Lecture 17 - max-product; junction tree; HMM

- Max-product alg.
  - Junction tree
  - HMM

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

\[
\begin{align*}
\tilde{p}_{i \rightarrow j}(x_j) &= \frac{1}{\alpha} \prod_{a \in \text{pa}(j)} \left( \frac{\tilde{p}_a(x_j) \tilde{p}_i(x_i)}{\tilde{p}_a(x_j)} \right) \\
\alpha &= \exp(-\lambda) \quad \text{"damping"}
\end{align*}
\]

\( \lambda \in [0, 1] \)

- This gives exact answer on trees
  - (fixed \( \lambda \)): yields correct marginals
- On (not too loopy) graphs: approximate solution

Getting conditionals:

\[
\tilde{p}(x_i | \bar{x}_E) \propto p(x_i, \bar{x}_E)
\]

Indicates values we are conditioning on

Keep this fused away marginalization

For each \( j \in E \)

(formal trick): redefine \( \tilde{p}_j(x_j) \equiv \tilde{p}_j(x_j) \delta(x_j, \bar{x}_j) \)

Kronecker delta

\( \delta(a, b) = 1 \) if \( a = b \)

\( \delta(0, 0\text{.}x_0) \)

Computing \( \rho_{i \rightarrow j}(x_i; x_j) \):

\[
\begin{align*}
\rho_{i \rightarrow j}(x_i; x_j) &= \frac{\tilde{p}_i(x_i) \tilde{p}_j(x_j)}{\tilde{p}_i(x_i)} \\
&= \tilde{p}_j(x_j) \delta(x_j, \bar{x}_j)
\end{align*}
\]

At the end, result of sum-product will give

\[
p(x_i; \bar{x}_E) = \prod_{j \in E} \rho_j(x_j; \bar{x}_j, x_i)
\]

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 we need is that \((IR, \oplus, \odot)\) is a **semi-ring**. We don't need additive inverses.

Can do "sum-product" in **other semi-rings**:

- \((IR, \max, +)\):
  \[ \max(a+b, a+c) = \max(b, c) \]
- \((IR^+, \max, \cdot)\):
  \[ \max(a \cdot b, a \cdot c) = a \cdot \max(b, c) \]

\( \rightarrow "\text{max-product}\)"

\[ \max \left( \frac{\prod f_i(x_i)}{\prod \mathbf{k}_i \mathbf{e}_j} \right) \]

For getting augmax:

Store argument of this max as a set of \( x_i \).

\[ \max_{x \in \mathbf{c}} \frac{\prod f_i(x_i)}{\prod \mathbf{k}_i \mathbf{e}_j} \]

Augmax for this value of \( x_j \).

Augmax \( f(x_i, x_j) \)

\( \rightarrow \text{to get augmax} p(x, \mathbf{e}) \) "decoding".

- Run max-product alg. (only forward messages).
- Backtrack the augmax-potentials to get full augmax.

\( \text{aka Viterbi algorithm} \)

**Property of tree case**:

\[ p \in \mathbf{E} \text{ (tree)} \]

\[ \mathbf{p} = \{ [x_i] \} \text{ with non-zero marginals} \]

\[ \mathbf{p}(x) = \prod_{i \in V} p(x_i) \prod_{\mathbf{f} \in \mathbf{E}} \frac{p(x_i; x_{\mathbf{f}})}{p(x_i) p(x_{\mathbf{f}})} \]

\( \overset{\text{if }}{=} \text{ similar to DOMS for any set of factors } \mathbf{f} = \{ f_i; x_i, x_j \} \text{, } f_i > 0 \)
similar to DOM; for any set of factors $f_1, f_2, f_3, f_4, f_5$, we have

\[ \sum_{x_i} \prod_{j \neq i} f_j(x_{j \neq i}) = \sum_{x_i} f_i(x_i) \quad \forall x_i \]

local consistency property

then if define joint

\[ p(x) = \prod_{j \neq i} \frac{f_j(x_j)}{\sum_{x_j} f_j(x_j)} \quad \forall x_j \neq x_i \]

then we get correct marginals i.e.,

\[ p(x_i) = f_i(x_i) \]

etc...

15h32

quintin tree algorithm: generalization of sum-product to a clique tree, (with JT property)

draw a clique tree with the running intersection property

by $j \in C_1 \cap C_2$, then $j \in C_j \forall j$ along path from $C_1$ to $C_2$

for a JT on a $\Delta$-graph

- use maximum weighted spanning tree alg. on clique graph

- separator sets as recognition edges $\Rightarrow$ has running intersection property

exists in triangulated graph /

decomposable graph

when have JT, one can show
when have J.T., one can show

\[
p(x_v) = \frac{\prod p(x_c)}{\prod p(x_s)}
\]

separates sets in the J.T.

J.T. alg.: reconstruct the above formulation

by starting with: \( p(x_v) = \frac{1}{Z} \prod p_c(x_c) \) \( \frac{1}{Z} p_s(x_s) \) where \( p(x_s) = 1 \) at initialization.

Do message passing on J.T. to update

\[
\begin{array}{c}
\text{new at the end} \\
(\text{new } p_s) \rightarrow p(x_v)
\end{array}
\]

\textbf{HMM} \\
(hidden Markov model)

\( Z_c \in \mathbb{R} \) \( \ldots \) \( K \) states denote \( Z_t \in c_k \), e.g., speech signal discrete, e.g., DNA sequence

\( \text{often } Z_t \approx \text{Gaussian} \) \( \rightarrow \text{Kalman filter} \)

\text{HMM \rightarrow generalization of mixture model}

\text{GMM}

\text{DGM: } p(x_{1:T}, z_{1:T}) = p(z_1) \prod_{t=1}^{T} \left( \sum_{c} p(x_t | z_t) \frac{1}{Z} p(z_t) \right)

\text{often, emission / transition prob. are homogeneous in time (i.e., do not depend on t)
\[ P_t(\mathbf{x}_t | \mathbf{z}_t) = \mathcal{F}(\mathbf{z}_t | \mathbf{z}_t) \]

\[ P_t(\mathbf{z}_t = \mathbf{i} | \mathbf{z}_{t-1} = \mathbf{j}) = A_{ij} \]

"stochastic matrix"

\[ A_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_t = \mathbf{i} \\ 0 & \text{otherwise} \end{cases} \]

Inference tasks:

- **Prediction**: \( p(\mathbf{z}_t | \mathbf{x}_{t+1}) \) "where next?"
- **Filtering**: \( p(\mathbf{z}_t | \mathbf{x}_{t:t}) \) "where now?"
- **Smoothing**: \( p(\mathbf{z}_t | \mathbf{x}_{1:t}) \) "where in the past?"

\[ \mathcal{O}_t - \text{Recursion} : \]

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

\[
p(\mathbf{z}_t | \mathbf{x}_{1:t}) = \frac{1}{Z_t} \cdot M_{\mathbf{x}_{t-1} \rightarrow \mathbf{z}_t}(\mathbf{z}_t) \cdot M_{\mathbf{z}_{t-1} \rightarrow \mathbf{z}_t}(\mathbf{z}_t) \quad (\text{hmm } Z_t = 1) \]

\[
M_{\mathbf{x}_{t-1} \rightarrow \mathbf{z}_t}(\mathbf{z}_t) = \sum_{\mathbf{z}_{t-1}} p(\mathbf{x}_t | \mathbf{z}_t) s(\mathbf{x}_t, \mathbf{x}_{t-1}) = p(\mathbf{x}_t | \mathbf{z}_t) \]

\[
M_{\mathbf{z}_{t-1} \rightarrow \mathbf{z}_t}(\mathbf{z}_t) = \sum_{\mathbf{z}_{t-1}} p(\mathbf{z}_t | \mathbf{z}_{t-1}) M_{\mathbf{z}_{t-1} \rightarrow \mathbf{z}_t}(\mathbf{z}_{t-1}) \cdot M_{\mathbf{z}_{t-1} \rightarrow \mathbf{z}_{t-1}}(\mathbf{z}_{t-1}) \]

\[ \alpha^t_t(\mathbf{z}_t) = p(\mathbf{x}_t | \mathbf{z}_t) \sum_{\mathbf{z}_{t-1}} p(\mathbf{z}_t | \mathbf{z}_{t-1}) \alpha^t_{t-1}(\mathbf{z}_{t-1}) \]
α-reussen aka. "forward recursion" like the "collect phase" in sum-product alg.