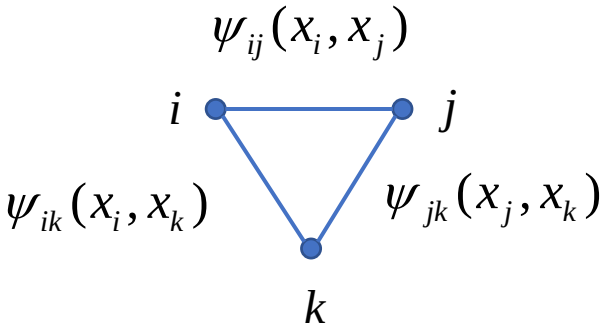
Deterministic Algorithms

$$P(\{x_i\}) = \frac{1}{Z}\prod_i \psi_i(x_i)\prod_{i,j}\psi_{ij}(x_i, x_j)$$ MRF



Suppose, we want to estimate the marginal distributions $\prod_i P_i(x_i)$

or $\mathbf{x}^* = \arg\max_{\{x_i\}} P(\{x_i\})$

If the graph is a polytree (i.e. no closed loops), then we can use dynamic programming

→ This cannot be applied if the graph has closed loops.

Here we describe a popular algorithm – **belief propagation** – that gives correct results on polytrees, and empirically good approximations most of time on graphs

And we will show the relation to MCMC

JOHNS HOPKINS UNIVERSITY

Belief Propagation (BP)

proceeds by passing messages between the graph nodes

$m_{ij}(x_j : t)$:message that node i passes to node j to affect state x_j

The messages gets updated as follows:

$$m_{ij}(x_j : t+1) = \sum_{x_i} \psi_{ij}(x_i, x_j)\psi_i(x_i)\prod_{k \neq j} m_{ki}(x_i : t)$$ Sum-product rule

Alternative: the max-product

$$m_{ij}(x_j : t+1) = \max_{x_i} \left\{ \psi_{ij}(x_i, x_j)\psi_i(x_i)\prod_{k \neq j} m_{ki}(x_i : t) \right\}$$

If the algorithm converges (it may not), then we compute approximations to the marginals:

$$b_i(x_i) \propto \psi_i(x_i)\prod_k m_{ki}(x_i)$$

$$b_{ij}(x_i, x_j) \propto \psi_i(x_i)\psi_{ij}(x_i, x_j)\psi_j(x_j)\prod_{k \neq j} m_{ki}(x_i)\prod_{l \neq i} m_{lj}(x_j)$$

Belief Propagation

BP (sum-product) was first proposed by Judea Pearl (C.S. UCLA) for performing inference on polytrees

The max-product algorithm was proposed earlier by Gallagher

The application was developed in the 1990's for decoding problems
- goal was to achieve Shannon's bound on information transmission

Experimentally, it was shown that BP usually converges to reasonable approximations
- Full understanding of when & why it converges is an open problem

On polytree, it is similar to dynamic programming – so in a sense, it is the way to extend DP to graphs with closed loops

Example of BP (sum-product)

$$\mathbf{x} = (x_1, x_2, x_3, x_4)$$

$$P(\mathbf{x}) = \frac{1}{Z}\psi_{12}(x_1, x_2)\psi_{23}(x_2, x_3)\psi_{34}(x_3, x_4)$$

Messages:

$m_{12}(x_2)$, from node 1 to node 2

$m_{21}(x_1)$, from node 2 to node 1

$m_{23}(x_3)$, from node 2 to node 3

$m_{32}(x_2)$, from node 3 to node 2

$m_{34}(x_4)$, from node 3 to node 4

$m_{43}(x_3)$, from node 4 to node 3

Update rule (from message passing equation)

$$m_{12}(x_2 : t+1) = \sum_{x_1} \psi_{12}(x_1, x_2) \quad \rightarrow \text{boundary}$$

$$m_{21}(x_1 : t+1) = \sum_{x_2} \psi_{12}(x_1, x_2) m_{32}(x_2 : t)$$

$$m_{23}(x_3 : t+1) = \sum_{x_2} \psi_{23}(x_2, x_3) m_{12}(x_2 : t)$$

$$m_{32}(x_2 : t+1) = \sum_{x_3} \psi_{23}(x_2, x_3) m_{43}(x_3 : t)$$

$$m_{34}(x_4 : t+1) = \sum_{x_3} \psi_{34}(x_3, x_4) m_{23}(x_3 : t)$$

$$m_{43}(x_3 : t+1) = \sum_{x_4} \psi_{34}(x_3, x_4) \quad \rightarrow \text{boundary}$$

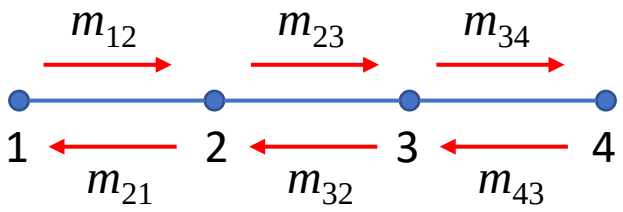
Many ways to run BP

(1) Update all messages in parallel

Note: BP can be parallelized but Dynamic Programming cannot)

(2) Start at boundaries

i.e.,



Calculate $m_{12}(x_2)$, then $m_{23}(x_3)$, then $m_{34}(x_4)$

➔ Forward pass (like Dynamic Programming)

$m_{43}(x_3)$, then $m_{32}(x_2)$, then $m_{21}(x_1)$

➔ Backward pass (like backward pass of DP)

Then read off estimates of unary marginals

$$b_1(x_1) = \frac{1}{Z_1} m_{21}(x_1), \quad Z_1 = \sum_{x_1} m_{21}(x_1) \text{ :normalization}$$

$$b_2(x_2) = \frac{1}{Z_2} m_{12}(x_2) m_{32}(x_2), \quad Z_2 \text{ :normalization}$$

$$b_3(x_3) = \frac{1}{Z_3} m_{23}(x_3) m_{43}(x_3), \quad Z_3 \text{ :normalization}$$

$$b_4(x_4) = \frac{1}{Z_4} m_{34}(x_4), \quad Z_4 \text{ :normalization}$$

$$b_{12}(x_1, x_2) = \frac{1}{Z_{12}} \psi_{12}(x_1, x_2) m_{32}(x_2), \quad \text{and so on}$$

Many ways to run BP

Alternatively, initialize the m's to take
- an initial value – e.g. $m_{ij}(x_j)=1$ for all i, j
- and update the messages in any order
  - will still converge for graph with no closed loops
- Then estimates (beliefs) will be the true marginals

$$\text{e.g.} \quad b_1(x_1) = \sum_{x_2,x_3,x_4} P(x_1, x_2, x_3, x_4)$$

$$b_{12}(x_1, x_2) = \sum_{x_3,x_4} P(x_1, x_2, x_3, x_4)$$

But, BP will often converge to a good approximation to the marginals for graphs which do not have closed loops
- This was discovered in the late 1980's

JOHNS HOPKINS UNIVERSITY

Advantages of BP over DP

(1) BP will converges (approximates) for many graphs with closed loops

(2) BP is parallelizable (nice if you have a parallel computer or GPU)

An alternative way to consider BP

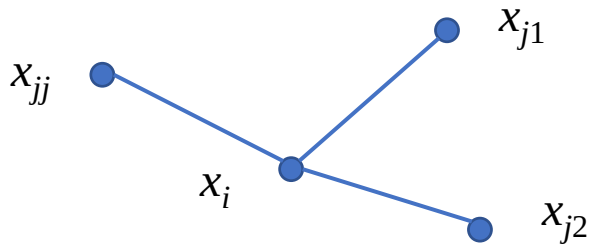
Make local approximations to the local distribution (BP w/o messages)

$$B(x_i, \mathbf{x}_{N(i)}) = \frac{1}{Z} b_i(x_i) \prod_{j \in N(i)} \frac{b_{ij}(x_i, x_j)}{b_i(x_i)}$$

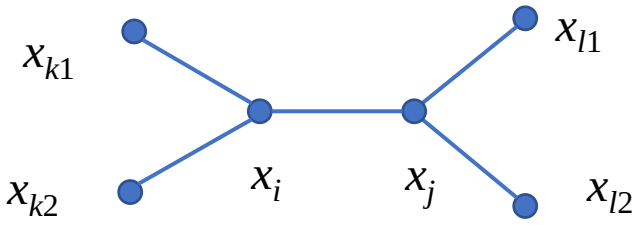
If the $b_{ij}(x_i, x_j)$ & $b_i(x_i)$ are the true marginal distribution, then $\frac{b_{ij}(x_i, x_j)}{b_i(x_i)} = b(x_j \mid x_i)$

An alternative way to consider BP

Cut
the rest of
the graph



$$B(x_i, x_j, \mathbf{x}_{N(i)}, \mathbf{x}_{N(j)}) = \frac{1}{Z_{ij}} b_{ij}(x_i, x_j) \prod_{k \in N(i)/j} \frac{b_{ik}(x_i, x_k)}{b_i(x_i)} \prod_{l \in N(j)/i} \frac{b_{jl}(x_j, x_l)}{b_j(x_j)}$$



Update Rule: Marginalization

$$b_{ij}(x_i, x_j : t+1) = \sum_{\mathbf{x}_{N(i,j)}} B(x_i, x_j, \mathbf{x}_{N(i)}, \mathbf{x}_{N(j)} : t)$$

$$b_{ij}(x_i : t+1) = \sum_{\mathbf{x}_{N(i)}} B(x_i, \mathbf{x}_{N(i)} : t)$$

Provably equivalent to BP

How does this relate to MCMC?

Chapman-Kolmogorov ⟵A deterministic form of Gibbs sampling

$$M_{t+1}(\mathbf{x}) = \sum_{\mathbf{x}'} K(\mathbf{x} \mid \mathbf{x}')\mu_t(\mathbf{x}') \qquad (K\text{: Transition kernel})$$

This converges to fixed point distribution $\Pi(\mathbf{x})$ s.t $\sum_{\mathbf{x}'} K(\mathbf{x} \mid \mathbf{x}')\mu(\mathbf{x}') = \Pi(\mathbf{x})$

MCMC estimates $\Pi(\mathbf{x})$ by repeatedly sampling from $K(\mathbf{x}, \mathbf{x}')$

Recall that Gibbs Sampler $K_\gamma(\mathbf{x} \mid \mathbf{x}') = P(x_\gamma \mid x'_{N(\gamma)})S_{\mathbf{x}/\gamma, \mathbf{x}'/\gamma}$

Substituting the Gibbs sampler into the Chapman-Kolmogorov equation

$$\mu_{t+1}(\mathbf{x}_\gamma) = \sum_{\mathbf{x}_{N(\gamma)}} P(\mathbf{x}_r \mid \mathbf{x}'_{N(\gamma)})\mu_t(\mathbf{x}_{N(\gamma)})$$

Replace $\mu_{t+1}(\mathbf{x}_\gamma)$ by $\sum_{x_i} B(x_i, x_{N(\gamma)}), \quad \mathbf{x}_r = y_i$

or $\sum_{x_i, x_j} B(x_i, x_j, x_{N(i,j)}), \quad \mathbf{x}_\gamma = (x_i, x_j)$

Bethe Free Energy

It can be shown that the fixed point of BP correspond to extreme of the Bethe Free Energy

$$F[b] = \sum_{ij} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \ln \frac{b_{ij}(x_i, x_j)}{\psi_i(x_i)\psi_j(x_j)\psi_{ij}(x_i, x_j)} - \sum_i (n_i - 1) \sum_{x_i} b_i(x_i) \ln \frac{b_i(x_i)}{\psi_i(x_i)}$$

This leads an alternative class of algorithm which seek to directly minimize $F[b]$

These algorithms are more complex than BP, more time consuming, and do not always give better results

Note M. Wainwright defines a class of convex free energies similar to Bethe

Note Junction trees allows DP to be applied to same graphs with closed loops (see Lauritzen and Spiegelhalter).

A range of alternative algorithms

The original is meanfield (MFT)

Kulback-Leibler: Define $B(\mathbf{x}) = \prod_i b_i(x_i)$ $KL(B) = \sum_{\mathbf{x}} B(\mathbf{x}) \ln \frac{B(\mathbf{x})}{P(\mathbf{x})}$

Seek to find the $B(\mathbf{x})$ that minimizes $KL(B)$

Equivalent to $\sum_{i,j} \sum_{x_i, x_j} b_i(x_i) b_j(x_j) \ln \frac{b_i(x_i) b_j(x_j)}{\psi_i(x_i) \psi_{ij}(x_i, x_j) \psi_j(x_j)} - \sum_i (n_i - 1) \sum_{x_i} b_i(x_i) \ln \frac{b_i(x_i)}{\psi_i(x_i)}$

Compare to Bethe Free Energy: $b_{ij}(x_i, x_j) \rightarrow b_i(x_i) b_j(x_j)$

Minimizing KL(B) is not straightforward, but it is straightforward to find a local minima

- These approaches are significantly faster than MCMC, but MCMC works when these do not.