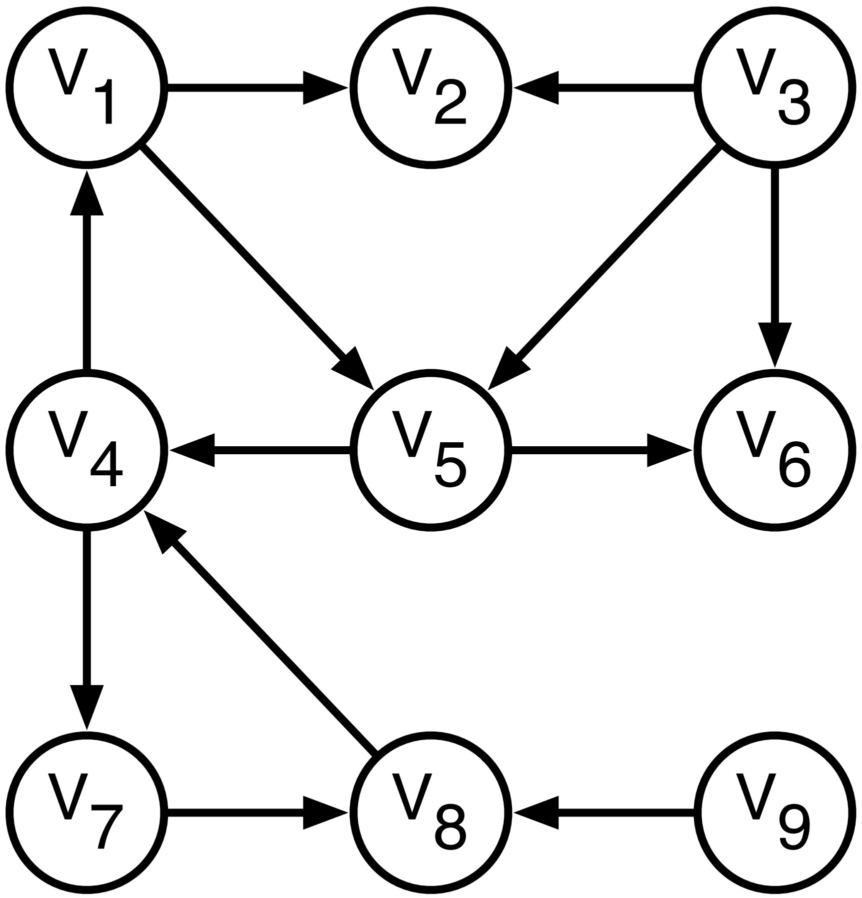
- vertices $V = \{V_1, \ldots, V_N\}$
- edges $E \in V \times V$
- $(V_1, V_2) = (V_2, V_1)$



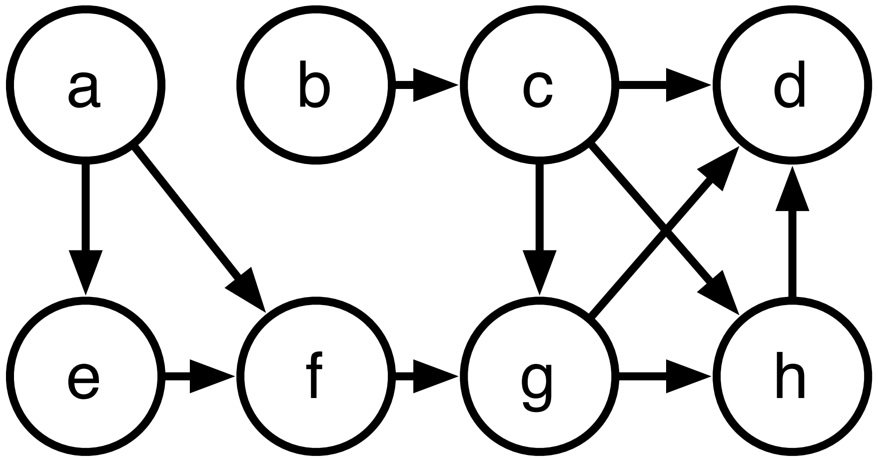
A directed graph

$G = (V, E)$

- vertices $V = \{V_1, \dots, V_N\}$
- edges $E \in V \times V$
- if $V_1 \rightarrow V_2$
  - V_1 is a parent of V_2
  - V_2 is a child of V_1



- A cycle is a path that starts and ends at the same node e.g V_1, V_5, V_4, V_1



Directed graphical models

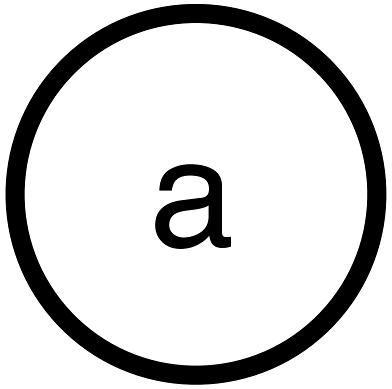
Sometimes referred to as Bayesian Networks

Directed graphical models

Pictorial notation of the (factorisation) of a probability mass/density function

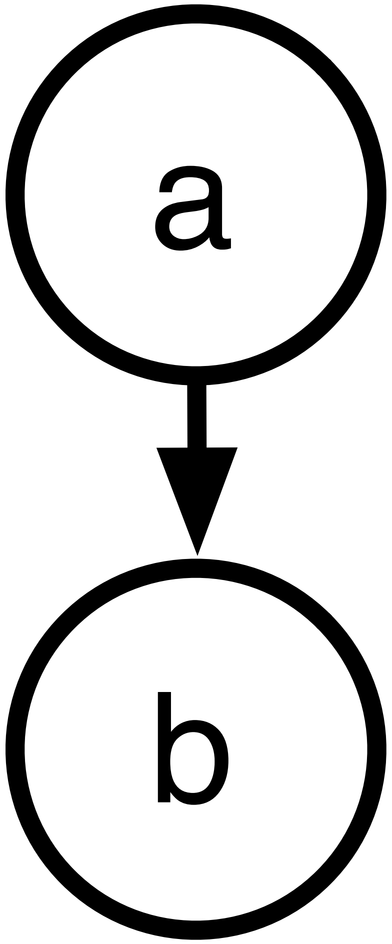
Single random variable A

$$P(A = a) = p(a)$$



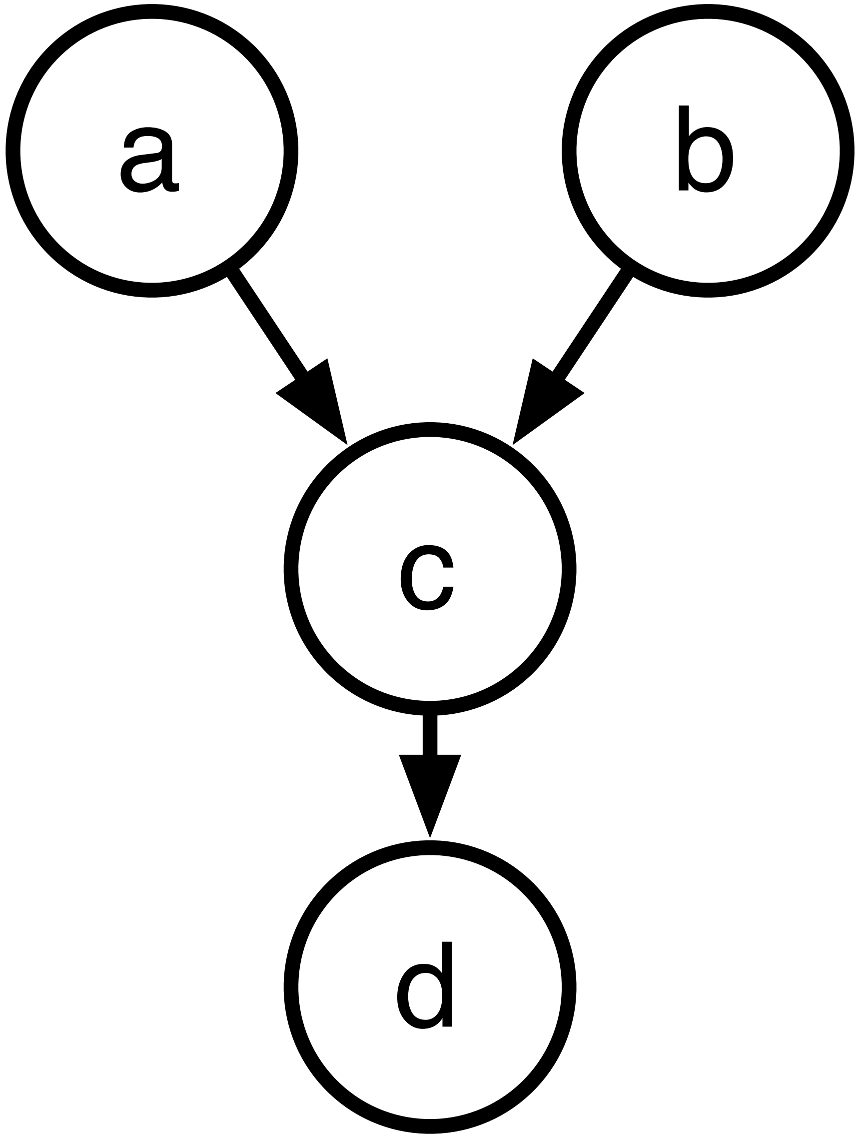
Conditional probability

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



One more example

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



Give me the graphical model for

$$p(a, b, c, x, y, z) = p(a)p(b)p(c|a, b)p(x|c)p(y|x, a)p(z|y)$$

Observed variables (often the data you measure)
B=b is observed

$$p(a|b)$$

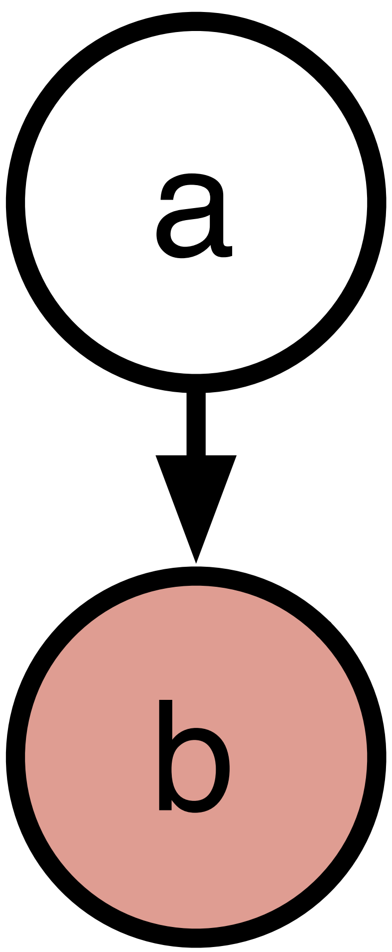
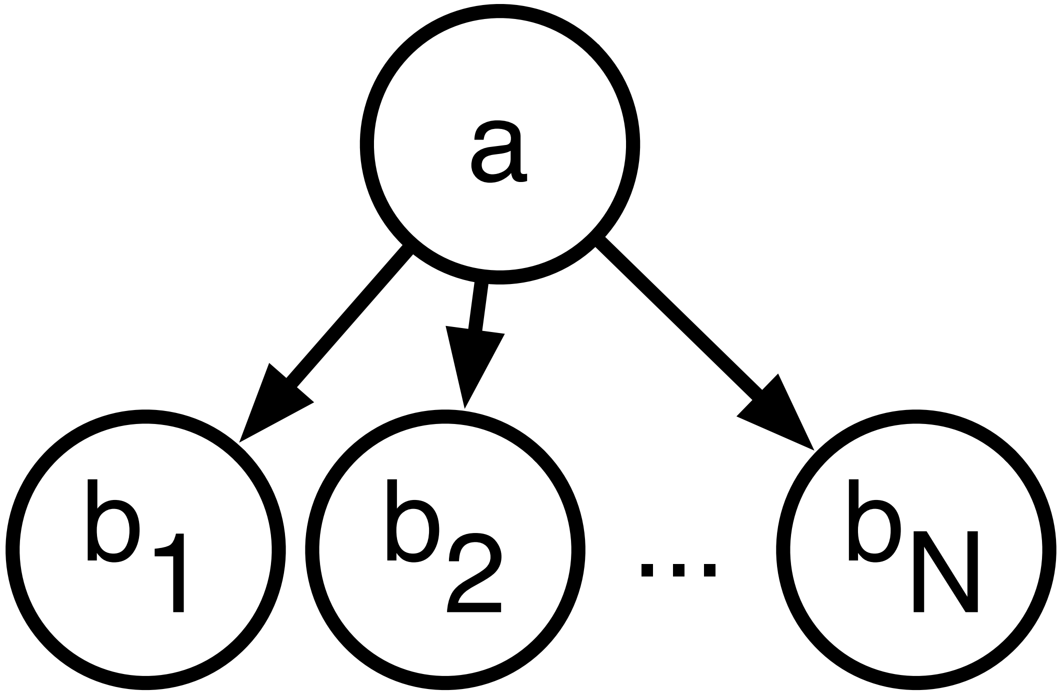
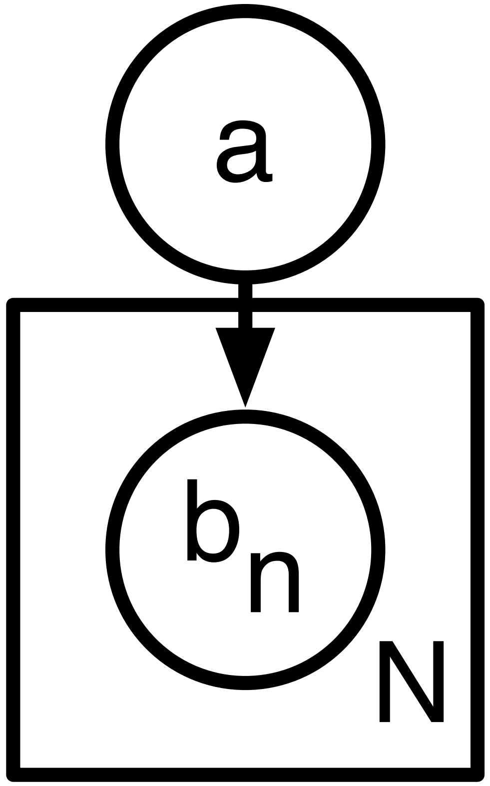


Plate notation (Repetition)

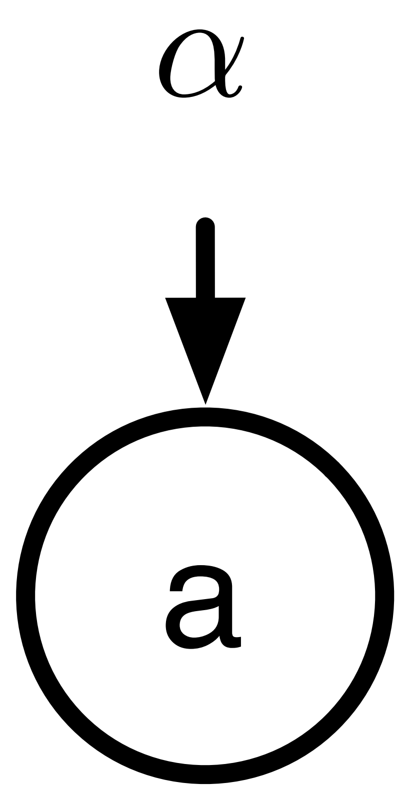
$$p(a, b_1, b_2, \ldots, b_N) = p(a) \prod_{n=1}^{N} p(b_n | a)$$





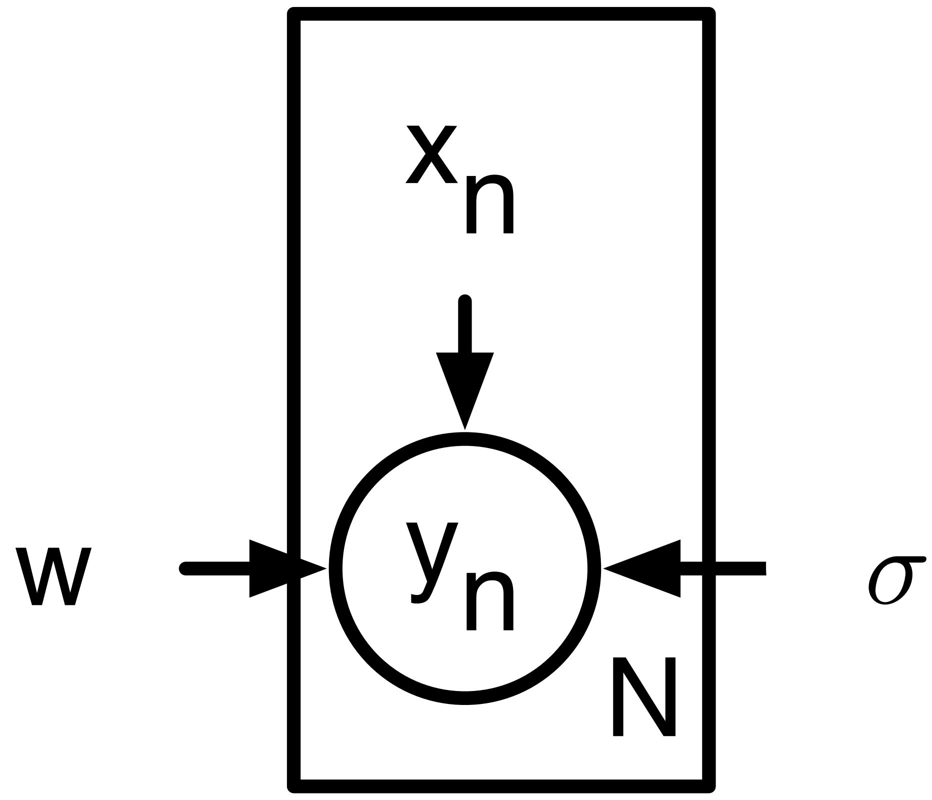
Making parameters/deterministic variables explicit (Bishop's notation)

$$p(a|\alpha)$$



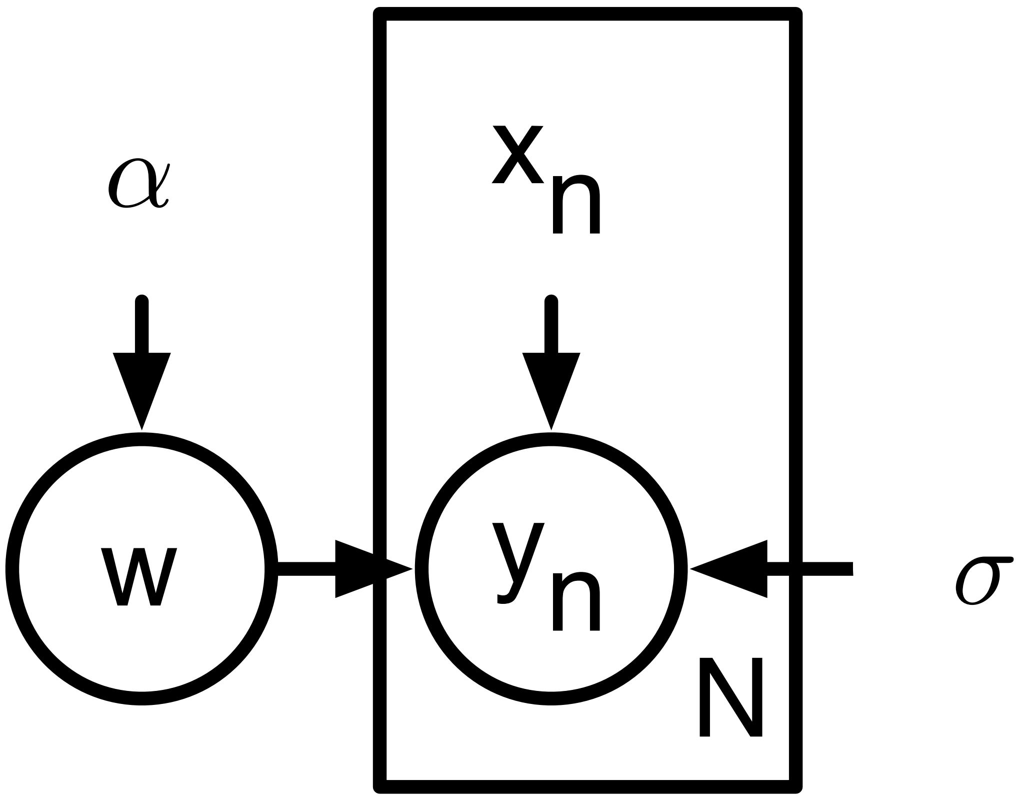
Linear regression

$$y_n \quad \sim \quad \mathcal{N}\left(\boldsymbol{w}^T \boldsymbol{x}_n, \sigma^2\right)$$



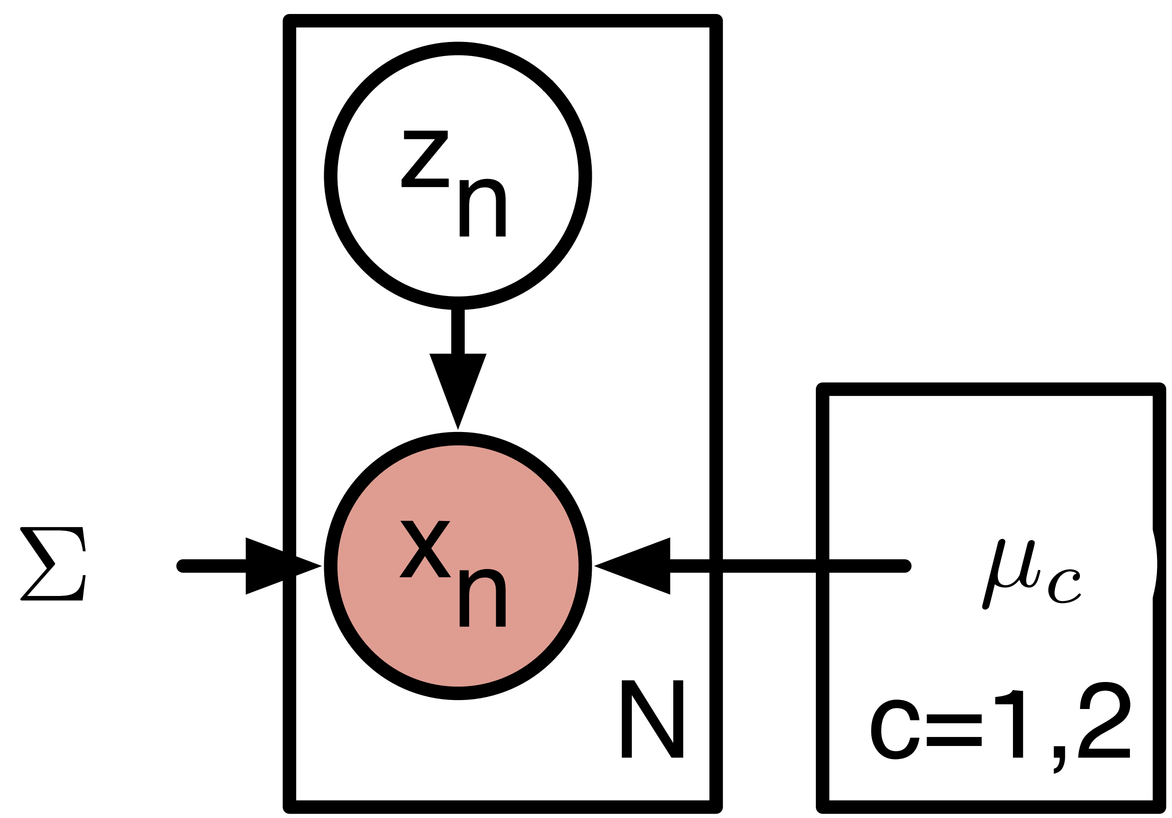
Bayesian Linear Regression

$$
\begin{aligned}
\boldsymbol{w} &\sim \mathcal{N}(\mathbf{0}, \alpha^2) \\
y_n | \boldsymbol{w} &\sim \mathcal{N}\left(\boldsymbol{w}^T \boldsymbol{x}_n, \sigma^2\right)
\end{aligned}
$$



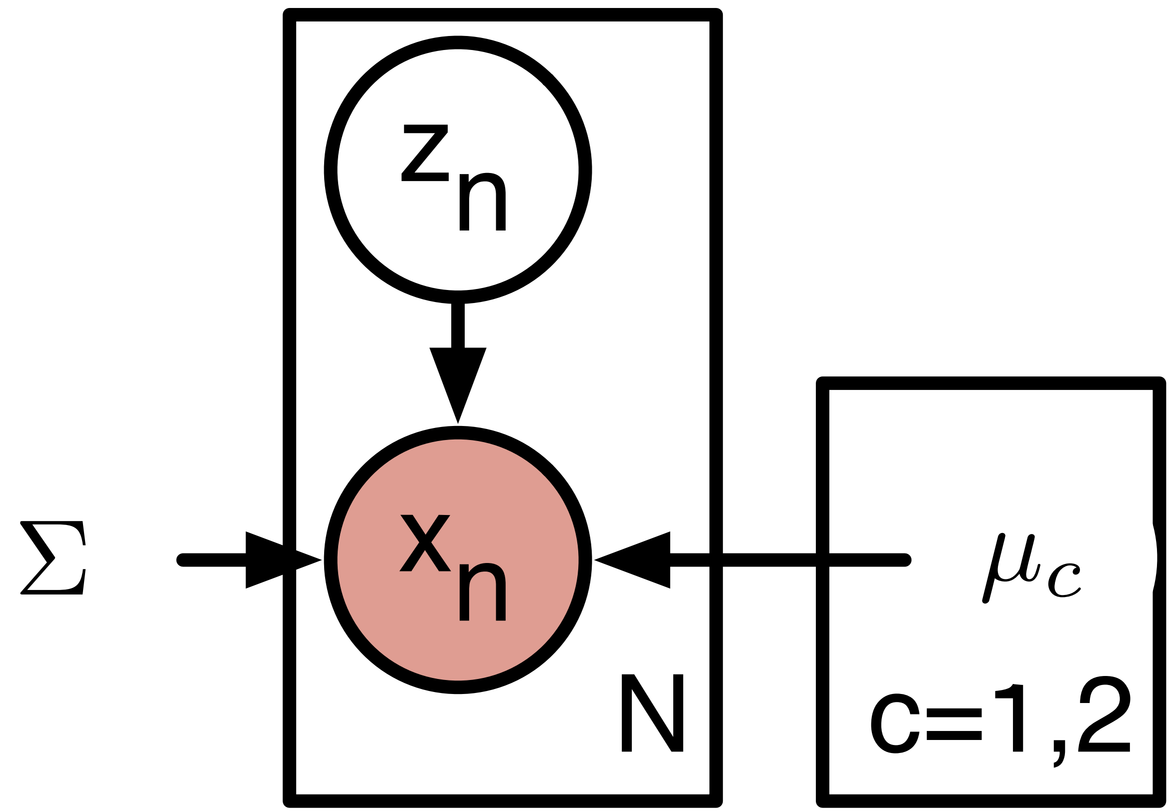
Linear Discriminant Analysis

$$
\begin{aligned}
z_n &\sim Bernouilli(\lambda) \\
\boldsymbol{x_n} | z_n &\sim \mathcal{N}(\boldsymbol{\mu}_{z_n}, \Sigma)
\end{aligned}
$$



Linear Discriminant Analysis

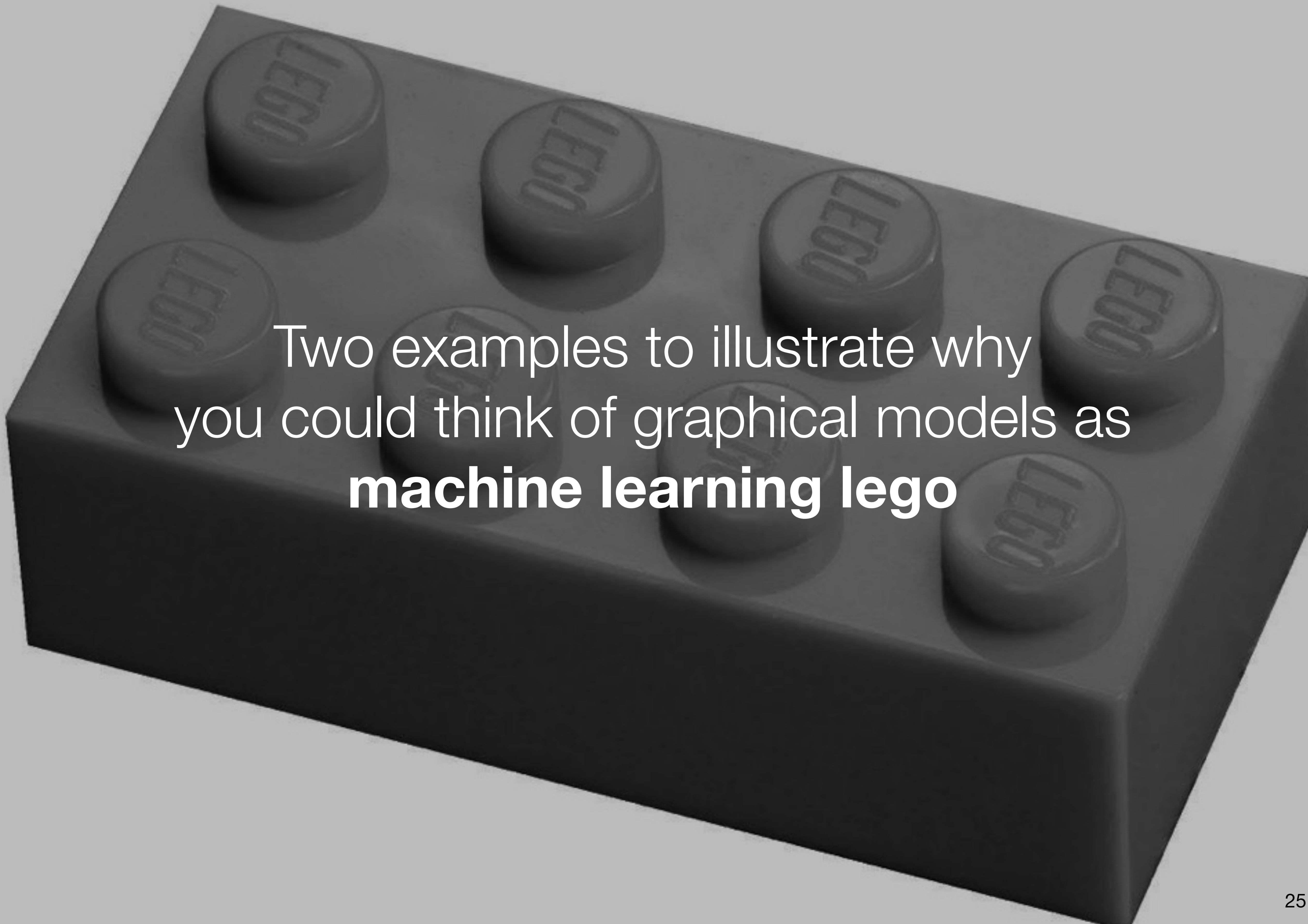
$$
\begin{aligned}
z_n &\sim Bernouilli(\lambda) \\
\boldsymbol{x_n}|z_n &\sim \mathcal{N}(\boldsymbol{\mu}_{z_n}, \Sigma)
\end{aligned}
$$



We can draw graphical representations of probability distributions on the blackboard.

What is the point?

A language to convey concepts or ideas about probabilistic machine learning models



Two examples to illustrate why
you could think of graphical models as
machine learning lego

Directed graphical models

Conditional independence

Conditional independence

We have three variables: *a*,*b* and *c*

a and *b* are conditionally independent given *c* if and

$$a \perp\!\!\!\perp b | c \iff p(a, b|c) = p(a|c)p(b|c)$$

special case: independence

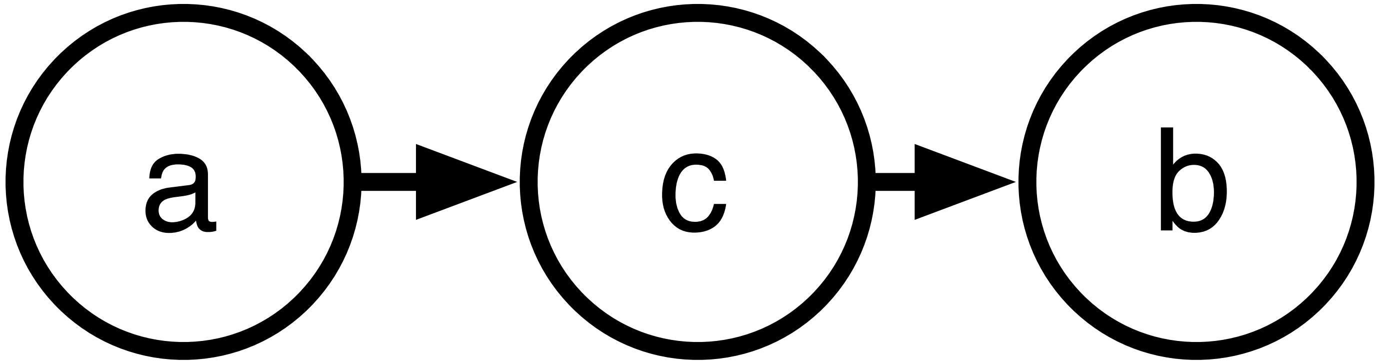
$$a \perp\!\!\!\perp b \iff p(a, b) = p(a)p(b)$$

Think of it as: the value of *a* does not directly influence the value of *b*

Head to tail

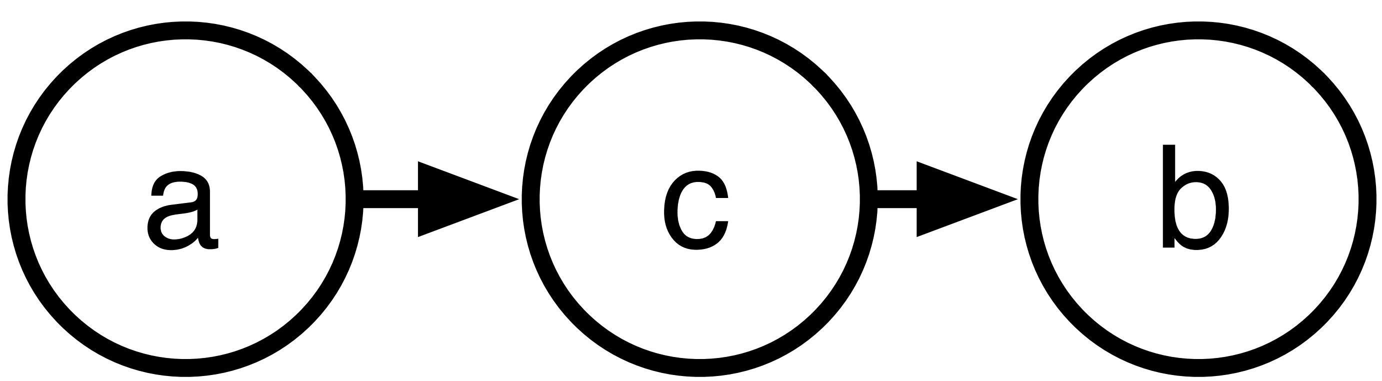
$$p(a,b,c) \quad = \quad p(a)p(c|a)p(b|c)$$

$$p(a,b)$$



Head to tail

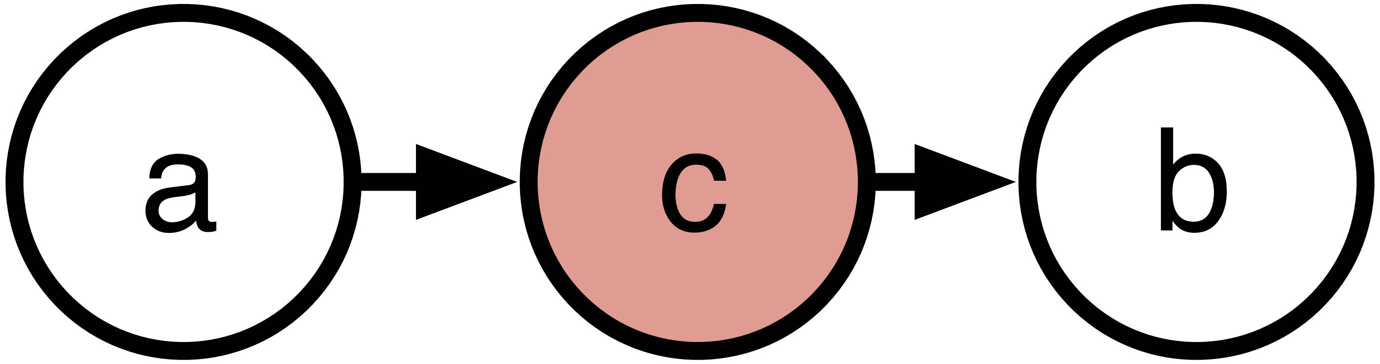
$$
\begin{aligned}
p(a, b, c) &= p(a)p(c|a)p(b|c) \\
\\
p(a, b) &= \sum_c p(a)p(c|a)p(b|c) \\
&= p(a) \sum_c p(c|a)p(b|c) \\
&= p(a)p(b|a)
\end{aligned}
$$



Head to tail

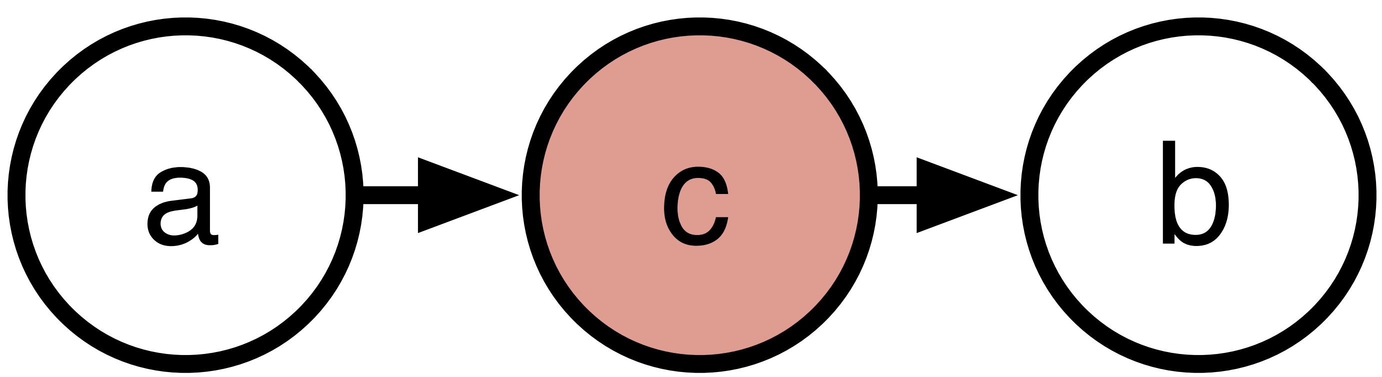
$$p(a, b, c) \quad = \quad p(a)p(c|a)p(b|c)$$

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



Head to tail

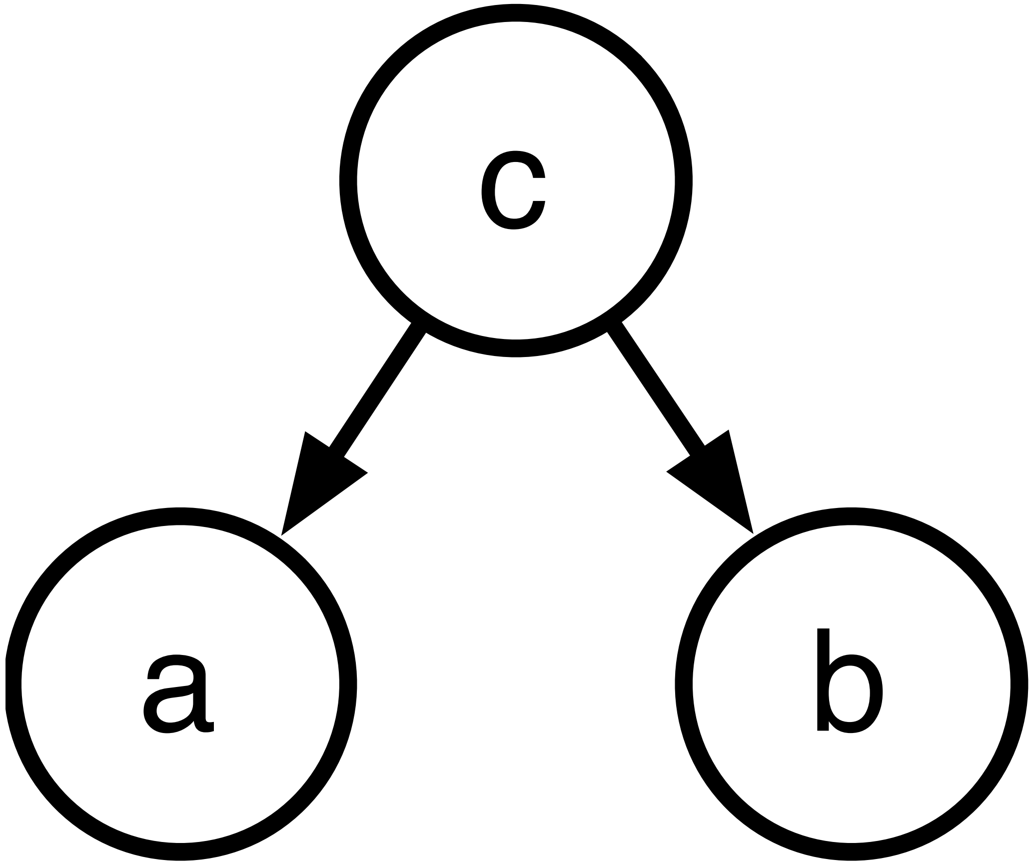
$$
\begin{aligned}
p(a, b, c) &= p(a)p(c|a)p(b|c) \\
\\
p(a, b|c) &= \frac{p(a, b, c)}{p(c)} \\
&= \frac{p(a)p(c|a)p(b|c)}{p(c)} \\
&= p(a|c)p(b|c) \\
&\Rightarrow a \perp\!\!\!\perp b|c
\end{aligned}
$$



Tail to tail

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

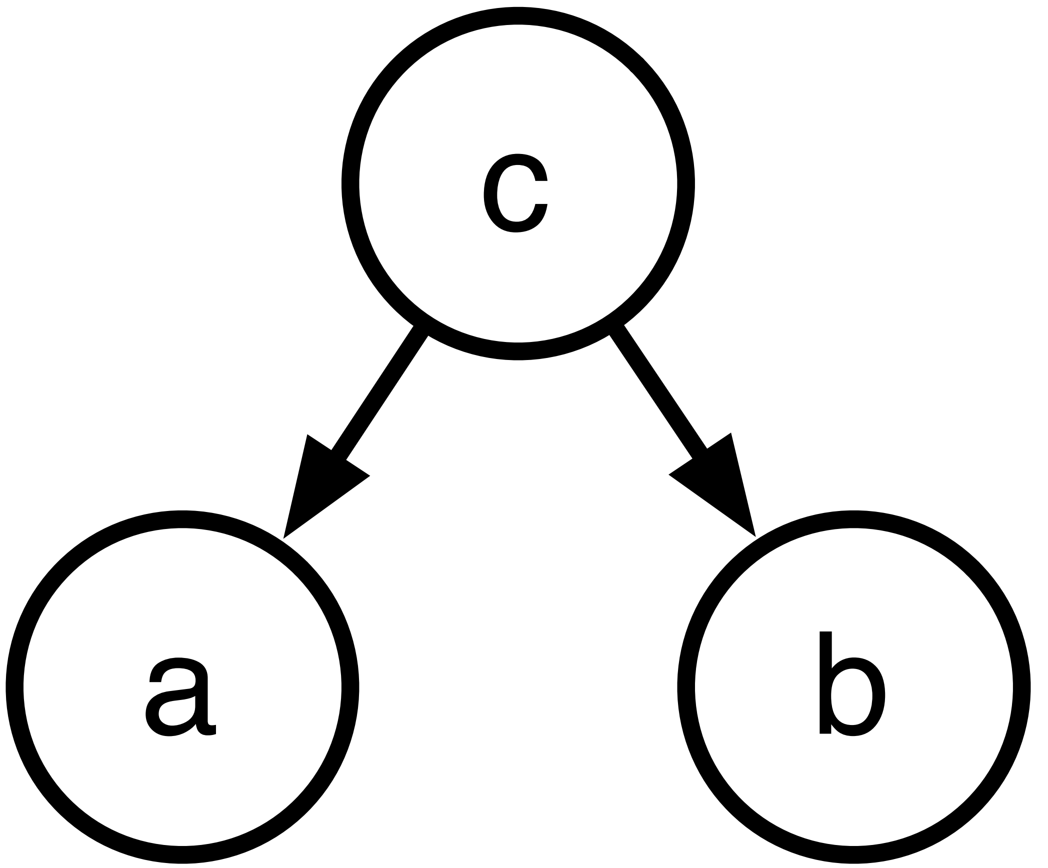
$$p(a, b)$$



Tail to tail

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

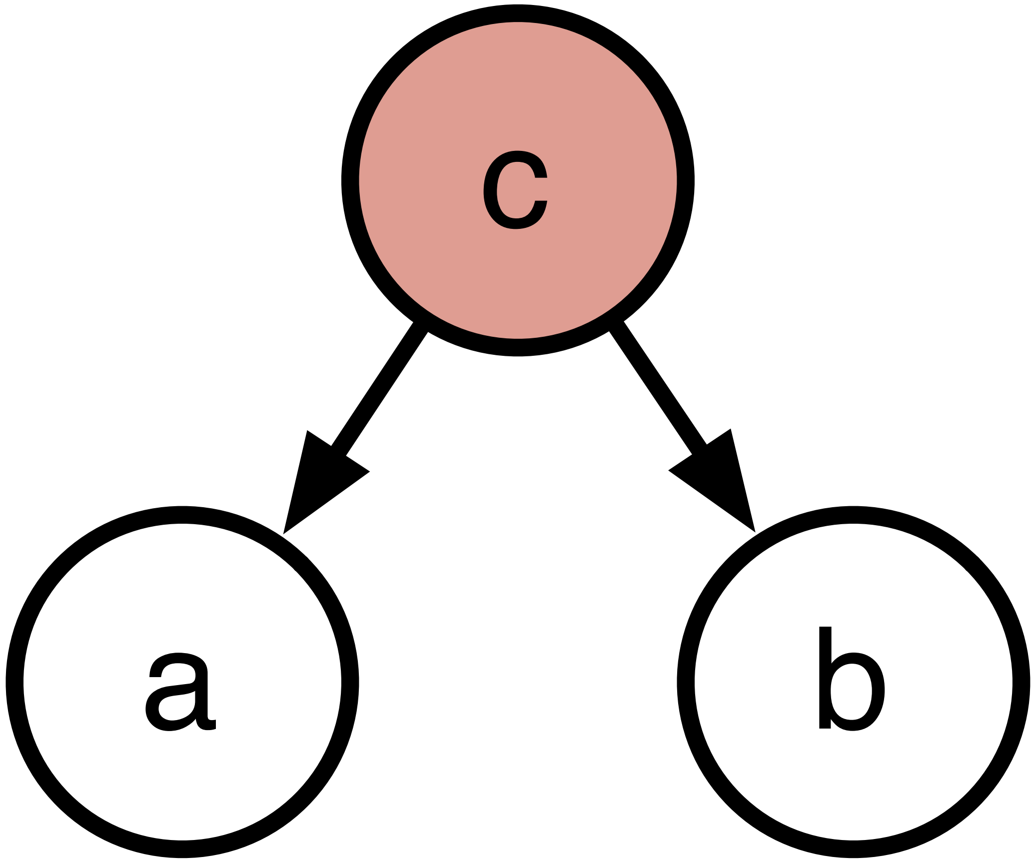
$$p(a, b) = \sum_c p(a|c)p(b|c)p(c)$$



Tail to tail

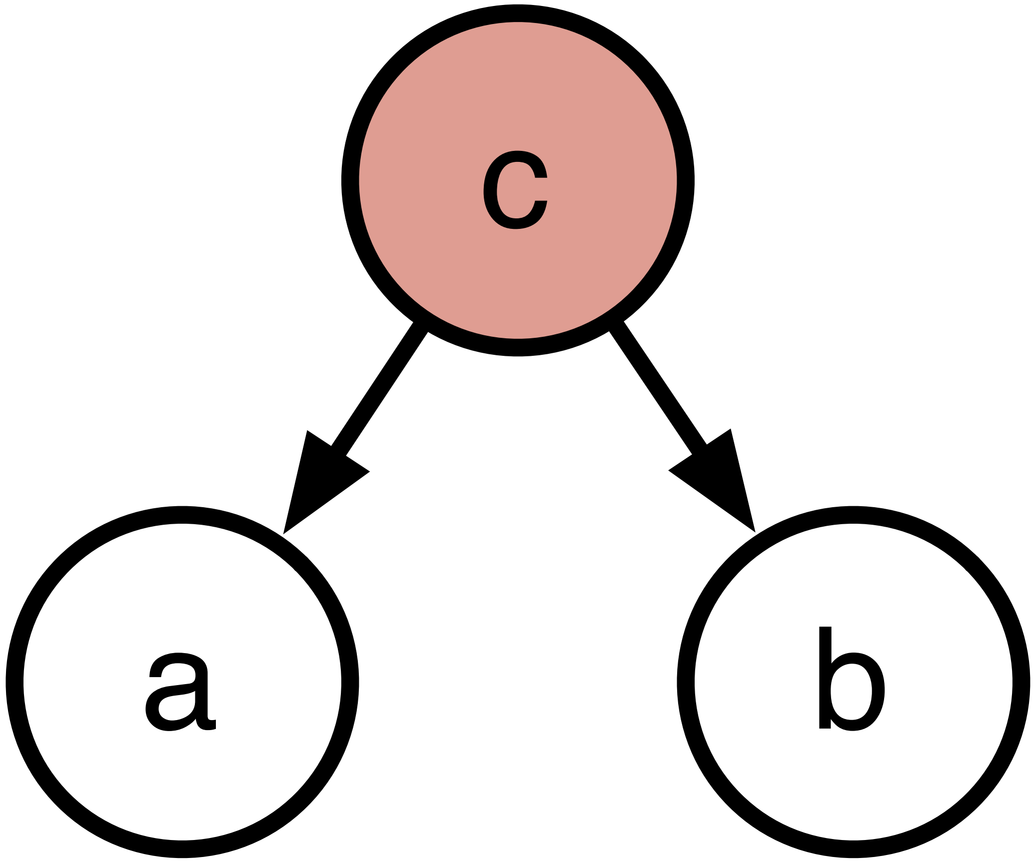
$$p(a,b,c) \quad = \quad p(a|c)p(b|c)p(c)$$

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



Tail to tail

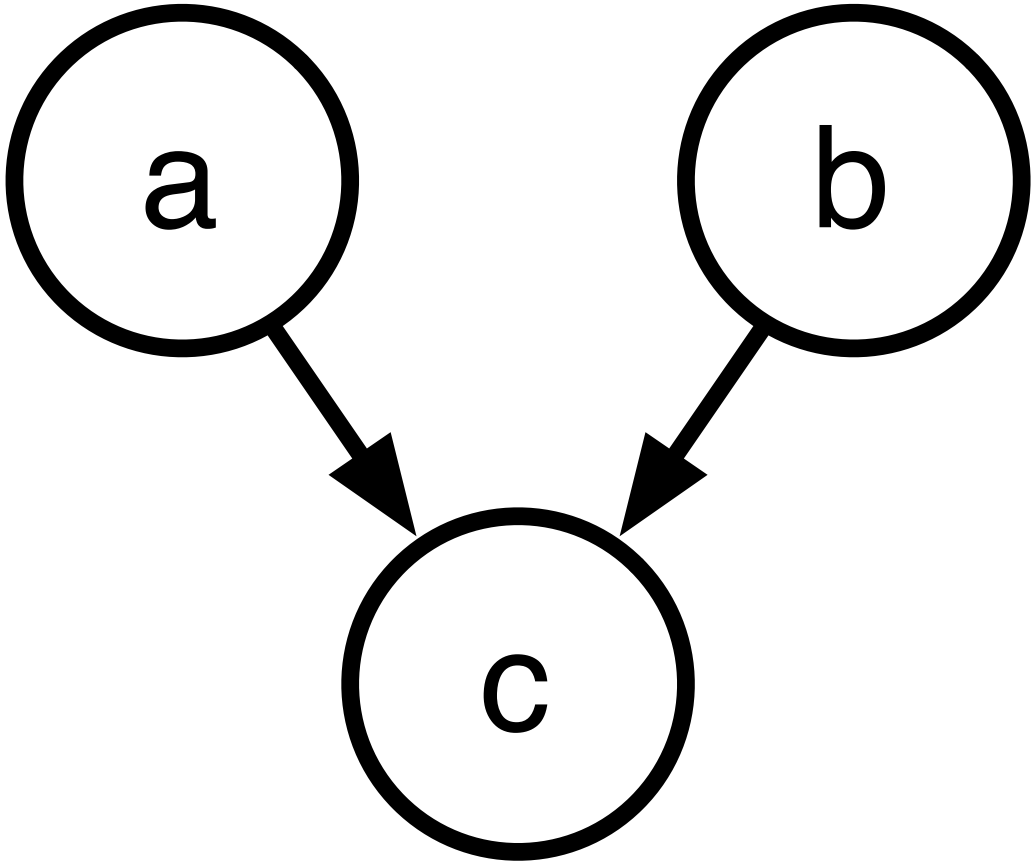
$$\begin{aligned} p(a,b,c) &= p(a|c)p(b|c)p(c) \\ p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\ &= \frac{p(a|c)p(b|c)p(c)}{p(c)} \\ &= p(a|c)p(b|c) \\ \Rightarrow & \quad a \perp\!\!\!\perp b|c \end{aligned}$$



Head to head

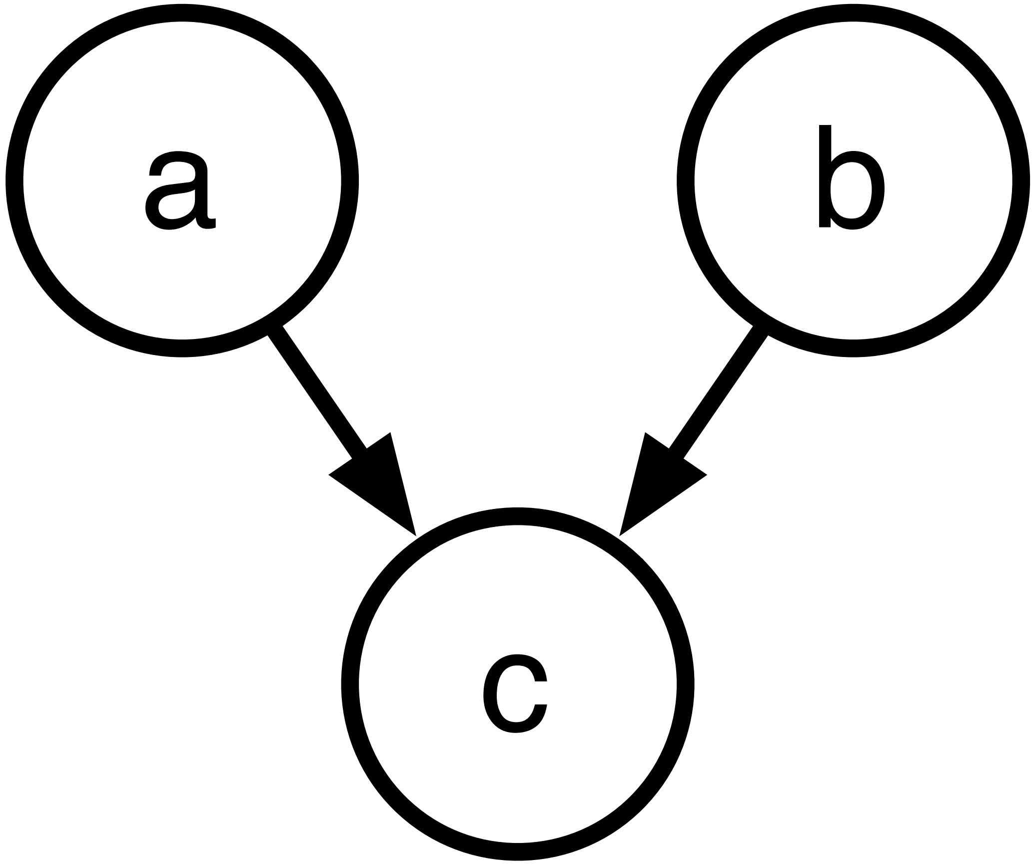
$$p(a,b,c) = p(a)p(b)p(c|a,b)$$

$$p(a,b)$$



Head to head

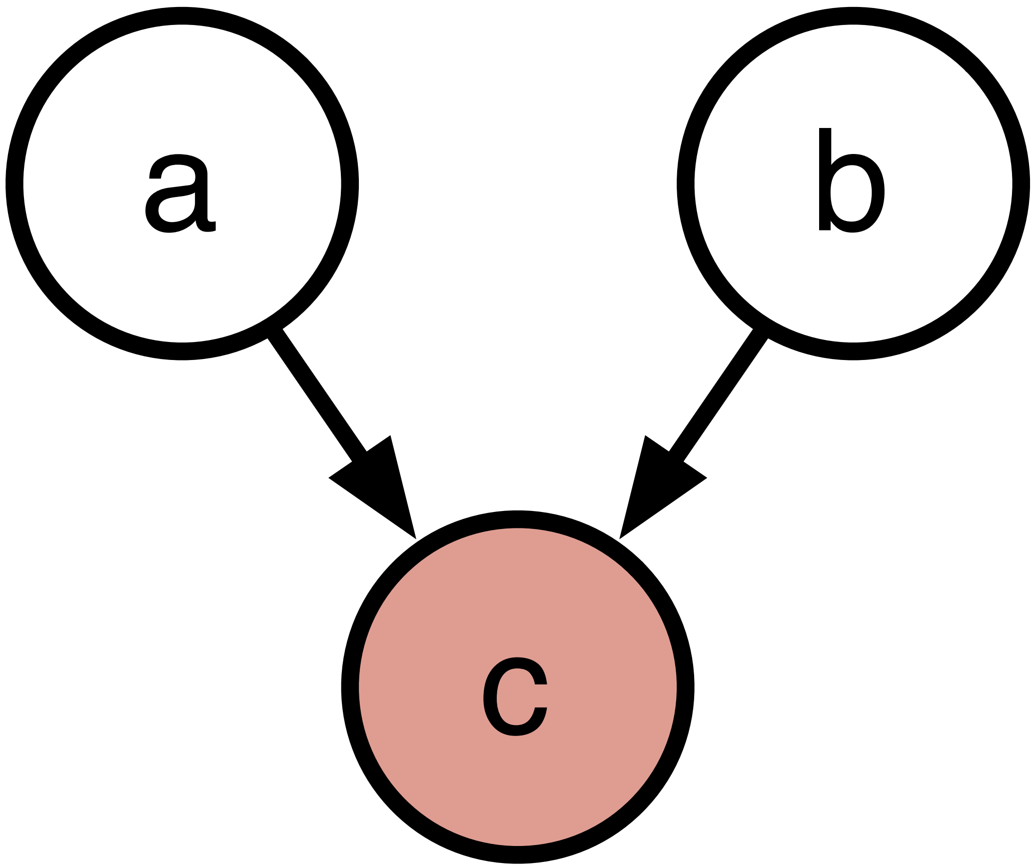
$$
\begin{aligned}
p(a,b,c) &= p(a)p(b)p(c|a,b) \\
p(a,b) &= p(a)p(b)\sum_c p(c|a,b) \\
&= p(a)p(b) \\
&\Rightarrow a \perp\!\!\!\perp b
\end{aligned}
$$



Head to head

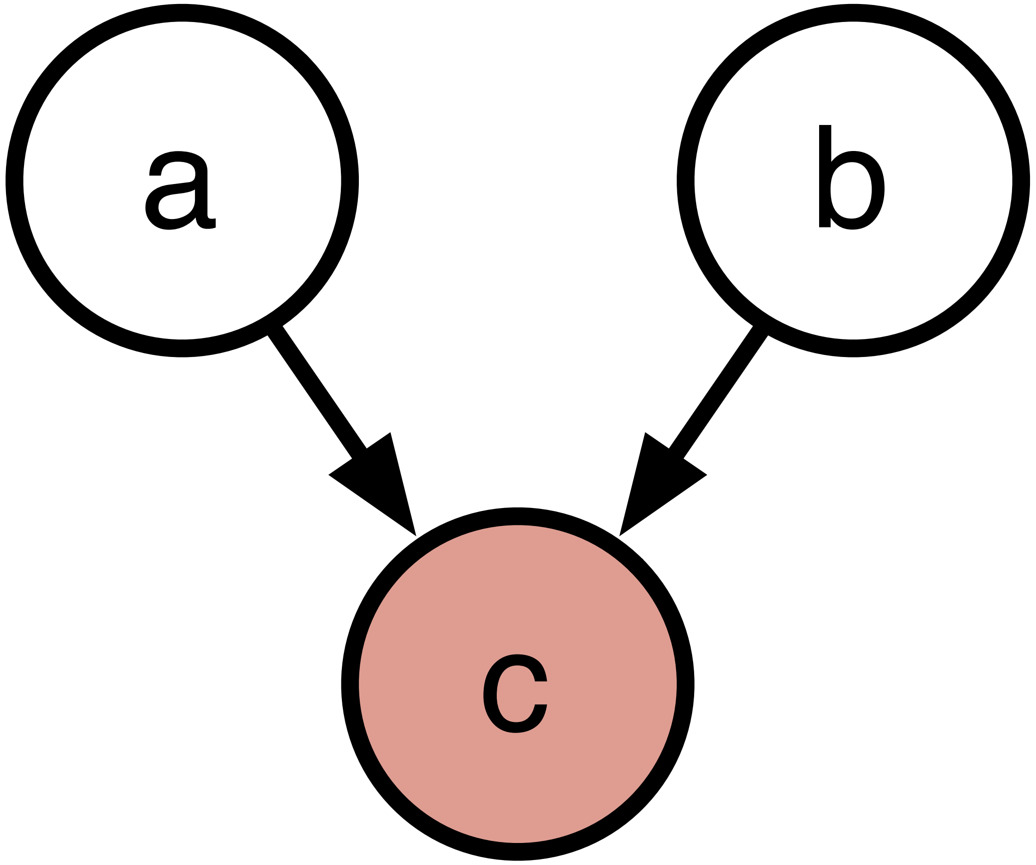
$$p(a,b,c) \quad = \quad p(a)p(b)p(c|a,b)$$

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



Head to head

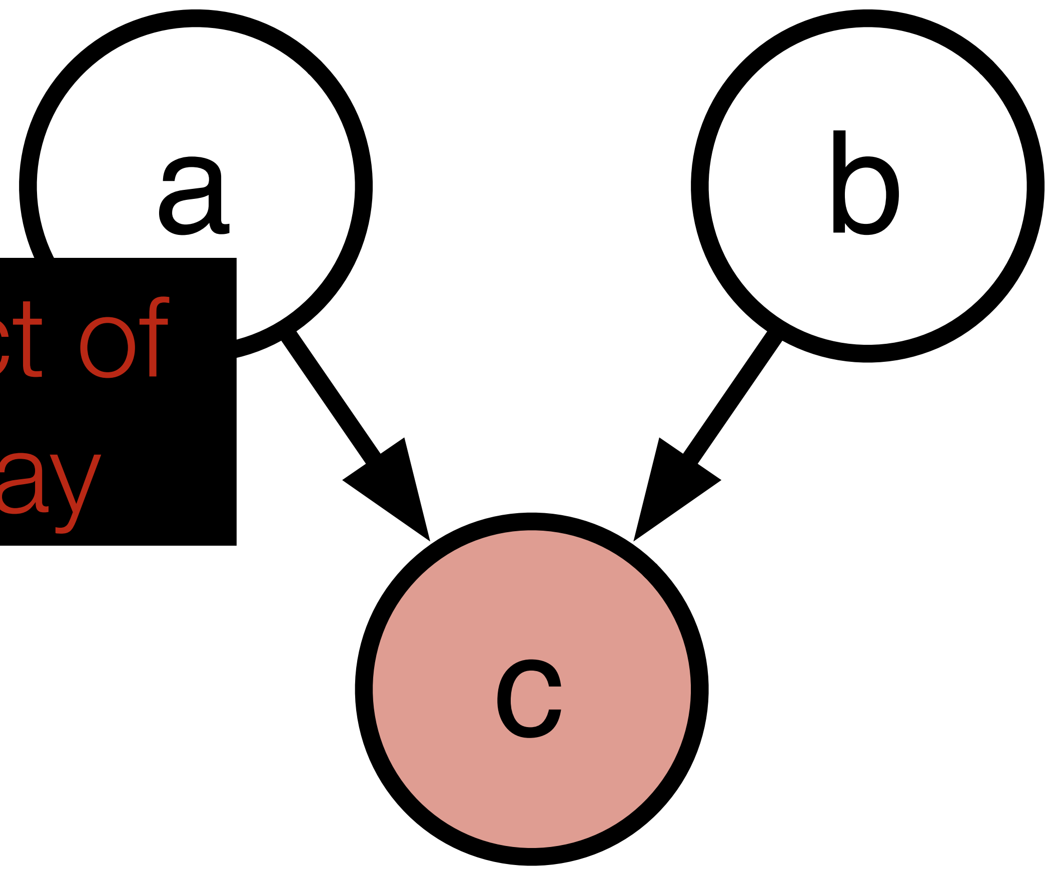
$$
\begin{aligned}
p(a,b,c) &= p(a)p(b)p(c|a,b) \\
p(a,b|c) &= \frac{p(a,b,c)}{p(c)} \\
&= \frac{p(a)p(b)p(c|a,b)}{p(c)} \\
&= p(a)p(b)\frac{p(c|a,b)}{p(c)}
\end{aligned}
$$



Head to head

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

This is the effect of Explaining away

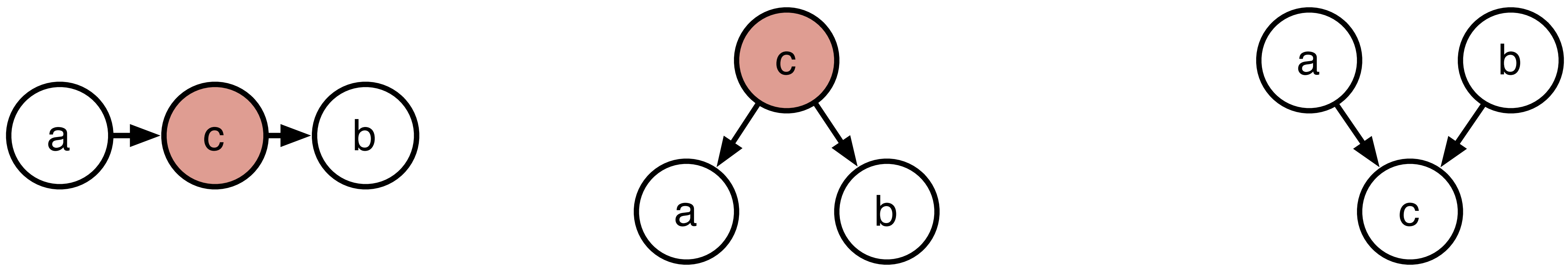


D-separation

The sets of random variables *A* and *B* are conditionally independent given *C* if every path between *A* and *B* is blocked.

a path is blocked when:

- the arrows meet tail to tail or head to tail and the node is in the set *C*
- the arrows meet head to head and neither the node, nor any of its descendants is in the set *C*

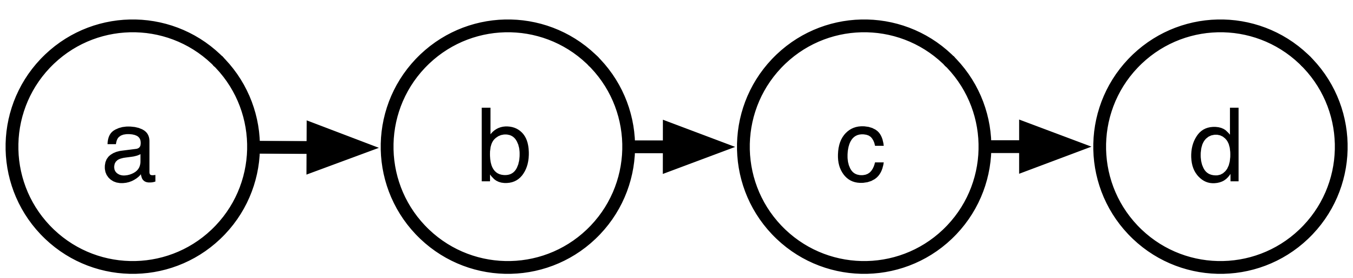


Directed graphical models:

Inference on a chain (marginal and most probable configuration)

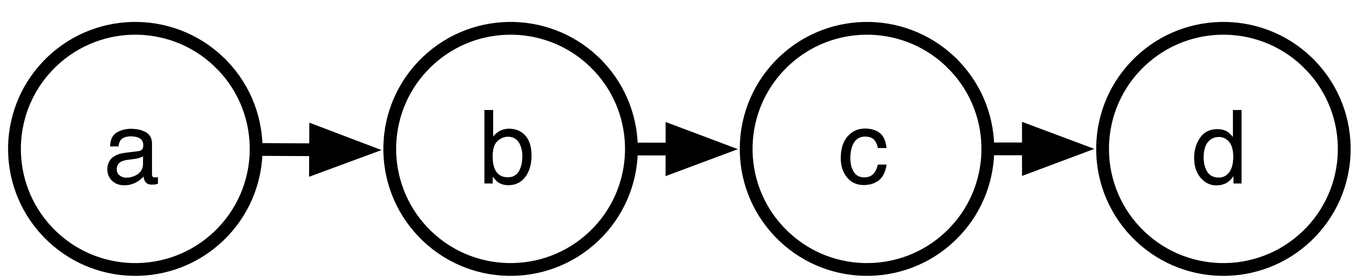
Marginal probability

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



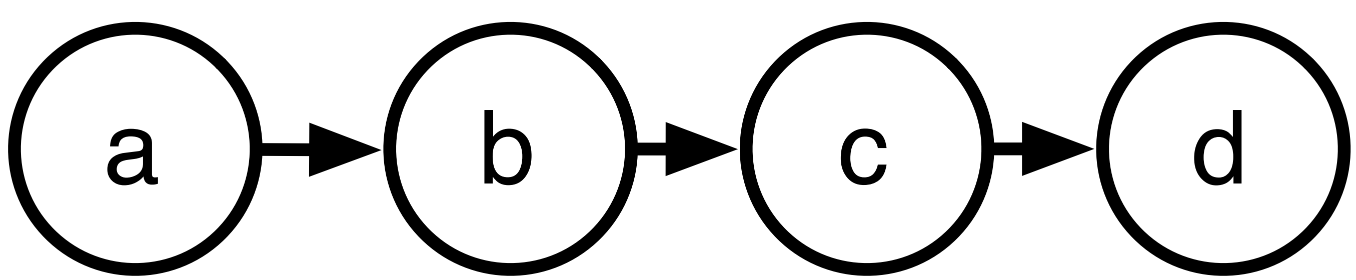
Marginal probability

$$
\begin{aligned}
p(a,b,c,d) &= p(a)p(b|a)p(c|b)p(d|c) \\
p(d) &= \sum_a \sum_b \sum_c p(a,b,c,d)
\end{aligned}
$$



Marginal probability

$$
\begin{aligned}
p(a,b,c,d) &= p(a)p(b|a)p(c|b)p(d|c) \\
p(d) &= \sum_c p(d|c) \sum_b p(c|b) \sum_a p(b|a)p(a) \\
&= \sum_c p(d|c) \sum_b p(c|b)p(b) \\
&= \sum_c p(d|c)p(c) \\
&= p(d)
\end{aligned}
$$



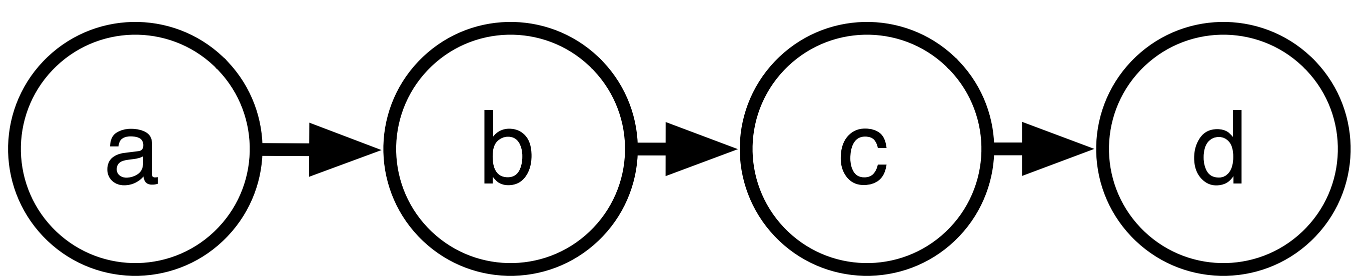
Marginal probability

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

$$p(d) = \sum p(d|c) \sum p(c|b) \sum p(b|a)p(a)$$

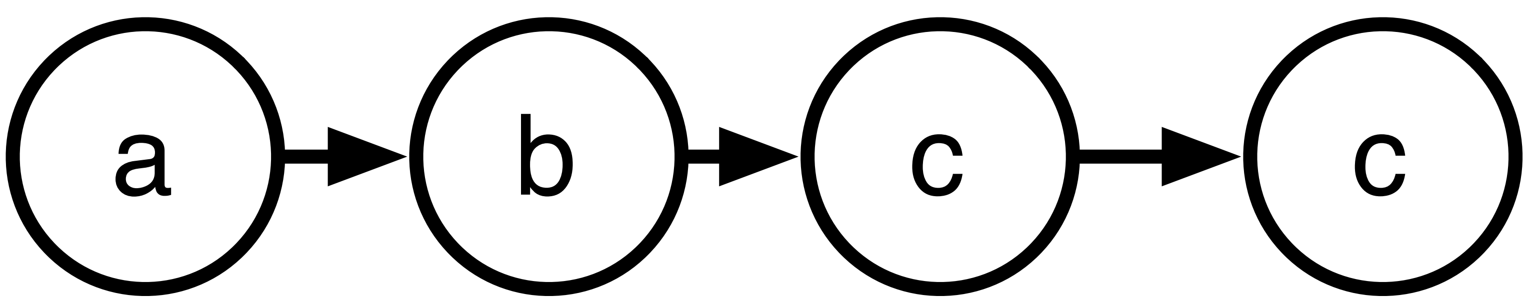
MATRIX NOTATION

$$= p(d)$$



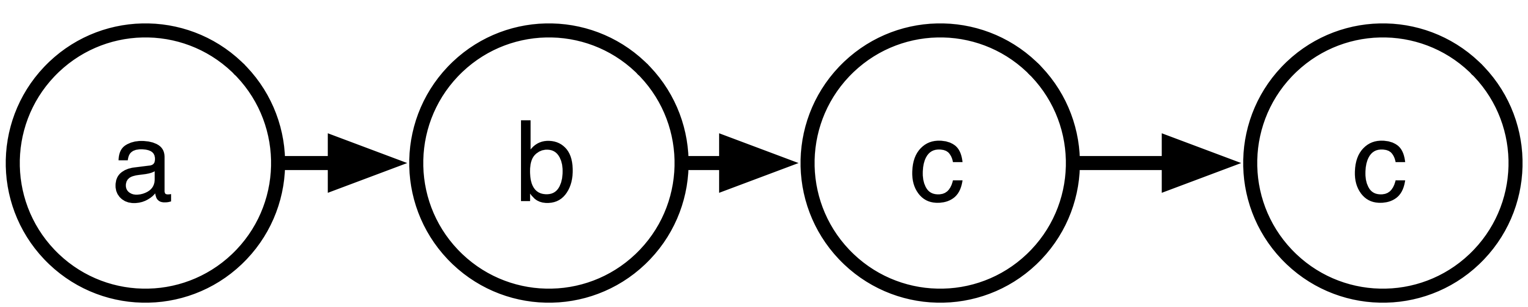
Most probable path

$$\arg\max_{a,b,c,d} p(a, b, c, d)$$



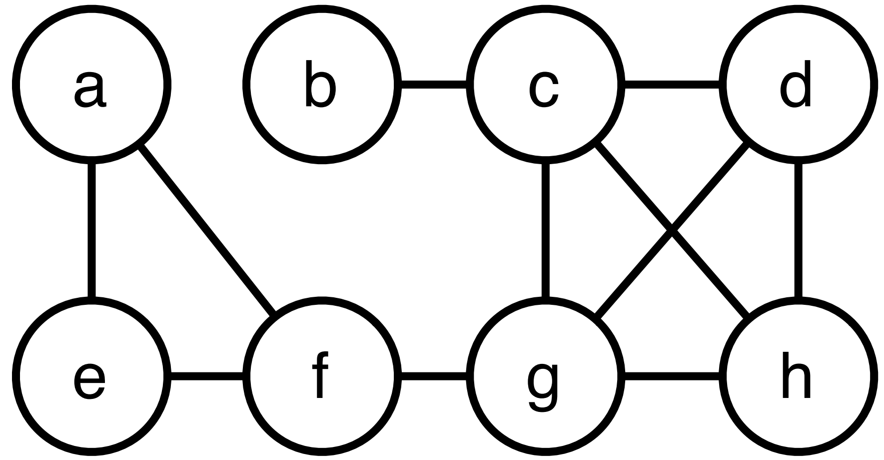
Most probable path

$$
\begin{aligned}
& \arg\max_{a,b,c,d} p(a,b,c,d) \\
= \quad & \arg\max_{a,b,c,d} p(d|c)p(c|b)p(b|a)p(a) \\
= \quad & \arg\left(\max_d \max_c \left[p(d|c) \max_b \left[p(c|b) \max_a p(b|a)p(a)\right]\right]\right)
\end{aligned}
$$



Directed graphical models

- Define a factorisation of the joint distribution in terms of conditional distributions
- Are a tool to convey information about a probabilistic model
- Enable us to quickly determine whether variables are conditionally independent
- Allow for efficient inference when the graphical model is a directed acyclic graph *(we only demonstrated this for a chain, this will be generalised later)*



Undirected graphical models

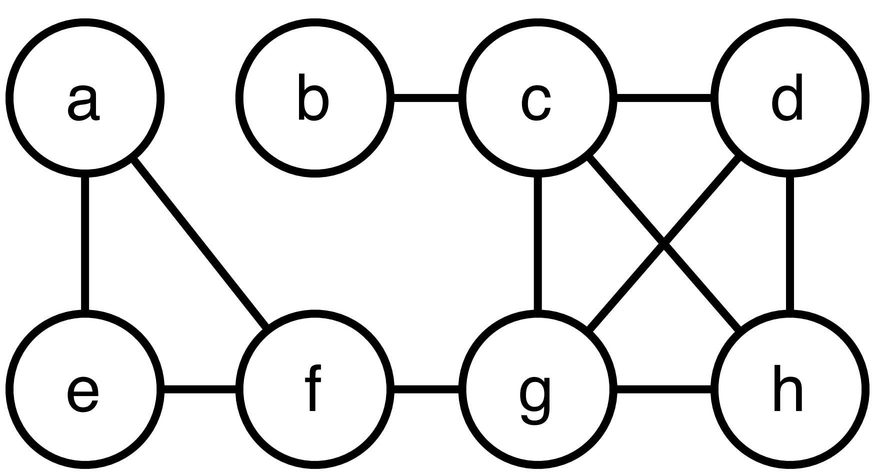
Also known as Markov Random Fields or Markov Networks

Undirected graphical models

- X: set containing all random variables
- X_i : subset of the random variables in X
- x_i : a specific random variable

Undirected models: cliques and maximal cliques

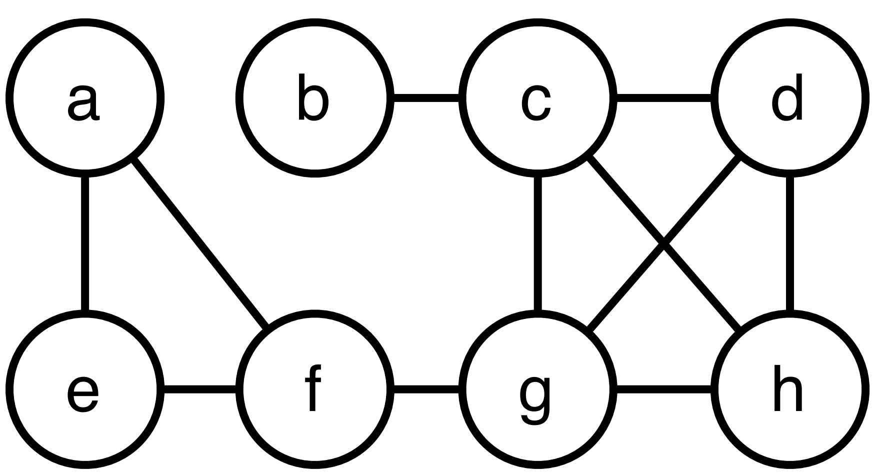
- A clique is a set of vertices that where all pairs of vertices are connected
- A maximal clique is a clique where adding any extra node would prevent it from being a clique.

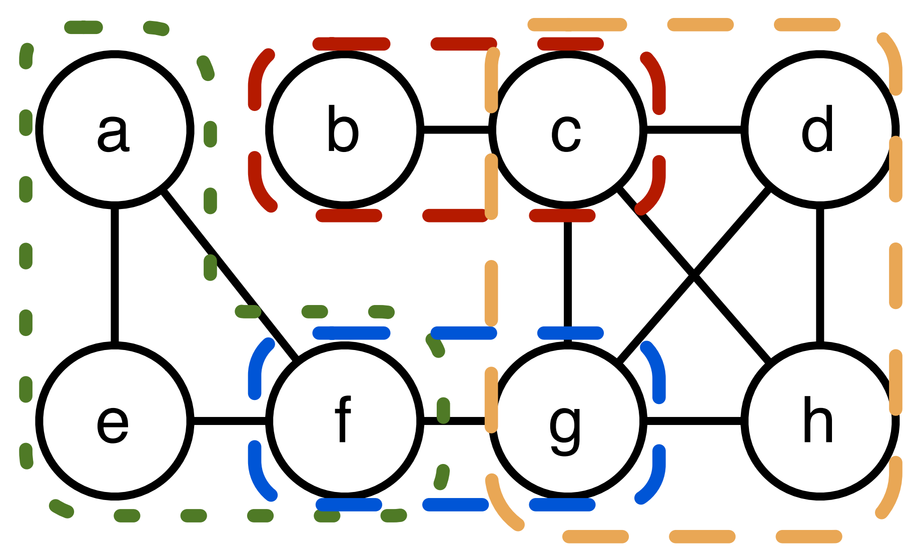


Undirected models: cliques and maximal cliques

- A clique is a set of vertices that where all pairs of vertices are connected

- A maximal clique is a clique where adding any extra node would prevent it from being a clique.



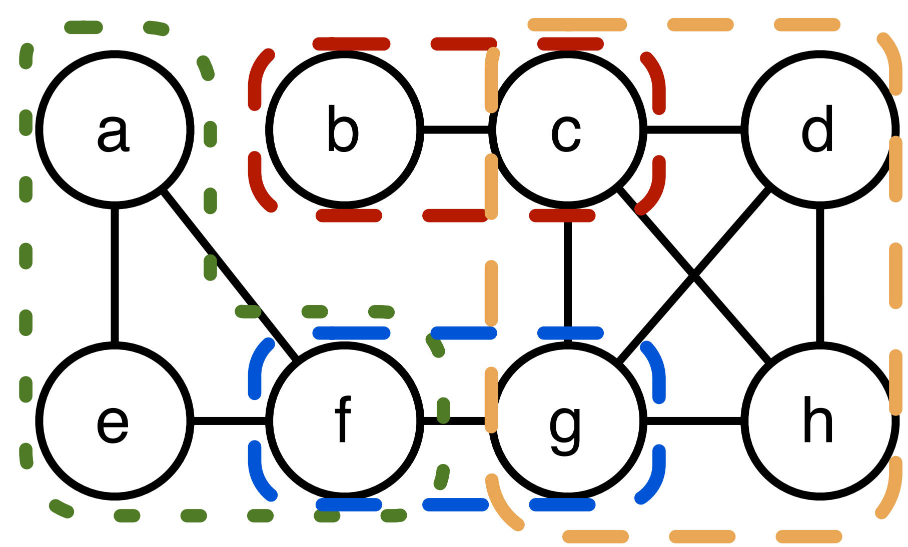


Undirected models: factorisation

- Define a factorisation based on the potential functions of the maximal cliques

- Potential functions are often not properly normalised probability functions (e.g. energy functions $f_i(X_i) = exp(-E(X_i))$)

$$p(X) = \frac{1}{Z}\prod_{i=1}^{C} f_i(X_i)$$

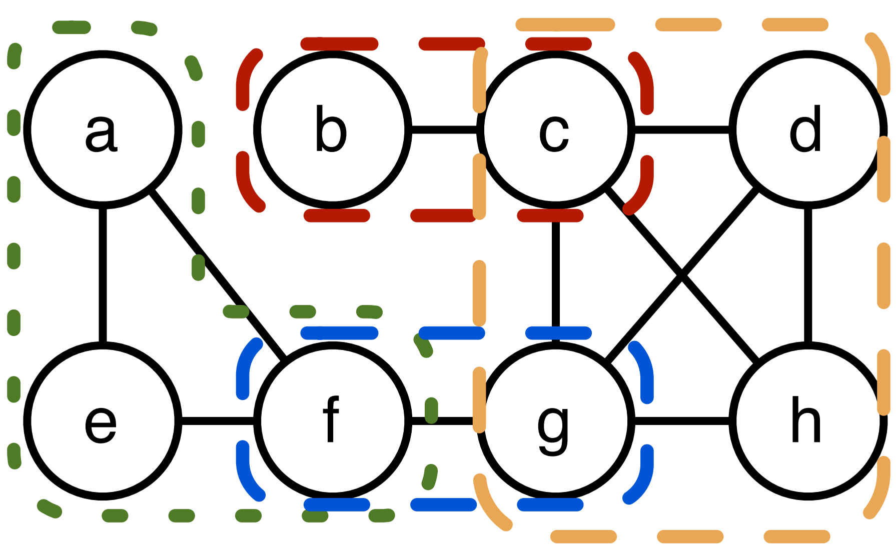


Undirected models: factorisation

- Define a factorisation based on the potential functions of the maximal cliques

- Potential functions are often not properly normalised probability functions (e.g. energy functions $f_i(X_i) = exp(-E(X_i))$)

$$
\begin{aligned}
p(X) &= \frac{1}{Z}\prod_{i=1}^{C} f_i(X_i) \\
X_1 &= \{a, e, f\} \\
X_2 &= \{b, c\} \\
X_3 &= \{f, g\} \\
X_4 &= \{c, d, g, h\}
\end{aligned}
$$

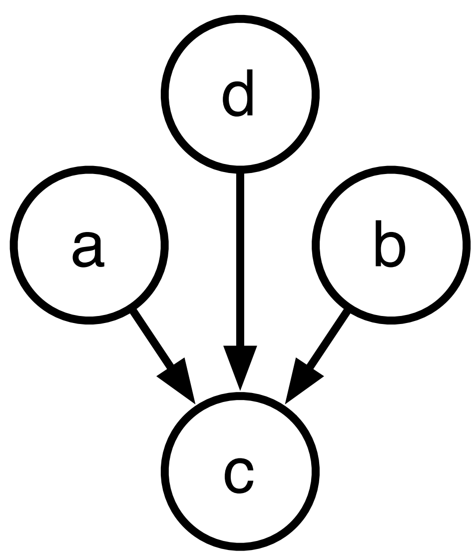


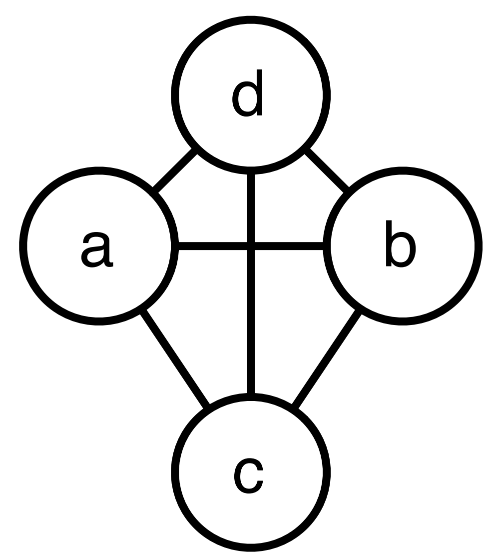
Undirected models: converting a directed one

Moralisation: converting a directed graph to an undirected one

- For a node x_i, add links between all parents of x_i
  (reason to add this link is the explaining away phenomenon)

- Remove all the arrows

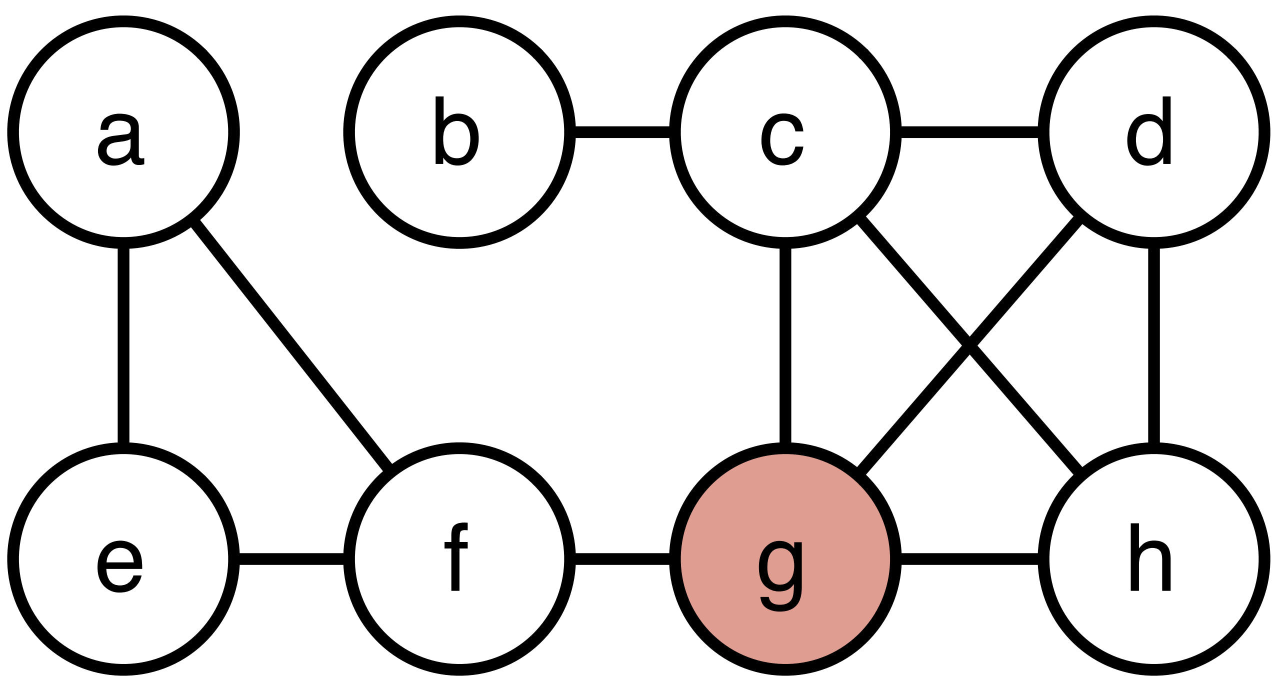




Undirected model: conditional independence

Two sets of nodes A, B, are conditionally independent give are blocked

- when all paths between them are blocked
    - A path is blocked when it contains an observed variable



Factor graphs

An additional generalisation to make the factorisation more explicit
Will be used in the programming exercise

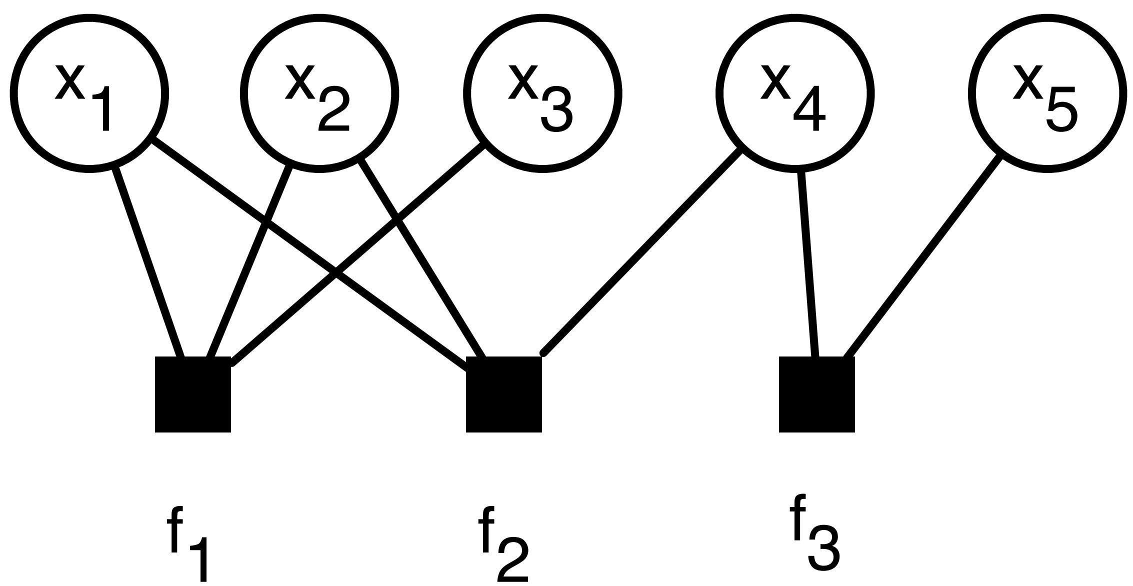
Factor graphs

Given a factorisation: $$p(X) = \frac{1}{Z}\prod_{i=1}^{C} f_i(X_i)$$

- Each variable x_j has an associated variable node
- Each factor $f_i(X_i)$ has an associated factor node and is connected to the variables used in the factor: $x_j \in X_i$
- Transforming a directed or undirected model to a factor graph is a straightforward application of these rules

Factor graphs

$$p(X) = \frac{1}{Z} f_1(x_1, x_2, x_3) f_2(x_1, x_2, x_4) f_3(x_4, x_5)$$



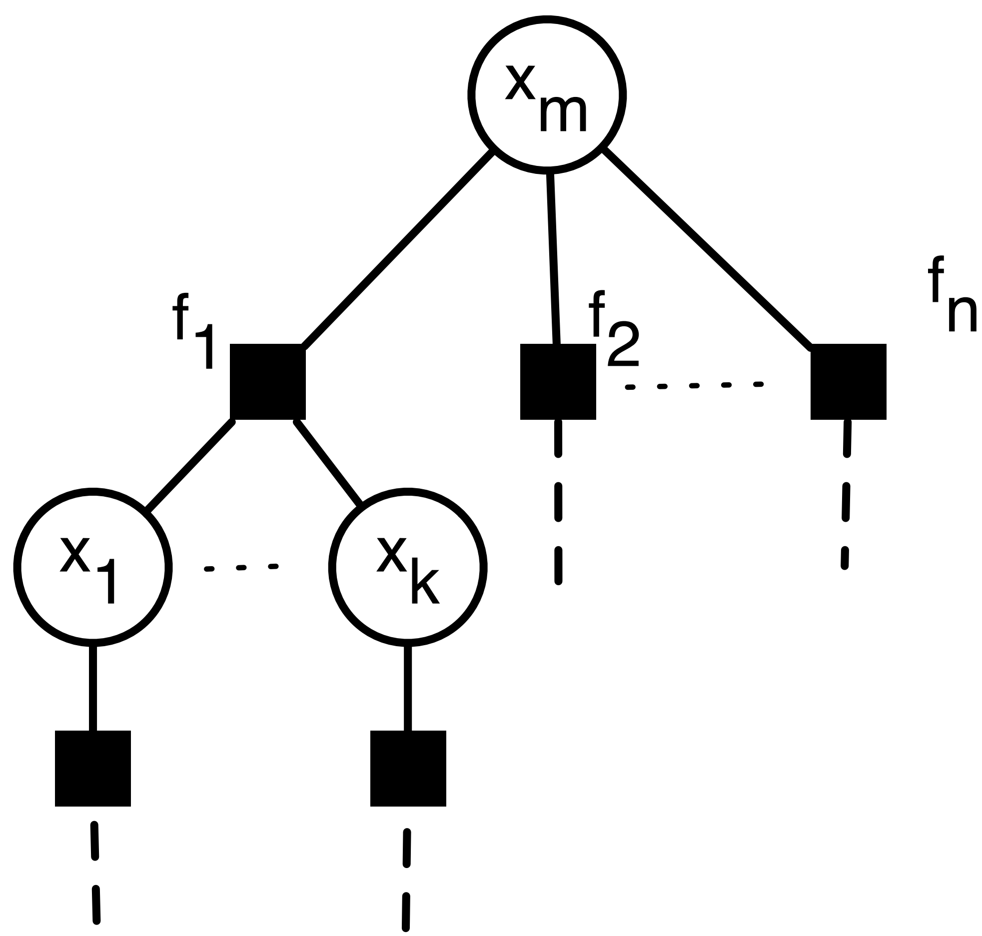
Inference in a factor graph tree: the sum-product algorithm

Conceptually quite easy, but the recursion will make the notation a bit cumbersome.

(I have not yet found a way to avoid the notational mess)

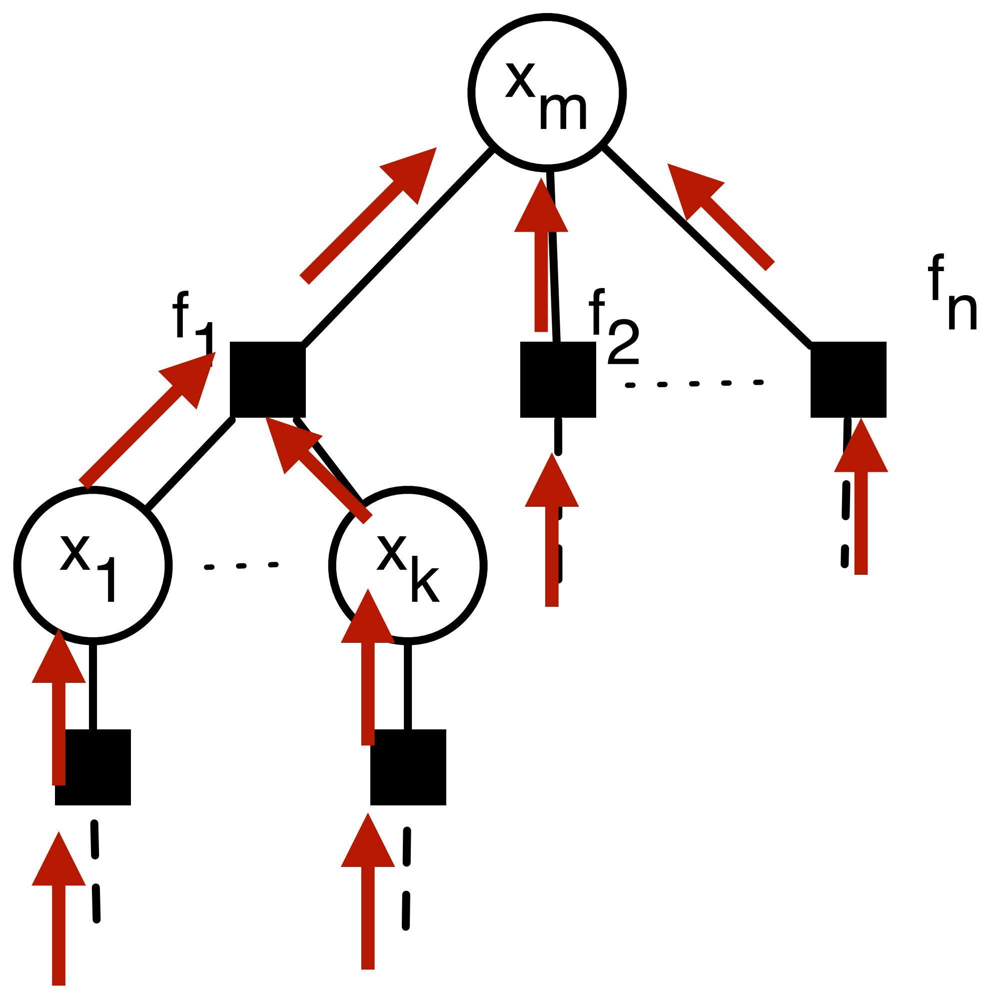
Sum-product algorithm: inference in a tree

$$p(x_m) = ?$$



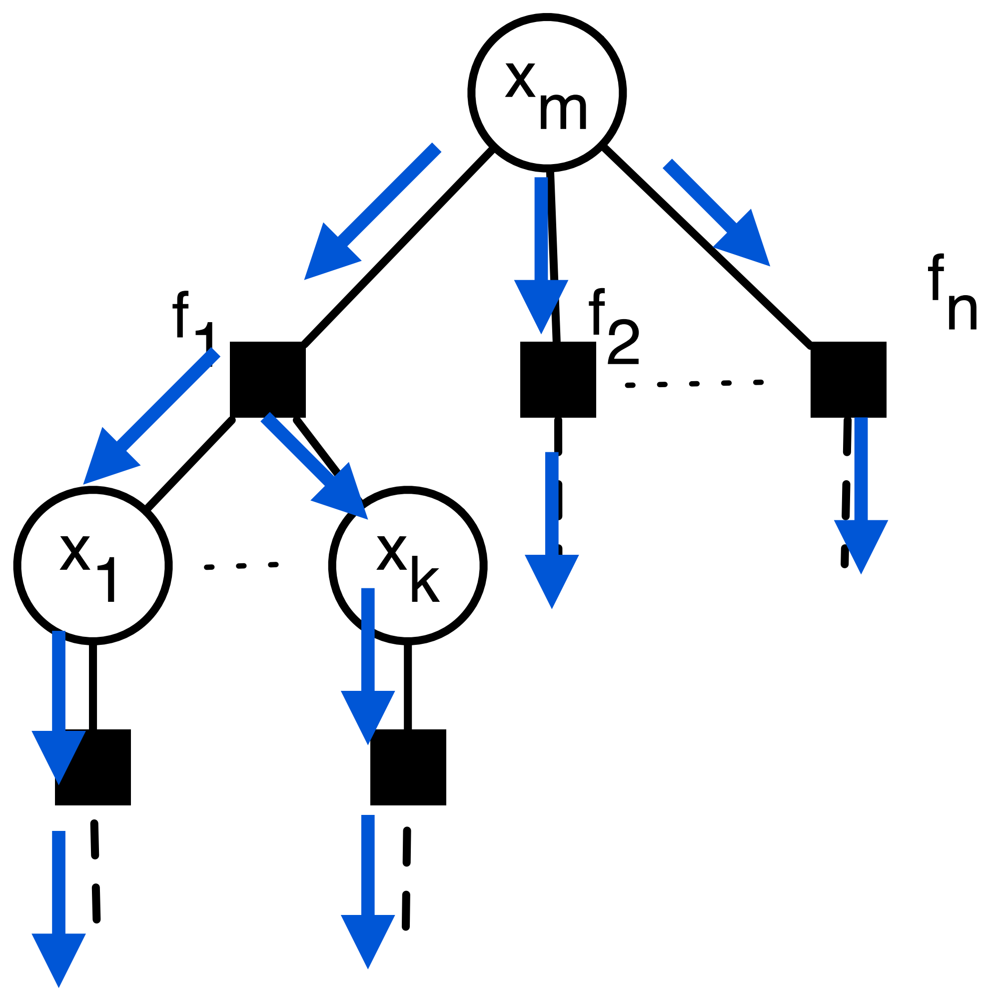
Sum-product algorithm: inference in a tree

You can compute the marginal for every node by passing twice through the tree



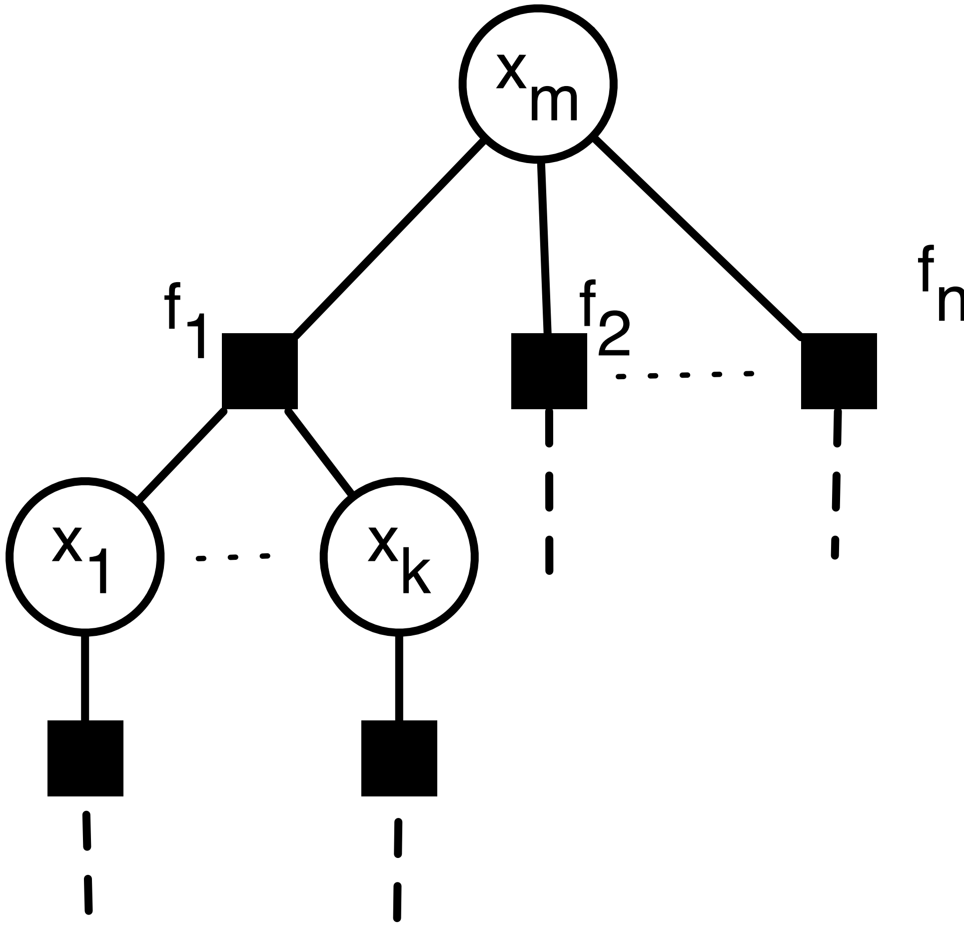
Sum-product algorithm: inference in a tree

You can compute the marginal for every node by passing twice through the tree



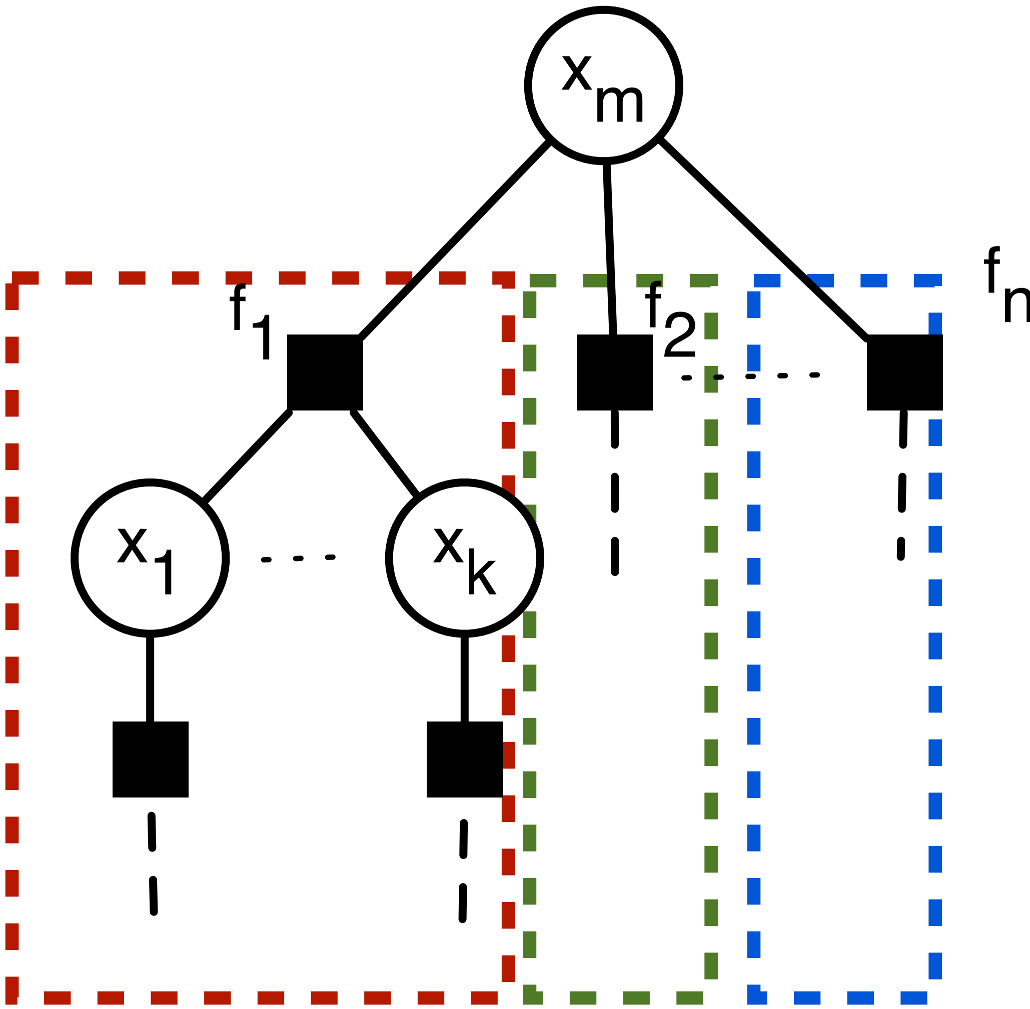
Sum-product algorithm: inference in a tree

$$p(x_m) = \frac{1}{Z} \sum_{X \setminus x_m} \prod_{c=1}^{C} f_c(X_c)$$



Sum-product algorithm: inference in a tree

$$p(x_m) = \frac{1}{Z} \sum_{X \setminus x_m} \prod_{c=1}^{C} f_c(X_c)$$



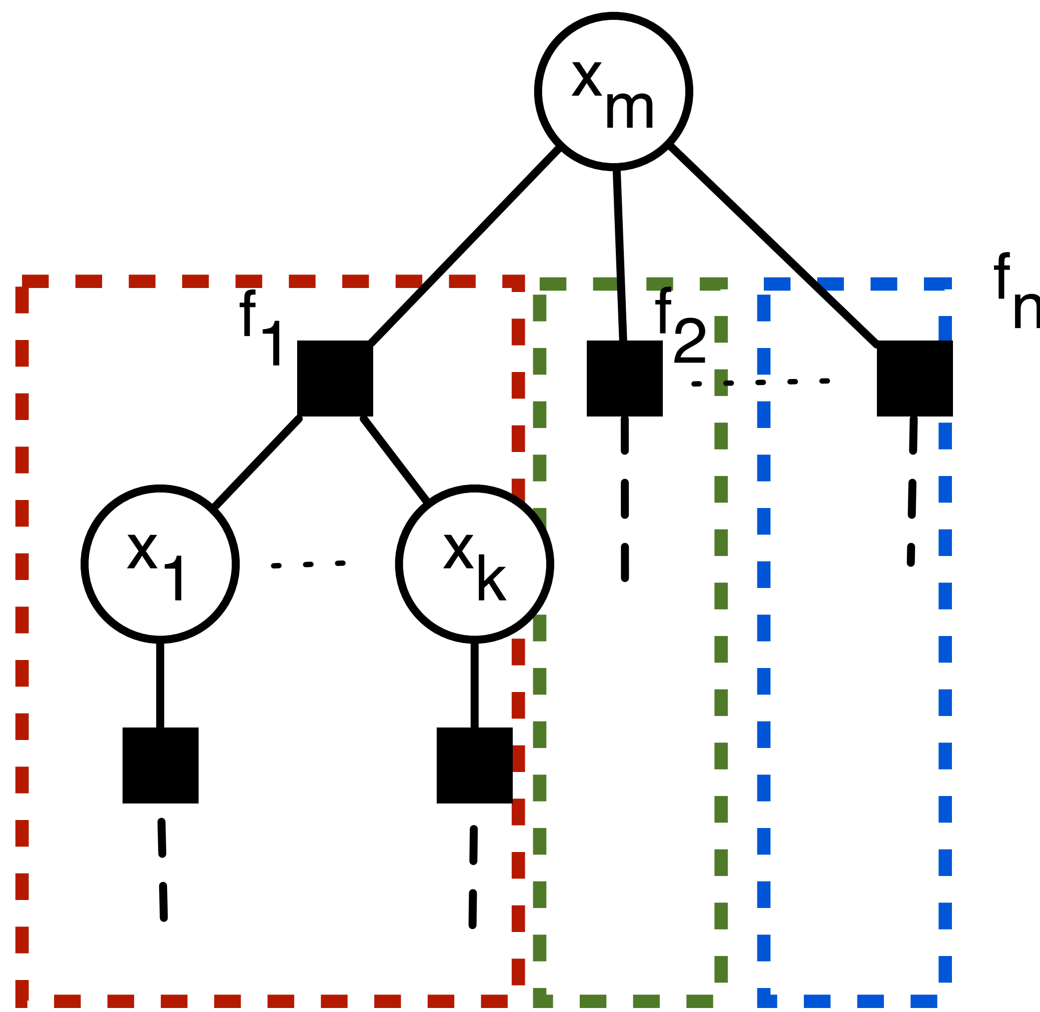
Sum-product algorithm: inference in a tree

Each subtree contains a non-overlapping subset of variables.

Let X_{t_i} be the subset of variables in a subtree.

Let $F_i(x_m, X_{t_i})$ be the factor defined by the subtree

$$p(x_m) \quad = \quad \frac{1}{Z} \sum_{X \setminus x_m} \prod_{c=1}^{C} f_c(X_c)$$



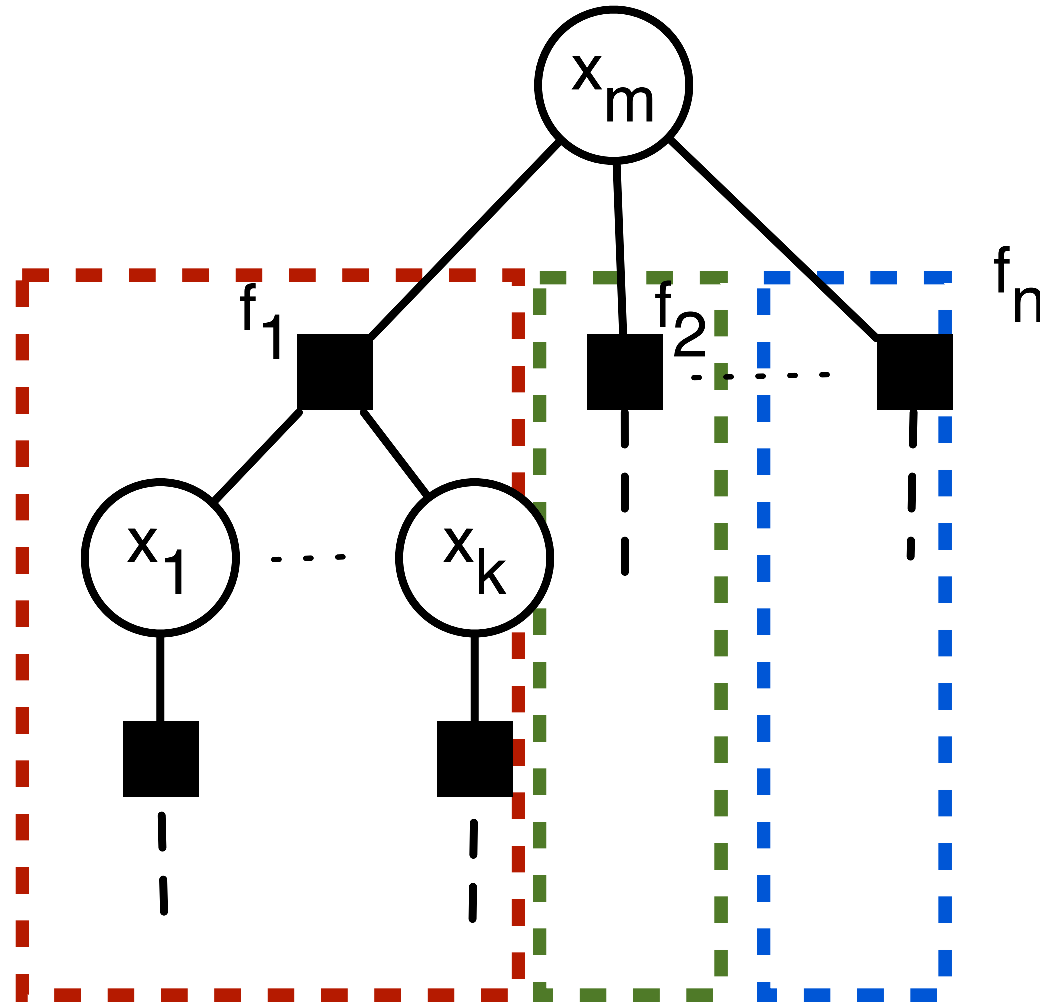
Sum-product algorithm: inference in a tree

Each subtree contains a non-overlapping subset of variables.

Let X_{t_i} be the subset of variables in a subtree.

Let $F_i(x_m, X_{t_i})$ be the factor defined by the subtree

$$
\begin{aligned}
p(x_m) &= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{c=1}^{C} f_c(X_c) \\
&= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{i=1}^{n} F(x_m, X_{t_i})
\end{aligned}
$$



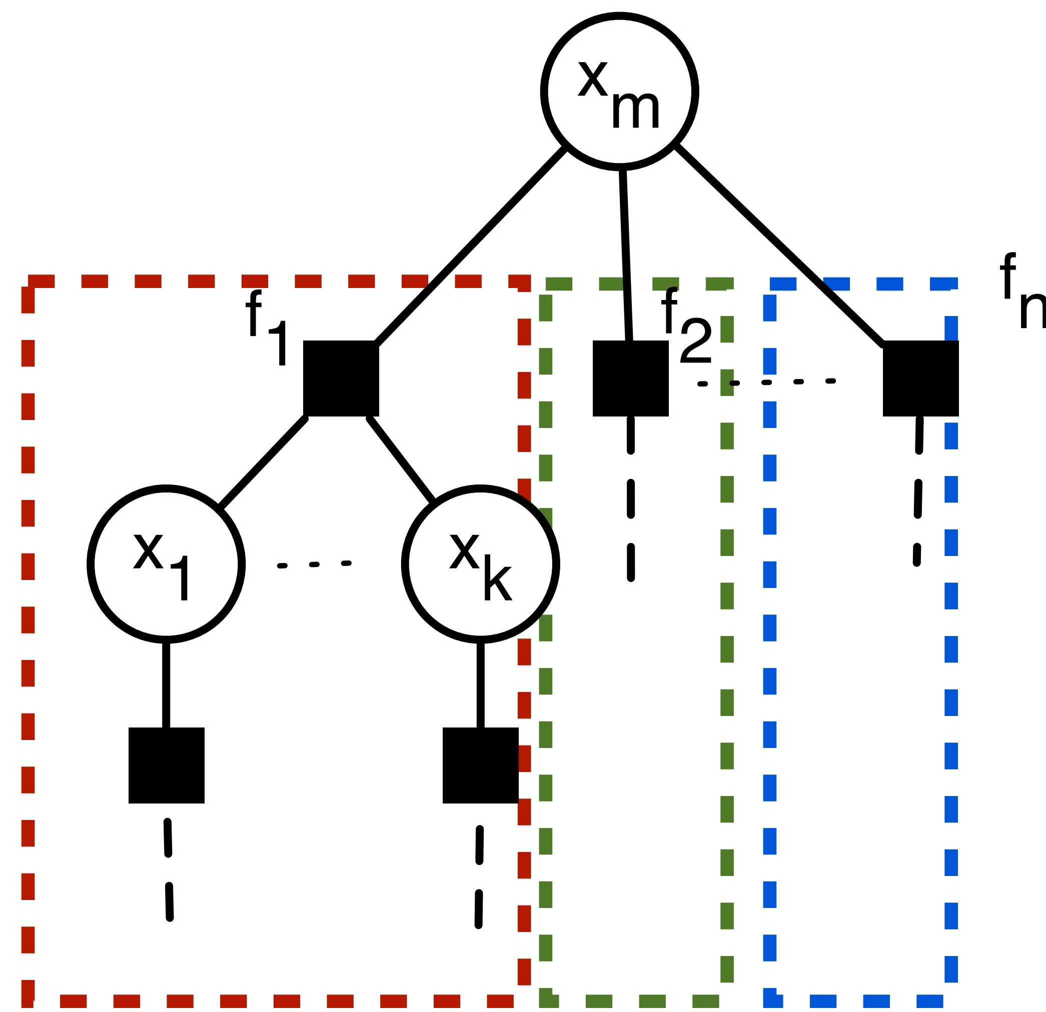
Sum-product algorithm: inference in a tree

Each subtree contains a non-overlapping subset of variables.

Let X_{t_i} be the subset of variables in a subtree.

Let $F_i(x_m, X_{t_i})$ be the factor defined by the subtree

$$
\begin{aligned}
p(x_m) &= \frac{1}{Z}\sum_{X\backslash x_m}\prod_{c=1}^{C} f_c(X_c) \\
&= \frac{1}{Z}\sum_{X\backslash x_m}\prod_{i=1}^{n} F(x_m, X_{t_i})
\end{aligned}
$$



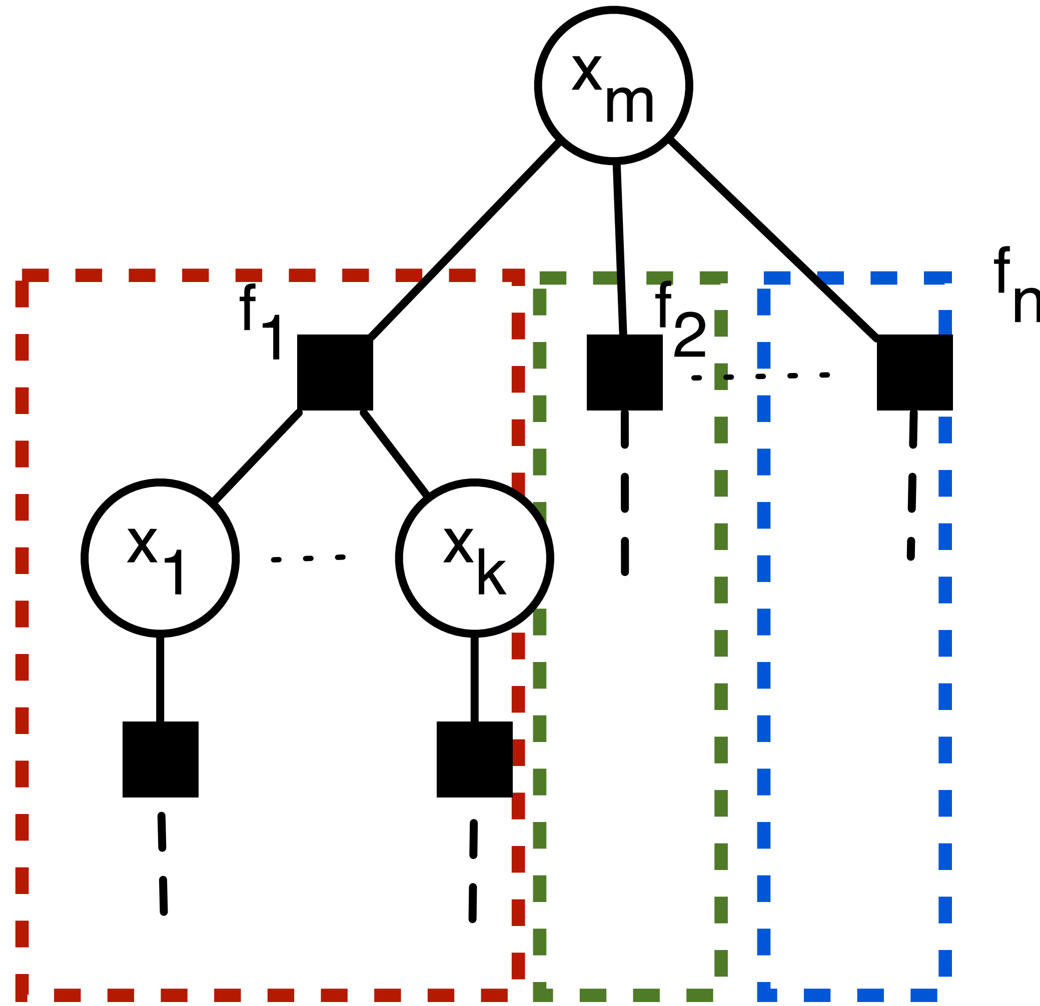
Sum-product algorithm: inference in a tree

Each subtree contains a non-overlapping subset of variables.

Let X_{t_i} be the subset of variables in a subtree.

Let $F_i(x_m, X_{t_i})$ be the factor defined by the subtree

$$
\begin{aligned}
p(x_m) &= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{c=1}^{C} f_c(X_c) \\
&= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{i=1}^{n} F(x_m, X_{t_i}) \\
&= \frac{1}{Z} \prod_{i=1}^{n} \sum_{X_{t_i}} F(x_m, X_{t_i})
\end{aligned}
$$



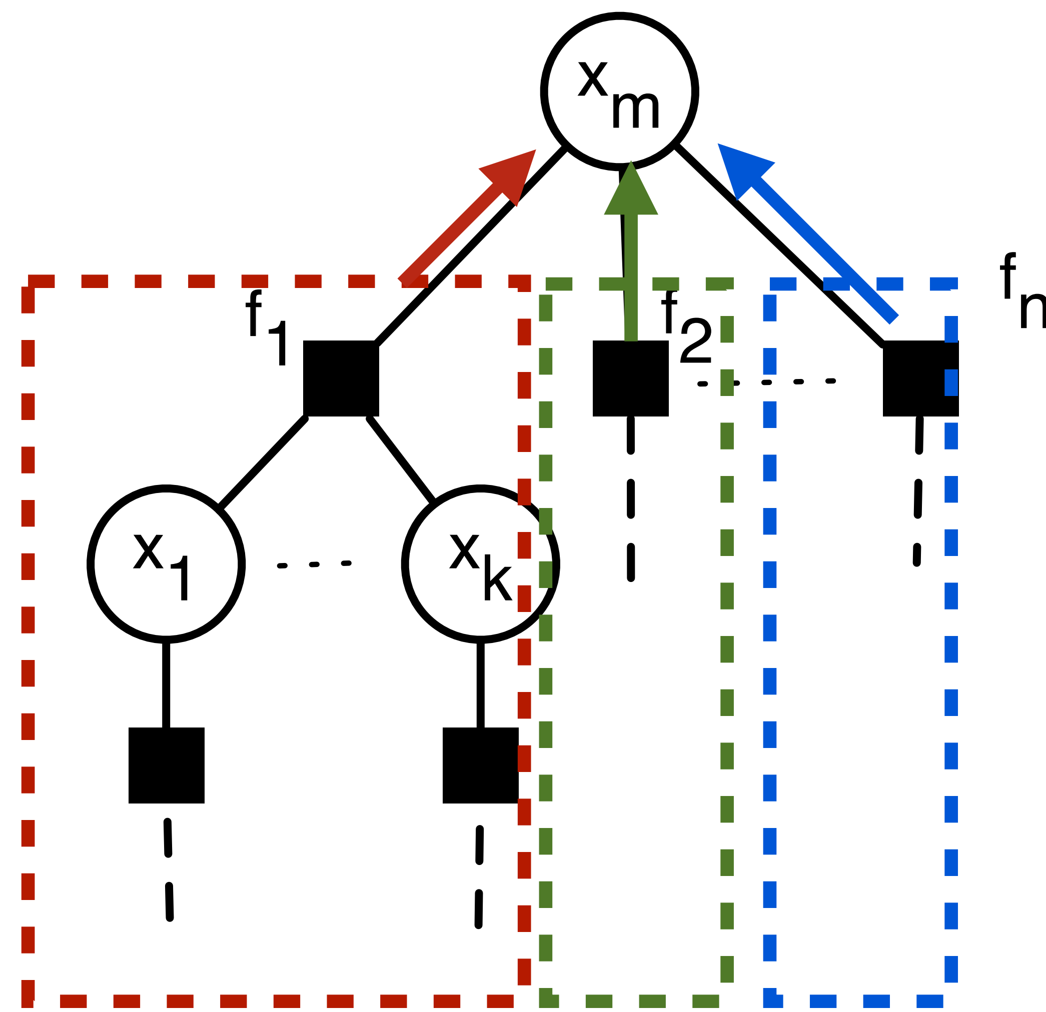
Sum-product algorithm: inference in a tree

Each subtree contains a non-overlapping subset of variables.

Let X_{t_i} be the subset of variables in a subtree.

Let $F_i(x_m, X_{t_i})$ be the factor defined by the subtree

$$
\begin{aligned}
p(x_m) &= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{c=1}^{C} f_c(X_c) \\
&= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{i=1}^{n} F(x_m, X_{t_i}) \\
&= \frac{1}{Z} \prod_{i=1}^{n} \sum_{X_{t_i}} F(x_m, X_{t_i}) \\
&= \frac{1}{Z} \prod_{i=1}^{n} \mu_{f_i \to x_m}(x_m)
\end{aligned}
$$



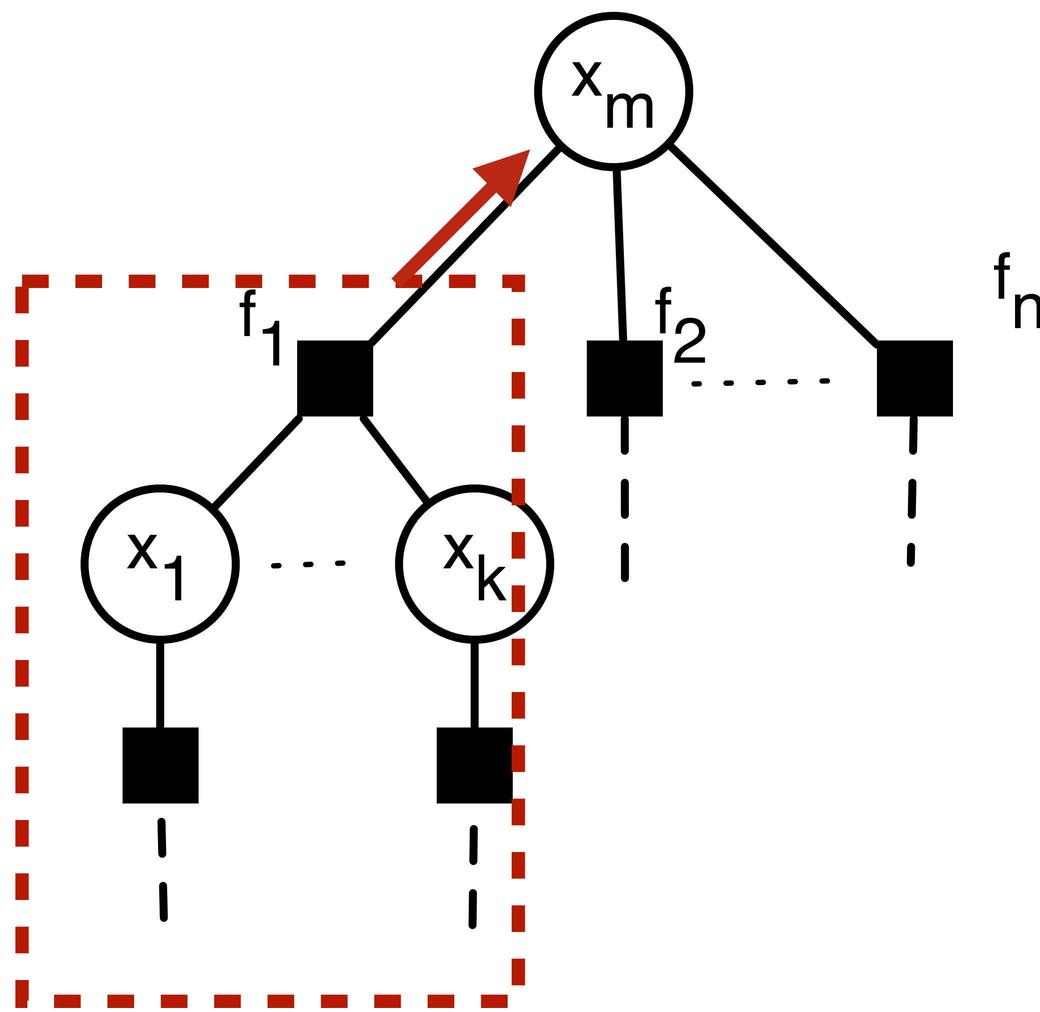
Sum-product algorithm: inference in a tree

Each subtree contains a non-overlapping subset of variables.

Let X_{t_i} be the subset of variables in a subtree.

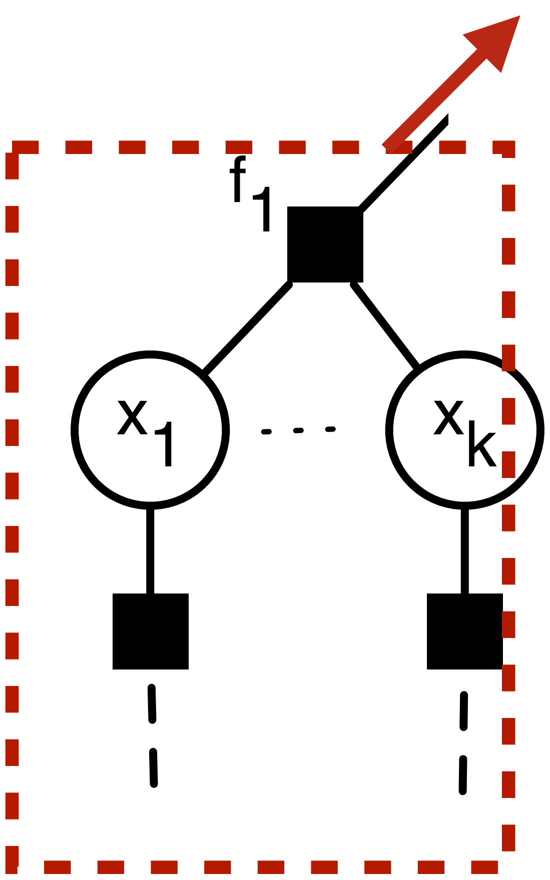
Let $F_i(x_m, X_{t_i})$ be the factor defined by the subtree

$$
\begin{aligned}
p(x_m) &= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{c=1}^{C} f_c(X_c) \\
&= \frac{1}{Z} \sum_{X \backslash x_m} \prod_{i=1}^{n} F(x_m, X_{t_i}) \\
&= \frac{1}{Z} \prod_{i=1}^{n} \sum_{X_{t_i}} F(x_m, X_{t_i}) \\
&= \frac{1}{Z} \prod_{i=1}^{n} \mu_{f_i \rightarrow x_m}(x_m)
\end{aligned}
$$



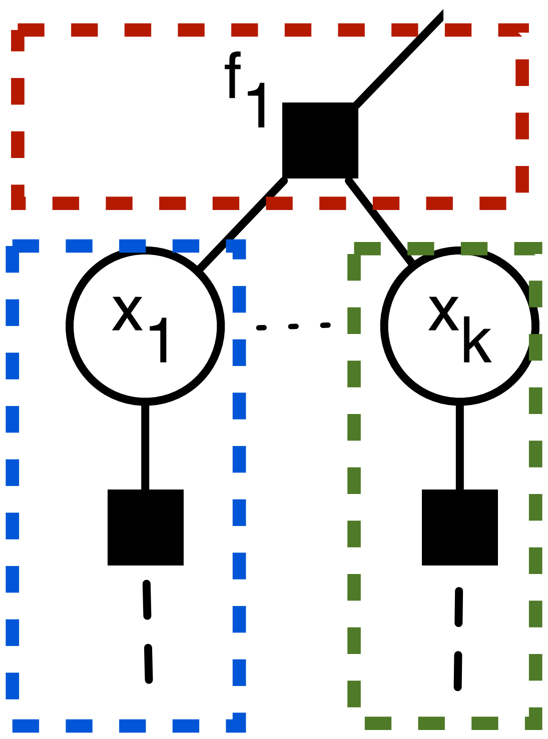
Sum-product algorithm: inference in a tree

$$\mu_{f_i \to x_m}(x_m) \quad = \quad \sum_{X_{t_i}} F(x_m, X_{t_i})$$



Sum-product algorithm: inference in a tree

$$\mu_{f_i \to x_m}(x_m) \quad = \quad \sum_{X_{t_i}} F(x_m, X_{t_i})$$

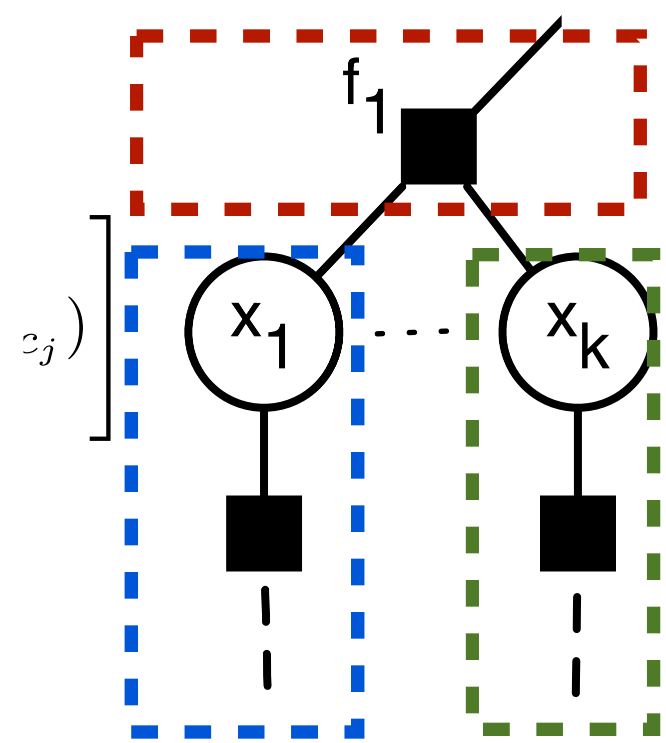


Sum-product algorithm: inference in a tree

X_i are the variables connected to factor f_i

Let $X_{c_j} \subset X_{t_i} \backslash X_i$ be the remaining variables in a **new** subtree

$$\mu_{f_i \to x_m}(x_m) \quad = \quad \sum_{X_{t_i}} F(x_m, X_{t_i})$$

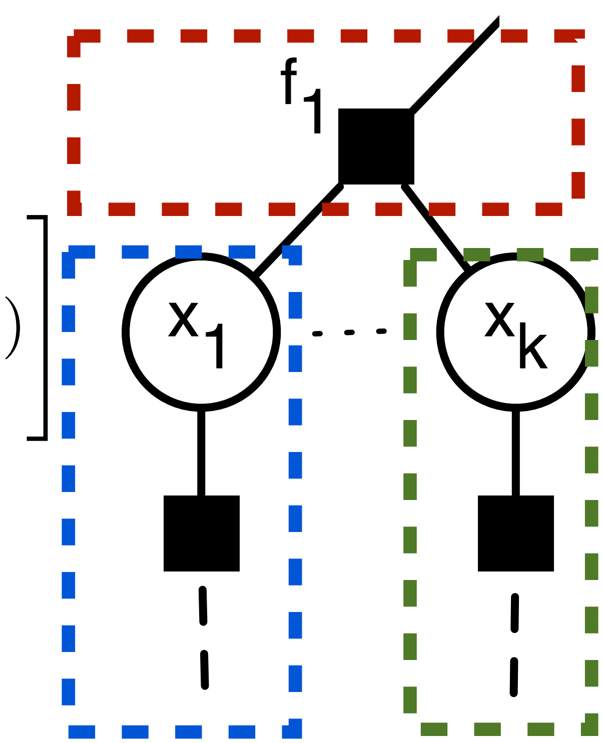


Sum-product algorithm: inference in a tree

X_i are the variables connected to factor f_i

Let $X_{c_j} \subset X_{t_i} \backslash X_i$ be the remaining variables in a **new** subtree

$$\begin{aligned}
\mu_{f_i \to x_m}(x_m) &= \sum_{X_{t_i}} F(x_m, X_{t_i}) \\
&= \sum_{X_{t_i}} f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} G_j(x_j, X_{c_j})
\end{aligned}$$

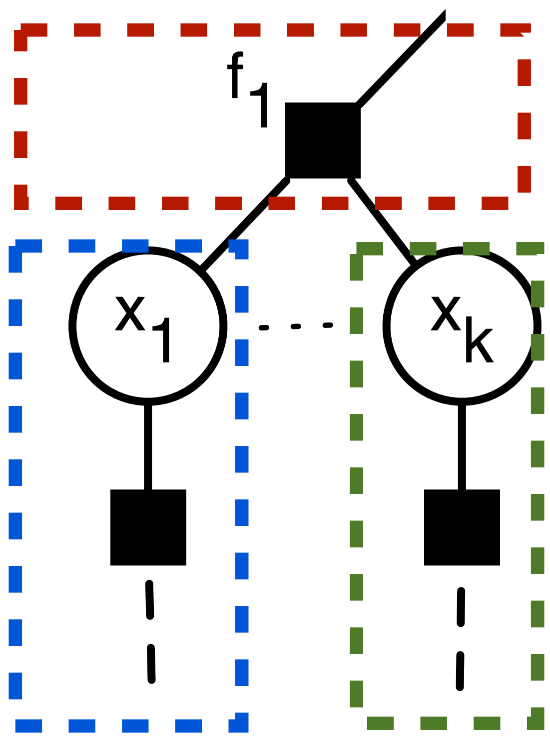


Sum-product algorithm: inference in a tree

X_i are the variables connected to factor f_i

Let $X_{c_j} \subset X_{t_i} \backslash X_i$ be the remaining variables in a **new** subtree

$$\begin{aligned}
\mu_{f_i \to x_m}(x_m) &= \sum_{X_{t_i}} F(x_m, X_{t_i}) \\
&= \sum_{X_{t_i}} f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} G_j(x_j, X_{c_j}) \\
&= \sum_{X_i \backslash x_m} \left[f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} \sum_{X_{c_j}} G_j(x_j, X_{c_j}) \right]
\end{aligned}$$

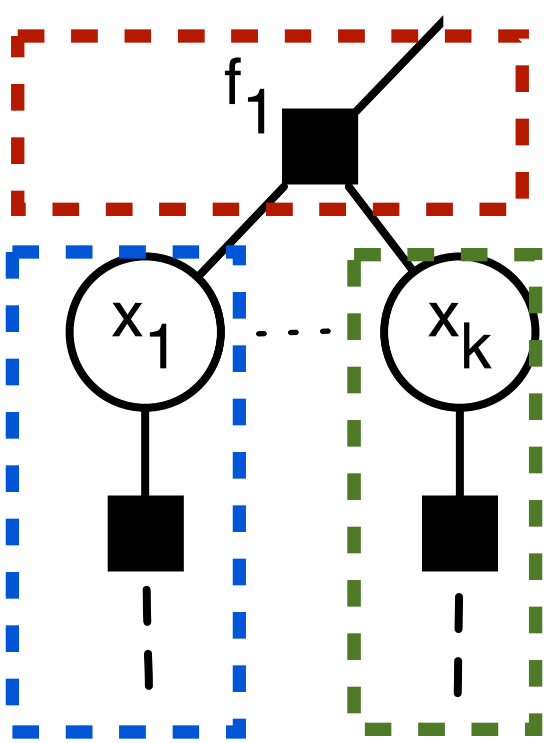


Sum-product algorithm: inference in a tree

X_i are the variables connected to factor f_i

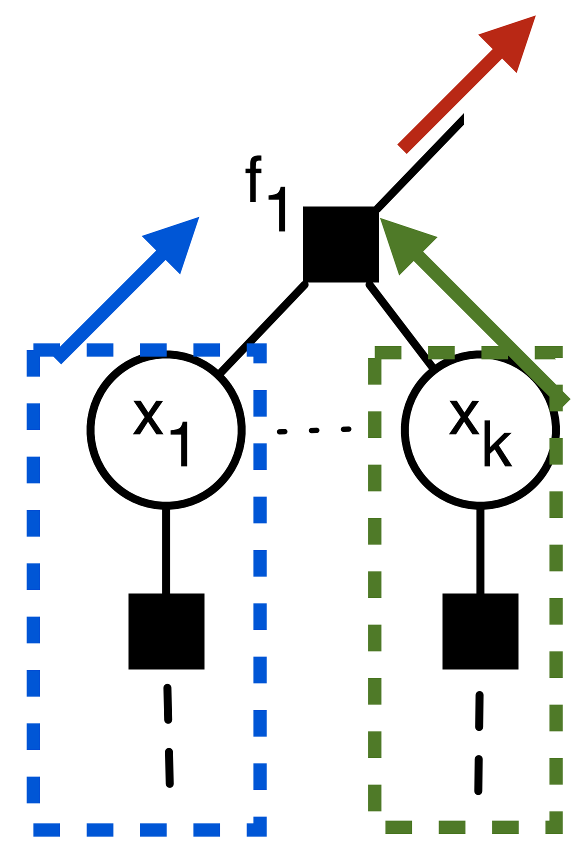
Let $X_{c_j} \subset X_{t_i} \backslash X_i$ be the remaining variables in a **new** subtree

$$
\begin{aligned}
\mu_{f_i \to x_m}(x_m) &= \sum_{X_{t_i}} F(x_m, X_{t_i}) \\
&= \sum_{X_{t_i}} f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} G_j(x_j, X_{c_j}) \\
&= \sum_{X_i \backslash x_m} \left[f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} \sum_{X_{c_j}} G_j(x_j, X_{c_j}) \right] \\
&= \sum_{X_i \backslash x_m} \left[f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} \mu_{x_j \to f_j}(x_m) \right]
\end{aligned}
$$

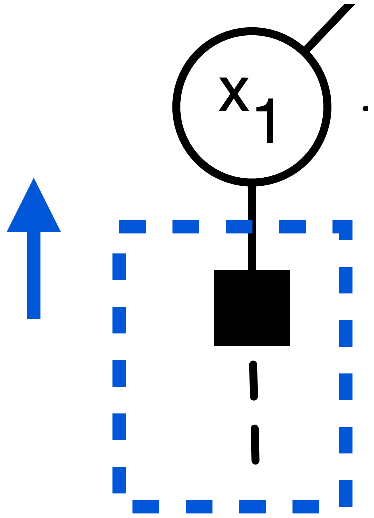


Sum-product algorithm: inference in a tree

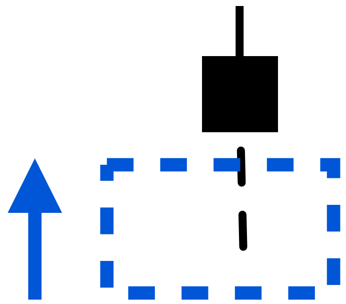
$$\mu_{f_i \to x_m}(x_m) \quad = \quad \sum_{X_i \backslash x_m} \left[f_i(x_m, X_i \backslash x_m) \prod_{j=1}^{k} \mu_{x_j \to f_j}(x_m) \right]$$



Sum-product algorithm: inference in a tree

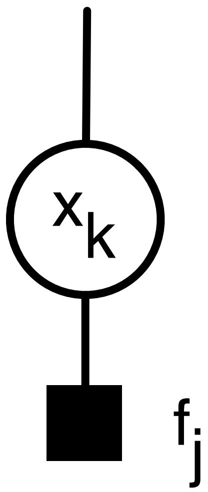


Sum-product algorithm: inference in a tree

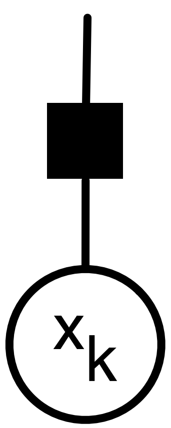


Sum-product algorithm: inference in a tree

Sum-product algorithm: inference in a tree



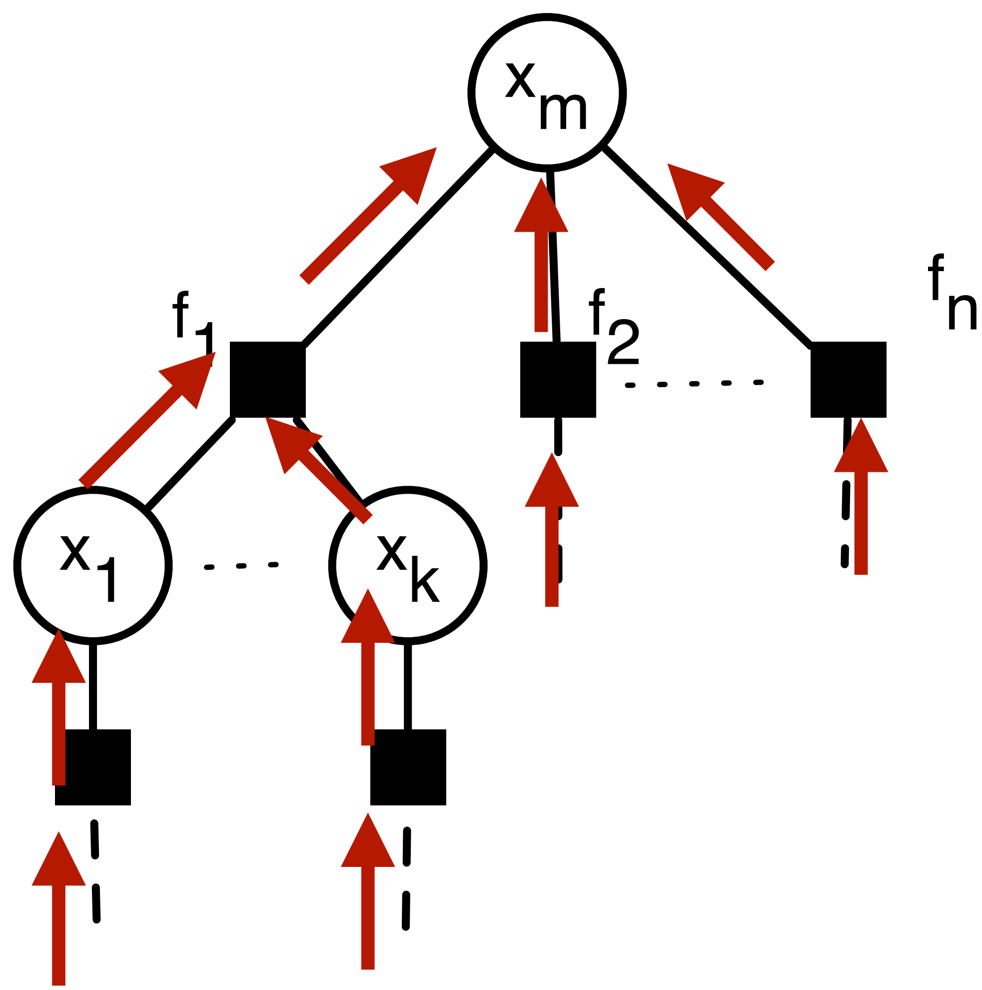
$$\mu_{f_j \to x_k}(x_k) = f_j(x_k)$$



$$\mu_{x_k \to f_j}(x_k) = 1$$

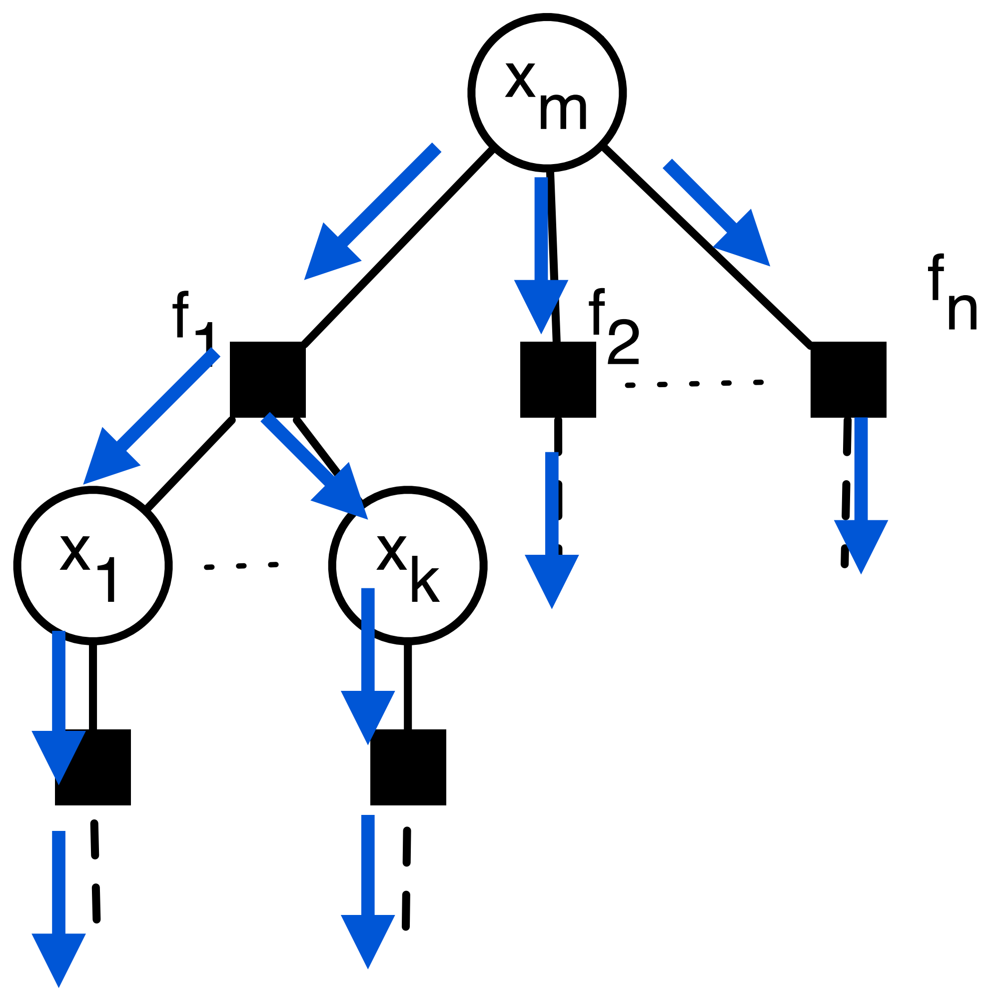
Sum-product algorithm: inference in a tree

You can compute the marginal for every node by passing twice through the tree



Sum-product algorithm: inference in a tree

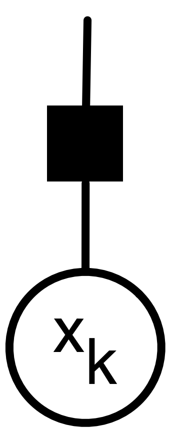
You can compute the marginal for every node by passing twice through the tree



Sum-product algorithm: inference in a tree

- Dealing with evidence corresponds to fixing the messages in the variable nodes.

- Subsequent re-normalisation of the computed conditional marginal is needed



$$\mu_{x_k \to f_j}(x_k) = 1 \quad \text{if} \quad X_k = x_k$$
$$\mu_{x_k \to f_j}(x_k) = 0 \quad \text{if} \quad X_k \neq x_k$$

Most probable solution

Replace the sum with max, feel free to verify it yourself

Optimising/fitting/training a model

- Maximum likelihood, EM, MCMC, Variational Inference, …
- Depends on the specific model
- Can become quite complicated
- You can easily fill an entire course on this topic

References

Bishop, chapter 8