Lecture 14 - scribbles

Friday, October 19, 2018 13:24

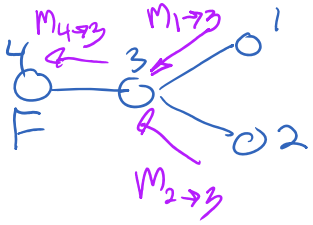
today: sum-product
max-product

Inference on trees

graphEliminate on a tree

good order: eliminate leaves first

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

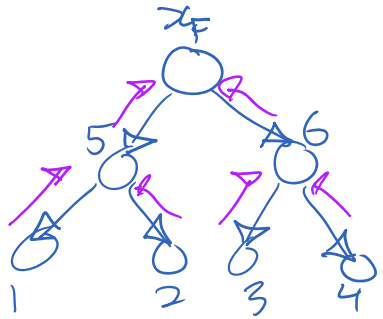


$$p(x_4) = \frac{\psi_4(x_4)}{Z} \sum_{x_3} \psi_3(x_3) \psi(x_4, x_3) \left(\sum_{x_2} \psi(x_2) \psi(x_3, x_2) \right) \left(\sum_{x_1} \psi(x_1) \psi(x_1, x_3) \right)$$

$m_{2\to3}(x_3)$ $m_{1\to3}(x_3)$

$m_{3\to4}(x_4)$

order: make a directed tree by using x_F as a root
(singleton)



start from leaves; go up towards root

$$m_{i\to j}(x_j) = \sum_{x_i} \psi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{K \in \text{children}(i)} m_{K\to i}(x_i)$$

child parent

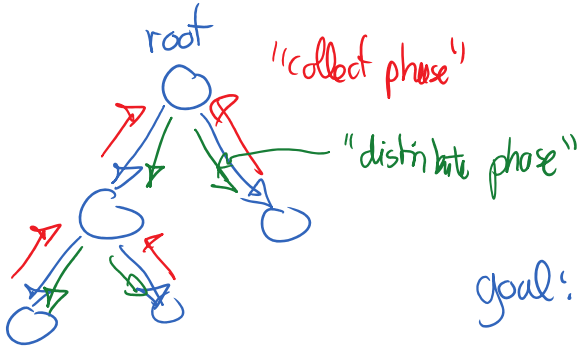
x_i

$k \in \text{children}(i)$

new factors containing i on actual list

sum-product alg. (for trees)

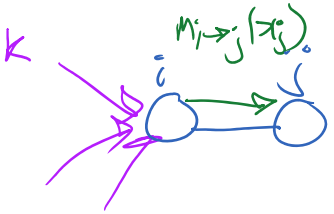
get all marginals cheaply by storing (caching) & re-using messages (dynamic programming)



goal: $\forall \{i,j\} \in E$, compute $m_{i \to j}(x_j)$

$m_{j \to i}(x_i)$

rule: i can only send message to neighbor j when it has received all messages from other neighbors



$$m_{i \to j}(x_j) = \sum_{x_i} \psi_i(x_i)\, \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus \{j\}} m_{k \to i}(x_i)$$

$m_{i \to j}$: child → parent

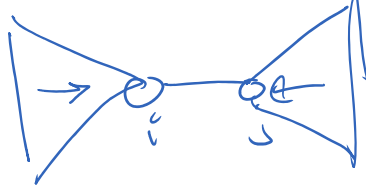
$N(i)$: neighbors

at end : $p(x_i) \propto \prod_{j \in N(i)} m_{j \to i}(x_i)\, \psi_i(x_i)$
(node marginal):

normalization $\sum_{x_i} \left(\quad \right)$

(edge marginal) $p(x_i, x_j) = \frac{1}{Z} \psi_i(x_i)\psi_j(x_j)\psi_{ij}(x_i,x_j) \Big/ \left(\prod_{k \in N(i) \setminus \{j\}} m_{k \to i}(x_i) \prod_{k' \in N(j) \setminus \{i\}} m_{k' \to j}(x_j) \right)$

normalization



Kth 10

sum-product schedules

a) above, distribute/collect schedule

b) (flooding) parallel schedule :

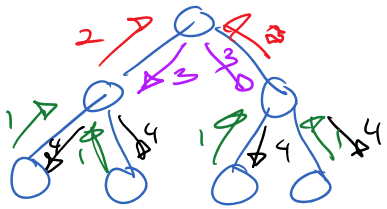
1) initialize $m_{i \to j}(x_j)$ messages to uniform dist. $\forall (i,j)$ s.t. $\{i,j\} \in E$

2) at every step (in parallel) compute $m^{new}_{i \to j}(x_j)$

as if the neighbor messages were already correctly computed

→ can prove that after "diameter of the tree" # of steps, all messages are correctly computed for a tree (and are fixed points)

for a tree (and the fixed points)



Loopy Belief Propagation (loopy BP) : approximate inference

$$m_{i\to j}^{(new)}(x_j) = \left(m_{i\to j}^{(x_j)}\right)^{\alpha} \left(\sum_{x_i} \psi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in N(i)\setminus\{j\}} m_{k\to i}^{old}(x_i) \right)^{1-\alpha}$$

$\alpha \in [0,1]$ "damping"

- this gives exact answer on tree (fixed pt. yields correct marginals)
- on (not too loopy) graphs $\rightsquigarrow$ approximate solution

getting conditionals:

$$p(x_i \mid \bar{x}_E) \propto p(x_i, \bar{x}_E)$$

indicate values we are conditioning on

keep this fixed during marginalization for each $j \in E$

(formal trick): redefine $\psi_j(x_j) \triangleq \psi_j(x_j)\cdot \delta(x_j, \bar{x}_j)$

Kronecker-delta δ s.t. $\delta(a,b) = \begin{cases} 1 & \text{if } a=b \\ 0 & \text{o.w.} \end{cases}$

computing $M_{j \to i}(x_i)$; $\sum_{x_j} \tilde{\psi}_j(x_j) \cdot \text{stuff}(x_j, x_i)$

$= \psi_j(\bar{x}_j) \text{stuff}(\bar{x}_j, x_i)$

at the end, result of sum-product

will give $p(x_i, \bar{x}_E) = \frac{1}{Z} \psi_i^e(x_i) \prod_{k \in N(i)} m_{k \to i}(x_i)$

14h3) $p(x_i | \bar{x}_E)$ renormalize over x_i

Max-product algorithm:

for sum-product, main property used was distributivity of $\oplus$ over $\odot$

$(\mathbb{R}, \oplus, \odot)$ is a semi-ring

↳ don't need additive inverse

can do "sum-product" on other semi-rings

$(\mathbb{R}, \max, \oplus)$ $\quad \max(a \oplus b, a \oplus c) = a \oplus \max(b, c)$

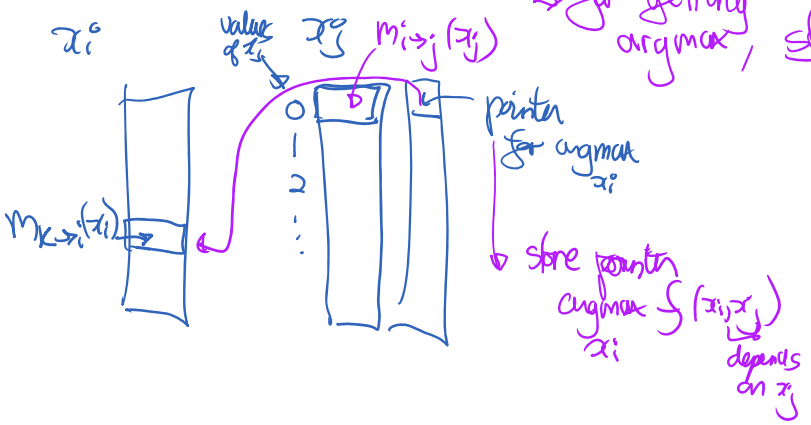
$(\mathbb{R}_+, \max, \otimes)$ $\quad \max(a \otimes b, a \otimes c) = a \otimes \max(b, c)$

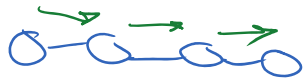
↳ "max-product"

$$\max_{x_{1:n}} \prod_i f_i(x_i) = \prod_i \max_{x_i} f_i(x_i)$$

$$m_{i \to j}(x_j) = \max_{x_i} \left[\psi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus \{j\}} m_{k \to i}(x_i) \right]$$

↳ for getting argmax, store argument of this max as fct. of x_j





to get $\underset{x_{1:n}}{\text{argmax}}\, p(x_{1:n})$ "decoding"

- run max-product alg (only forward messages)
- backtrack the argmax pointers to get full argmax

aka. Viterbi algorithm

property: $p \in \mathcal{G}(\text{tree})$ $\Rightarrow$ $p(x) = \prod_{i \in V} p(x_i) \prod_{\{i,j\} \in E} \frac{p(x_i, x_j)}{p(x_i) p(x_j)}$

with non-zero marginals

with non-zero marginals

(proof: simple exercise)

* similar to DGM; for any set of factors $\{f_{ij}(x_i,x_j)\}$, $\{f_i(x_i)\}$ $f_{ij} \geq 0$, $f_i \geq 0$ s.t. "local consistency property"

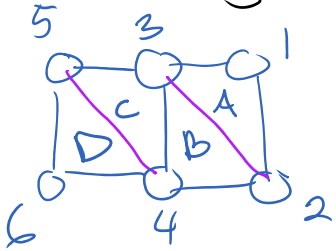
$$\begin{cases} \sum_{x_j} f_{ij}(x_i,x_j) = f_i(x_i) & \forall x_i \\ \sum_{x_i} f_{ij}(x_i,x_j) = f_j(x_j) \\ \sum_{x_i} f_i(x_i) = 1 \end{cases}$$

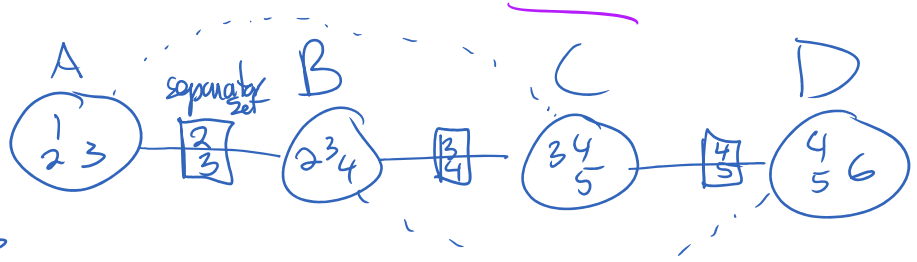
then if define joint (for tree)

$$p(x) = \prod_i f_i(x_i) \prod_{\{i,j\} \in E} \frac{f_{ij}(x_i,x_j)}{f_i(x_i) f_j(x_j)}$$

we get correct marginals i.e. $p(x_i) = f_i(x_i)$ etc...

Junction tree alg.: generalization of sum-product to a clique tree





above is a clique tree with the "running intersection property" "junction tree"

to build JT: • use max weighted spanning tree alg. on clique graph from a Δ-graph (with size of separator sets as weight on the edges)

JT $\Leftrightarrow$ triangulated graph / decomposable graph $\rightsquigarrow$ running graph Eliminate

when have J.T., you can show

$$p(x_V) = \frac{\prod_C p(x_C)}{\prod_S p(x_S)}$$

S ← separator set in my J.T.

J.T. tree alg.: reconstruit the above formulation $p(x_V) = \sum \frac{\prod_C \psi_C(x_C)}{\prod_S \varphi_S(x_S)}$ where $\varphi_S(x_S)=1$ at beginning

do message passing to update ψ_C^{new}, φ_S^{new} $\rightsquigarrow$ at end $p(x_C)$, $p(x_S)$ ...