Probabilistic Graphical Models (2): Inference

Qinfeng (Javen) Shi

The Australian Centre for Visual Technologies,
The University of Adelaide, Australia

6 May 2011

Course Outline

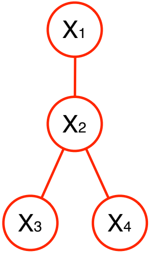
Probabilistic Graphical Models:

1. Representation
2. Inference (Today)
3. Learning
4. Sampling-based approximate inference
5. Temporal models
6. . . .

Inference

- Marginals and MAP
- Variable Elimination (covered in the previous talk)
- Max/sum-product (Message Passing, (Loopy) BP)
- Junction Tree Algorithm
- Linear Programming (LP) Relaxations
- Graph Cut
- ...

Marginal and MAP



Marginal inference: $P(x_i) = \sum_{x_j : j \neq i} P(x_1, x_2, x_3, x_4)$

MAP inference: $(x_1^*, x_2^*, x_3^*, x_4^*) = \operatorname*{argmax}_{x_1, x_2, x_3, x_4} P(x_1, x_2, x_3, x_4)$

In general, $x_i^* \neq \operatorname*{argmax}_{x_i} P(x_i)$

Marginals

When do we need marginals? Recall sum-product gives marginals by $q(x_i) = \psi(x_i) \prod_{j \in Ne(i)} m_{j \to i}(x_i)$, for $P(x_i) = \frac{1}{Z} q(x_i)$. Marginals are used to compute

- normalisation constant
  $Z = \sum_{x_i} q(x_i) = \sum_{x_j} q(x_j) \;\; \forall i, j = 1, \ldots$.
  log loss in CRFs is $-\log P(x_1, \ldots, x_n) = \log(Z) + \ldots$
- expectations like $\mathbb{E}_{P(x_i)}[\phi(x_i)]$ and $\mathbb{E}_{P(x_i, x_j)}[\phi(x_i, x_j)]$, where $\psi(x_i) = \langle \phi(x_i), w \rangle$ and $\psi(x_i, x_j) = \langle \phi(x_i, x_j), w \rangle$
  Gradient of CRFs risk contains above expectations.

MAP

When do we need MAP?

- find the most likely configuration for $(x_i)_{i \in \mathcal{V}}$ in testing.
- find the most violated constraint generated by $(x_i^\dagger)_{i \in \mathcal{V}}$ in training (*i.e.* learning), *e.g.* by cutting plane method (used in SVM-Struct) or by Bundle method for Risk Minimisation (Teo JMLR2010).

Max-product

$$P(x_1^*, x_2^*, x_3^*, x_4^*) = \max_{x_1,x_2,x_3,x_4} P(x_1, x_2, x_3, x_4)$$

$$= \max_{x_1,x_2,x_3,x_4} \psi(x_1, x_2)\psi(x_2, x_3)\psi(x_2, x_4)\psi(x_1)\psi(x_2)\psi(x_3)\psi(x_4)$$

$$= \max_{x_1,x_2} \left[\dots \max_{x_3} \Big(\psi(x_2, x_3)\psi(x_3) \Big) \max_{x_4} \Big(\psi(x_2, x_4)\psi(x_4) \Big) \right]$$

$$= \max_{x_1} \left[\psi(x_1) \max_{x_2} \Big(\psi(x_2)\psi(x_1, x_2) m_{3\to 2}(x_2) m_{4\to 2}(x_2) \Big) \right]$$

$$= \max_{x_1} \Big(\psi(x_1) m_{2\to 1}(x_1) \Big) \Rightarrow x_1^* = \operatorname*{argmax}_{x_1} \Big(\psi(x_1) m_{2\to 1}(x_1) \Big)$$

$$x_i^* = \operatorname*{argmax}_{x_i} \Big(\psi(x_i) \prod_{j\in Ne(i)} m_{j\to i}(x_i) \Big)$$

$$m_{j\to i}(x_i) = \max_{x_j} \Big(\psi(x_j)\psi(x_i, x_j) \prod_{k\in Ne(j)\setminus\{i\}} m_{k\to j}(x_j) \Big)$$

Max-product

Max/sum-product is also known as Message Passing and Belief Propagation (BP).

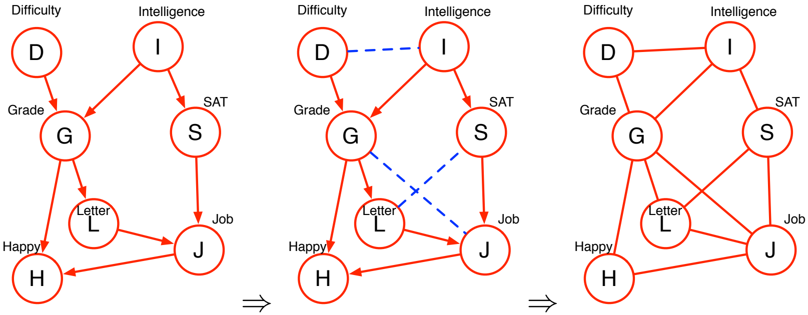
In graphs with loops, running BP for several iterations is known as Loopy BP (neither convergence nor optimal guarantee in general).

Junction Tree Algorithm

- moralise (directed acyclic graph only)
- triangulate (turn unchordal graphs to chordal ones)
- construct junction tree (clique tree)
- pick a clique as the root clique.
- send message from the root to leaves, and send messages from leaves to the root.
- read marginals from junction tree and messages.

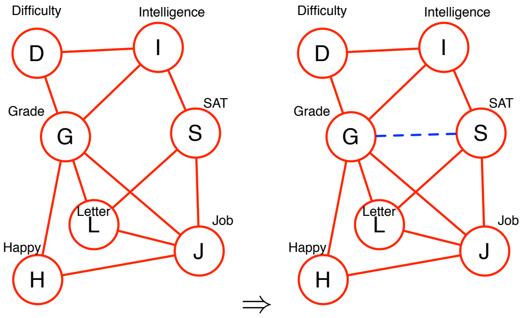
Junction Tree Algorithm - moralise

Moralisation: connect the common parents, and turn all edges to undirected ones.



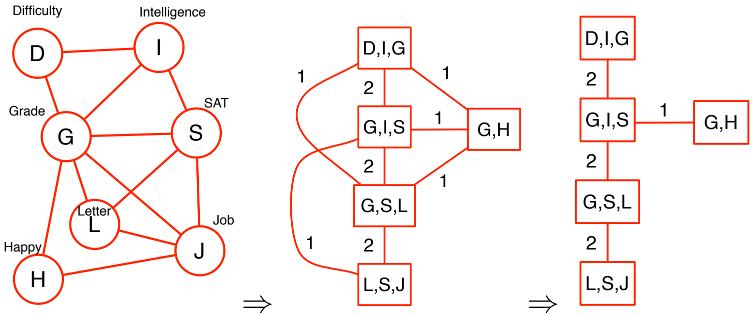
Junction Tree Algorithm - triangulate

Chordal if there is no cycle of length > 3.
Triangulation: keep adding short cut edges to cycles until the graph's chordal.

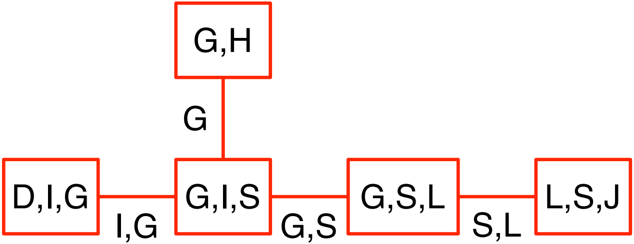


Junction Tree Algorithm - construct junction tree

build a clique tree and then find the maximal spanning tree



Junction Tree Algorithm - message passing

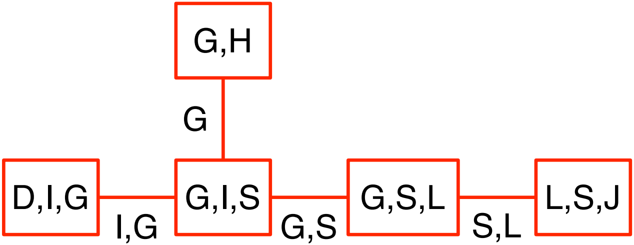


$$P(\mathbf{C}) = \frac{1}{Z}\prod_{c\in\mathbf{C}}\psi(c), \quad c_1 = \{D, I, G\}, c_2 = \{G, I, S\}, \ldots$$

$$P(H) = \frac{1}{Z}\sum_{D,I,G,S,L,J}\prod_{c\in\mathcal{C}}\psi(c)$$

re-arrange $\sum$ to eliminate variables

Junction Tree Algorithm - message passing



$$P(c_r) = \frac{1}{Z} \sum_{\mathbf{C} \setminus c_r} \Big(\prod_{c \in Ne(c_r)} m_{c \to c_r}(c_r) \Big)$$

$$m_{c_s \to c_t}(c_t) = \sum_{c_s \setminus (c_s \cap c_t)} \Big(\prod_{c \in Ne(c_s) \setminus c_t} m_{c \to c_s}(c_s) \Big)$$

LP Relaxations

Assume pairwise MRFs with graph $G(\mathcal{V}, \mathcal{E})$

$$
\begin{aligned}
P(\mathbf{X} \mid \mathbf{Y}) &= \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} \psi_{i,j}(x_i, x_j) \prod_{i\in\mathcal{V}} \psi_i(x_i) \\
&= \frac{1}{Z} \exp\left(- \sum_{(i,j)\in\mathcal{E}} E_{i,j}(x_i, x_j) - \sum_{i\in\mathcal{V}} E_i(x_i) \right)
\end{aligned}
$$

$$
\begin{aligned}
\text{MAP } \mathbf{X}^* &= \operatorname*{argmax}_{\mathbf{X}} \prod_{(i,j)\in\mathcal{E}} \psi_{i,j}(x_i, x_j) \prod_{i\in\mathcal{V}} \psi_i(x_i) \\
&= \operatorname*{argmin}_{\mathbf{X}} \sum_{(i,j)\in\mathcal{E}} E_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} E_i(x_i)
\end{aligned}
$$

LP Relaxations

$$\underset{\mathbf{x}}{\text{argmin}} \sum_{(i,j)\in\mathcal{E}} E_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} E_i(x_i)$$

$\Leftrightarrow$ the following Integer Program:

$$\underset{\{q\}}{\text{argmin}} \sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} q_{i,j}(x_i, x_j) E_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} \sum_{x_i} q_i(x_i) E_i(x_i)$$

$$\text{s.t.} \quad q_{i,j}(x_i, x_j) \in \{0, 1\}, \sum_{x_i,x_j} q_{i,j}(x_i, x_j) = 1, \sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j).$$

Relax to Linear Program:

$$\underset{\{q\}}{\text{argmin}} \sum_{(i,j)\in\mathcal{E}} \sum_{x_i,x_j} q_{i,j}(x_i, x_j) E_{i,j}(x_i, x_j) + \sum_{i\in\mathcal{V}} \sum_{x_i} q_i(x_i) E_i(x_i)$$

$$\text{s.t.} \quad q_{i,j}(x_i, x_j) \in [0, 1], \sum_{x_i,x_j} q_{i,j}(x_i, x_j) = 1, \sum_{x_i} q_{i,j}(x_i, x_j) = q_j(x_j).$$

Examples using PGM inference

Show papers in

- Image scene understanding
- Semantic video understanding

More inference methods including graph cut will be covered in Advanced Topics or in discussion.
Next talk: Learning in graphical models.