Inference and Representation

David Sontag

New York University

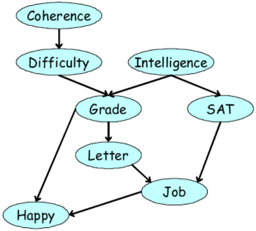
Lecture 5, Sept. 30, 2014

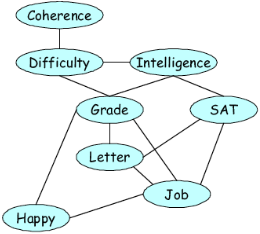
Today's lecture

1. Running-time of variable elimination
   - Elimination as graph transformation
   - Fill edges, width, treewidth
2. Sum-product belief propagation (BP)
   Done on blackboard
3. Max-product belief propagation
4. Loopy belief propagation

Running time of VE in graph-theoretic concepts

- Let's try to analyze the complexity in terms of the graph structure
- G_Φ is the undirected graph with one node per variable, where there is an edge (X_i, X_j) if these appear together in the scope of some factor ϕ
- Ignoring evidence, this is either the original MRF (for sum-product VE on MRFs) or the moralized Bayesian network:





Elimination as graph transformation

When a variable X is eliminated,

- We create a single factor ψ that contains X and all of the variables $\mathbf{Y}$ with which it appears in factors
- We eliminate X from ψ, replacing it with a new factor τ that contains all of the variables $\mathbf{Y}$, but not X. Let's call the new set of factors Φ_X

How does this modify the graph, going from G_Φ to G_{Φ_X}?

- Constructing ψ generates edges between all of the variables $Y \in \mathbf{Y}$
- Some of these edges were already in G_Φ, some are new
- The new edges are called **fill edges**
- The step of removing X from Φ to construct Φ_X removes X and all its incident edges from the graph

Induced graph

- We can summarize the computation cost using a single graph that is the union of all the graphs resulting from each step of the elimination
- We call this the **induced graph** $\mathcal{I}_{\Phi,\prec}$, where $\prec$ is the elimination ordering

Chordal Graphs

Graph is **chordal**, or triangulated, if every cycle of length ≥ 3 has a shortcut (called a "chord")

Theorem: Every induced graph is chordal
Proof: (by contradiction)

- Assume we have a chordless cycle $X_1 - X_2 - X_3 - X_4 - X_1$ in the induced graph
- Suppose X_1 was the first variable that we eliminated (of these 4)
- After a node is eliminated, no fill edges can be added to it. Thus, $X_1 - X_2$ and $X_1 - X_4$ must have pre-existed
- Eliminating X_1 introduces the edge $X_2 - X_4$, contradicting our assumption

Chordal graphs

- **Thm:** Every induced graph is chordal
- **Thm:** Any chordal graph has an elimination ordering that does not introduce any fill edges

Algorithm 9.3 Maximum Cardinality Algorithm for constructing an elimination ordering

Procedure Max-Cardinality (
  $\mathcal{H}$ // An undirected graph over $\mathcal{X}$
)
  Initialize all nodes in $\mathcal{X}$ as unmarked
  for $k = |\mathcal{X}| \ldots 1$
    $X \leftarrow$ unmarked variable in $\mathcal{X}$ with largest number of marked neighbors
    $\pi(X) \leftarrow k$
    Mark X
  return π

(The elimination ordering is REVERSE)

- **Conclusion:** Finding a good elimination ordering is equivalent to making graph chordal with minimal width

Today's lecture

1. Running-time of variable elimination
   - Elimination as graph transformation
   - Fill edges, width, treewidth
2. Sum-product belief propagation (BP)
   Done on blackboard
3. Max-product belief propagation
4. Loopy belief propagation

MAP inference

- Recall the MAP inference task,

$$\arg\max_{\mathbf{x}} p(\mathbf{x}), \qquad p(\mathbf{x}) = \frac{1}{Z}\prod_{c \in C} \phi_c(\mathbf{x}_c)$$

  (we assume any evidence has been subsumed into the potentials, as discussed in the last lecture)

- Since the normalization term is simply a constant, this is equivalent to

$$\arg\max_{\mathbf{x}} \prod_{c \in C} \phi_c(\mathbf{x}_c)$$

  (called the *max-product* inference task)

- Furthermore, since log is monotonic, letting $\theta_c(\mathbf{x_c}) = \lg \phi_c(\mathbf{x_c})$, we have that this is equivalent to

$$\arg\max_{\mathbf{x}} \sum_{c \in C} \theta_c(\mathbf{x}_c)$$

  (called *max-sum*)

Semi-rings

- Compare the sum-product problem with the max-product (equivalently, max-sum in log space):

$$\text{sum-product} \qquad \sum_{\mathbf{x}} \prod_{c \in C} \phi_c(\mathbf{x}_c)$$

$$\text{max-sum} \qquad \max_{\mathbf{x}} \sum_{c \in C} \theta_c(\mathbf{x}_c)$$

- Can exchange operators $(+, *)$ for $(\max, +)$ and, because both are semirings satisfying associativity and commutativity, everything works!
- We get "max-product variable elimination" and "max-product belief propagation"

Simple example

- Suppose we have a simple chain, $A - B - C - D$, and we want to find the MAP assignment,

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

- Just as we did before, we can push the maximizations inside to obtain:

$$\max_{a,b} \phi_{AB}(a,b) \max_{c} \phi_{BC}(b,c) \max_{d} \phi_{CD}(c,d)$$

  or, equivalently,

$$\max_{a,b} \theta_{AB}(a,b) + \max_{c} \theta_{BC}(b,c) + \max_{d} \theta_{CD}(c,d)$$

  [*Illustrate factor max-marginalization on board.*]

- To find the actual maximizing assignment, we do a traceback (or keep back pointers)

Max-product variable elimination

Procedure Max-Product-VE (
  Φ, // Set of factors over $\boldsymbol{X}$
  $\prec$ // Ordering on $\boldsymbol{X}$
)
  Let $X_1, \ldots, X_k$ be an ordering of $\boldsymbol{X}$ such that
    $X_i \prec X_j$ iff $i < j$
    for $i = 1, \ldots, k$
      $(\Phi, \phi_{X_i}) \leftarrow$ Max-Product-Eliminate-Var(Φ, X_i)
    $\boldsymbol{x}^* \leftarrow$ Traceback-MAP$(\{\phi_{X_i} \,:\, i = 1, \ldots, k\})$
    return $\boldsymbol{x}^*, \Phi$ // Φ contains the probability of the MAP

Procedure Max-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)
  $\Phi' \leftarrow \{\phi \in \Phi \,:\, Z \in \mathit{Scope}[\phi]\}$
  $\Phi'' \leftarrow \Phi - \Phi'$
  $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
  $\tau \leftarrow \max_Z \psi$
  return $(\Phi'' \cup \{\tau\}, \psi)$

Procedure Traceback-MAP (
  $\{\phi_{X_i} \,:\, i = 1, \ldots, k\}$
)
  for $i = k, \ldots, 1$
    $\boldsymbol{u}_i \leftarrow (x^*_{i+1}, \ldots, x^*_k)\langle \mathit{Scope}[\phi_{X_i}] - \{X_i\}\rangle$
      // The maximizing assignment to the variables eliminated after X_i
    $x^*_i \leftarrow \arg\max_{x_i} \phi_{X_i}(x_i, \boldsymbol{u}_i)$
      // x^*_i is chosen so as to maximize the corresponding entry in the factor, relative to the previous choices $\boldsymbol{u}_i$
  return $\boldsymbol{x}^*$

Max-product belief propagation (for tree-structured MRFs)

- Same as sum-product BP except that the messages are now:

$$m_{j\to i}(x_i) = \max_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- After passing all messages, can compute single node *max-marginals*,

$$m_i(x_i) = \phi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i) \qquad \propto \qquad \max_{\mathbf{x}_{V\setminus i}} p(\mathbf{x}_{V\setminus i}, x_i)$$

- If the MAP assignment $\mathbf{x}^*$ is **unique**, can find it by locally decoding each of the single node max-marginals, i.e.

$$x_i^* = \arg\max_{x_i} m_i(x_i)$$

Max-sum belief propagation (for tree-structured MRFs)

- Same as sum-product BP except that the messages are now:

$$m_{j\to i}(x_i) = \max_{x_j} \theta_j(x_j) + \theta_{ij}(x_i, x_j) + \sum_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- After passing all messages, can compute single node *max-marginals*,

$$m_i(x_i) = \theta_i(x_i) + \sum_{j\in N(i)} m_{j\to i}(x_i) \qquad = \qquad \max_{\mathbf{x}_{V\setminus i}} \log p(\mathbf{x}_{V\setminus i}, x_i) + C$$

- If the MAP assignment $\mathbf{x}^*$ is **unique**, can find it by locally decoding each of the single node max-marginals, i.e.

$$x_i^* = \arg\max_{x_i} m_i(x_i)$$

- Working in log-space prevents numerical underflow/overflow

Implementing sum-product in log-space

- Recall the sum-product messages:

$$m_{j\to i}(x_i) = \sum_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- Making the messages in log-space corresponds to the update:

$$m_{j\to i}(x_i) = \log \sum_{x_j} \exp(\theta_j(x_j) + \theta_{ij}(x_i, x_j) + \sum_{k\in N(j)\setminus i} m_{k\to j}(x_j))$$

$$= \log \sum_{x_j} \exp(T(x_i, x_j)),$$

where $T(x_i, x_j) = \theta_j(x_j) + \theta_{ij}(x_i, x_j) + \sum_{k\in N(j)\setminus i} m_{k\to j}(x_j)$

- Letting $c_{x_i} = \max_{x_j} T(x_i, x_j)$, this is equivalent to

$$= c_{x_i} + \log \sum_{x_j} \exp(T(x_i, x_j) - c_{x_i}),$$

Exactly solving MAP, beyond trees

- MAP as a discrete optimization problem is

$$\arg\max_{\mathbf{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j)$$

- Very general discrete optimization problem – many hard combinatorial optimization problems can be written as this (e.g., 3-SAT)
- Studied in operations research communities, theoretical computer science, AI (constraint satisfaction, weighted SAT), etc.
- Very fast moving field, both for theory and heuristics