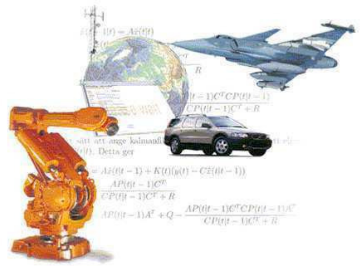
Machine Learning, Lecture 9
Graphical models



Thomas Schön

Division of Automatic Control
Linköping University
Linköping, Sweden.

Email: schon@isy.liu.se,
Phone: 013 - 281373,
Office: House B, Entrance 27.

Outline of lecture 9

1. Summary of lecture 8
2. Directed acyclic graphs
   - General properties
   - Conditional independence
3. Undirected graphs
   - General properties
   - Conditional independence
   - Relation with directed graphs
4. Factor graphs
   - Inference using belief propagation (BP)
     - Sum-product algorithm
     - Max-sum algorithm

(Chapter 8)

Summary of lecture 8 (I/III)

In **boosting** we train a sequence of M models $y_m(x)$, where the error function used to train a certain model depends on the performance of the previous models.

The models are then combined to produce the resulting classifier (for the two class problem) according to

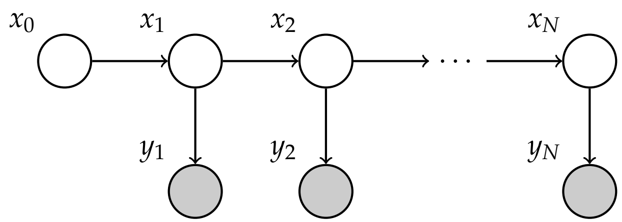
$$Y_M(x) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m y_m(x)\right)$$

We saw that the AdaBoost algorithm can be **interpreted** as a sequential minimization of an exponential cost function.

Summary of lecture 8 (II/III)

We started introducing some basic concepts for **probabilistic graphical models** $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ consisting of

1. a set of **nodes** $\mathcal{V}$ (a.k.a. vertices) representing the random variables and
2. a set of **links** $\mathcal{L}$ (a.k.a. edges or arcs) containing elements $(i, j) \in \mathcal{L}$ connecting a pair of nodes $(i, j) \in \mathcal{V}$ and thereby encoding the probabilistic relations between nodes.



Summary of lecture 8 (III/III)

The set of parents to node j is defined as

$$\mathcal{P}(j) \triangleq \{i \in \mathcal{V} \mid (i,j) \in \mathcal{E}\}$$

The directed graph describes how the joint distribution $p(x)$ **factors** into a product of factors $p(x_i \mid x_{\mathcal{P}(i)})$ only depending on a subset of the variables,

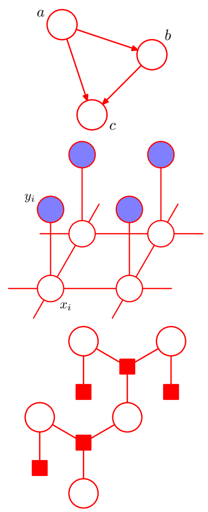
$$p(x_{\mathcal{V}}) = \prod_{i \in \mathcal{V}} p(x_i \mid x_{\mathcal{P}(i)}).$$

Hence, for the state space model on the previous slide, we have

$$p(X,Y) = p(x_0) \prod_{t=1}^{N} p(x_t \mid x_{t-1}) \prod_{t=1}^{N} p(y_t \mid x_t)$$

Why graphs?

- Simple visualization of probabilistic relationships.
- Can be used to design and motivate new models.
- They can provide some insights into the properties of the model, such as conditional independence properties.
- Some complex computations for inference and learning can be expressed and visualized.



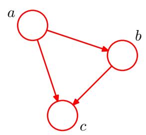
We are going to consider three types of graphs:

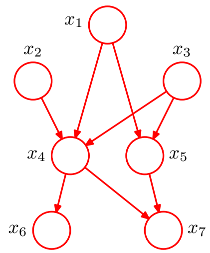
- **Directed graphs** a.k.a. Bayesian networks
- **Undirected graphs** a.k.a. Markov random fields
- **Factor graphs** are a more convenient form that can be obtained from the above two for the purposes of inference and learning.

Directed acyclic graphs

- Suppose we have K random variables $x_{1:K} = \{x_1, \dots, x_K\}$.
- The most general decomposition of the joint density of these variables is

$$p(x_{1:K}) = p(x_1) \prod_{k=2}^{K} p(x_k | x_{0:k-1})$$





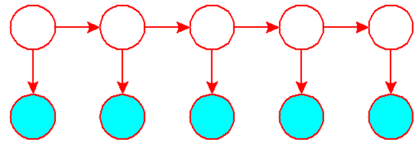
- With a directed acyclic graph, we have the following model.

$$p(x_{1:K}) = \prod_{k=1}^{K} p(x_k | x_{\mathcal{P}(k)})$$

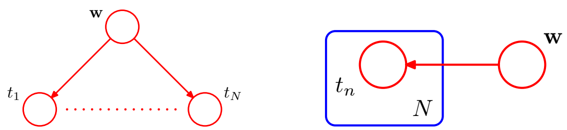
where $\mathcal{P}(k)$ is the parents of node k.

Measured, hidden and multiple variables

- The measured variables are shown with shaded nodes.



- If one has identical nodes, **plates** can be used to simplify the graph.



The variable N in the lower right corner gives the number of the identical nodes.

Suppose we have $x_{1:N}$ i.i.d. and distributed as

$$x_i \sim p(x|\pi_{1:K}, \mu_{1:K}, \Lambda_{1:K}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x; \mu_k, \Lambda_k^{-1}\right)$$

for $i = 1, \dots, N$.

In a Bayesian model, all the unknowns $\{\pi_{1:K}, \mu_{1:K}, \Lambda_{1:K}\}$ are modelled as random variables.

$$\pi_{1:K} \sim \mathrm{Dir}(\pi_{1:K}|\alpha_0) \overset{\triangle}{\propto} \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1}$$

$$\mu_{1:K}, \Lambda_{1:K} \sim p(\mu_{1:K}, \Lambda_{1:K}) \triangleq \prod_{k=1}^{K} \mathcal{N}(\mu_k; m_0, (\beta_0 \Lambda_k)^{-1}) \mathcal{W}(\Lambda_k | W_0, \nu_0)$$

Define the latent variables $z_n \triangleq [z_{n1}, \cdots, z_{nK}]^{\mathrm{T}}$ for $n = 1, \dots, N$ as we did in the construction used for EM and VB.

Then the joint density can be written as

$$p(x_{1:N}, z_{1:N}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}\left(x; \mu_k, \Lambda_k^{-1}\right)^{z_{nk}}$$

Example: Polynomial regression.

- Let $t_{1:N}$ be the values of a function at the points $x_{1:N}$.
- We would like to find the Kth degree polynomial approximating this function whose coefficients are shown as $\mathbf{w} \in \mathbb{R}^{K+1}$.
- $\mathbf{w} \sim \mathcal{N}(0, \Sigma)$
- Then the model can be written as

$$t_n = \phi(x_n)\mathbf{w} + v_n$$

  where $\phi(x) = \left[1, x, x^2, \dots, x^K\right]$.
- $\{v_n\}_{n=1}^{N}$ is i.i.d. and $v_n \sim \mathcal{N}(0, R)$.

Example: Polynomial regression

$$t_n = \phi(x_n)\mathbf{w} + v_n$$

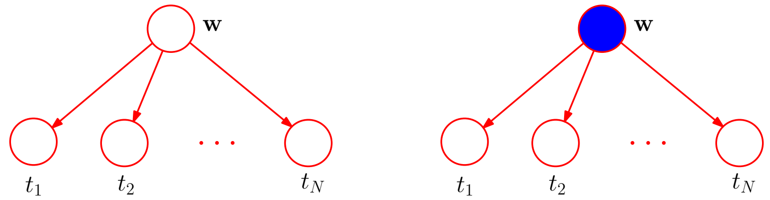
- The joint density for the problem can be written as

$$p(t_{1:N}, \mathbf{w}) = p(t_{1:N}|\mathbf{w})p(\mathbf{w}) = p(\mathbf{w}) \prod_{i=1}^{N} p(t_i|\mathbf{w})$$

- What is the reason for the equality $p(t_{1:N}|\mathbf{w}) = \prod_{i=1}^{N} p(t_i|\mathbf{w})$?

Observing conditional independence (III/III)

Example: Polynomial regression

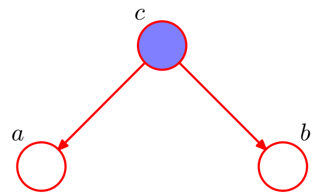


When $\mathbf{w}$ is assumed known it is said to "**block** the path", rendering all the variables $\{t_n\}_{n=1}^N$ conditionally independent.

Important question: Can this be formalized, i.e., can we discern CI properties directly from the graph?

CI from DAGs – Ex 1

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



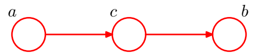
$$\implies \quad a \perp b|c$$

CI rule for tail-to-tail nodes

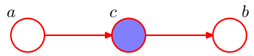
For conditional independence of two nodes, the tail-to-tail nodes between them must be observed, which blocks the path.

CI from DAGs – Ex 2

Head-to-tail nodes:



- Are a and b independent $a \overset{?}{\perp} b$?
- How about when c is given; $a \overset{?}{\perp} b|c$?



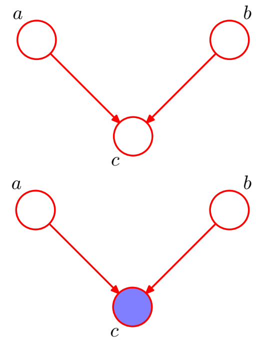
CI rule for head-to-tail nodes

For conditional independence of two nodes, the head-to-tail nodes between them must be observed, which blocks the path.

CI from DAGs – Ex 3

Head-to-head nodes:

- Are a and b independent $a \overset{?}{\perp} b$? Yes, since $\int p(a,b,c)\mathrm{d}c = \int p(a)p(b)p(c \mid a,b)\mathrm{d}c = p(a)p(b)$.
- How about when c is given; $a \overset{?}{\perp} b|c$? No, since $p(a,b \mid c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a)p(b)p(c|a,b)}{p(c)} \neq p(a \mid c)p(b \mid c)$.



CI rule for head-to-head nodes

For conditional independence of two nodes, the head-to-head nodes between them must be unobserved, which blocks the path.

D-separation

D-separation for Directed Acyclic Graphs

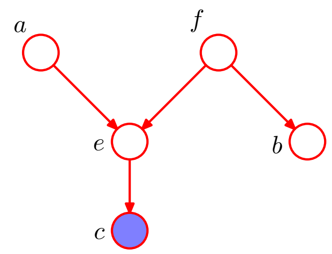
Consider a directed acyclic graph in which A, B and C are arbitrary non-intersecting sets of nodes. We have the property

$$A \perp B | C$$

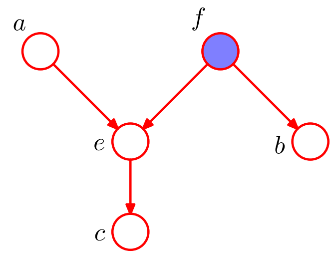
if, on all possible paths from any node in A to any node in B,

- all tail-to-tail and head-to-tail nodes are in C;
- neither head-to-head nodes nor any of their descendants are in C.

D-separation examples



- The path from a to b is not blocked by f, since it is a tail-to-tail node and f not observed.
- Nor is it blocked by e, which is a head-to-head node, with an observed node c as descendant.
- Hence, CI $(a \perp b \mid c)$ does **not follow** from this graph.



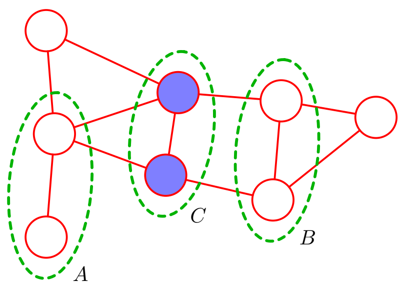
- The path from a to b is blocked by f, since it is a tail-to-tail node and f is observed.
- It is also blocked by e, head-to-head node and neither it not its descendants are observed.
- Hence, CI $(a \perp b \mid c)$ **follows** from this graph.

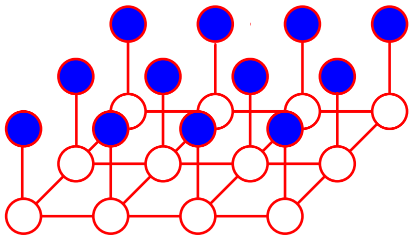
Undirected graphical model (Markov random fields)

- Nodes and edges carry similar meanings.
- Conditional independence is determined by graphical separation.

$$A \perp B | C$$

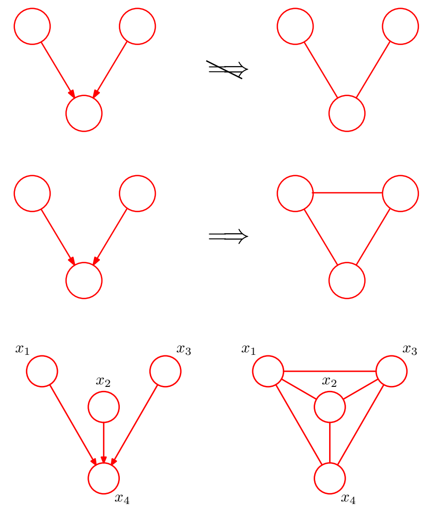
- A more natural representation for some models, e.g., images.
- One must take special care while converting directed graphs to undirected ones.





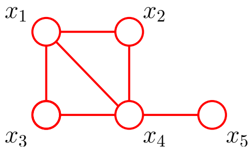
Conversion from directed to undirected

- When conversion is done directly some correlations that would be present in the original model can be lost.
- One must "marry" the parents to get those correlations back, this is called **moralization**.
- Moralization has to be performed for all the pairs of parents.



PDF for an undirected graphical model

The core result is given by the so-called Hammersley-Clifford theorem using the concept of **cliques**.



Definition (Clique)

A clique C is a subset of nodes $\{1, \ldots, N\}$ of an undirected graph such that there exists a link between all pairs.

Hammersley-Clifford Theorem (a basic version)

The joint probability distribution $p(x_{1:N})$ of an undirected graph for variables $\{x_1, \ldots, x_N\}$ is given by

$$p(x_{1:N}) = \frac{1}{Z}\prod_C \psi_C(x_C) \quad \text{where} \quad Z = \sum_{x_{1:N}}\prod_C \psi_C(x_C).$$

Physics interpretation

- The Hammersley-Clifford theorem has a physics interpretation when the functions $\psi_C(x_C)$ are non-zero everywhere.
- In this case, we can write

$$\psi_C(x_C) = \exp(-E(x_C))$$

  where $E(\cdot)$ is called an **energy function**.
- The overall graph can then be considered as a lattice with a potential energy function described by $E(x_C)$.
- Finding the maximum of the density can then be considered as finding the point where the total potential energy is minimized.

$$p(x_{1:N}) = \frac{1}{Z}\prod_C \exp(-E(x_C)) = \frac{1}{Z}\exp\left(-\sum_C E(x_C)\right)$$

- A local maximum then corresponds to an equilibrium.

Application – image de-noising (I/II)

Suppose we have a noisy image and want to remove the noise.

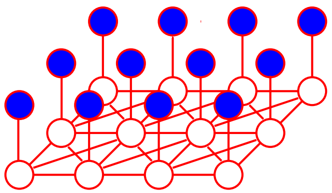
- Model the true pixel values as $x_{i,j}$.
- Model the measured image pixel values as

$$y_{i,j} = x_{i,j} + v_{i,j}, \quad v_{i,j} \sim \mathcal{N}(0, \beta^2).$$

- Choose the energy functions as

$$E_y(x_{i,j}, y_{i,j}) = \frac{1}{\beta^2}(y_{i,j} - x_{i,j})^2$$

$$E_x(x_{i_1,j_1}, x_{i_2,j_2}) = \min\left(\frac{1}{\alpha^2}(x_{i_1,j_1} - x_{i_2,j_2})^2, \gamma\right)$$

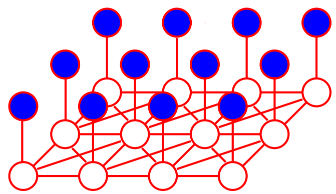


Application – image de-noising (II/II)

- The density is then

$$\begin{aligned}-\log p(x_{1:N_x,1:N_y}, y_{1:N_x,1:N_y}) = &\sum_{i,j} E_y(x_{i,j}, y_{i,j}) \\ &+ E_x(x_{i,j}, x_{i+1,j+1}) + E_x(x_{i,j}, x_{i-1,j-1}) \\ &+ E_x(x_{i,j}, x_{i-1,j+1}) + E_x(x_{i,j}, x_{i+1,j-1}) + C\end{aligned}$$

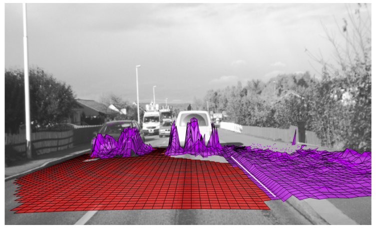
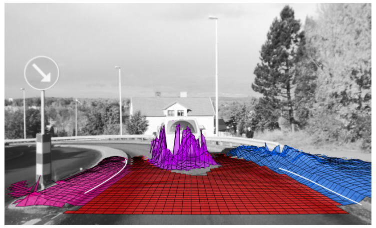
- If the image is 8 bit grayscale, maximization in general requires the calculation of $256^{(N_x \times N_y)}$ different combinations.
- We instead maximize w.r.t. only one pixel keeping the others fixed at their last values.
- This is called **Iterative Conditional Modes (ICM)**.



Road surface estimation

Aim: Estimate the road surface using images from a stereo camera.

Solved using a Conditional Random Field (CRF) model and message passing.

Lorentzon, M. and Andersson, T. **Road surface modeling using stereo vision**, Master's thesis, LiTH-ISY-EX–12/4582–SE, Linköping university, Sweden, 2012.

http://liu.diva-portal.org/smash/record.jsf?searchId=2&pid=diva2:532767

Inference in graphical models

Inference in graphical models amounts to computing the posterior distribution of one or more of the nodes that are not observed.

The **structure** in the graphical model is exploited in finding inference algorithms.

Most inference algorithms can be expressed in terms of **message passing** algorithms, where local messages are propagated around the graph.

Inference on a chain

The message $\mu_\alpha(x_n)$ can be evaluated **recursively**

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

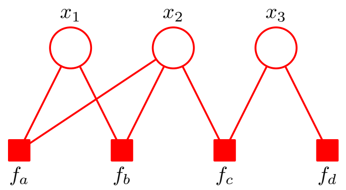
where the recursion is started by $\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$. Similarly, for the message $\mu_\beta(x_n)$ we have

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

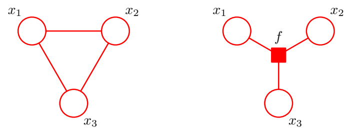
The generalization of this message passing idea to trees is referred to as the **sum-product algorithm**.

Factor graphs

- Both directed and undirected graphs give a factorial representation for the joint density.
- Factor graphs make this factorization more explicit by adding nodes for each factor.
- Both directed and undirected graphs can be converted into factor graphs.



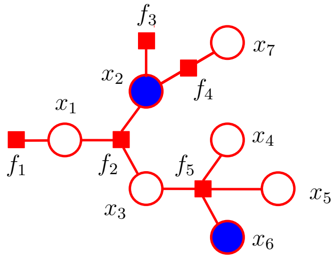
$$p(x_{1:4}) = f_a(x_1, x_2) f_b(x_1, x_2) f_c(x_2, x_3) f_d(x_3)$$



Inference in factor graphs

- We have the joint density for the graph on the right given as

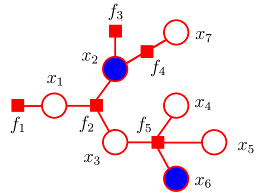
$$p(x_{1:7}) \propto f_1(x_1) f_2(x_{1:3}) f_3(x_2) f_4(x_2, x_7) f_5(x_{3:6})$$



- When we have measurements of some variables, we might need the posteriors of some or all unobserved variables.

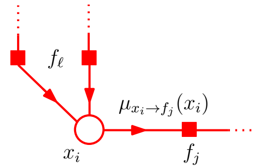
$$p(x_1, x_3, x_4, x_5, x_7 | x_2, x_6) = \frac{p(x_{1:7})}{p(x_2, x_6)} = \frac{p(x_{1:7})}{\sum_{x_1, x_3, x_4, x_5, x_7} p(x_{1:7})}$$

Inference in factor graphs



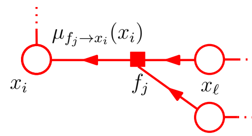
- Making inference requires marginals.
- It is possible to calculate the marginals on a graph efficiently by passing local messages along the graph.
- Two interconnected types of messages are considered
  - Messages from variable nodes to factor nodes

$$\mu_{x_i \to f_j}(x_i) = \prod_{f_\ell \in \mathrm{ne}(x_i) \setminus f_j} \mu_{f_\ell \to x_i}(x_i)$$



  - Messages from factor nodes to variable nodes

$$\mu_{f_j \to x_i}(x_i) = \sum_{x_\ell \in \mathrm{ne}(f_j) \setminus x_i} f_j \prod_{x_\ell \in \mathrm{ne}(f_j) \setminus x_i} \mu_{x_\ell \to f_j}(x_\ell)$$



Inference in factor graphs

Sum-Product Algorithm

- Calculate messages from variable nodes to factor nodes

$$\mu_{x_i \to f_j}(x_i) = \prod_{f_\ell \in \mathrm{ne}(x_i) \setminus f_j} \mu_{f_\ell \to x_i}(x_i)$$

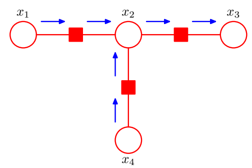
- Calculate messages from factor nodes to variable nodes

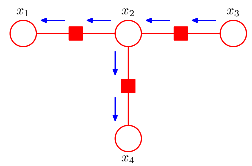
$$\mu_{f_j \to x_i}(x_i) = \sum_{x_\ell \in \mathrm{ne}(f_j) \setminus x_i} f_j \prod_{x_\ell \in \mathrm{ne}(f_j) \setminus x_i} \mu_{x_\ell \to f_j}(x_\ell)$$

- Iterate messages until convergence. (Different iteration schemes can be designed.)
- After convergence, the marginals are calculated as

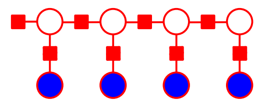
$$p(x_i) \propto \prod_{f_\ell \in \mathrm{ne}(x_i)} \mu_{f_\ell \to x_i}(x_i)$$

Inference in factor graphs



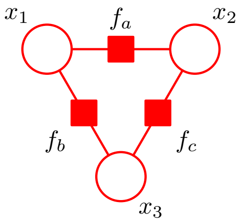


- The values in the observed nodes are just substituted into the factors and not integrated out.
- If the graph is a tree, the algorithm can calculate all the marginals by making
  - a forward pass from the root to the leaves
  - a backward pass from the leaves to the root.
- The sum-product algorithm gives the exact results in a tree structured graph.
- The sum-product algorithm is **equivalent to a Kalman smoother** for linear Gaussian dynamical systems.



Inference in factor graphs

- When the sum-product algorithm is applied to directed graphs without loops the resulting algorithm is sometimes referred to as **belief propagation**.
- In a graph with loops, the sum-product algorithm is not exact and actually might not converge.
- People anyway apply it to the graphs with loops also, which is called **loopy belief propagation**.
- Even in this form, it has important applications in communications (decoding of error correcting codes).



A few concepts to summarize lecture 9

Directed graphs: A graphical description of a probabilistic model where the conditional probabilities correspond to edges.

D-separation: Checking for conditional independence is somewhat troublesome for directed graphs requiring a condition called D-separation to be satisfied.

Undirected graphs: Another graphical representation where conditional independence is given by simple graph separation.

Factor graphs: An extension of directed and undirected graphs which makes the probabilistic factors explicit.

Belief propagation: A probabilistic inference type using graphs where local messages are propagated among the graph nodes.

Sum-product algorithm: A form of belief propagation which gives exact results only for trees but also applied to graphs with loops anyway.