MAP Inference: Exact Algorithms

Probabilistic Graphical Models
Sharif University of Technology
Spring 2017

Soleymani
MAP query

- $X = X \cup E$
- **MAP query:**
  
  $$X^* = \arg\max_X P(X|e)$$

  Most probable assignment for all of (non-evidence) variables

- **Marginal MAP query** ($X = Y \cup Z$):
  
  $$Y^* = \arg\max_Y P(Y|e)$$

  Most probable assignment for some variables of interest given an evidence $E = e$

- **Marginal MAP is a much harder problem**
MAP queries: application

- MAP queries are often used as a way of “filling in” unknown information
  - Using the probabilistic model, it finds the most probable assignment

- Example: speech recognition
  - we are trying to decode the most likely words given the (noisy) acoustic signal
Max-product: distributive law

- Max operator distributes over product:

\[
\max(a \times b, a \times c) = a \times \max(b, c)
\]

- Distributive law: If \(X \notin \text{Scope}(\phi_1)\) then

\[
\max_X \phi_1 \cdot \phi_2 = \phi_1 \cdot \max_X \phi_2
\]
Max-Sum and Max-Product

Max-product inference:

\[
\max \prod_k \phi_k(x_{c_k})
\]

Max-product is equivalent to the following max-sum:

\[
\max \sum_k \log \phi_k(x_{c_k})
\]

Max operator distributes over sum:

\[
\max(a + b, a + c) = a + \max(b, c)
\]
MAP-Elimination

\[
\max_{a,b,c} P(a, b, c, d) = \max_a \max_b \max_c P(a)P(b|a)P(c|b)P(d|c)
\]

\[
= \max_c \max_b P(c|b)P(d|c) \max_a P(a)P(b|a)
\]

\[
= \max_c P(d|c) \max_b P(c|b) m_a^{\text{max}}(b)
\]

\[
= \max_c P(d|c) m_b^{\text{max}}(c)
\]

\[
= m_c^{\text{max}}(d)
\]
Max-product on trees

\[ m_{ij}^{\text{max}}(x_i) = \max_{x_i} \left( \phi(x_i) \phi(x_i, x_j) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}^{\text{max}}(x_i) \right) \]

\[ \max_{x} P(x|e) \propto \max_{x} P(x, e) \]
\[ = \max_{x_r} \phi(x_r) \prod_{k \in \mathcal{N}(r)} m_{kr}^{\text{max}}(x_r) \]

- For finding the map probability \( \max_{x} P(x, e) \) the first pass (i.e., inward pass to the root) of message-passing algorithm is sufficient
Finding maximizing configuration (most probable assignment):

$$x^* \in \arg\max_x P(x|e)$$

We need second pass (i.e., outward pass) to find most probable configuration:

$$x_r^* \in \arg\max_{x_r} \phi(x_r) \prod_{i \in N(r)} m_{ir}^{\text{max}}(x_r)$$

$$x_i^* \in \arg\max_{x_i} \phi(x_i) \phi(x_i, x_r^*) \prod_{k \in N(i) \setminus r} m_{ki}^{\text{max}}(x_i)$$
Maximum Posterior Assignment

- We can use a traceback procedure that incrementally builds a MAP assignment, one variable at a time to decode the MAP assignment
  - The root variable is the first variable that can be decoded.
  - Other variables are decoded in the outward pass.
Maximum Posterior Assignment: Bookkeaping and traceback

- We choose an $x_r^*$ such that:

$$x_r^* \in \arg\max_{x_r} \phi(x_r) \prod_{i \in \mathcal{N}(r)} m_{ir}^{\max}(x_r)$$

- To have a consistent maximizing configuration, we define $c_{ij}(x_j)$ and save it during the inward pass:

$$c_{ij}(x_j) \in \arg\max_{x_i} \left( \phi(x_i) \phi(x_i, x_j) \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}^{\max}(x_i) \right)$$

- Then, in the outward pass, we can find for each $i \in \mathcal{N}(r)$:

$$x_i^* = c_{ir}(x_r^*)$$

- This procedure continues outward to the leaves.
Junction-tree for MAP Inference: Message passing view

\[ m_{ij}^{\text{max}}(S_{ij}) = \max_{c_i \setminus S_{ij}} \left( \psi_i \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}^{\text{max}}(S_{ki}) \right) \]

\[ \max_x P(x|e) \propto \max_x P(x, e) = \max_{x_r} \psi_r \prod_{k \in \mathcal{N}(r)} m_{kr}^{\text{max}}(S_{kr}) \]
Junction-tree for MAP Inference: Belief update view

- Replace sum with max in the update equations:

\[
\phi^*_S = \max_{V \setminus S} \psi_V
\]

\[
\psi^*_W = \frac{\phi^*_S}{\phi_S} \psi_W
\]

- For each clique, we can found at the end of the algorithm:

\[
\psi_C = \max_{V \setminus C} P(x)
\]
Computational complexity of marginal MAP

Unfortunately, the max and sum operations do not commute

Marginal MAP is computationally much more expensive than VE for pure sum-product or pure max-product.
   Indeed, we are not free to choose an arbitrary elimination ordering.

Constrained elimination ordering
   a smaller range of legal elimination orderings
   Indeed, we *must* perform all the variable summations before we can perform any of the variable maximizations.
   Constraints might cause significantly larger tree width than a good unconstrained ordering.
Marginal MAP is harder than pure sum-product or max-product: example

- Even on very simple polytree networks, elimination algorithms can require exponential time to solve a marginal MAP query

\[ y^{m-map} = \arg \max_{Y_1, \ldots, Y_n} \sum_{X_1, \ldots, X_n} P(Y_1, \ldots, Y_n, X_1, \ldots, X_n) \]

- Size of the largest maximal clique (for each legal elimination order) will be \( n \) instead of 3 (that can be obtained when we have no constraint on the order)
References