Lecture 14 - Inference in UGM

Today: DGM vs. UGM

**DGM vs. UGM**

- **Definition:** Markov blanket for $i$ (for graph $G$) is the smallest set of nodes $M$ such that $X_i \perp \! \! \! \perp \text{rest} | M$.

- For UGM: $M = \{j: \exists i, j \in E \}$ is the set of neighbors of $i$.

- For DGM: $M = \pi_i \cup \text{children}(i) \cup \text{children}(i)$ similar to UGM.

**Property:**

<table>
<thead>
<tr>
<th>DGM</th>
<th>UGM</th>
</tr>
</thead>
<tbody>
<tr>
<td>factorization: $p(x) = \prod p_{x_i</td>
<td>\text{pa}(i)}$</td>
</tr>
<tr>
<td>cond. indep.: $d$-separation</td>
<td>$\mathbb{C}$-separation</td>
</tr>
<tr>
<td>marginalization: not closed in general</td>
<td>closed (connect all neighbors of removed node)</td>
</tr>
<tr>
<td>Cannot</td>
<td>Cannot</td>
</tr>
<tr>
<td>exactly</td>
<td>exactly</td>
</tr>
<tr>
<td>capture some</td>
<td>capture some</td>
</tr>
<tr>
<td>families</td>
<td>families</td>
</tr>
</tbody>
</table>

**Moralization:**

- Let $G$ be a DAG; when can we get an equivalent UGM?

- Def: For $G$ a DAG, we call $\tilde{G}$ the moralized graph of $G$

  where $\tilde{G}$ is an undirected graph with same $V$

  and $E = \{ x_i, x_j | (i,j) \in E \}$ undirected version of $E$

  $U \{ x_k, x_l | x_k \in \pi_i$ for some $i \}$ moralization

- Connect all the parents of $i$ with $i$ in big clique

  only needed if $|\pi(i)| > 1$
for a DAG $G$ with no v-shape:

\[ Z(G) = Z(\overline{G}) \]

but in general, can only say $Z(G) \leq Z(\overline{G})$

\[ \text{note that } G \text{ is the minimal undirected graph s.t. } Z(G) \leq Z(\overline{G}) \]

\[ p(x) = \frac{\Pi_{x_i \in V} p(x_i | x_{\Pi_i})}{\Pi_{x_i \in C} p(x_i)} \]

where $C = \emptyset \cup \Pi_i$.

**General themes in this class**

A) representation $\rightarrow$ DGM $\rightarrow$ UGM

\[ \text{parameterization} \rightarrow \text{exponential family} \]

B) inference $p(x) \rightarrow$ today: elimination algorithm

\[ \text{"easy" variable} \rightarrow \text{next: sum-product / belief propagation} \]

\[ \text{later: approximate inference e.g. MCMC} \]

C) statistical estimation $\rightarrow$ MLE

\[ \text{learning} \rightarrow \text{maximum entropy} \]

\[ \text{method of moments} \]

---

Inference:

want to compute

a) marginal $p(x_F)$ for some $F \subseteq V$
b) conditional: \( p(X = |X_E) \)

\text{query: "latent"}

\( C \) for UGM: partition function \( Z = \sum_{X_e} (\mathcal{Z} \cdot \mathcal{y}_c(x)) \)

why? - missing data: \( p(x_{\text{inds | xds}}) \)

\( L \rightarrow \text{example} \)

\( \text{prediction} p(x_{\text{value | xds}}) \)

\text{"latent cause"}

\( p(x_{\text{cause | xds}}) \)

\( \star \) also related instance

\( \arg \max_{X_E} p(x_f | X_E) \)

\( \text{could be big (UTIibi dig.) (e.g. speech recognition)} \)

\( \star \) inference is also needed during estimation (parameter fitting ACE)

[eg during E-step \( p(z | x) \) ]

\( \star \) present inference alg for UGM [for simplicity and more generality]

\( \text{but note that sampling it’s more efficient to work directly with DGM} \)

make DGM \( \rightarrow \subseteq \text{UGM via moralize} \)

\text{i.e. } p \in \text{DGM }; \quad p(z) = \prod_i p(x_i | z_{\text{pa}(x_i)})

\text{moralize: } G^\circ \subseteq G^\circ \cup \zeta

\begin{align*}
p(x_i) &= \sum_{z | x_i:z} p(z) = \sum_{x_i:z} \frac{1}{Z} \cdot \mathcal{y}_c(x_i) \\
Z &= 1
\end{align*}

\text{Graph elimination alg. (for inference in generic UGM)}

\text{consider } p \in S(\zeta) \quad p(x) = \frac{1}{Z} \cdot \mathcal{Z}(\mathcal{y}_c(x))

\text{undirected}

\text{say want to compute } p(x_F) \text{ for FSV "query nodes"}

\text{main trick: use distributivity of } \otimes \text{ over } \otimes \rightarrow \mathcal{Z}(a \otimes b) = \mathcal{Z}(a) \otimes \mathcal{Z}(b)
\[ f(x_1)g(x_2) = \left( \sum_{x_1} f(x_1) \right) \left( \sum_{x_2} g(x_2) \right) \]

more generally
\[ \sum_{x_i \in \mathbb{X}} f_i(x_i) = \prod_{i=1}^n \left( \sum_{x_i} f_i(x_i) \right) \]
\[ = \mathcal{O}(\mathbb{X}^n) \]
\[ = \mathcal{O}(\mathbb{X}^n) \]

\[ p(x_4) = \frac{1}{Z} \sum_{x_1, x_2, x_3} \psi(x_1, x_2) \psi(x_2, x_3) \psi(x_3, x_4) \psi(x_1, x_3) \psi(x_2, x_4) \]
\[ = \frac{1}{Z} \left( \sum_{x_3} \psi(x_3, x_4) \psi(x_1, x_3) \psi(x_2, x_4) \right) \]
\[ = \frac{1}{Z} \sum_{x_3} \psi(x_3, x_4) \psi(x_1, x_3) \psi(x_2, x_4) \]
\[ \Rightarrow m_3(x_4) \]
\[ \Rightarrow \text{joint } \]
\[ \Rightarrow \text{marginal } \]
\[ \Rightarrow p(x_4) = \frac{m_3(x_4)}{Z} \]

---

**general alg.: graph Eliminate**

1. choose an elimination ordering s.t. \( F \) are the last nodes
2. put all \( \psi(x_i) \) on "active list"
3. repeat in order of variables to eliminate
   - (say \( x_f \) is variable to eliminate)
     1. remove all factors from active active list with \( x_i \) in it (take their product)
     2. sum over \( x_f \) to get a new factor \( m_f(x_f) \) (think as \( \psi_{x_f}(x_f) \))

\[ \prod_{x_i \in \text{st.}} \psi(x_i(x_f)) \]
\[ \prod_{x_i \in \text{st.}} \psi(x_i(x_f)) \]
\[ \prod_{x_i \in \text{st.}} \psi(x_i(x_f)) \]
get \( M_i(x_{S_i}) \leq \frac{1}{x_i} \sum_{s \in S_i} \sum_{v \in o(x_{S_i})} m_i (x_v) \)

\( S_i = (U_d) \setminus \text{slice}\)

3) put back \( m_i(x_{S_i}) \) in active list \( M_S / x_{S_i} \)

"normalize" of last product of factors left has only \( x_T \Rightarrow \) proportional \( p(x_T) \)

memory needed \( \approx 2 \cdot \frac{|f| \cdot |S_i| + 1}{n} \)

computational cost \( \approx 2 \cdot \frac{1}{n} \)

later, related "treewidth" of a graph

"augmented graph"

graph obtained by running graph eliminate + keeping track of all edges added

for a fixed ordering

note: augmented graph after graph eliminate is always a triangulated graph

def: graph with no cycle of size 4 or more that cannot be broken by a "clique"

\( \text{treewidth of a graph} \leq \min \) of all possible clique in augmented graph

\( \text{size of largest -1} \)

\( \rightarrow \) conventional treewidth (tree) = 1

\( \text{both memory} \& \text{running time of graph eliminate is dominated by} \ 2^{\text{size of clique}} \)

best ordering gives \( 2^{\text{treewidth} + 1} \)

not all orderings are good

bad news:

\( \text{we need to compute treewidth of a graph search (articulation point)} \)
a) NP hard to compute treewidth of general graph (or find best ordering)
b) NP hard to do (exact) inference in general DAG

\[ \Rightarrow \text{approximate methods} \]

Example:
- treewidth of a grid

\[ \approx \sqrt{N} \]

\[ \text{Soda} \]

\[ \text{treewidth} \]

\[ \text{Good news} \]

- inference is linear time for trees (treewidth = 1) (**sum-product alg**)

\[ |V| + |E| \]

(\text{HMM, hidden chain, etc.})

- efficient for \text{"small treewidth graph"}

\[ \Rightarrow \text{we join tree alg} \]