COMPSCI 688: Probabilistic Graphical Models
Lecture 8: Undirected Graphical Models: Inference

- HW 2, due April 5

- Quiz 5, due Friday (posted v. soon)

Dan Sheldon
Manning College of Information and Computer Sciences
University of Massachusetts Amherst

Based on materials by Benjamin M. Marlin (marlin@cs.umass.edu) and Justin Domke (domke@cs.umass.edu)

---

**Markov Random Fields**

\[ p(x_i, x_j, x_k, x_l) = \frac{1}{Z} \sum_{x_{\neg C}} \phi_{i23}(x_i, x_j, x_k) \phi_{24}(x_j, x_k, x_l) \]

- Markov random field
  \[ p(x) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \]

- Dependence graph \( G \): where nodes \( i \) and \( j \) are connected by an edge if they appear together in some factor

---

**Inference in Markov Networks**

\[ p(x_e, x_u, x_Q) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \]

- Compute probability queries of the form...

  \[ p(x_Q | x_E) = \sum_{x_U} p(x_Q, x_U | x_E) \]

  - condition on evidence variables \( x_E \)
  - marginalize unobserved variables \( x_U \)
  - compute the joint distribution over query variables \( x_Q \)

  ... by transforming Markov network into one with fewer or simpler factors

  - Conditioning is easy
  - Marginalization is hard! (NP-hard in general, but sometimes efficient depending on \( G \))
### Conditioning

Reduce every factor by hard-coding the evidence variables

### Factor Reduction: Example

**Step 1**

\[
\Phi'_1(Y_1, Y_2) = \Phi_1(Y_1, Y_2)
\]

<table>
<thead>
<tr>
<th>( Y_1 = 0 )</th>
<th>( Y_1 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_2 = 0 )</td>
<td>1</td>
</tr>
<tr>
<td>( Y_2 = 1 )</td>
<td>7</td>
</tr>
</tbody>
</table>

\[
\Phi'_2(X_1) = \Phi_2(Y_1, X_1)
\]

<table>
<thead>
<tr>
<th>( X_1 = 0 )</th>
<th>( X_1 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_2 = 0 )</td>
<td>3</td>
</tr>
<tr>
<td>( Y_2 = 1 )</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
\Phi'_3(Y_2) = \Phi_3(Y_2)
\]

<table>
<thead>
<tr>
<th>( Y_2 = 0 )</th>
<th>( Y_2 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 = 0 )</td>
<td>6</td>
</tr>
<tr>
<td>( X_1 = 1 )</td>
<td>2</td>
</tr>
</tbody>
</table>

Query: \( P(Y_1, Y_2 \mid X_1 = 0, X_2 = 1) \)

**Step 2**

\[
\Phi'_1(Y_1, Y_2) = \Phi_1(Y_1, Y_2)
\]

<table>
<thead>
<tr>
<th>( Y_1 = 0 )</th>
<th>( Y_1 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_2 = 0 )</td>
<td>1</td>
</tr>
<tr>
<td>( Y_2 = 1 )</td>
<td>7</td>
</tr>
</tbody>
</table>

\[
\Phi'_2(Y_1) = \Phi_2(Y_1)
\]

<table>
<thead>
<tr>
<th>( Y_1 = 0 )</th>
<th>( Y_1 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_2 = 0 )</td>
<td>3</td>
</tr>
<tr>
<td>( Y_2 = 1 )</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
\Phi'_3(Y_2) = \Phi_3(Y_2)
\]

<table>
<thead>
<tr>
<th>( Y_2 = 0 )</th>
<th>( Y_2 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 = 0 )</td>
<td>6</td>
</tr>
<tr>
<td>( X_1 = 1 )</td>
<td>2</td>
</tr>
</tbody>
</table>

Query: \( P(Y_1, Y_2 \mid X_1 = 0, X_2 = 1) \)
Inference: Marginalization

Marginalization

Marginalization is the process of summing over some of the variables to get the marginal distribution of the remaining variables, or the partition function.

For example, the partition function is

\[
Z = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} \prod_{c \in C} \phi_c(x_c)
\]

Naively, this takes exponential time, but we can sometimes use the factorization structure to speed it up.

Example: Variable Elimination on a Chain

Consider the following MRF on a four-node “chain” graph:

\[
p(x_1, x_2, x_3, x_4) = \frac{1}{Z} \phi_1(x_1) \phi_2(x_2) \phi_3(x_3) \phi_4(x_4) \phi_{12}(x_1, x_2) \phi_{23}(x_2, x_3) \phi_{34}(x_3, x_4)
\]

\[
\begin{array}{c|c|c}
  x_i & \phi_i(x_i) & x_j \\
  \hline
  0 & 1 & 0 0 2 \\
  1 & 2 & 0 1 1 \\
  \end{array}
\]

Query: \(P(Y_1, Y_2 | X_1=0, X_2=1) \propto \phi_1(Y_1, Y_2) \phi_2'(Y_1) \phi_3'(Y_2)\)
Let's compute $Z$:

$$Z = \sum_{x_1} \sum_{x_2} \sum_{x_3} \sum_{x_4} \Phi_1(x_1) \Phi_2(x_2) \Phi_3(x_3) \Phi_4(x_4) \Phi_5(x_1) \Phi_6(x_2) \Phi_7(x_3) \Phi_8(x_4) \Phi_9(x_2) \Phi_{10}(x_1) \Phi_{11}(x_2) \Phi_{12}(x_3) \Phi_{13}(x_4)$$

Above, we eliminated $x_4, x_3, x_2, x_1$

What if we want to compute the unnormalized marginal $\hat{p}(x_1)$?

$$\hat{p}(x_1) = \sum_{x_2} \sum_{x_3} \sum_{x_4} \Phi_1(x_1) \Phi_2(x_2) \Phi_3(x_3) \Phi_4(x_4) \Phi_5(x_1) \Phi_6(x_2) \Phi_7(x_3) \Phi_8(x_4) \Phi_9(x_2) \Phi_{10}(x_1) \Phi_{11}(x_2) \Phi_{12}(x_3) \Phi_{13}(x_4)$$

Pictorially, this is how we changed the MRF

What if we want to compute the actual marginal $p(x_1)$?

Take $\hat{p}(x_1)$ and normalize it

$$Z = \sum_{x_1} \hat{p}(x_1), \quad p(x_1) = \frac{1}{Z} \hat{p}(x_1)$$

**Lesson**: always normalize at the end
Variable Elimination Discussion

The Variable Elimination Algorithm

Variable elimination is an algorithm to compute any marginal distribution in any MRF. In words: pick a variable \(x_i\) to eliminate, multiply together all factors containing \(x_i\) to get an intermediate factor, then sum out \(x_i\).

- Let \(F = \{\phi_c : c \in C\}\) be the set of factors.
- For each variable \(i\) in some elimination order (may not include all variables):
  - Let \(A = \{\phi_i \in F : i \in c\}\) be the set of factors whose scope contains \(i\).
  - Let \(\phi_i(x_i) = \prod_{c \in A} \phi_c(x_c)\) be the product of factors in \(A\) with scope \(i\) equal to the union of the scopes of the individual factors.
  - Let \(\psi(x_{c(i)}) = \sum_{x_i} \phi_i(x_{c(i)}, x_i)\) be the result of summing out \(x_i\).
  - Let \(F = F \setminus A \cup \{\psi\}\).

The final set of factors forms an MRF for the marginal distribution of the variables that were not eliminated.

What if we eliminate \(x_3\) first?

\[
Z = \sum_{x_1} \sum_{x_2} \sum_{x_4} \phi_1(x_1) \times \phi_2(x_2, x_3) \times \phi_4(x_4, x_3)
\]

\[
= \sum_{x_1} \sum_{x_2} \phi_1(x_1) \times \phi_2(x_2, x_3) \times (\phi_3(x_3) \phi_4(x_4, x_3)) \times \tau_5(x_3, x_4)
\]

Correct, but less efficient due to larger intermediate factors.

What if our graph is a star graph?

If we eliminate leaves first, it is very efficient. If we eliminate the hub node first, it creates a factor with size exponential in the number of nodes.

The efficiency of variable elimination depends on the maximum size of the intermediate factors created, which depends on the elimination ordering.

- Inference in MRFs is NP-hard, so we can’t always find a good elimination ordering.
- Finding the best elimination ordering for a given MRF is also NP-hard!
- It’s always efficient to eliminate leaves if present (intermediate factors are no larger than original ones).
  - \(\implies\) for trees, we can find an efficient elimination ordering.
- In fact, because the elimination ordering is predictable in trees, we can realize extra efficiencies when answering multiple queries through a dynamic programming approach known as message passing.
Message Passing Derivation

Let's go back to our chain example. Suppose we want to compute $p(x_4)$? Which variables should we eliminate, and in what order?

What if we want to compute $p(x_3)$? Which variables should we eliminate, and in what order?
Message Passing Derivation

When doing "leaf-first" variable elimination to compute any marginal \( p(x_i) \), there are only 6 different intermediate factors:

\[ m_{1 \to 2}, m_{2 \to 3}, m_{3 \to 4}, m_{4 \to 3}, m_{3 \to 2}, m_{2 \to 1} \]

Let’s call \( m_{j \to i} \) the “message” from \( j \) to \( i \).

We can compute \( Z \) by “collecting” messages at any node:

\[ Z = \sum_{x_i} \phi_i(x_i) \prod_{j \in \text{nb}(i)} m_{j \to i}(x_i) \]

The general formula for a marginal is similar, but we omit the final summation and normalize:

\[ p(x_i) = \frac{1}{Z} \phi_i(x_i) \prod_{j \in \text{nb}(i)} m_{j \to i}(x_i) \]

Message Passing in a Chain

- Initialize \( m_{0 \to 1}(x_1) = 1 \), \( m_{n \to n}(x_n) = 1 \).
- For \( i = 2 \) to \( n \):
  - Let \( k = i - 2 \), pass message from \( j = i - 1 \) to \( i \)
  - Let \( m_{j \to i}(x_i) = \sum_{x_j} m_{k \to j}(x_j) \phi_j(x_j) \phi_{j}(x_i, x_j) \)
- For \( i = n - 1 \) down to 1:
  - Let \( k = i + 2 \), pass message from \( j = i + 1 \) to \( i \)
  - Let \( m_{j \to i}(x_i) = \sum_{x_j} m_{k \to j}(x_j) \phi_j(x_j) \phi_{j}(x_i, x_j) \)
- Compute each unnormalized marginal as \( \hat{p}(x_i) = m_{i \to n}(x_i) \phi_i(x_i) m_{i+1 \to i}(x_i) \)
- Compute \( Z = \sum_{x_i} \hat{p}(x_i) \) for any \( i \), and normalize each marginal: \( p(x_i) = \frac{1}{Z} \hat{p}(x_i) \)

Pairwise Marginals

- Correct formula for a pairwise marginal \( p(x_i, x_{i+1}) \)?

The messages satisfy recurrences, e.g.:

\[ m_{2 \to 3}(x_3) = \sum_{x_2} m_{1 \to 2}(x_2) \phi_2(x_2) \phi_{23}(x_2, x_3) \]

The message \( m_{i - 1 \to i}(x_i) \) sums out all variables from the product of all factors “to the left” of \( x_i \).

The message \( m_{i+1 \to i}(x_i) \) has a similar recurrence, and sums out variables/factors “to the right”.

Using the recurrences, we can compute all messages, and therefore all marginals in two passes through the chain, one in each direction.
Discussion: Message Passing vs. Variable Elimination

- Variable elimination can compute marginals and $Z$ exponentially faster than direct summation for nice enough graphs (e.g. chains, trees)
- Naively, to compute all single-node marginals you would have to run variable elimination $n$ times, once per node (but this would repeat work)
- Message passing can compute all the marginals for the same cost as running variable elimination twice, so is a factor of $\approx n/2$ faster than naive variable elimination
- (Message passing is nice, but you could say variable elimination did the heavy lifting.)