Lecture 15 - max-product, junction tree; HMM

Tuesday, October 31, 2023 2:19 PM

today: • max product alg.
• junction tree
• HMM

getting conditionals

$x_E = \bar{x}_E$

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

$\bar{x}_E$: indicates values we are conditioning on

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

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

Kronecker delta fct.

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

computing $m_{j \to i}(x_i) = \sum_{x_j} \tilde{\psi}_j(x_j) \, \text{stuff}(x_i, x_j)$

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

at the end, result of sum-product

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

Z: renormalize over x_i to get $p(x_i | \bar{x}_E)$

max-product alg.

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

all need is that $(\mathbb{R}, \oplus, \odot)$ is $[a \odot b \oplus a \odot c = a \odot (b \oplus c)]$

semi-ring → ring that doesn't require additive inverses

(*) can do "sum-product" on other semi-rings

(*) can do "sum-product" on other semi-rings

$(\mathbb{R} \cup \{-\infty\}, \max, +)$ $\max(a+b, a+c) = a + \max(b,c)$

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

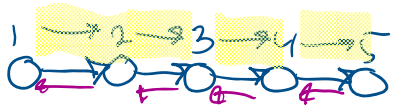
↳ "max-product"

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

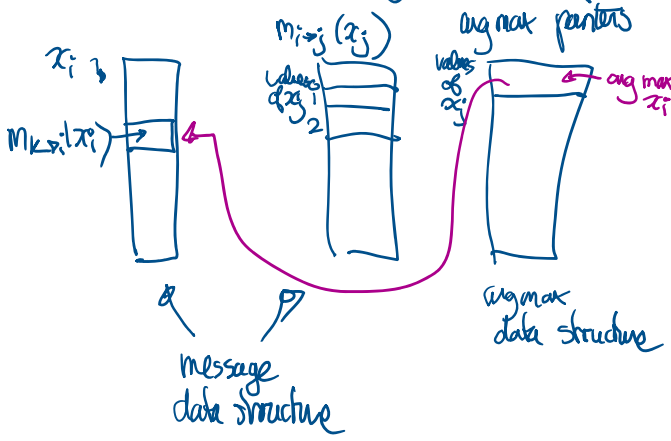
$$m_{i \to j}(x_j) = \max_{x_i} \left\{ \psi_i(x_i) \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus \{j\}} m_{k \to i}(x_i) \right\}$$

x_i ↳ for getting argmax

store argument of this max as a fct. of x_j



$$\max_{x_{1:5}} \frac{1}{Z} \prod_c \psi_c(x_c) = \frac{1}{Z} \max_{x_5} \left[m_{4 \to 5}(x_5) \psi_5(x_5) \right]$$



(*) to get $\operatorname{argmax}_{x_{1:n}} p(x_{1:n})$ "decoding"

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

aka. Viterbi alg.

property of tree UGM

$p \in \mathcal{G}(\text{tree})$ with non-zero marginals $\Rightarrow$

$$p(x) = \prod_{i \in V} p(x_i) \prod_{\{i,j\}} \frac{p(x_i, x_j)}{p(x_i) p(x_j)}$$

$\psi_i(x_i)$ $\psi_{ij}(x_i, x_j)$

$$\frac{p(x_i, x_j)}{p(x_i) p(x_j)} = \frac{p(x_i | x_j)}{p(x_i)}$$

$\frac{p(x_i|x_j)}{p(x_i)}$

⊛ Similar to DGM; for any set of factors $\{f_{ij}(x_i, x_j)\}$

$\{f_i(x_i)\}$ $\quad f_{ij} \geq 0$

$f_i \geq 0$

s.t. "local consistency property"

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

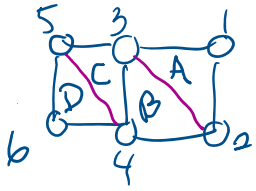
$$\sum_{x_i} f_i(x_i) = 1$$

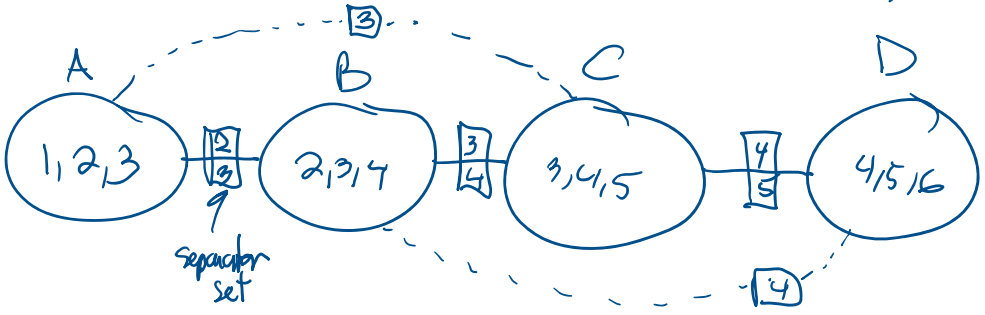
then if define joint $p(x) = \frac{1}{Z} \prod_i f_i(x_i) \prod_{i,j \in E} \frac{f_{ij}(x_i, x_j)}{f_i(x_i) f_j(x_j)}$

(for tree) → then can show that $Z = 1$
and f_{ij}'s are correct marginals
f_i's

e.g. $p(x_i) = f_i(x_i)$
etc.. //

junction tree alg: generalization of sum-product to a clique tree (with J.T. property)





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

↳ if $j \in C_1 \cap C_2$, then $j \in C_k \; \forall C_k$ along path from C_1 to C_2

to build a J.T. on a Δ-graph:
- use maximum weight spanning tree alg on clique graph (with size of separator set as weight on edges) ⇒ has running intersection property

$\exists$ a J.T. ⇔ triangulated / decomposable graph ⇔ running

$\exists$ a J.T. $\Leftrightarrow$ triangulated / decomposable graph $\leftrightarrow$ running graph-Elimination

when have J.T., one can show

$$p(x_V) = \frac{\prod_{C \in \mathcal{C}} p(x_C)}{\prod_{S} p(x_S)}$$

S: separator sets of J.T.

$V = \{1,2,3\}$
$C_1 = \{1,2\}$
$C_2 = \{2,3\}$

15h27

J.T. alg.: reconstruct the above formulation

by starting with $p(x_V) = \frac{1}{Z} \frac{\prod_C \psi_C(x_C)}{\prod_S \varphi_S(x_S)}$ where $\varphi_S(x_S) = 1$ at initialization

do message passing on J.T. to update

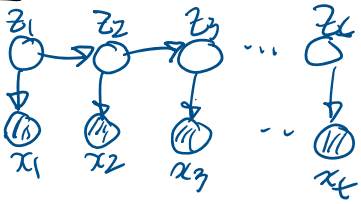
ψ_C^{new}
φ_S^{new}

at end of Message passing: $\psi_C^{new} \to p(x_C)$, $\varphi_S \sim p(x_S)$

- which leaves the joint invariant

(see Mike's book ch. 17 for details)

HMM (Hidden Markov model)

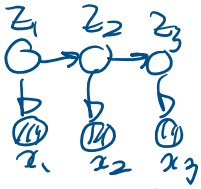


$z_t \in \{1, \dots, k\}$ discrete

x_t < cfs. eg. speech sound waves
discrete eg. DNA sequence

(later $z_t \sim$ Gaussian $\to$ Kalman filter)

HMM $\to$ generalization of mixture model

GMM  add dependence on time  ....

DGM: $p(x_{1:T}, z_{1:T}) = p(z_1) \prod_{t=2}^{T} p(x_t|z_t) \prod_{t=2}^{T} p(z_t|z_{t-1})$

DGM: $p(x_{1:T}, z_{1:T}) = p(z_1) \prod_{t=1}^{T} p(x_t|z_t) \prod_{t=2}^{T} p(z_t|z_{t-1})$

$p(x_t|z_t)$: emission prob. $p(z_t|z_{t-1})$: transition prob.

often: emission & trans. prob. are homogeneous in time (i.e. do not depend on t)

$p_t(x_t|z_t) = f(x_t|z_t)$

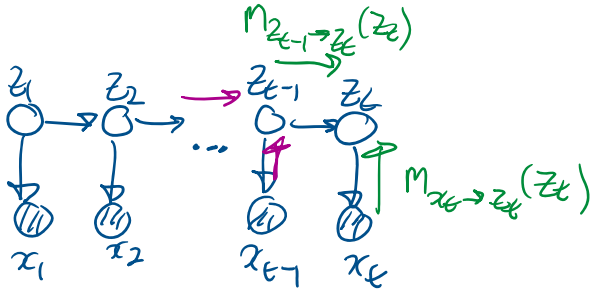
$p_t(z_t = i \mid z_{t-1} = j) = A_{ij}$ ("stochastic matrix")

A: i (dist. over z_t), j; $\sum_i A_{ij} = 1$

inference tasks:
- prediction $p(z_t|x_{1:t-1})$ "where next"
- filtering $p(z_t|x_{1:t})$ "where now"
- smoothing $p(z_t|x_{1:T})$ "where in the past" $T > t$

α-recursion

Let's run sum-product alg. to derive recursion to compute $p(z_t|x_{1:t})$



to compute $p(z_t, \bar{x}_{1:t})$ or $p(z_t|\bar{x}_{1:t})$

$p(z_t, \bar{x}_{1:t}) = \frac{1}{Z} 1 \; m_{z_{t-1} \to z_t}(z_t) \; m_{x_t \to z_t}(z_t)$ (here $Z = 1$)

$= \alpha_t(z_t)$

$m_{x_t \to z_t}(z_t) = \sum_{x_t} p(x_t|z_t) \delta(x_t, \bar{x}_t) = p(\bar{x}_t|z_t)$ (conditioning trick)

$m_{z_{t-1} \to z_t}(z_t) = \sum_{z_{t-1}} p(z_t|z_{t-1}) \; m_{z_{t-2} \to z_{t-1}}(z_{t-1}) \; m_{x_{t-1} \to z_{t-1}}(z_{t-1})$

$= p(z_{t-1}, \bar{x}_{1:t-1}) = \alpha_{t-1}(z_{t-1})$

$p(z_t, \bar{x}_{1:t})$

$$\alpha_t(z_t) = p(\bar{x}_t|z_t) \sum_{z_{t-1}} p(z_t|z_{t-1}) \alpha_{t-1}(z_{t-1})$$

$$\alpha_t(z_t) = p(\bar{x}_t \mid z_t) \sum_{z_{t-1}} p(z_t \mid z_{t-1}) \alpha_{t-1}(z_{t-1})$$

vector · matrix · vector

$\alpha_{t-1}(z_{t-1})$

α-recursion aka "forward recursion"

like the "collect phase" in sum-product alg.

initialization: $\alpha_1(z_1) = p(z_1, \bar{x}_1) = p(\bar{x}_1 \mid z_1)\, p(z_1)$

let $\alpha_t(z_t) \triangleq p(\bar{x}_t \mid z_t)$

$$\alpha_t = \alpha_t \odot (A \alpha_{t-1})$$

$\odot$ Hadamard product

$\tilde{\alpha}_t(z_t) \triangleq p(z_t \mid \bar{x}_{1:t})$ "filtering distribution"

$= \alpha_t(z_t)$

$\left\{ \sum_{z_t} \alpha_t(z_t) \right\} \left\{ \sum_{z_t} p(z_t, \bar{x}_{1:t}) \right\}$

$= p(\bar{x}_{1:t})$ "evidence prob."

space complexity: $O(K)$ extra storage

time complexity: $O(tK^2)$