COMPSCI 773 S1C

Global Energy Minimisation: Belief Propagation Stereo

Prof. Georgy Gimel'farb

Stereo Matching: MAP, ML, MPM, and MWM

- Maximum a posteriori (MAP) decision:

$$D^*_{\mathrm{MAP}} = \arg\max_D \Pr(D|G_1, G_2) \equiv \arg\max_D \Pr(D)\Pr(G_1, G_2|D)$$

  - Unknown or uniform prior $P(D) \Rightarrow$ Maximum likelihood (ML) decision:

$$D^*_{\mathrm{ML}} = \arg\max_D \Pr(G_1, G_2|D)$$

- Maximum posterior marginals (MPM):

$$d^*(x, y) = \arg\max_d \Pr_{x,y}(d|G_1, G_2);\ (x, y) \in \mathbf{S}$$

- Marginal weighed means (MWM):

$$d^*(x, y) = \sum_d d \cdot \Pr_{x,y}(d|G_1, G_2); (x, y) \in \mathbf{S}$$

Main problems: maximising multivariate distributions of disparity maps and images or computing their marginals

Basic Simplification: Pairwise Interactions

MAP,ML: Minimum energy $\min_D\{\log \Pr(D) + \log \Pr(G_1, G_2|D)\}$:

$$D^* = \arg\min_{d_{0,0},\ldots,d_{X-1,Y-1}} \left\{ \sum_{x=0}^{X-1}\sum_{y=0}^{Y-1} \varphi_{x,y}(d_{x,y}) \quad + \right.$$
$$\left. \sum_{x=1}^{X-1}\sum_{y=0}^{Y-1} \psi_{\mathrm{h}:x,y}(d_{x-1,y}, d_{x,y}) + \sum_{x=0}^{X-1}\sum_{y=1}^{Y-1} \psi_{\mathrm{v}:x,y}(d_{x,y-1}, d_{x,y}) \right\}$$

MPM, MWM: Marginal distributions $\Pr_{x,y}(d_{x,y}|G_1, G_2)$

$$f_{i,j}(d_{i,j}) = \sum_{\substack{(\xi,\eta)=(0,0)\\(\xi,\eta)\neq(i,j)}}^{(X-1,Y-1)} \sum_{d_{\xi,\eta}=d_{\min}}^{d_{\max}} \left[\prod_{(x,y)=(0,0)}^{(X-1,Y-1)} \Phi_{x,y}(d_{x,y}) \quad \times \right.$$
$$\left. \prod_{(x,y)=(1,0)}^{(X-1,Y-1)} \Psi_{\mathrm{h}:x,y}(d_{x-1,y}, d_{x,y}) \prod_{(x,y)=(0,1)}^{(X-1,Y-1)} \Phi_{\mathrm{v}:x,y}(d_{x,y-1}, d_{x,y}) \right]$$

1D cases ($Y = 1$): exact solutions;

2D cases (NP-hard): only approximate solutions

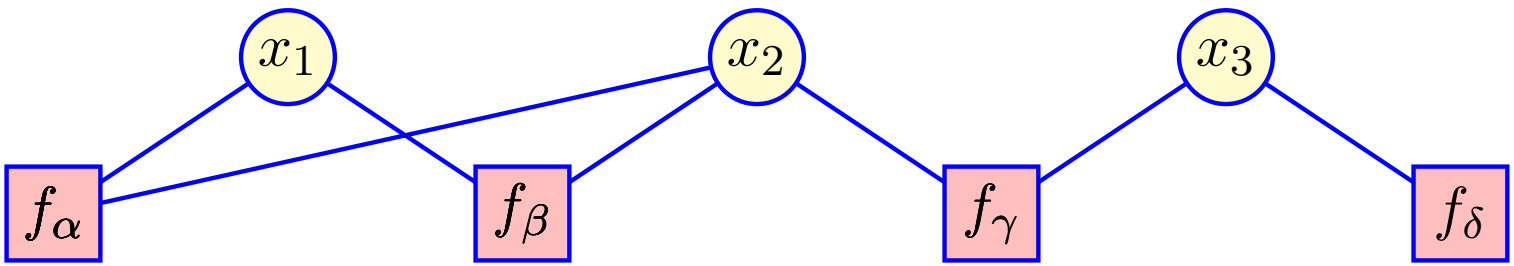
Graphical Models: Joint Probability Distributions

Joint p.d. over a set of variables D: a product of factors – functions depending each on a subset of variables $D_s \subset D$:

$$D = (d_{0,0}, \ldots, d_{X-1,Y-1}) \quad \Rightarrow \quad \Pr(D) = \prod_s f_s(D_s)$$

Factor graph: a node for every variable d_i and a node for each factor $f_s(D_s)$ in the joint p.d.

- Factor graph for
  $p(d_1, d_2, d_3) = f_\alpha(d_1, d_2) f_\beta(d_1, d_2) f_\gamma(d_2, d_3) f_\delta(d_3)$



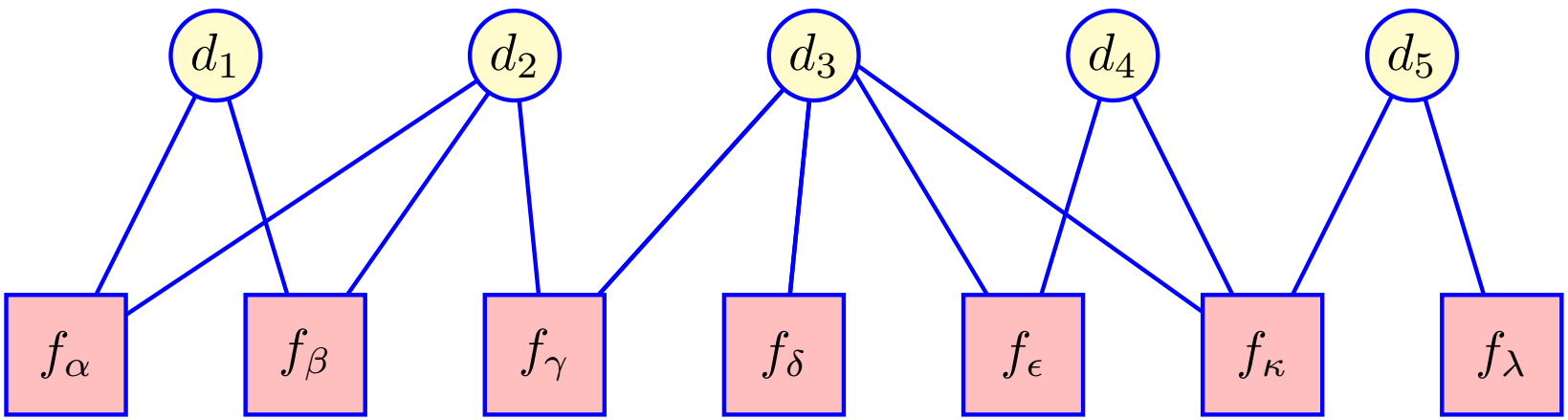
Bipartite Factor Graphs

- **Goal**: to simplify and generalise the sum-product algorithm for computing marginal probabilities given a joint p.d. $\Pr(d_1, \ldots, d_n)$:

$$p_i(d_i) = \sum_{d_1} \cdots \sum_{d_{i-1}} \sum_{d_{i+1}} \cdots \sum_{d_n} \Pr(d_1, \ldots, d_{i-1}, d_i, d_{i+1}, \ldots, d_n)$$

- Explicitly represent a function factored over subsets of variables
  - Nodes for factors in addition to the nodes for variables:

$$\Pr(D) = f_\alpha(d_1, d_2) f_\beta(d_1, d_2) f_\gamma(d_2, d_3) f_\delta(d_3) f_\epsilon(d_3, d_4) f_\kappa(d_3, d_4, d_5) f_\lambda(d_5)$$

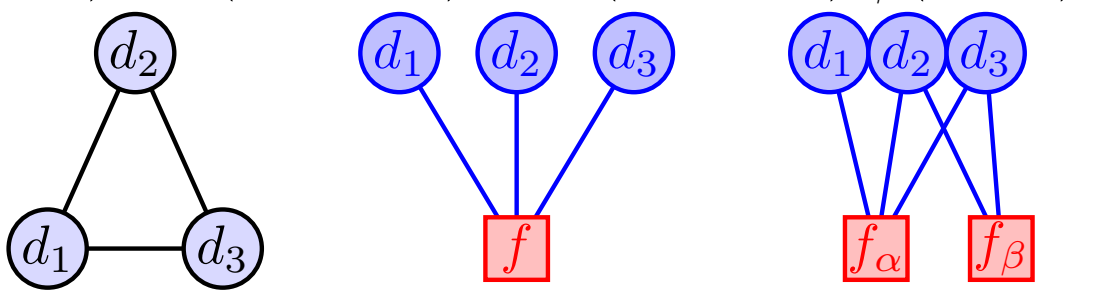


Bipartite Factor Graphs

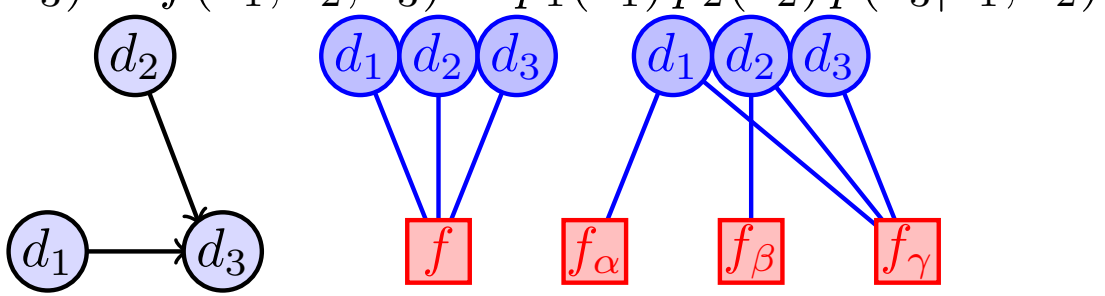
- Two distinct kinds of nodes
- All links go between the nodes of opposite type

Many different factor graphs for the same undirected or directed graph

$$\Pr(d_1, d_2, d_3) = f(d_1, d_2, d_3) = f_\alpha(d_1, d_2, d_3) f_\beta(d_2, d_3)$$

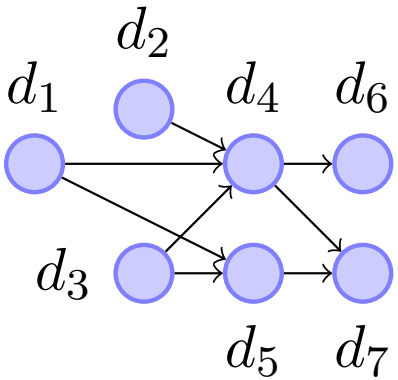


$$\Pr(d_1, d_2, d_3) = f(d_1, d_2, d_3) = \overbrace{p_1(d_1)}^{f_\alpha(d_1)} \overbrace{p_2(d_2)}^{f_\beta(d_2)} \overbrace{p(d_3|d_1, d_2)}^{f_\gamma(d_1,d_2,d_3)}$$



Forming a Factor Graph for a Directed Graph $\Gamma = \{\mathbb{N}, \mathbb{E}\}$

$$\Pr(d_1,\ldots,d_7)=p_1(d_1)p_2(d_2)p_3(d_3)p_4(d_4|d_1,d_2,d_3)p_5(d_1,d_3)p_6(d_6|d_4)p_7(d_7|d_4,d_5)$$

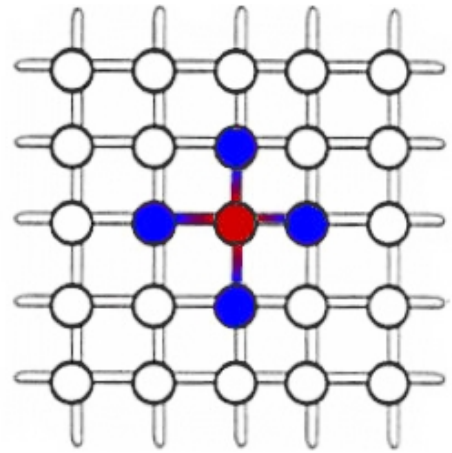


Markov chains, or trees, or Bayesian networks: $\Pr(d_1,\ldots,d_n) = \prod_i p_i(d_i|d_j : j \in S_i)$ for all $i = 1,\ldots,n$; $S_i \subset \{1,\ldots,n\}\backslash i$

1. Create variable nodes for all the nodes $i \in \mathbb{N}$
2. Create factor nodes corresponding to conditional distributions $p_i(d_i|d_j : j \in S_i)$
3. Add the links between the variable and factor nodes

Factor graphs have a tree structure for any original tree – undirected, directed, or polytree

Forming a Factor Graph for an Undirected Graph (optional)



Markov random field (MRF) model:

$$\Pr(d_1, \ldots, d_n) = \frac{1}{Z} \prod_{c \in C} f_c(d_j : \ j \in S_c)$$

1. Create variable nodes d_i for all the nodes $i \in \mathbb{N}$
2. Create factor nodes corresponding to the maximal complete subgraphs, or cliques c containing each the nodes $S_c \subset \mathbb{N}$
3. Set the factors $f_c(d_j : \ j \in S_c)$ equal to the same potential functions for the MRF
4. Add the links between the variable and factor nodes

- Many possible factor graphs for the same model

Computing Marginals: The Sum–Product Algorithm

Marginal probabilities $p_i(d_i) = \sum_{\mathbb{N}\setminus i} \Pr(d_j : j \in \mathbb{N})$ for a node (or several nodes):

- Substitute for $\Pr(d_1, \ldots, d_N)$ using the factor graph expression:

$$\Pr(D) = \frac{1}{Z} \prod_c f_c(d_j : j \in S_c)$$

  where $S_c \subset \{1, \ldots, N\}$ is a subset of the nodes

- Then interchange the summations and products for obtaining a computationally efficient algorithm: $p_i(d_i) =$

$$\sum_{d_j:\, j \in \mathbb{N}\setminus i} \prod_c f_c(d_j : j \in S_c) \Rightarrow \prod_{\substack{\text{neighbours of} \\ i \in S_c}} \sum_{\substack{d_j \\ j \in S_c}} f_{\mathbf{c}}(d_i, d_j : j \in S_c \setminus i)$$

Special case: **Belief propagation** for exact inference on digraphs without loops

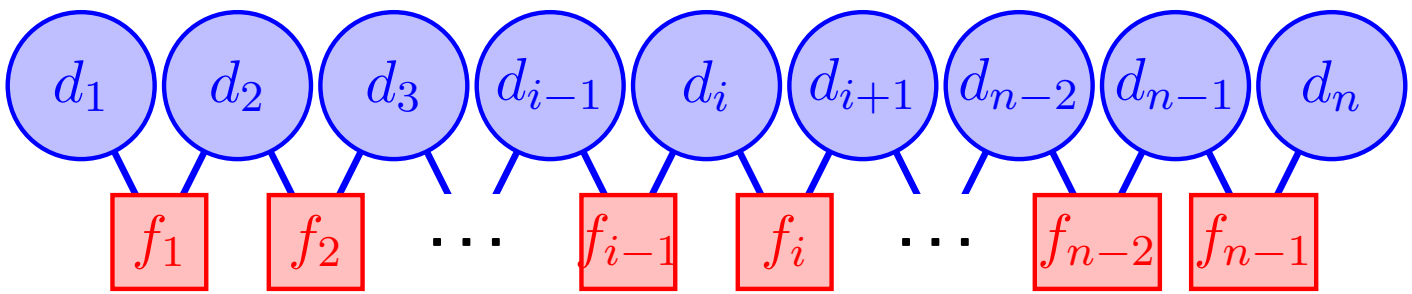
Computing Marginals

- Joint distribution of n discrete variables $\Pr(d_1, d_2, \ldots, d_n)$
- K-valued variables: all d_i in $\mathbb{K} = \{0, \ldots, K-1\}$
- Marginal p.d. of d_i (K^n operations in the general case!):

$$p_i(x_i) = \sum_{d_1=0}^{K-1} \cdots \sum_{d_{i-1}=0}^{K-1} \sum_{d_{i+1}=0}^{K-1} \cdots \sum_{d_n=0}^{K-1} \Pr(d_1, \ldots, d_n)$$

- A simple case: an undirected Markov chain

$$\Pr(d_1, \ldots, d_n) = f_1(d_1, d_2) f_2(d_2, d_3) \cdots f_{n-1}(d_{n-1}, d_n)$$



Computing Marginals $p_i(d_i)$: a Simple Case

For $\Pr(d_1, \ldots, d_n) = f_1(d_1, d_2) f_2(d_2, d_3) \cdots f_{n-1}(d_{n-1}, d_n)$ – only $(n-1)K^2$ operations by interchanging the sums and the products:

$$
\begin{aligned}
p_i(d_i) \quad = \quad & \overbrace{\sum_{d_{i-1}=0}^{K-1} f_{i-1}(d_{i-1}, d_i) \cdots \overbrace{\sum_{d_2=0}^{K-1} f_2(d_2, d_3) \overbrace{\sum_{d_1=0}^{K-1} f_1(d_1, d_2)}^{\varphi_1(d_2)}}^{\varphi_2(d_3)}}^{\varphi_{i-1}(d_i)} \\
\times \quad & \underbrace{\sum_{d_{i+1}=0}^{K-1} f_i(d_i, d_{i+1}) \cdots \underbrace{\sum_{d_{n-1}=0}^{K-1} f_{n-2}(d_{n-2}, d_{n-1}) \underbrace{\sum_{d_n=0}^{K-1} f_{n-1}(d_{n-1}, d_n)}_{\psi_{n-1}(d_{n-1})}}_{\psi_{n-2}(d_{n-2})}}_{\psi_i(d_i)}
\end{aligned}
$$

Computing Marginals: a Simple Case

Sequential computations ($d_j \in \mathbb{K} = \{0, 1, \ldots, K-1\}$; $j = 1, \ldots, n$):

$$\begin{aligned}
\varphi_1(d_2) &= \sum_{d_1=0}^{K-1} f_1(d_1, d_2) && \text{for each } d_2 \in \mathbb{K} \\
\varphi_2(d_3) &= \sum_{d_2=0}^{K-1} f_2(d_2, d_3)\varphi_1(d_2) && \text{for each } d_3 \in \mathbf{K} \\
\ldots &\quad \ldots && \ldots \\
\varphi_{i-1}(d_i) &= \sum_{d_{i-1}=0}^{K-1} f_{i-1}(d_{i-1}, d_i)\varphi_{i-2}(d_{i-1}) && \text{for each } d_i \in \mathbb{K}
\end{aligned}$$

$$\begin{aligned}
\psi_{n-1}(d_{n-1}) &= \sum_{d_n=0}^{K-1} f_{n-1}(d_{n-1}, d_n) && \text{for each } d_{n-1} \in \mathbb{K} \\
\psi_{n-2}(d_{n-2}) &= \sum_{d_{n-1}=0}^{K-1} f_{n-2}(d_{n-2}, d_{n-1})\psi_{n-1}(d_{n-1}) && \text{for each } d_{n-2} \in \mathbb{K} \\
\ldots &\quad \ldots && \ldots \\
\psi_i(d_i) &= \sum_{d_{i+1}=0}^{K-1} f_i(d_i, d_{i+1})\psi_{i+1}(d_{i+1}) && \text{for each } d_i \in \mathbb{K}
\end{aligned}$$

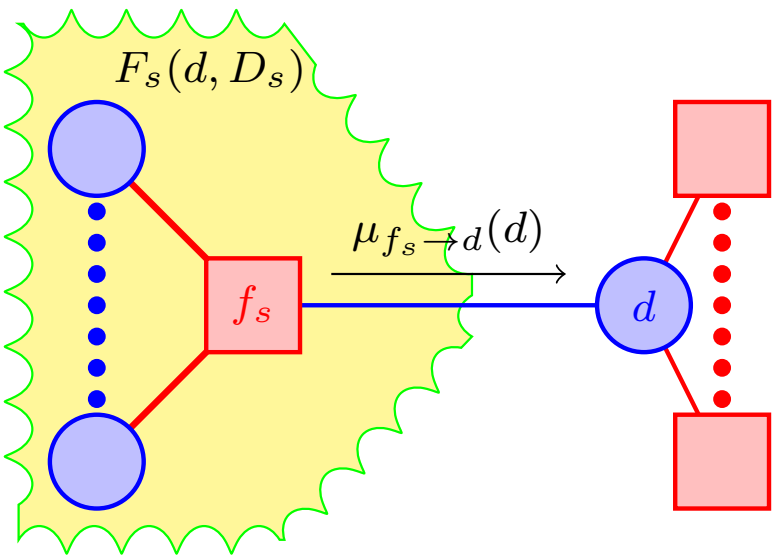
$$p_i(d_i) = \varphi_{i-1}(d_i)\psi_i(d_i)$$

The Sum–Product Algorithm

A fragment of a factor graph: evaluating the marginal $p(d)$

$$\Pr(D) = \prod_{s \in \mathbb{N}(d)} F_s(d, D_s)$$

$$\Rightarrow \; p(d) = \sum_{d_s : s \in \mathbb{N} \setminus d} \Pr(D)$$



Message from f_s to d:

$$\mu_{f_s \to d}(d) \equiv \sum_{D_s} F_s(d, D_s)$$

Partitioning factors in the joint p.d. $\Pr(D)$ into groups according to the tree structure of the graph

- One group per each of the factor nodes being a *neighbour* of the variable node d
- $\mathbb{N}(d))$ – the set of the factor nodes that are neighbours of d
- D_s – the set of all variables in a subtree connected to the node d via the factor node f_s
- $F_s(d, D_s)$ – the product of all the factors in the group associated with the node f_s

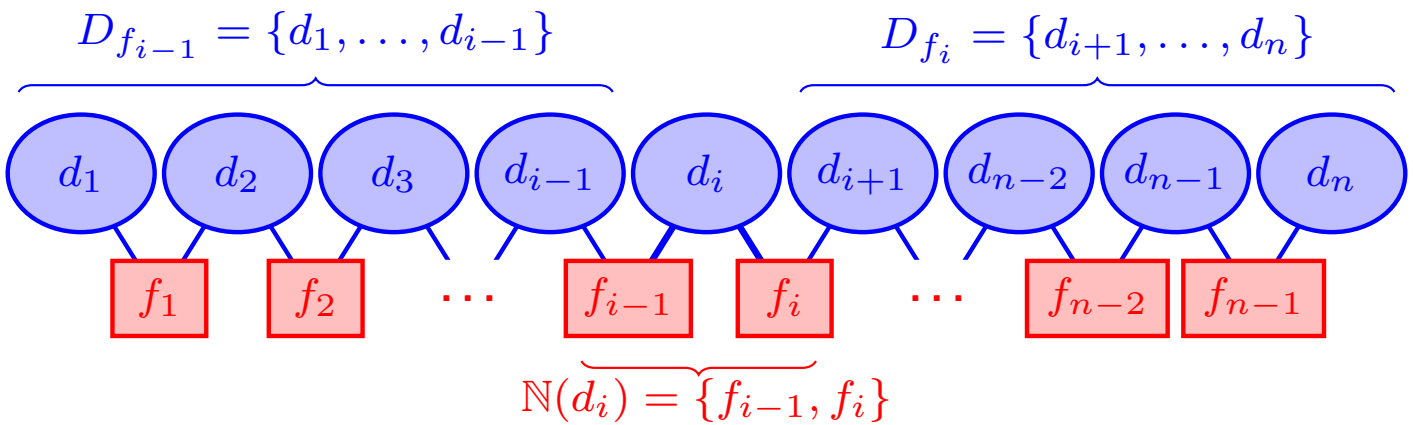
Computing Marginals: a Simple Case

$$\Pr(D) = \prod_{\substack{s\in\mathbb{N}(d_i)\\ =\{f_{i-1},f_i\}}} F_s(d_i, D_s) = \underbrace{\prod_{j=1}^{i-1} f_j(d_j, d_{j+1})}_{F_{f_{i-1}}(d_i, D_{f_{i-1}})} \underbrace{\prod_{j=i}^{n-1} f_j(d_j, d_{j+1})}_{F_{f_i}(d_i, D_{f_i})}$$

$$p_i(d_i) = \prod_{s\in\mathbb{N}(d_i)} \sum_{D_s} F_s(d_i, D_s) = \mu_{f_{i-1}\to d_i}(d_i)\mu_{f_i\to d_i}(d_i)$$

$$\mu_{f_{i-1}\to d_i}(d_i) = \sum_{D_{f_{i-1}}=\{d_1,\ldots,d_{i-1}\}} F_{f_{i-1}}(d_i,\ D_{f_{i-1}} = \{d_1,\ldots,d_{i-1}\})$$

$$\mu_{f_i\to d_i}(d_i) = \sum_{D_{f_i}=\{d_{i+1},\ldots,d_n\}} F_{f_i}(d_i,\ D_{f_i} = \{d_{i+1},\ldots,d_n\})$$



The Sum–Product Algorithm

The goal marginal is the product of all the incoming **messages**:

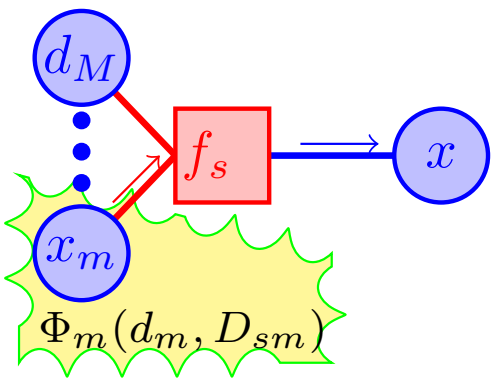
$$\begin{aligned} p(d) &= \sum_{D\setminus d} \Pr(D) = \sum_{D\setminus d} \prod_{s\in\mathbb{N}(d)} F_s(d, D_s) = \prod_{s\in\mathbb{N}(d)} \left[\sum_{D_s} F_s(d, D_s) \right] \\ &= \prod_{s\in\mathbb{N}(d)} \mu_{f_s\to d}(d) \end{aligned}$$

Each factor $F_s(d, D_s)$ is described by a factor (sub)-graph and so can itself be factored:

$$F_s(d, D_s) = f_s(d, d_1, \ldots, d_M)\Phi_1(d_1, D_{s_1}) \cdots \Phi_M(d_M, D_{s_M})$$

$d_1, \ldots, d_M$ – variables associated with factor f_s in addition to d

- Messages from factor nodes to variable nodes:
  $\mu_{f_s\to d_i}(d_i) \equiv \sum_{D_s} F_s(d_i, D_s)$; $s \in \mathbb{N}(d_i)$
- Messages from variable nodes to factor nodes:
  $\mu_{d_m\to f_s}(d_m) \equiv \sum_{D_m} |\Phi_m(d_m, D_m)$; $f_s \in \mathbb{N}(d_m)$



Computing Messages from Factor Nodes to Variable Nodes

$$\mu_{f_s \to d}(d) \equiv \sum_{D_s} F(d, D_s)$$

$$= \sum_{d_1} \cdots \sum_{d_M} f_s(d, \{d_1, \ldots, d_M\}) \prod_{m \in \mathbb{N}(f_s) \setminus d} \overbrace{\sum_{D_m} \Phi_m(d_m, D_m)}^{\mu_{d_m \to f_s}(d_m)}$$

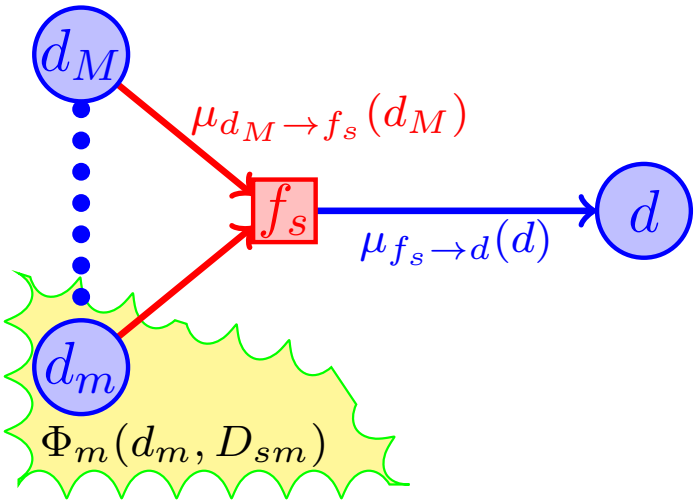
$$= \sum_{d_1} \cdots \sum_{d_M} f_s(d, \{d_1, \ldots, d_M\}) \prod_{m \in \mathbb{N}(f_s) \setminus d} \mu_{d_m \to f_s}(d_m)$$

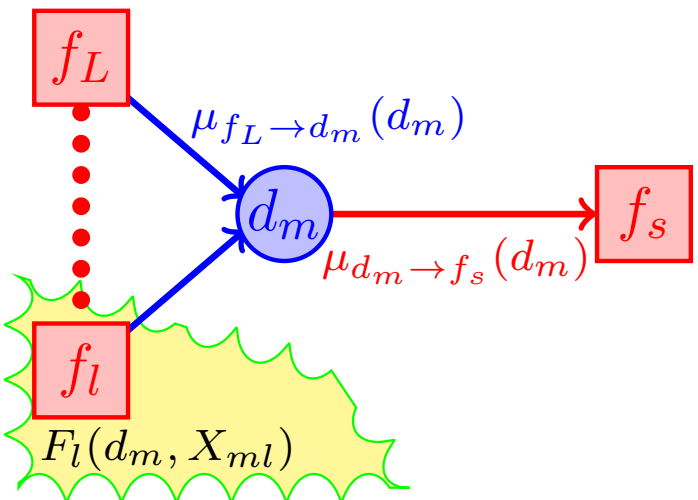
- $\mathbb{N}(f_s)$ – the set of variable nodes being neighbours of the factor node f_s
- $\mathbb{N}(f_s) \setminus d$ – the same set but without the node d

Evaluating the message sent by a factor node to a variable node along their connecting link:

1. Take the product of the incoming messages along all other links coming to the factor node
2. Multiply by the factor associated with that node
3. Marginalise over all the variables associated with the incoming messages

Message Evaluation





$$\mu_{d_m \to f_s}(d_m) = \sum_{D_m} \Phi_m(d_m, D_{sm})$$

- Sending a factor-to-variable message after receiving incoming messages from all other neighbouring variable nodes
- Evaluating the variable-to factor-messages – again by the (sub)-graph factoring
  - The term $\Phi_m(d_m, D_{sm})$ associated with a node d_m is a product of the terms $F_l(d_m, D_{ml})$, associated each with one of the factor nodes f_l linked to node d_m (excluding the node f_s):

$$\Phi_m(d_m, D_{sm}) = \prod_{l \in \mathbb{N}(d_m) \setminus f_s} F_l(d_m, D_{ml})$$

  - The product over all neighbours of the node d_m excepting the node f_s
  - Each of the factors $F_l(d_m, D_{ml})$: a subtree of the original graph of the same kind as the joint $\Pr(D)$

Message Evaluation

Sending a variable-to-factor message along the connecting link:

$$
\begin{aligned}
\mu_{d_m \to f_s}(d_m) &= \prod_{l \in \mathbb{N}(d_m) \setminus f_s} \overbrace{\left[\sum_{D_{ml}} F_l(d_m, D_{ml}) \right]}^{\mu_{f_l \to d_m}(d_m)} \\
&= \prod_{l \in \mathbb{N}(d_m) \setminus f_s} \mu_{f_l \to d_m}(d_m) \\
\mu_{d_m \to f_s}(d_m) &= \sum_{D_{sm}} \Phi_m(d_m, D_{sm}) \\
&= \sum_{D_{sm}} \left[\prod_{l \in \mathbb{N}(d_m) \setminus f_s} F_l(d_m, D_{ml}) \right]
\end{aligned}
$$

- Product of the incoming messages along all of the other links:
  - A variable node with only two 2 neighbours passes messages through unchanged
  - A variable-to-factor message can be sent once incoming messages from all other neighbouring factor nodes are received

Message Evaluation

Marginal for a variable node d: the product of incoming messages along all of the links arriving at that node:

$$p(d) = \prod_{s \in \mathbb{N}(d)} \mu_{f_s \to d}(d);$$

$$p(D_s) = f_s(D_s) \prod_{i \in \mathbb{N}(f_s)} \mu_{d_i \to f_s}(d_i)$$

- Each of these messages can be computed recursively from other messages by viewing d as the root of the tree and starting at the leaf nodes
- A variable leaf node: $\mu_{d \to f}(d) = 1$
- A factor leaf node: $\mu_{f \to d}(d) = f(d)$

General Sum–Product Algorithm for a Tree

- Pick any arbitrary (variable or factor) node to serve as the root
- Propagate messages from the leaves to the root until the root node will have received messages from all of its neighbours
- Once this message propagation is complete, then propagate messages from the root to all of its neighbours and further along all of the links outwards from the root all the way to the leaves
  - Now a message passed in both directions across every link in the graph, and every node received a message from all its neighbours
  - Marginal distribution is readily calculated for every variable in the graph because every variable node has received the messages from all its neighbours
  - After one message has passed in each direction across each link, the marginal distributions are $p(d) = \prod_{s \in \mathbb{N}(d)} \mu_{f_s \to d}(d)$

The Sum–Product Algorithm: Normalising the Marginals

For a factor graph derived from a directed graph, the joint distribution $\Pr(D)$ is already correctly normalised

- Therefore, the marginals will be similarly normalised

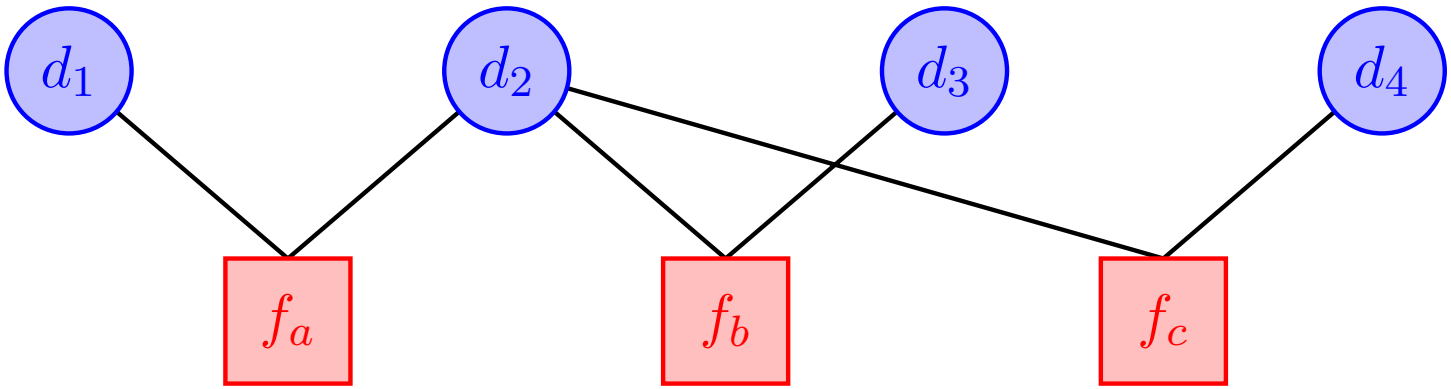
For an undirected graph, a factor(the partition function) $\frac{1}{Z}$ normalising $\Pr(D)$ is generally unknown

- The sum-product algorithm is run first with a non-normalised joint distribution in order to find the non-normalised marginals
- The factor $\frac{1}{Z}$ is then easily obtained by normalising any one of these marginals
  - Computationally efficient procedure: because the normalisation is done over a single variable rather than over the entire set of variables $D = (d_1, d_2, \ldots, d_n)$

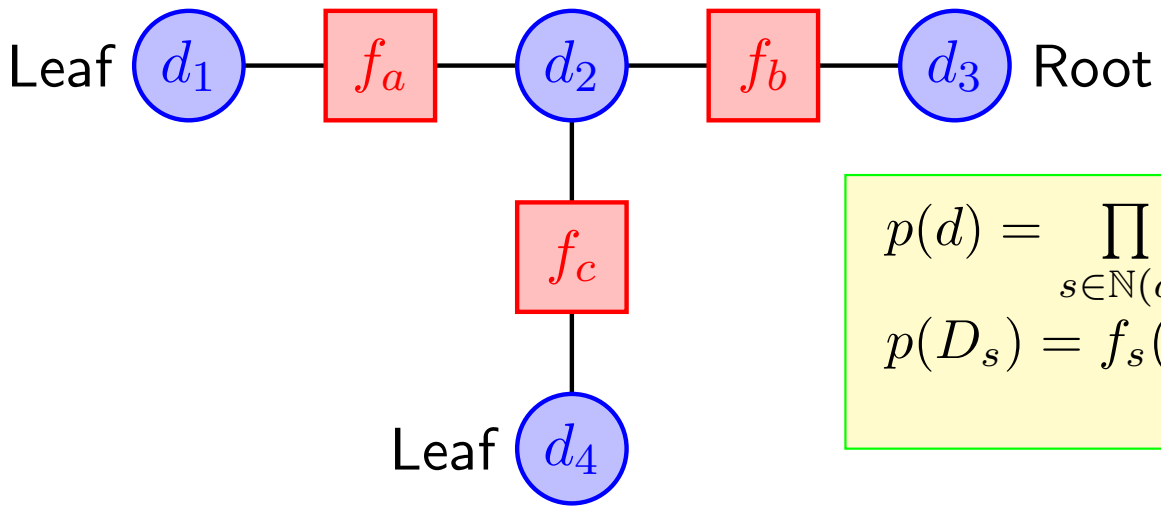
The Sum–Product Algorithm: An Example

The non-normalised joint distribution:

$$\Pr(d_1, d_2, d_3, d_4) = f_a(d_1, d_2) f_b(d_2, d_3) f_c(d_2, d_4)$$



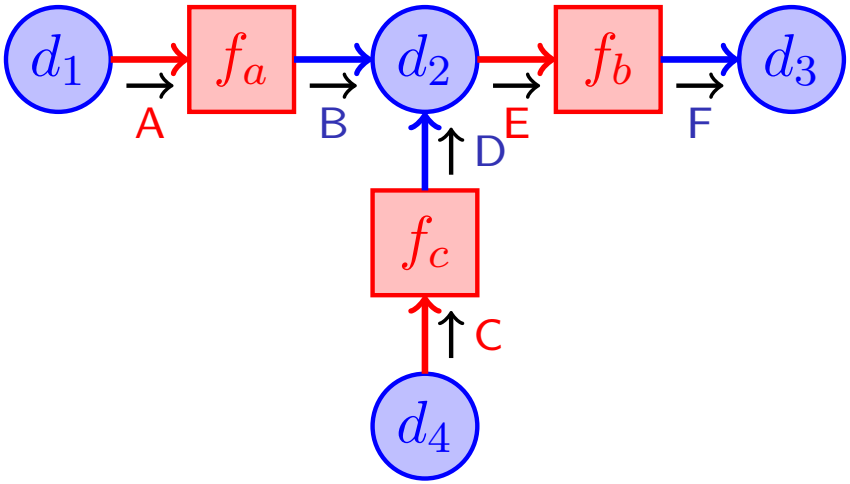
Tree structure:



$$p(d) = \prod_{s \in \mathbb{N}(d)} \mu_{f_s \to d}(d)$$

$$p(D_s) = f_s(D_s) \prod_{i \in \mathbb{N}(f_s)} \mu_{d_i \to f_s}(d_i)$$

Leaves–to–Root and Root-to-Leaves Flows of Messages



A $\mu_{d_1 \to f_a} = 1$

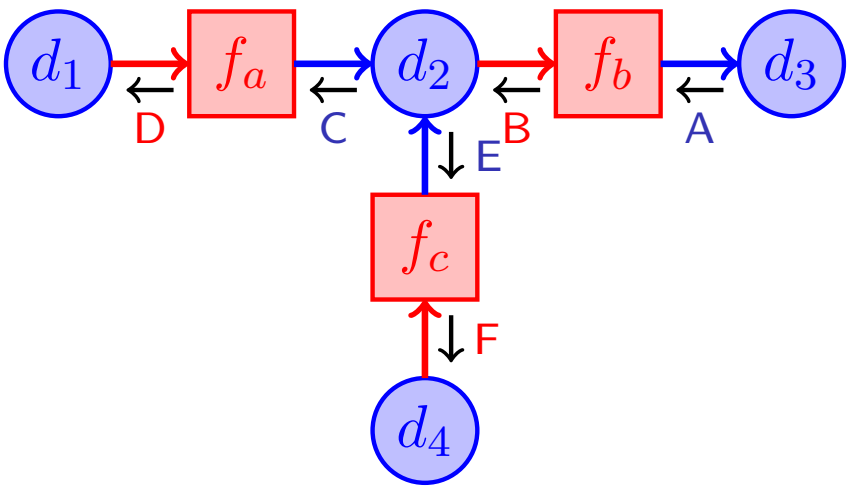
B $\mu_{f_a \to d_2}(d_2) = \sum_{d_1} f_a(d_1, d_2)$

C $\mu_{d_4 \to f_c} = 1$

D $\mu_{f_c \to d_2}(d_2) = \sum_{d_4} f_c(d_2, d_4)$

E $\mu_{d_2 \to f_b}(d_2) = \mu_{f_a \to d_2}(d_2)\mu_{f_c \to d_2}(d_2)$

F $\mu_{f_b \to d_3}(d_3) = \sum_{d_2} f_b(d_2, d_3)\mu_{d_2 \to f_b}(d_2)$



A $\mu_{d_3 \to f_b} = 1$

B $\mu_{f_b \to d_2}(d_2) = \sum_{d_3} f_b(d_2, d_3)$

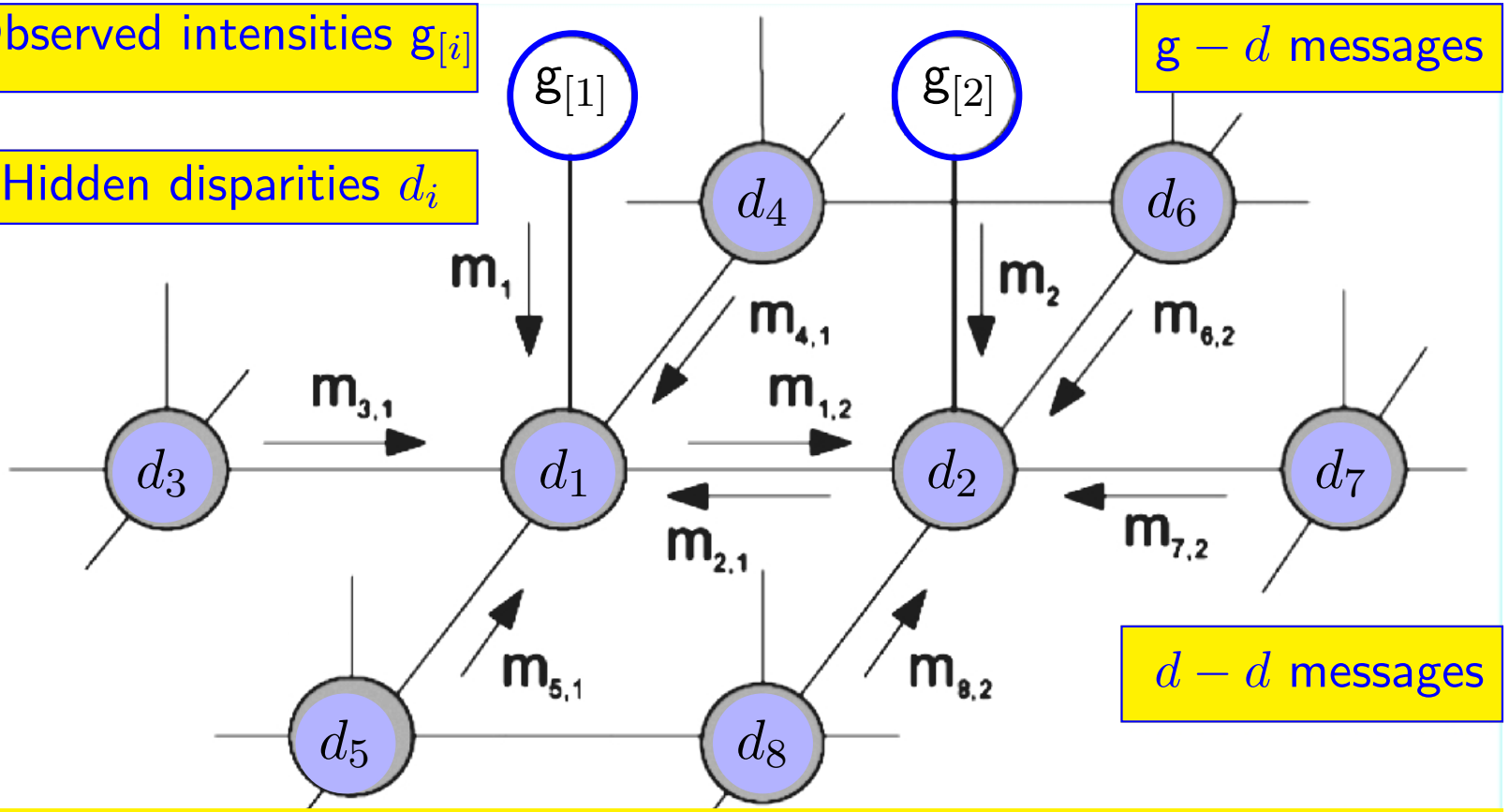
C $\mu_{d_2 \to f_a}(d_2) = \mu_{f_b \to d_2}(d_2)\mu_{f_c \to d_2}(d_2)$

D $\mu_{f_a \to d_1}(d_1) = \sum_{d_2} f_a(d_1, d_2)\mu_{d_2 \to f_a}(d_2)$

E $\mu_{d_2 \to f_c}(d_2) = \mu_{f_a \to d_2}(d_2)\mu_{f_b \to d_2}(d_2)$

F $\mu_{f_c \to d_4}(d_4) = \sum_{d_2} f_c(d_2, d_4)\mu_{d_2 \to f_c}(d_2)$

MRF Model for Stereo Matching



Posterior model: $\Pr(D|\mathsf{G}) \propto \prod_i \left(\psi_i(d_i, g_{[i]}) \prod_{j \in \mathbb{N}_i} \varphi_{j,i}(d_j, d_i) \right)$

J.Sun e.a.: Stereo Matching Using Belief Propagation, *IEEE T.PAMI*, vol.25(7), 787–800, 2003

Loopy Belief Propagation

The sum-product algorithm provides efficient and exact inference in tree-structured graphs, but graphs in many practical applications have loops

- The message passing framework can be generalised to arbitrary graphs topologies to give an exact inference
- But in the case of discrete variables, its computational complexity grows exponentially with the maximum number of interdependent variables

Thus it is not feasible to use the exact inference: effective approximate methods such as **loopy belief propagation** (LBP) has to be exploited

- LBP is possible because the message passing rules for the sum-product algorithm are purely local

Loopy Belief Propagation

Because of the graph cycles, the information can flow many times around the graph

- For some models, the LBP algorithm converges, whereas for others it will not

Message passing schedule in LBP:

- Each node should send a message across the link from the node after receiving messages from all other links
- Transmit only **pending** messages: after a node receives a message on one of its links
- Convergence to the exact marginal if there are no more pending messages

Energy Minimisation: The Min–Sum BP Algorithm

- Factor graph expression $D^* = \arg\min_D \{E(D) = \log \Pr(D)\}$: $\min_{d_1,\ldots,d_N} \sum_{c\in\mathbb{C}} f_c(d_j : j \in S_c)$ where $S_c \subset \mathbb{N} = \{1,\ldots,N\}$ is a subset of the nodes
- Then interchange the minimisation and sums for obtaining a computationally efficient minimisation algorithm:

$$\min_{d_j:\, j\in\mathbb{N}} \sum_{c\in\mathbb{C}} f_c(d_j : j \in S_c)$$

$$\Rightarrow \sum_{i\in\mathbb{N}} \sum_{\substack{\text{neighbours of} \\ i\in S_c}} \min_{d_j:\, j\in S_c} f_{\mathbf{c}}(d_i, d_j : j \in S_c \backslash i)$$

- Special case: **Belief propagation** for exact computations of the minimum energy on digraphs without loops
  - An implicit decision D^* contrary to an explicit one in dynamic programming: ambiguities for multiple equivalent cases