15-750: Graduate Algorithms	February 03, 2017

Lecture 8: Dynamic Programming II: Inference on Graphical Models

Lecturer: David Witmer *Scribe: Zhengbo Li*

1 Recall: Dynamic programming steps from last class

1. Define subproblems.
2. Write solution to subproblem recursively in terms of solutions to smaller subproblems.
3. Prove that this recurrence is correct using induction.
4. Determine runtime.

2 Lecture outline

- Factor graphs and examples.
- Inference tasks for factor graphs.
- Efficient inference on trees using dynamic programming.

Reference: Mezard and Montanari, *Information, Physics, and Computation*, Chapters 9 and 14. Available online

3 Factor graphs

Let P be a probability distribution on $\{0,1\}^n$ with the following form:

$$P(x) = \frac{1}{Z}\prod_{a=1}^{m} \psi_a(X_{\partial a})$$

where:

- Z is a normalization factor, i.e, $Z = \sum_{X \in \{0,1\}^n} \prod_{a=1}^{m} \psi_a(X_{\partial a})$
- $\partial a \subseteq [n]$
- $|\partial a| = k_a$
- $X_{\partial a} = (X_{(\partial a)_1}, X_{(\partial a)_2}, \ldots, X_{(\partial a)_{k_n}})$
- $\psi_a : \{0,1\}^{\partial a} \to \mathbb{R}^{\geq 0}$, capture dependencies, relationships among variables.

3.1 Examples

Example 3.1 (Medical diagnosis)**.**

Variables

Notation	1	0
c	if I have a cold	otherwise
s	if I have a sore throat	otherwise
r	if I have a runny noise	otherwise
p	if there is pollen in the are	otherwise

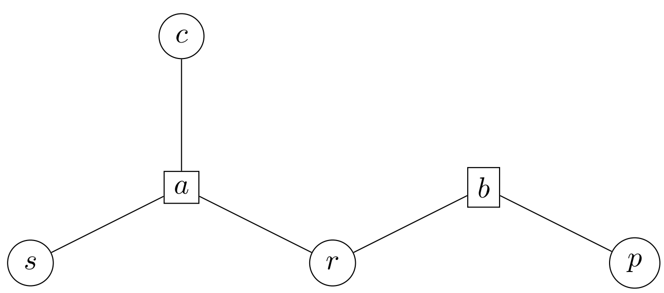
Functions c, s, and r are related: If I have a cold, I am more likely to have a runny noise.

c	s	r	$\psi_a(c,s,r)$
1	1	1	0.1
0	0	1	0.3
0	0	1	0.6

r, p are related: If there is pollen in the air, I am more likely to have a runny noise.

r	p	$\psi_b(r,p)$
0	1	0.1
1	0	0.2
1	1	0.3
0	0	0.4

Factor graph



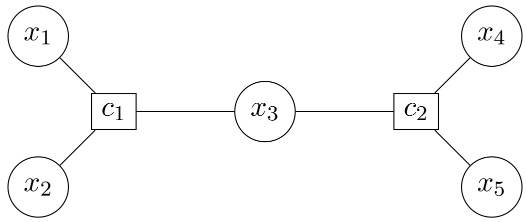
$$P(c,s,r,p) = \frac{1}{Z}\psi_a(c,s,r)\psi_b(r,p)$$

We can ask the following questions:

- What is the probability that I have a cold given that my nose is running? $P(c = 1 \mid r = 1)$.
- What is the probability that I have cold given that my noise is running and there is pollen? $P(c = 1 \mid r = 1, p = 1)$.

Example 3.2 (3-SAT)**.** We have n variables $x_i \in \{0,1\}$, $2n$ literals $\{x_i, \bar{x_i}\}$, and m clauses, e.g., $x_i \vee x_j \vee \bar{x_k}$. We want to know whether there is an assignment to variables satifying all m clauses, e.g., $(x_1 \vee \bar{x_2} \vee x_3) \wedge (x_3 \vee \bar{x_4} \vee \bar{x_5})$.

Represent 3-SAT instance by a factor graph as follows:



ψ_{c_1} is indicator for clause 1 being satisfied: $\psi_{c_1}(x_1, x_2, x_3) = x_1 \vee \bar{x_2} \vee x_3$.
ψ_{c_2} is indicator for clause 2 being satisfied: $\psi_{c_2}(x_3, x_4, x_5) = x_3 \vee x_4 \vee x_5$.
Then

$$P(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z}\psi_{c_1}(x_1, x_2, x_3)\psi_{c_2}(x_3, x_4, x_5).$$

We have the following possible tasks:

- Compute $Z = \sum_{x \in \{0,1\}^5} \psi_{c_1}(x_1, x_2, x_3)\psi_{c_2}(x_3, x_4, x_5)$, which is the number of satisfying assignments.
- Compute $P(x_i = 1)$, which is the probability that a satisfying assignment sets $x_i = 1$.
- Sample a satisfying assignment.

3.2 Definition

Factor graphs are bipartite graphs composed of two sets of nodes: variable nodes $[n]$ and factor nodes $[m]$. Let ∂v denote the set of v's neighbours and we have $\psi_a : \{0,1\}^{\partial a} \to \mathbb{R}^{\geq 0}$ for each factor node. The corresponding distribution is:

$$P(x) = \frac{1}{Z}\prod_{a=1}^{m} \psi_a(x_{\partial a}).$$

3.3 Tasks:

- Compute marginals $P(x_i = 1)$.
- Compute conditional marginals $P(x_i = 1 \mid x_j = 0)$.
- Sample from distribution.
- Find mode: $\text{argmax}_x P(x)$

3.4 Assumptions:

- $k_a = |\partial a| = O(1)$.
- Factor graph is a tree. In this case we can use dynamic programming to do all of the above efficiently.

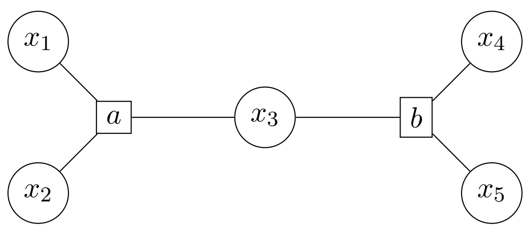
4 Compute marginals

Say we want to compute marginal distribution of x_1,

$$P(x_i = 1) = \frac{1}{Z} \sum_{\substack{x \in \{0,1\}^n \\ x_1 = 1}} \prod_{a=1}^{m} \psi_a(x_{\partial a}).$$

The naive algorithm computes all 2^{n-1} terms of the sum and adds them up. We will use dynamic programming to do better.

Consider the following example:



$$P(x_1 = 1) = \frac{1}{Z} \sum_{x_2, x_3, x_4, x_5 \in \{0,1\}} \psi_a(1, x_2, x_3) \psi_b(x_3, x_4, x_5)$$

To compute this, we need to compute the 16 terms of the sum, requiring 32 computations of the ψ's. On the other hand, consider

$$P(x_1 = 1) = \frac{1}{Z} \sum_{x_2, x_3 \in \{0,1\}} \psi_a(1, x_2, x_3) \sum_{x_4, x_5 \in \{0,1\}} \psi_b(x_3, x_4, x_5)$$

.

We can compute $\sum_{x_4, x_5 \in \{0,1\}} \psi_b(x_3, x_4, x_5)$ for $x_3 = 0$ and $x_3 = 1$, respectively, by doing 8 computations of ψ's. We can then do 4 additional computations of ψ's to get $P(x_1 = 1)$, for a total of 12 computations of ψ's.

Idea: Reorder sums and products to reuse computation (dynamic programming).

4.1 Define subproblems

Pick an arbitrary variable as the root and we get a tree. Define the subproblem $\nu_i(b)$ as the marginal distribution for $x_i = b$ in the factor graph corresponding to subtree rooted at x_i.

$$\nu_i(b) = \sum_{y_i = b} \prod_a \psi_a(y_{\partial a})$$

Where y is assignment for the subtree rooted at x_i and a is factor in the subtree rooted at x_i.

More formally, we define:

- T_w to be subtree rooted at w. w can be variable for factor node.
- V_w to be all variables in T_w.
- F_w to be all factors in T_w.
- $\text{Ch}(w)$ to be children of w.

There are two types of subtrees:

- **Variable rooted**

$$\nu_i(b) = \sum_{\substack{y \in \{0,1\}^{V_i} \\ y_i = b}} \prod_{a \in F_i} \psi_a(y_{\partial a})$$

- **Factor rooted**

$$\hat{\nu}_a(b) = \sum_{y \in \{0,1\}^{V_a}} \psi_a(b, y_{\partial a - \{i\}}) \prod_{a' \in F_a - \{a\}} \psi_{a'}(y_{\partial a'})$$

4.2 Write a recurrence

We need to write recurrence for both variable nodes and factor nodes:

$$\nu_i(b) = \begin{cases} \prod_{a \in \mathrm{Ch}(i)} \hat{\nu}_a(b) & \text{if } i \text{ is not a leaf,} \\ 1 & \text{if } i \text{ is a leaf.} \end{cases} \tag{1}$$

$$\hat{\nu}_a(b) = \begin{cases} \sum_{y \in \{0,1\}^{\mathrm{Ch}(a)}} \psi_a(b, y) \prod_{j \in \mathrm{Ch}(a)} \nu_j(y_j) & \text{if } i \text{ is not a leaf,} \\ \psi_a(b) & \text{if } a \text{ is a leaf.} \end{cases} \tag{2}$$

This is called the "Sum-Product Algorithm".

4.3 Prove recurrence is correct

We prove that the recurrence is correct by induction on the height of the tree.

Base case: Leaves (height 0). (1) and (2) hold.

Inductive case: Assume (1) and (2) hold for variable and factor nodes of height $\leq h$. Want to prove that (1) and (2) hold for variable and factor nodes of height $h + 1$.

Subcase 1: i is a variable node of height $h + 1$.

$$\begin{aligned} \nu_i(b) &= \prod_{a \in \mathrm{Ch}(i)} \hat{\nu}_a(b) \\ &= \prod_{a \in \mathrm{Ch}(i)} \sum_{y \in \{0,1\}^{V_a}} \psi_a(b, y_{\partial a - \{i\}}) \prod_{a' \in F_a - \{a\}} \psi_{a'}(y_{\partial a'}) \\ &= \sum_{y \in \{0,1\}^{V_i}} \prod_{a \in \mathrm{Ch}(i)} \left(\psi_a(b, y_{\partial a - \{i\}}) \prod_{a' \in F_a - \{a\}} \psi_{a'}(y_{\partial a'}) \right) \\ &= \sum_{\substack{y \in \{0,1\}^{V_i} \\ y_i = b}} \prod_{a \in F_i} \psi_a(y_{\partial a}) \end{aligned}$$

Subcase 2: a is a factor node of height $h + 1$.

$$
\begin{aligned}
\hat{\nu}_a(b) &= \sum_{y \in \{0,1\}^{\mathrm{Ch}(a)}} \psi_a(b,y) \prod_{j \in \mathrm{Ch}(a)} \nu_j(y_j) \\
&= \sum_{y \in \{0,1\}^{\mathrm{Ch}(a)}} \psi_a(b,y) \prod_{j \in \mathrm{Ch}(a)} \left(\sum_{\substack{z \in \{0,1\}^{V_j} \\ z_j = y_j}} \prod_{a' \in F_j} \psi_{a'}(Z_{\partial a'}) \right) \\
&= \sum_{y \in \{0,1\}^{\mathrm{Ch}(a)}} \psi_a(b,y) \sum_{\substack{z \in \{0,1\} \\ z_{\mathrm{Ch}(a)} = y}} \prod_{j \in \mathrm{Ch}(a)} \left(\prod_{a' \in F_j} \psi_{a'}(z_{\partial a'}) \right) \\
&= \sum_{y \in \{0,1\}^{V_a}} \psi_a(b, y_{\partial a - \{i\}}) \prod_{a' \in F_a - \{a\}} \psi_{a'}(y_{\partial a'})
\end{aligned}
$$

4.4 Runtime

From (1) and (2):

- Computing ν_i requires time $O(\deg(i))$ given subproblem solutions.
- Computing $\hat{\nu}_a$ requires time $O(2^{\deg(a)} \deg(a)) = O(1)$ given subproblem solutions.

Total runtime:

$$
\sum_{a=1}^{m} O(1) + \sum_{i=1}^{n} O(\deg(i)) = O(m) + O(|E|) = O(|E|)
$$

5 Other tasks

5.1 Computing conditional marginals

For example, if we want to compute $P(x_1 = b_1 \mid x_2 = b_2)$, we can simply add another factor ψ_a for x_2, where $\psi_a(b_2) = 1$ and $\psi_a(1 - b_2) = 0$. Thus the problem is reduced to computing unconditioned marginal of x_1.

5.2 Sampling

- Compute $P(x_1)$, assign x_1 to b_1 with probability $P(x_1 = b_1)$, $b_1 \in \{0,1\}$.
- Compute $P(x_2 \mid x_1 = b_1)$, assign x_2 to b_2 with probability $P(x_2 = b_2 \mid x_1 = b_1)$, $b_2 \in \{0,1\}$.
- Continue for $x_3, \ldots, x_n$ in the same way.

5.3 Optimization: Compute $\operatorname{argmax}_x P(x)$

Definition 5.1. For variable $i \in [n]$ and $b \in \{0,1\}$, the max marginal of i, denoted $M_i(b)$, is $\max_{x \in \{0,1\}^n} \{P(x) : x_i = b\}$.

Given an algorithm computing max marginals, we can compute $\operatorname{argmax}_x P(x)$ as follows:

- Compute M_1. Set $x_1 = \text{argmax}_b M_1(b)$. Say $x_1 = b_1$.
- Fix $x_1 = b_1$ and compute M_2. Set $x_2 = \text{argmax}_b M_2(b)$. Say $x_2 = b_2$.
- Repeat for $x_3, \ldots, x_n$.

5.4 Computing max marginals

We use a similar dynamic programming approach called the Max Product Algorithm.

$$\nu_i(b) = \prod_{a \in \text{Ch}(i)} \hat{\nu}_a(b)$$

$$\hat{\nu}_a(b) = \max_{y \in \{0,1\}^{\text{Ch}(a)}} \left\{ \psi_a(b, y) \prod_{j \in \text{Ch}(a)} \nu_j(y_j) \right\}$$

We can prove that this works using similar inductive argument.