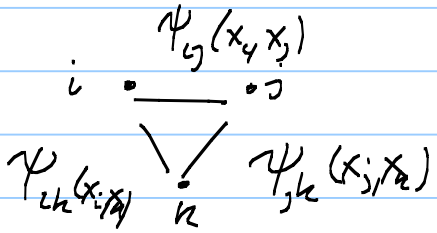
Lecture 25: Deterministic Algorithms

Note Title 6/4/2006

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

· MRF.



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

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

If the graph is a poly-tree (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 poly-trees, and empirically good approximations most of time on graphs.

And we will show the relation to MCMC.

page 2

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:

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

Called sum-product rule.

Alternative the max-product

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

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

The max-product algorithm was proposed earlier by Gallager.

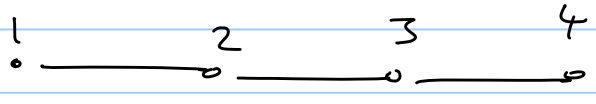
The application was developed in the 1990's for decoding problems. (goal to achieve Shannon's limit of information transmission).

Experimentally, it was shown that BP usually converges to reasonable approximates.

Full understanding of when & why it converges is an open problem.

On polytrees, 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)



$\underline{X} = (X_1, X_2, X_3, X_4)$

$$P(\underline{X}) = \frac{1}{Z} \psi_{12}(X_1, X_2)\, \psi_{23}(X_2, X_3)\, \psi_{34}(X_3, X_4)$$

Messages:

$m_{12}(X_2)$, from node 1 to node 2
$m_{21}(X_1)$, " " 2 " " 1
$m_{23}(X_3)$, " " 2 " " 3
$m_{32}(X_2)$, " " 3 " " 2
$m_{34}(X_4)$, " " 3 " " 4
$m_{43}(X_3)$, " " 4 " " 3

Update Rule (from message passing equations)

$$m_{12}(X_2; t+1) = \sum_{X_1} \psi_{12}(X_1, X_2) \quad \rightarrow \text{boundary}$$

$$m_{21}(X_1; t+1) = \sum_{X_2} \psi_{12}(X_1, X_2)\, m_{32}(X_2; t)$$

$$m_{23}(X_3; t+1) = \sum_{X_2} \psi_{23}(X_2, X_3)\, m_{12}(X_2; t)$$

$$m_{32}(X_2; t+1) = \sum_{X_3} \psi_{23}(X_2, X_3)\, m_{43}(X_3; t)$$

$$m_{34}(X_4; t+1) = \sum_{X_3} \psi_{34}(X_3, X_4)\, m_{23}(X_3; t)$$

$$m_{43}(X_3; t+1) = \sum_{X_4} \psi_{34}(X_3, X_4) \quad \rightarrow \text{boundary}$$

Page 5:

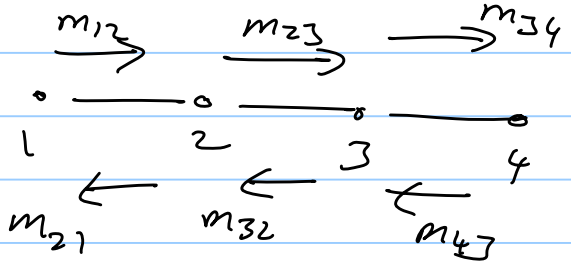
Many ways to run BP.

(1) Update all messages in parallel.

(note: BP can be parallelized – but Dynamic Programming cannot)

(2) Start at boundaries

→ ie.



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 (not like backward pass of DP)

Then read off estimates of unary marginals

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

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

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

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

$$b_{12}(x_1, x_2) = \frac{1}{Z_{12}} \psi_{12}(x_1, x_2) m_{32}(x_2),$$

and so on.

Page 6:

Alternately, 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.

The estimates (beliefs) will be the true marginals → e.g. $b_1(x_1) = \sum_{x_2,x_3,x_4} P(x_1,x_2,x_3,x_4)$

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

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 1990's)

Advantages of BP over DP

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

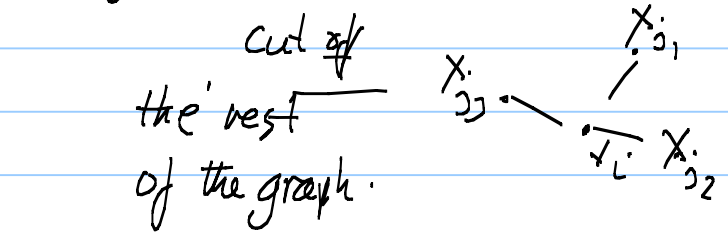
(2) BP is parallelizable (nice if you have a parallel computer or a GPU).

An alternative way to consider BP.

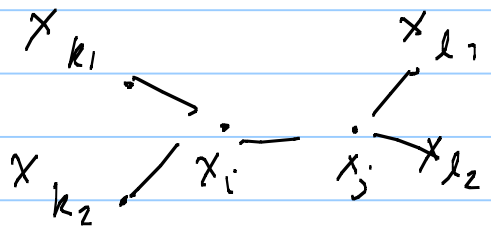
Make local approximations to the local distribution (BP without messages).

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

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



$$B(x_i, x_j, \underline{x}_{N(i)}, \underline{x}_{N(j)}) = \frac{1}{Z_{ij}} b_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} \frac{b_{ik}(x_i, x_k)}{b_i(x_i)} \prod_{\ell \in N(j) \setminus i} \frac{b_{j\ell}(x_j, x_\ell)}{b_j(x_j)}$$



Update Rule: Marginalization.

$$\begin{cases} b_{ij}(x_i, x_j : t+1) = \sum_{\underline{x}_{N(i,j)}} B(x_i, x_j, \underline{x}_{N(i)}, \underline{x}_{N(j)} : t) \\ b_i(x_i : t+1) = \sum_{\underline{x}_{N(i)}} B(x_i, \underline{x}_{N(i)} : t) \end{cases}$$

Provably equivalent to BP.

How does this relate to MCMC?

A deterministic form of Gibbs sampling.

Chapman-Kolmogorov

$$\mu_{t+1}(\underline{x}) = \sum_{\underline{x}'} K(\underline{x}|\underline{x}')\,\mu_t(\underline{x}')$$

K: transition kernel.

This converges to fixed point distribution $\pi(\underline{x})$ s.t. $\sum_{\underline{x}'} K(\underline{x}|\underline{x}')\,\pi(\underline{x}') = \pi(\underline{x})$.

MCMC estimates $\pi(\underline{x})$ by repeatedly sampling from $K(\underline{x}|\underline{x}')$.

Recall the Gibbs Sampler

$$K_v(\underline{x}|\underline{x}') = p(x_v|\underline{x}_{N(v)})\,\delta_{\underline{x}_{\setminus v},\underline{x}'_{\setminus v}}$$

Substituting the Gibbs sampler into the Chapman-Kolmogorov equations

$$\mu_{t+1}(\underline{x}_v) = \sum_{\underline{x}_{N(v)}} p(\underline{x}_v|\underline{x}_{N(v)})\,\mu_t(\underline{x}_{N(v)})$$

Replace $\mu_t(\underline{x}_{N(v)})$ by $\sum_{x_i} B(x_i, \underline{x}_{N(v)})$ $\quad \underline{x}_v = x_i$

or $\sum_{x_i,x_j} B(x_i,x_j,\underline{x}_{N(i,j)})$ $\quad \underline{x}_v = (x_i,x_j)$

This is BP

Bethe Free Energy.

It can be shown that the fixed points of BP correspond to extrema of the Bethe free energy.

$$F[b] = \sum_{ij} \sum_{x_i, x_j} b_{ij}(x_i, x_j) \log \frac{b_{ij}(x_i, x_j)}{\psi_i(x_i)\psi_j(x_j)\psi_{ij}(x_i, x_j)}$$

$$- \sum_i (n_i - 1) \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\psi_i(x_i)}$$

This leads an alternative class of algorithms 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: related work (Wainwright) defines a class of convex free energies similar to Bethe.

Note: junction trees allows PP to be applied to some graphs with closed loops.

A range of alternative algorithms, the original is mean field MFT

Kullback-Leibler: Define $B(\underline{x}) = \prod_i b_i(x_i)$

$$KL(B) = \sum_{\underline{x}} B(\underline{x}) \log \frac{B(\underline{x})}{P(\underline{x})}$$

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

Equivalent to:

$$\sum_{i,j} \sum_{x_i, x_j} b_i(x_i) b_j(x_j) \log \frac{b_i(x_i) b_j(x_j)}{\psi_i(x_i) \psi_{ij}(x_i, x_j) \psi_j(x_j)}$$

$$- \sum_i (n_i - 1) \sum_{x_i} b_i(x_i) \log \frac{b_i(x_i)}{\psi_i(x_i)}.$$

Compare to Bethe Free Energy:

$$b_{ij}(x_i, x_j) \to b_i(x_i) b_j(x_j).$$

Minimizing $KL(B)$ is not straightforward, but it is straightforward to find a local minimia.

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

Recent work, combine proposals based on BP or MFT ~ evaluate by Metropolis.