Master in Computer Vision Barcelona

Module: Optimization methods in CV
Inference algorithms I: BP and LBP
Lecturer: O. Ramos Terrades

Goals of this Lecture & Tools

Goal

- Belief propagation (BP): sum-prod
- Loopy belief propagation (LBP): sum-prod

Tools

- UGM library at `http://www.cs.ubc.ca/~schmidtm/Software/UGM.html`
- OpenGM at `http://hci.iwr.uni-heidelberg.de/opengm2/`
  - Matlab and Python
  - Public benchmark at `http://hci.iwr.uni-heidelberg.de/opengm2/?l0=benchmark`

Outline

Representation

Inference algorithms
- Belief Propagation
- Loopy belief propagation (LBP)

Representation: factor graphs

- Interactions defined on maximal cliques
- Factors defined on cliques
- Join pdfs factorizes on cliques:

$$p(x) = \frac{1}{Z} \prod_{\alpha} \phi_\alpha(x_\alpha) \qquad (1)$$

where $Z = \int p(x)dx$ is the partition function

Partition function: Z

$x = (x_1, \ldots, x_N)$, x_i discrete r.v. with domain $\{0, \ldots, L-1\}$:

$$Z = \sum_{l_1=0}^{L-1} \cdots \sum_{l_N=0}^{L-1} p(x_1 = l_1, \ldots, x_1 = l_N) \qquad (2)$$

Change of notation

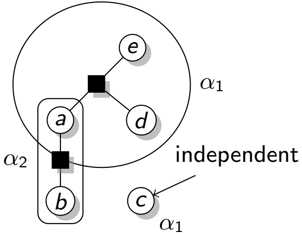
- rvs: $x_1, \ldots, x_N$. subset of rvs: $x_{i_1}, \ldots, x_{i_s}$, $1 \leq i_1 < \cdots < i_s \leq N$
- define index set: $\alpha = \{i_1, \ldots, i_s\}$
- complementary index set: $\bar{\alpha} = \{1, \ldots, N\} \setminus \alpha$

$$\sum_{\bar{x}_\alpha} p(x) \Rightarrow p(x_\alpha) = \int p(x) d\bar{x}_\alpha \qquad (3)$$

Representation: feature functions

$$p(a, b, c, d, e) = \frac{1}{Z}\phi_{\alpha_1}(a, e, d)\phi_{\alpha_2}(a, b)\phi_{\alpha_3}(c)$$

- Factor functions are the model *parameters*
- We can define it by means of *feature functions*: $f_{i,l}(x_{\alpha_i})$
- Feature functions are sufficient statistics



feature functions

$$\log \phi_\alpha(x_\alpha) = \sum_l \theta_{\alpha,l} f_l(x_\alpha) \qquad (4)$$

$\theta_{\alpha,l}$ model parameter

Example (feature functions)

- Data value: $x \in \mathbb{R}$.
- Indicator function: $\mathbb{1}_{\{x=a\}}(x)$, $a, x \in \mathbb{N}$
- L^p-norm: $\| x \|_p^p$, $x \in \mathbb{R}^m$
- L^p-distance: $\| x - y \|_p^p$, $x, y \in \mathbb{R}^m$

Representation: log-linear models

Log-linear model

$$p(x) = \frac{1}{Z} \exp \left\{ \sum_{\alpha,l} \theta_{\alpha,l} f_{\alpha,l}(x_\alpha) \right\} \qquad (5)$$

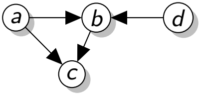
Energy

$$E(x) = -\sum_{\alpha,l} \theta_{\alpha,l} f_{\alpha,l}(x_\alpha) \qquad (6)$$

$$p(x) = \frac{1}{Z} \exp\{-E(x)\} \qquad (7)$$

- How can we estimate model parameters? Answer: learning algorithms.
- Solve MAP inference problem $\Leftrightarrow$ minimize energy E
  $\Rightarrow$ We can apply methods from the first part of this module

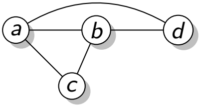
Representation: Bayes networks and factor graphs

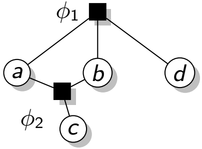


$$p(a,b,c,d) = p(c|a,b)p(a)p(b|a,d)p(d)$$

“Moralization”=marrying parents

- $p(c|a,b)$ a factor have to contain rvs a, b and c.
- $p(b|a,d)$ a factor have to contain rvs a, b and d.

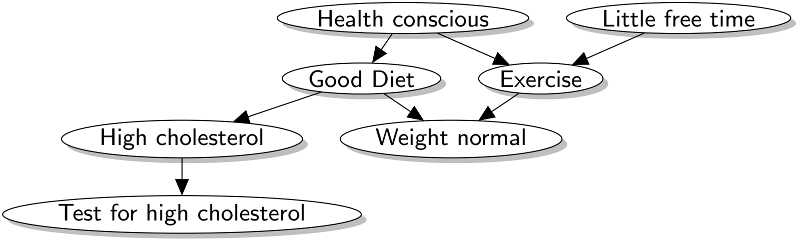




$$p(a,b,c,d) = \frac{1}{Z}\phi(c,a,b)\phi(b,a,d) \qquad p(a,b,c,d) = \frac{1}{Z}\phi_1(c,a,b)\phi_2(b,a,d)$$

Representation: exercise

Moralize the following Bayes network and draw the associated factor graph:



Outline

Representation

Inference algorithms

- Belief Propagation
- Loopy belief propagation (LBP)

Belief Propagation: sum-product

Goal:

Given a factor graph representing some directed, or undirected, model compute marginals $p(x_n)$.

Assumptions:

- acyclic factor graph (tree)
- all factor functions are known (parameters)
- discrete variables

Benefits:

- exact inference of $p(x_n)$: $O(NK^2)$
- all marginals $p(x_n)$, $n = 1, \ldots, N$ in 2 computations

Belief Propagation: sum-product

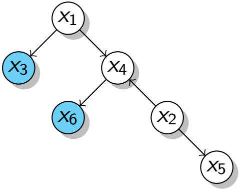
Main idea:

$$p(x_n) = \int p(x) d\bar{x}_n = \int \prod_s \phi_s(x_s) dx_{s\setminus\{n\}} \quad (8)$$

ϕ_s factor function. s index set of variables connected to factor s

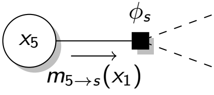
- interchange $\int$ and $\prod$ for those x_s without x_n
- express the algorithm as passing messages between graph nodes

1. Consider x_n as the root of a graph

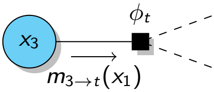


Belief Propagation: sum-product

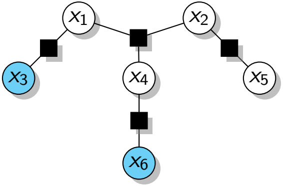
2. Initialize variable messages from (tree) leaves :



$$m_{5\to s} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

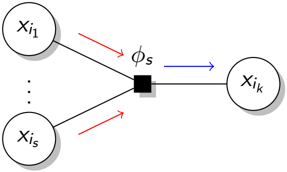


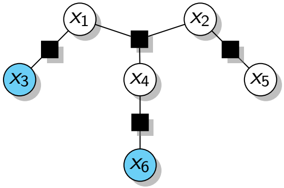
$$m_{3\to t} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$



Belief Propagation: sum-product

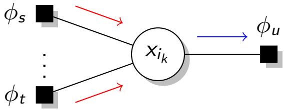
3. Send messages from factors to variables:

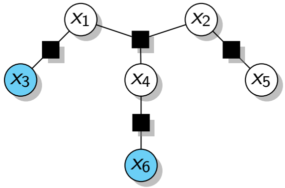




$$m_{i_k \leftarrow s}(x_{i_k}) = \int \phi_s(x_s) \prod_{j \in s \setminus \{i_k\}} m_{j \rightarrow s}(x_j) dx_{s \setminus \{i_k\}}$$

4. Send messages from variables to factors :





$$m_{i_k \to u}(x_{i_k}) = \prod_{\substack{t \ni i_k \\ t \neq u}} m_{i_k \leftarrow t}(x_{i_k})$$

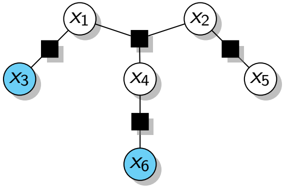
Belief Propagation: sum-product

5. repeat steps 3 and 4 until all variables have received and sent messages.

6. **Belief** estimation:

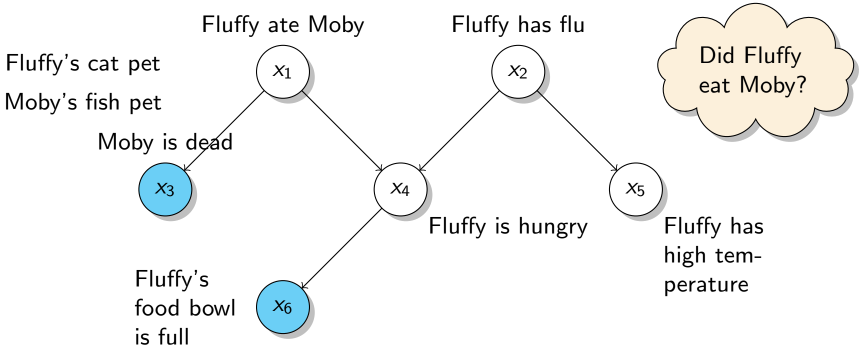
$$b(x_n) = \frac{1}{Z_n} \prod_{s \ni n} m_{n \leftarrow s}(x_n)$$

where Z_n is the partition function associated to belief $b_n(x_n)$.



Belief Propagation: sum-product

A silly numerical example*



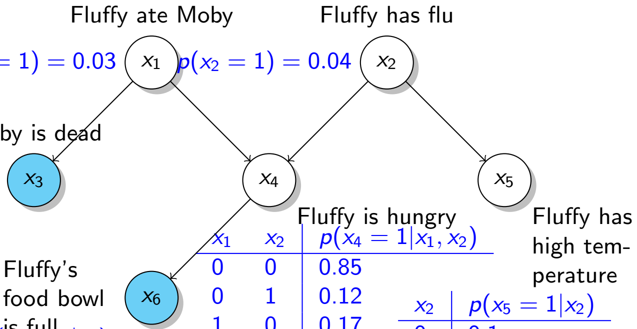
$$p(x) = p(x_3|x_1)p(x_6|x_4)p(x_4|x_1,x_2)p(x_5|x_2)p(x_1)p(x_2)$$

* Machine learning course 4F13 by Z. Grahramani and C.E. Rasmussenn, Dep. of Engineering, University of Cambridge,

http://mlg.eng.cam.ac.uk/teaching/4f13/0708/

Belief Propagation: sum-product

Conditional probabilities:



x_1	$p(x_3 = 1 \mid x_1)$
0	0.16
1	1

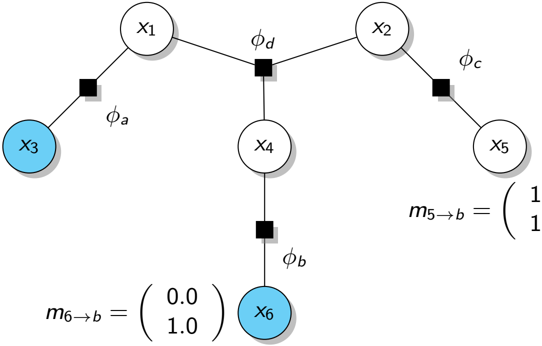
x_1	x_2	$p(x_4 = 1 \mid x_1, x_2)$
0	0	0.85
0	1	0.12
1	0	0.17
1	1	0.02

x_2	$p(x_5 = 1 \mid x_2)$
0	0.1
1	0.9

x_4	$p(x_6 = 1 \mid x_4)$
0	0.95
1	0.1

Belief Propagation: sum-product

1. Convert the Bayes net to a factor graph.
2. Initialize variable messages from leaves :



$$m_{3\to a} = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}$$

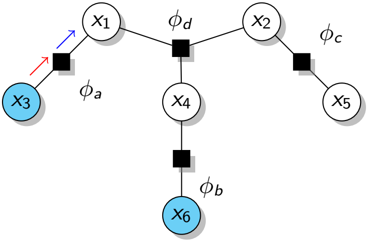
$$m_{5\to b} = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}$$

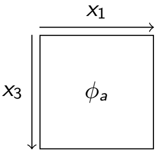
$$m_{6\to b} = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}$$

Belief Propagation: sum-product

3. propagate messages from factors to variables:

$$\phi_a(x_1, x_3) = \begin{pmatrix} 0.84 & 0.0 \\ 0.16 & 1.0 \end{pmatrix}$$



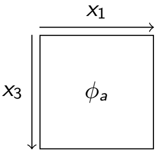


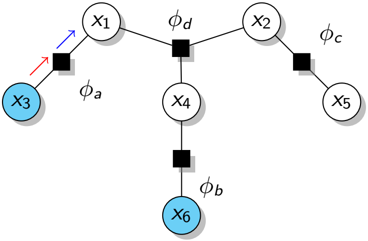
$$m_{1\leftarrow a} = \int \phi_a(x_1, x_3) m_{3\rightarrow a}(x_3) dx_3 =$$

Belief Propagation: sum-product

3. propagate messages from factors to variables:

$$\phi_a(x_1, x_3) = \begin{pmatrix} 0.84 & 0.0 \\ 0.16 & 1.0 \end{pmatrix}$$



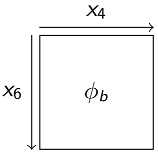


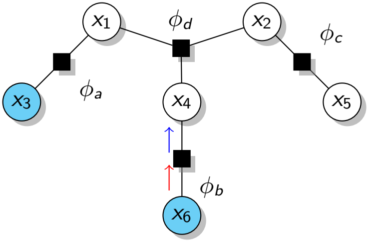
$$m_{1 \leftarrow a} = \int \phi_a(x_1, x_3) m_{3 \rightarrow a}(x_3) dx_3 = \begin{pmatrix} 0.84 & 0.16 \\ 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} = \begin{pmatrix} 0.16 \\ 1.0 \end{pmatrix}$$

Belief Propagation: sum-product

3. propagate messages from factors to variables:

$$\phi_b(x_4, x_6) = p(x_4|x_6) = \begin{pmatrix} 0.05 & 0.9 \\ 0.95 & 0.1 \end{pmatrix}$$



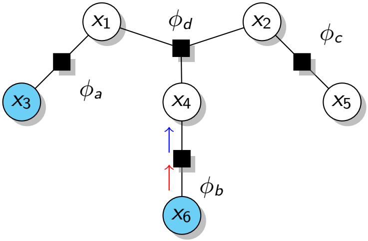


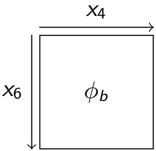
$$m_{4\leftarrow b} = \int \phi_b(x_4, x_6) m_{6\rightarrow b}(x_6) dx_6 =$$

Belief Propagation: sum-product

3. propagate messages from factors to variables:

$$\phi_b(x_4, x_6) = p(x_4|x_6) = \begin{pmatrix} 0.05 & 0.9 \\ 0.95 & 0.1 \end{pmatrix}$$



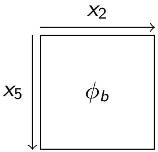


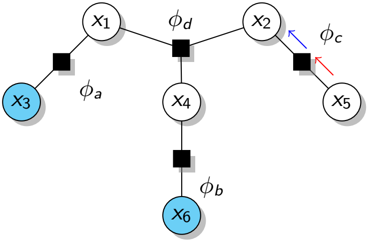
$$m_{4\leftarrow b} = \int \phi_b(x_4, x_6) m_{6\rightarrow b}(x_6) dx_6 = \begin{pmatrix} 0.05 & 0.95 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix} = \begin{pmatrix} 0.95 \\ 0.1 \end{pmatrix}$$

Belief Propagation: sum-product

3. propagate messages from factors to variables:

$$\phi_c(x_2, x_5) = p(x_5|x_2) = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$



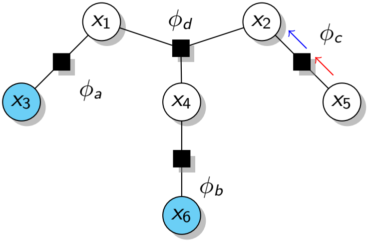


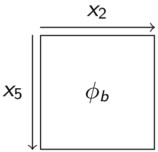
$$m_{2\leftarrow c} = \int \phi_c(x_2, x_5) m_{5\rightarrow c}(x_5) dx_5 =$$

Belief Propagation: sum-product

3. propagate messages from factors to variables:

$$\phi_c(x_2, x_5) = p(x_5|x_2) = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$$





$$m_{2\leftarrow c} = \int \phi_c(x_2, x_5) m_{5\rightarrow c}(x_5) dx_5 = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix} \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}$$

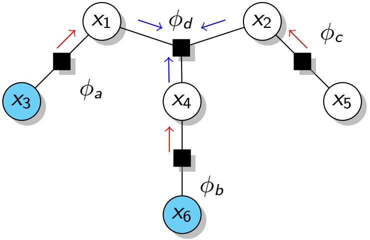
Belief Propagation: sum-product

4. propagate message from variables to factors:

$$m_{1\to d} = m_{1\leftarrow a} = \begin{pmatrix} 0.16 \\ 1.0 \end{pmatrix}$$

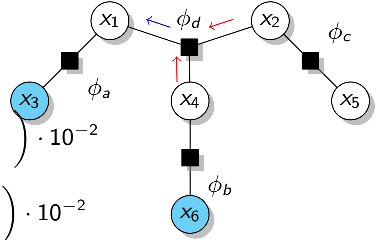
$$m_{4\to d} = m_{4\leftarrow b} = \begin{pmatrix} 0.95 \\ 0.1 \end{pmatrix}$$

$$m_{2\to d} = m_{2\leftarrow c} = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}$$



Belief Propagation: sum-product

5. Repeat steps 2 and 3: propagate messages from factors to variables:



$$\phi_d(0, x_2, x_4) = \begin{pmatrix} 13.96 & 3.41 \\ 79.15 & 0.46 \end{pmatrix} \cdot 10^{-2}$$

$$\phi_d(1, x_2, x_4) = \begin{pmatrix} 2.39 & 0.12 \\ 0.49 & 0.24 \end{pmatrix} \cdot 10^{-2}$$

$$\phi_d(x_1, x_2, x_4) = p(x_4|x_1, x_2)p(x_1)p(x_2)$$

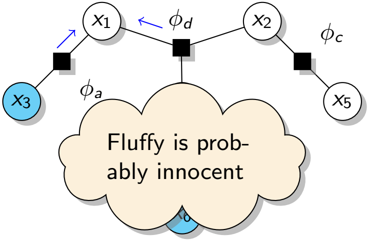
$$m_{1\leftarrow d} = \int \phi_d(x_1, x_2, x_4) m_{2\rightarrow d}(x_2) m_{4\rightarrow d}(x_4) dx_2 dx_4 = \begin{pmatrix} 0.25 \\ 0.02 \end{pmatrix}$$

$$m_{1\leftarrow d}(0) = \begin{pmatrix} 0.95 & 0.1 \end{pmatrix} \begin{pmatrix} 13.96 & 3.41 \\ 79.15 & 0.46 \end{pmatrix} \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} \cdot 10^{-2} = 0.25$$

Belief Propagation: sum-product

6. Estimate belief of x_1:

$$b_1(x_1) = \frac{1}{Z} m_{1\leftarrow a}(x_1) \circ m_{1\leftarrow d}(x_1) =$$
$$= \frac{1}{Z} \begin{pmatrix} 0.16 \\ 1.0 \end{pmatrix} \circ \begin{pmatrix} 0.25 \\ 0.02 \end{pmatrix} =$$
$$= \begin{pmatrix} 0.62 \\ 0.38 \end{pmatrix}$$



Belief Propagation: sum-product

To compute marginals for *all* rvs: $x_n, 1, \ldots, N$:

1. pick any node as a root
2. propagate messages from leaves to root
3. the root has received a message from all children $\rightarrow$ it can sent back a message to them
4. Children have also receive a message from all neighboring rvs $\rightarrow$ they can sent a message away from the root
5. repeat 4. until *all* leaves have receive a message

A message has passed in both directions across every link $\rightarrow O(2NK^2)$

Belief Propagation: Summary

Advantages of BP:

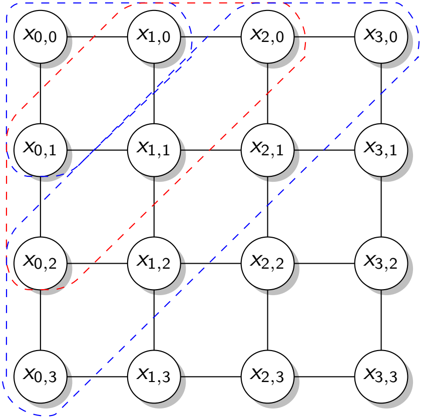
- *simplicity* of message passing
- *exact* inference on tree-structured graphs
- time and memory $O(NK^2)$ *linear* in the number of nodes

Shortcomings:

- graphs with *loops*: approximate solution. Convergence?
- $O(NK^2)$ *quadratic* in the number of labels
- $O(NK^c)$, c =size of largest clique : precludes efficiency on *high order* models

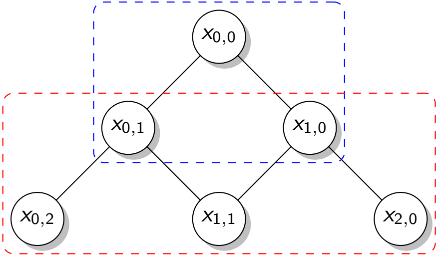
Inference algorithms: graphs with loops

- Can we estimate exact marginals on loopy graphs?
  - **clustering variables**: Junction tree, convex energies

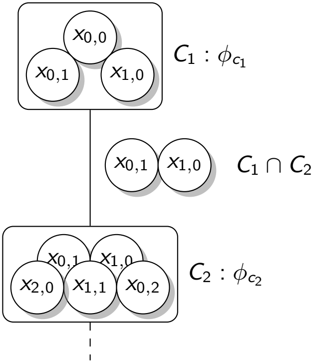


$$p(x) = \frac{1}{Z} \prod_{i,j} \phi_{i,j}(x_i, x_j) \tag{9}$$

Inference algorithms: junction tree



Junction tree



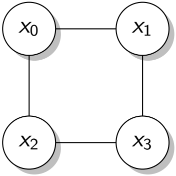
Applying usual definitions of conditional probabilities:

$$p(x) = \frac{p(x_{0,0}, x_{0,1}, x_{1,0})p(x_{0,1}, x_{1,0}, x_{2,0}, \dots)}{p(x_{0,1}, x_{1,0})} \cdots \tag{10}$$

Inference algorithms: exercise

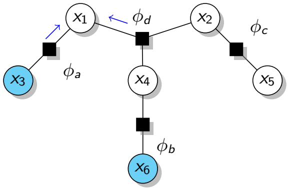
- Given the PGM of the image. Show that p factorise as:

$$p(x_0, x_2, x_3, x_4) = \frac{p(x_0, x_1, x_2)p(x_2, x_3, x_4)}{p(x_1)p(x_2)}$$

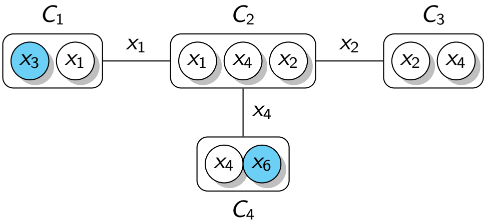


Inference algorithms: cluster graph

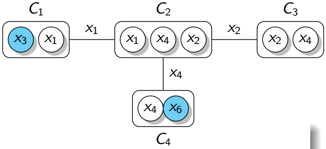
factor graph



Cluster graph



Clique tree



Running intersection property

$\mathcal{T}$ a cluster tree; $(\mathcal{V}_\mathcal{T}, \mathcal{E}_\mathcal{T})$ nodes and edges, respectively. $\mathcal{T}$ has the *running intersection property* if, whenever there is a variable x such that $x \in C_i$ and $x \in C_j$, then x is also in every cluster in the (unique) path in $\mathcal{T}$ between C_i and C_j

Clique tree

A cluster tree that satisfies the *running intersection property* is called a **clique tree**. Sometimes also called a **junction tree**

Inference algorithms: Junction tree

- Junction tree algorithm: minimize energy
- messages from *outer* factor to inner factors & vice versa
- Exact inference if graph contains **loops**
- Convergence is **guarantee**
- Complexity exponential with the biggest cluster size $\Rightarrow$ It can not be applied in many real situations

Loopy belief propagation (LBP): algorithm

Initialize messages: $m_{n \to \alpha} = 1$
Repeat:
    for all node variables x_n
        for all factors ϕ_α containing x_n:
            send a message from factor ϕ_α to x_n : $m_{n \leftarrow \alpha}$
        endfor
    endfor
    for all factors ϕ_α:
        for all variables x_n in ϕ_α:
            send a message from variables x_n to ϕ_α: $m_{n \to \alpha}$
        endfor
        Update $b_\alpha(x_\alpha)$.
    endfor
While not converged

Loopy belief propagation (LBP): algorithm

belief at factor level α:

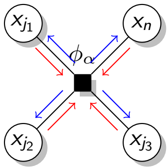
$$b_\alpha(x_\alpha) = \phi_\alpha(x_\alpha) \prod_{j \in \alpha} m_{j \to \alpha}(x_j) \tag{11}$$

- send a message from factor ϕ_α to x_n : $m_{n \leftarrow \alpha}$

$$m_{n \leftarrow \alpha}(x_n) = \int \phi_\alpha(x_\alpha) \prod_{j \in \alpha \setminus \{n\}} m_{j \to \alpha}(x_j) dx_{\alpha \setminus \{n\}} \tag{12}$$

$$= \frac{1}{m_{n \to \alpha}(x_n)} \int \phi_\alpha(x_\alpha) \prod_{j \in \alpha} m_{j \to \alpha}(x_j) dx_{\alpha \setminus \{n\}} \tag{13}$$

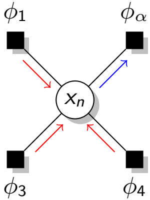
$$= \frac{\int b_\alpha(x_\alpha) dx_{\alpha \setminus n}}{m_{n \to \alpha}(x_n)} \tag{14}$$



Loopy belief propagation (LBP): algorithm

- send a message from variables x_n to ϕ_α:

$$m_{n\to\alpha}(x_n) = \prod_{k\in\alpha\setminus\{n\}} m_{k\leftarrow\alpha}(x_k) \qquad (15)$$



belief of rv x_n:

$$b_n(x_n) = \frac{1}{Z_n} \underbrace{\prod_{\alpha\ni n} \prod_{k\in\alpha\setminus\{n\}}}_{} m_{k\leftarrow\alpha}(x_k)^{\frac{1}{q_n-1}} = \frac{1}{Z_n} \prod_{\alpha\ni n} m_{k\leftarrow\alpha}(x_k)$$

$q_n - 1$ factors for each $k \neq n$,

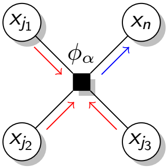
Inference algorithms: LBP vs Exact BP (tree)

Factor function

$$\phi_\alpha(x_\alpha) = \exp\{-\sum_l \theta_{\alpha,l} f_{\alpha,l}(x_\alpha)\} \tag{16}$$

LBP: Message from factor to variable

$$m_{n\leftarrow\alpha}(x_n) = \frac{\int b_\alpha(x_\alpha) dx_{\alpha\backslash n}}{m_{n\rightarrow\alpha}(x_n)} \tag{17}$$



Exact BP (chain & tree) & LBP

$$m_{n\leftarrow\alpha}(x_n) = \int \phi_\alpha(x_\alpha) \prod_{j\in\alpha\backslash\{n\}} m_{j\rightarrow\alpha}(x_j) dx_{\alpha\backslash\{n\}} \tag{18}$$

Loopy belief propagation (LBP): summary

- LBP algorithm $\sim$ minimize free energy
- Messages from variables to factors & vice versa the same than in BP algorithm
- Approximate if graph contains **loops**
- Convergence is **not guarantee**
- variants of LBP $\Rightarrow$ messages are sent differently

Exercise: Maximum entropy optimization [Jay57]

From where Log-linear model comes?

- Maximizing entropy

$$\operatorname*{argmax}_p H(p) = \operatorname*{argmax}_p - \int p(x) \log p(x) dx \tag{19}$$

- subject to:

$$\mu_{l,\alpha} = \int f_l(x_\alpha) p(x_\alpha) dx \qquad \text{(matching moment)}$$

$$1 = \int p(x) dx \qquad \text{(normalization)}$$

$\mu_{l,\alpha}$ empirical moments: $\frac{1}{N} \sum_n f_{l,\alpha}(x_\alpha^{(n)})$

Bibliography I

[Jay57] E. T. Jaynes.
Information theory and statistical mechanics.
Physical Review, 106(4):620–630, 1957.