Lecture 16 - Inference; GraphEliminate

Friday, October 30, 2020 13:31

today: inference → graph eliminate

general themes in this class

A) representation → DGM, UGM
parameterization → exponential family
} model high. dim. distribution

B) inference $p(x_Q | x_E)$ ("query", "evidence")
→ today: elimination algorithm
→ next: sum-product / belief propagation (for trees)
later: approximate inference e.g. MCMC variational

C) statistical estimation / learning → MLE, maximum entropy, method of moments

Inference

want to compute

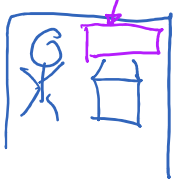
a) marginal: $p(x_F)$ for some $F \subseteq V$

b) conditional: $p(x_F | x_E)$ ("query", "evidence")

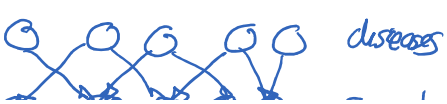
c) for UGM: partition function $Z = \sum_{x_V} \left(\prod_C \psi_C(x_C) \right)$ ← exponential sum

why?

- missing data $p(x_{unobs.} | x_{obs.})$ — example: "infilling task", missing pixels



- prediction $p(x_{future} | x_{past})$
- "latent cause" $p(x_{cause} | x_{obs.})$ e.g. diseases



Symptoms

* also related to inference:

$$\underset{x_F}{\text{argmax}}\ p(x_F \mid x_E)$$

F could be big here (e.g. speech recognition)

* inference is also needed during estimation (parameter fitting MLE)

[e.g. during E-step $p(z|x)$]

* present inference alg. for UGM [for simplicity and more generality]

but note that sometimes it's more efficient to work directly on DGM

make DGM $\rightarrow \subseteq$ UGM via moralization

i.e. $p \in$ DGM; $p(x) = \prod_i p(x_i \mid x_{\pi_i})$ moralize: $C_i \triangleq \{i\} \cup \pi_i$

$$= \frac{1}{Z} \prod_i \psi_{C_i}(x_{C_i}) \qquad \psi_{C_i}(x_{C_i}) \triangleq p(x_i \mid x_{\pi_i})$$

$Z = 1$

Graph elimination alg. (for inference)

- consider $p \in \mathcal{G}(\mathcal{G})$ (undirected) $\qquad p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$

Say want to compute $p(x_F)$ for $F \subseteq V$ "query nodes"

main trick: use distributivity of $\oplus$ over $\otimes$ $\rightarrow c \otimes (a \oplus b) = c \otimes a \oplus c \otimes b$

$$\sum_{x_1, x_2} f(x_1) g(x_2) = \left(\sum_{x_1} f(x_1)\right)\left(\sum_{x_2} g(x_2)\right)$$ [convince yourself]

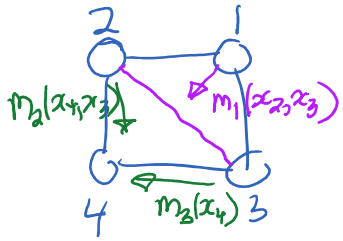
↖ sum term of product

e.g. $\sum_{x_2} f(x_1 = 1)\, g(x_2)$

more generally

$$\sum_{x_{1:n}} \prod_i f_i(x_i) = \prod_{i=1}^{n} \left(\sum_{x_i} f_i(x_i)\right)$$

$O(k^n \cdot n) \qquad O(k \cdot n)$



$F = \{4\}$

$$p(x_4) = \frac{1}{Z} \sum_{x_1,x_2,x_3} \psi(x_1,x_2)\psi(x_2,x_4)\psi(x_1,x_3)\psi(x_3,x_4)$$

$$= \frac{1}{Z} \left(\sum_{x_3} \psi(x_3,x_4) \left(\sum_{x_2} \psi(x_2,x_4) \left(\sum_{x_1} \psi(x_1,x_2)\psi(x_1,x_3) \right) \right) \right)$$

$\sum_{x_1} \psi(x_1,x_2)\psi(x_1,x_3) = m_1(x_2,x_3)$ "message"

m_1 stored as table

$\sum_{x_2} \psi(x_2,x_4) m_1(x_2,x_3) = m_2(x_3,x_4)$

$$= \frac{1}{Z} m_3(x_4)$$

→ last message is proportional to marginal $p(x_4)$

$\sum_{x_4} m_3(x_4) = Z$

general alg.: graphEliminate

general alg.: graphEliminate

init. a) choose an elimination ordering s.t. F are the last nodes

b) put all $\psi_c(x_c)$ on "active list"

"update" c) repeat in order of variables to eliminate

(say x_i is variable to eliminate)

1) remove all factors from active list with x_i in it & take their product

i.e. $\prod_{\alpha \text{ s.t. } i \in \alpha} \psi_\alpha(x_\alpha)$

2) sum over x_i to get a new factor $m_i(x_{S_i})$ (think as $\psi_{S_i}(x_{S_i})$)

i.e. S_i are all variables in those factors except i

$$\text{get } m_i(x_{S_i}) \triangleq \sum_{x_i} \prod_{\alpha \text{ s.t. } i \in \alpha} \psi_\alpha(x_\alpha)$$

new clique to sum over $S_i \cup \{i\}$

$$S_i \triangleq \left(\bigcup_{\alpha \text{ s.t. } i \in \alpha} \alpha\right) \setminus \{i\}$$

3) put back $m_i(x_{S_i})$ in active list $(\psi_{S_i}(x_{S_i}))$

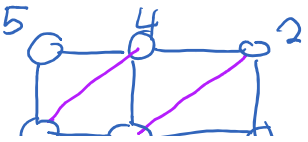
"normalize" d) last product of factors left has only $x_F \Rightarrow$ proportional $p(x_F)$

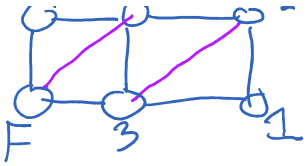
memory needed? $\simeq 2^{\max_i |S_i|} \cdot$ (# of factors)

computational cost $\simeq \underbrace{2^{\max_i |S_i|+1}}_{\text{later, related "treewidth" of a graph}} \cdot n$

14h34

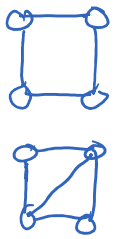
(*) "augmented graph" → graph obtained by running graphEliminate + keeping track of all edges added (for a fixed ordering)





"augmented graph" after graphEliminate is always a **triangulated graph**

def.: graph with no cycle of size 4 or more that cannot be broken by a "chord"
↳ an edge between two non-neighboring nodes in the cycle



← not triangulated

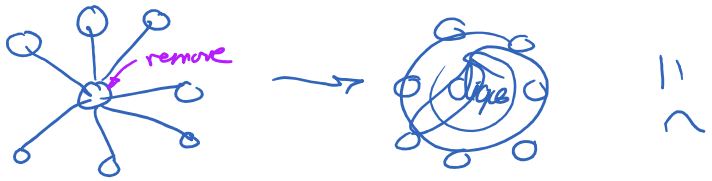
treewidth of a graph $\triangleq$ min over all possible elimination orderings { size of biggest clique in augmented graph − 1 }

convention: treewidth(tree) = 1

both memory & running time of graphEliminate is dominated by $2^{\text{size of biggest clique}}$

best ordering give $\approx 2^{\text{treewidth}+1}$

not all orderings are good



bad news:

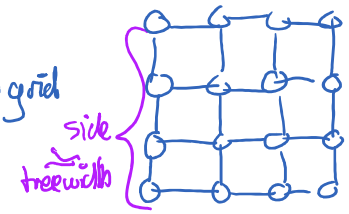
a) NP hard to compute treewidth of general graph (or find best ordering)

b) NP hard to do (exact) inference in general UGM

$\Rightarrow$ approximate methods

example:

treewidth of a grid $\approx \sqrt{|V|}$



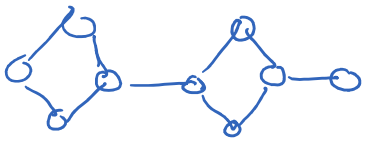
good news

* inference is linear time for trees (treewidth = 1) ("sum-product alg.")
↳ $N + |E|$

(HMM, Markov chain, etc.)

* efficient for "small treewidth graph"

⇝ use junction tree alg.

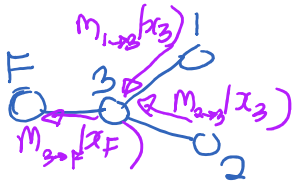


inference on trees

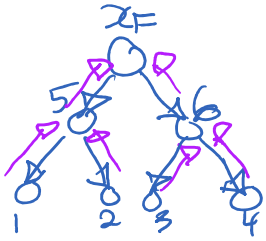
graphEliminate on a tree

good order is to eliminate leaves first

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



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

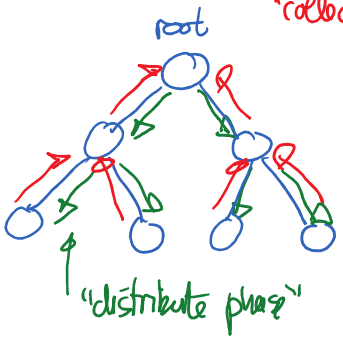


$$m_{\underset{\text{child}}{i} \to \underset{\text{parent}}{j}}(x_j) = \sum_{x_i} \psi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{K \in \text{children}(i)} m_{K\to i}(x_i)$$

new factors containing i and other vars

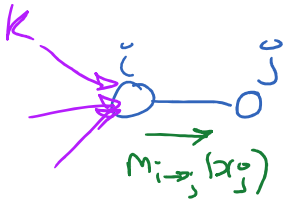
Sum-product alg. (for trees)

alg. to get all node/edge 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



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

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

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

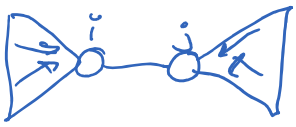
neighbors

at end (node marginal)

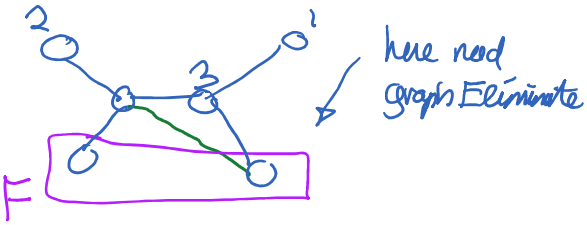
$$p(x_i) \propto \prod_{j\in N(i)} m_{j\to i}(x_i)\,\psi_i(x_i)$$

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

(edge marginal)



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



Sum-product schedules:

a) above, collect/distribute schedule

b) (flooding) parallel schedule:

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

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

as if the neighbor messages were correctly computed

→ one can prove that "diameter of tree" # of steps

all messages are correctly computed (for a tree) (and are fixed pt.)