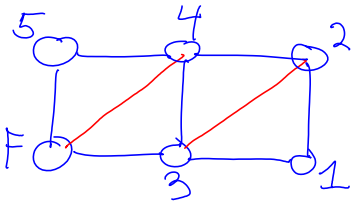
Lecture 13 - scribbles

Friday, October 14, 2016
13:17

today: Sum-product alg.
HMM

clique tree:

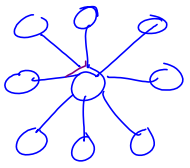


* running graphEliminate;
+ keep track of all added edges

→ get a triangulated graph

graph with no cycle of size 4 and more that cannot be broken by a chord

not all ordering are good:



convention treewidth(tree) = 1

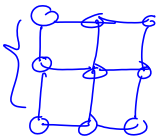
treewidth of graph = min { size of biggest clique formed − 1 }
elimination orderings

bad news;

a) NP hard to compute treewidth (or find best ordering)

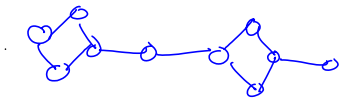
b) NP hard to do inference in general UGM
(need approximate methods)

example: tree width of a grid $\approx \sqrt{|V|}$



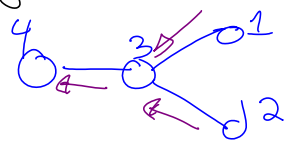
good news:

- inference in linear time for trees (treewidth=1)
(HMM, markov chain)
- efficient for "small treewidth graph"
→ "junction alg."



Inference in a tree:

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

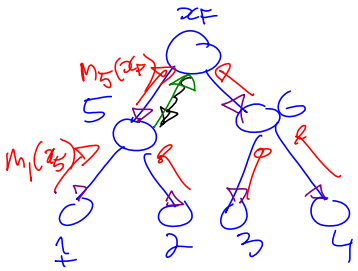


$$p(x_4) = \frac{\psi_4(x_4)}{Z} \sum_{x_3} \psi_3(x_3) \psi(x_4, x_3) \left(\sum_{x_2} \psi(x_2) \psi(x_3, x_2) \right) \left(\sum_{x_1} \psi(x_1) \psi(x_1, x_3) \right)$$

$m_{2 \to 3}(x_3)$ $m_{1 \to 3}(x_3)$

$m_{3 \to 4}(x_4)$

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



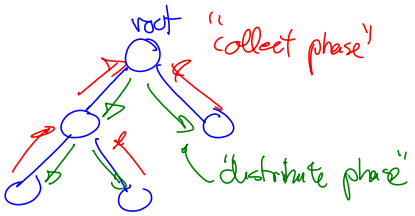
$$m_{i \to j}(x_j) = \sum_{x_i} \psi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in \text{children}(i)} m_{k \to i}(x_i)$$

i: child, j: parent

new factors containing i on active list

Sum-product alg (for trees)

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



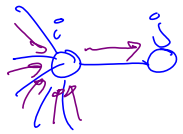
goal is to have $\forall \{i,j\} \in E$, want to have

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

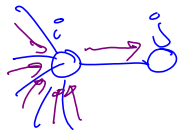
$m_{j \to i}(x_i)$

rule: i can only send message to neighbor j $m_{i \to j}(x_j)$

when it has received all messages from other neighbors



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

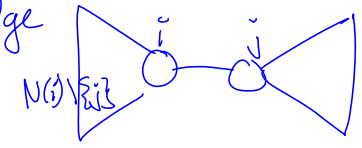


$$m_{i\to j}(x_j) = \sum_{x_i} \psi_i(x_i)\,\psi_{ij}(x_i,x_j) \prod_{k\in N(i)\setminus\{j\}} m_{k\to i}(x_i) \quad (\star)$$

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

$$Z = \sum_{x_i} (\quad\quad)$$

edge



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

above: distribute / collect schedule

(flooding) parallel schedule:

1) initialize $m_{i\to j}(x_j)$ to uniform distribution $\forall j$

2) at every step (in parallel) compute $m_{i\to j}(x_j)$ $(\star)$

as if neighbor messages were correctly computed

→ can prove after diameter (tree) steps (longest path in tree), all messages are correctly computed.

Loopy BP: approximate inference

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

$0 \le \alpha \le 1$ "damping"

this on tree → exact
on (not too loopy) graph → approximate

conditionals:

$$p(x_i \mid \underbrace{x_E}_{\text{given}}) \propto p(x_i, x_E)$$

keep it fixed during sum-product

(formal trick): redefine $\tilde{\psi}_j(x_j) = \psi_j(x_j)\,\delta(x_j, \bar{x}_j)$

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

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

when have $\sum_{x_j} \tilde{\psi}_j(x_j) \cdot \text{stuff}(x_j, x_i)$

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

$$p(x_i \mid \bar{x}_E) = \frac{p(x_i, \bar{x}_E)}{\sum_{x_i} p(x_i, \bar{x}_E)}$$

max-product?

only property used: distributivity of $\oplus$ with $\odot$

$(\mathbb{R}, \oplus, \odot)$ is semi-ring (don't need inverse to addition)

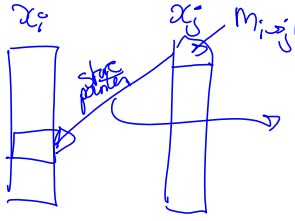
can do "sum-product" on other semi-rings

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

$(\mathbb{R}, \max, \oplus)$

→ "max-product" $\quad \max_{x_{1:n}} \prod_i f_i(x_i) = \prod_i \max_{x_i} f_i(x_i)$

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



↳ for argmax, store argument as function of x_j

max-product aka Viterbi algorithm → decoding $\operatorname{argmax}_{x_{1:n}} p(x_{1:n})$

⊛ implementation comment $\prod m_{k\to i}(x_i)$

K (small #)

$$= \exp\left(\sum_K \log m_{k\to i}(x_i)\right)$$

$$\sum_x f(x) = \sum_x \exp(\log(f(x)))$$

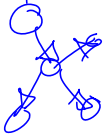
$$= \exp(\log f(x_{max}))\left[1 + \sum_{x \neq x_{max}} \exp\left(\log\left(\frac{f(x)}{f(x_{max})}\right)\right)\right]$$

proposition:

$p \in \mathcal{G}(\text{Tree})$ with non-zero marginals $\Rightarrow p(x) = \prod_i p(x_i) \prod_{\{i,j\}\in E} \frac{p(x_i,x_j)}{p(x_i)p(x_j)}$

proof:

$p \in \mathcal{G}(\text{undirected tree}) \Leftrightarrow p \in \mathcal{G}(\text{any orientation of tree})$



$\Rightarrow p(x) = \prod_i p(x_i | x_{\pi_i})$ for some orientation of tree

say x_n is root

$$= \left(\prod_{i<n} p(x_i | x_{\pi_i})\right) p(x_n)$$

$$= \left(\prod_{i<n} \frac{p(x_i)p(x_i, x_{\pi_i})}{p(x_i)p(x_{\pi_i})}\right) p(x_n)$$

$$= \prod_i p(x_i) \prod_{\{i,j\}\in E} \frac{p(x_i,x_j)}{p(x_i)p(x_j)} \quad //$$

as before, 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")

$$\begin{cases} \sum_{x_j} f_{ij}(x_i,x_j) = f_i(x_i) \\ \sum_{x_i} f_{ij}(x_i,x_j) = f_j(x_j) \end{cases}$$ ("marginalization consistency")

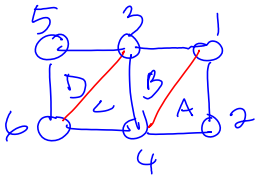
$$\sum_{x_i} f_i(x_i) = 1$$

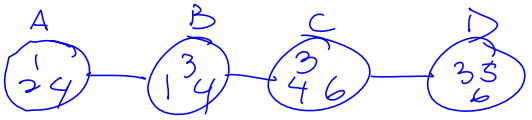
define a joint $p(x) = \prod_i f_i(x_i) \prod_{(i,j)\in E} \frac{f_{ij}(x_i,x_j)}{f_i(x_i)f_j(x_j)}$

with correct marginals ie. $p(x_i) = f_i(x_i)$ etc...

junction tree alg:

generalization of Sum-product to clique trees.





"running intersection property"

triangulated graph / decomposable graph

running Graph Elimination