Lecture 17 - max-product; junction tree; HMM

Tuesday, November 3, 2020 14:28

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

loopy belief propagation (loopy BP): approximate inference for graphs with cycles

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

$\alpha \in [0,1]$ "damping" parameter / step-size

* this gives exact answer on trees (fixed pt. → yields correct marginals)
* on (not too loopy) graphs → approximate solution

getting conditionals:

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

$\bar{x}_E$: indicates values we are conditioning on; $x_E = \bar{x}_E$: 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 def. $\delta(a,b) \triangleq \begin{cases} 1 & \text{if } a=b \\ 0 & \text{o.w.} \end{cases}$

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

$$\sum_{x_j} \tilde{\psi}_j(x_j)\,\text{stuff}(x_j,x_i) = \psi_j(\bar{x}_j)\,\text{stuff}(\bar{x}_j,x_i)$$

at the end, result of sum-product will give $p(x_i,\bar{x}_E) = \frac{1}{Z}\psi_i(x_i)\prod_{k\to i} m_{k\to i}(x_i)$

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

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 semi-ring
↳ don't need additive inverses

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

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

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

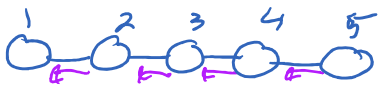
↳ "max product"

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

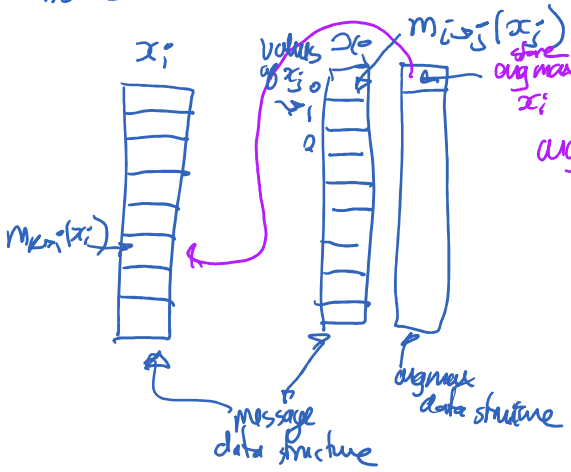
$$m_{i\to j}(x_j) = \max_{x_i} \left[\psi_i(x_i)\psi_{ij}(x_i,x_j) \prod_{k\in N(i)\setminus\{j\}} m_{k\to i}(x_i) \right]$$

for getting argmax

store argument of this max as a fct. of x_j



$$\max_{x_{1:5}} \frac{1}{Z} \prod_c \psi_c(x_c) = \frac{1}{Z} \max_{x_1} m_{2\to 1}(x_1)$$



(*) to get $\arg\max_{x_{1:n}} p(x_{1:n})$ "decoding"

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

aka Viterbi algorithm

Property of tree UGM

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

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

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

(*) similar to DGM; for any set of factors $\{f_i(x_i,x_j)\}, \{f_i(x_i)\}$ $f_i \geq 0$

⊛ 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"

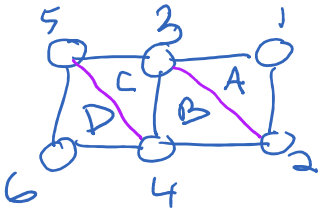
$$\begin{cases} \sum_{x_j} f_{ij}(x_i,x_j) = f_i(x_i) & \forall x_i \\ \sum_{x_i} f_{ij}(x_i,x_j) = f_j(x_j) & \forall x_j \\ \sum_{x_i} f_i(x_i) = 1 \end{cases}$$

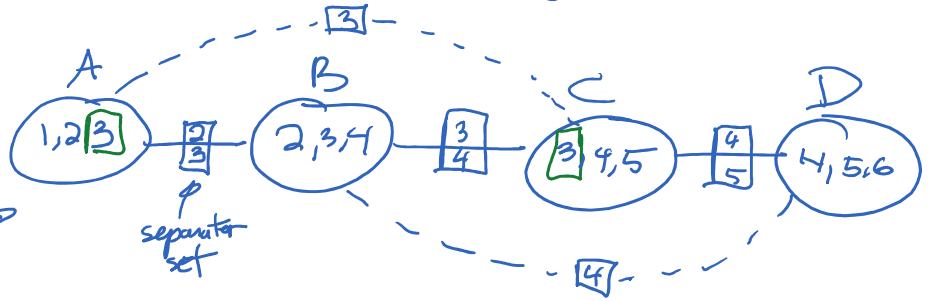
then if define joint (for tree) $p(x) = \prod_i f_i(x_i) \prod_{i \sim j \in E} \frac{f_{ij}(x_i,x_j)}{f_i(x_i) f_j(x_j)}$

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

15h32

junction tree algorithm : 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 $j \in C_j$ $\forall C_j$ along path from C_1 to C_2

to build a J.T. or a A-graph

- use maximum weight spanning tree alg. on clique graph (with size of separator sets as weight on edges) ⇒ has running intersection property

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

⊛ when have J.T., one can show

⊛ when have J.T., one can show

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

S: separator sets in the J.T.

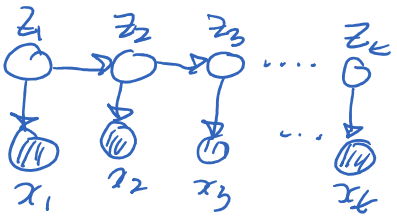
J.T. alg.: reconstruct the above formulation

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

do message passing on J.T. to update

ψ_C^{new}, ψ_S^{new} $\xrightarrow{\text{at the end}}$ $p(x_C)$, $p(x_S)$

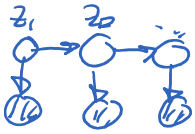
HMM (hidden Markov model)



$z_t \in \{1, \dots, K\}$ discrete (later $z_t \sim$ Gaussian $\rightsquigarrow$ Kalman filter)

x_t ← cts, e.g. speech signal; discrete e.g. DNA sequence

HMM → generalization of mixture model

GMM  add dependence on time 

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

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

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

$p_t(x_t|z_t) = f(x_t|z_t)$ ∀ t

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

$p(z_t = i | z_{t-1} = j) = A_{ij}$ — "stochastic matrix"

A i (} dist. over z_t

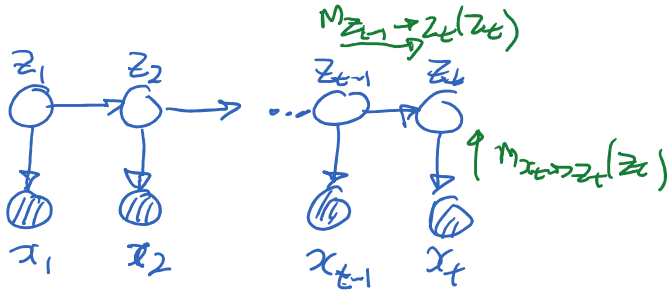
$\sum_i A_{ij} = 1 \quad \forall j$

inference tasks:

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

α-recursion:

Let's run sum-product here to derive recursions to compute prob.



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

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

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

$m_{z_{t-1} \to z_t}(z_t) = \sum_{z_{t-1}} p(z_t \mid z_{t-1}) \underbrace{m_{z_{t-2} \to z_{t-1}}(z_{t-1}) \cdot m_{x_{t-1} \to z_{t-1}}(z_{t-1})}_{p(z_{t-1}, \bar{x}_{1:t-1}) \;=\; \alpha_{t-1}(z_{t-1})}$

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

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

α-recursion aka "forward recursion" (also the "rollout phase" in ...

α-recursion aka. "forward recursion" like the "collect phase" in sum-product alg.