CS281A/Stat241A: Statistical Learning Theory

Lecture 4: The Elimination Algorithm

Lecturer: Michael I. Jordan *Scribes: Evan Chang and Hoeteck Wee*

1 Review

Recall the parameterizations of a family of distributions that we associate with a particular graph.

- Directed Graphs

$$p(x) = \prod_{i=1}^{n} p(x_i \mid x_{\pi_i})$$

- Undirected Graphs

$$p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi^E_{X_C}(x_C)$$

where $\mathcal{C}$ = the set of the maximal cliques. Note, defining the parameterization in terms of the maximal cliques is solely for theoretical convenience. It is also correct to include smaller cliques, which is often useful in practice.

2 Inference on Graphs

Today, we discuss the elimination algorithm for performing inference on graphs. This algorithm has also surfaced in other fields; it is known peeling in genetics, filtering/smoothing in electrical engineering, and decimation in statistical physics.

The goal of inference is to compute the conditional probability $p(x_F \mid x_E)$ where F denotes the set of query nodes and E denotes the set of evidence nodes. We write R for the remaining nodes that are marginalized out. In other words, we partition the vertices of the graph into E, F, and R.

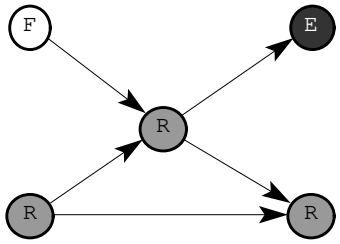


Figure 1: The vertices in the graph $\mathcal{V}$ are partitioned into the evidence nodes E, the query nodes F, and the remaining nodes R.

The elimination algorithm for inference works on general graphs, but is limited in that F must be a singleton.

3 Directed Graphs

3.1 Example

We would like to compute $p(x_1 \mid \bar{x}_6)$ in Figure 2 where $\bar{x}_6$ denotes a specific value of x_6. Note that

$$p(x_1 \mid \bar{x}_6) = \frac{p(x_1, \bar{x}_6)}{p(\bar{x}_6)} \propto p(x_1, \bar{x}_6)$$

and that $p(\bar{x}_6)$ is a constant. Thus, we view $p(x_1, \bar{x}_6)$ as an unnormalized representation of the conditional probability $p(x_1 \mid \bar{x}_6)$. Marginalizing over x_2, x_3, x_4, and x_5 on the parameterization induced by the graph, we have:

$$\begin{aligned}
p(x_1, \bar{x}_6) &= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} p(x_1)\cdot p(x_2 \mid x_1)\cdot p(x_3 \mid x_1)\cdot p(x_4 \mid x_2)\cdot p(x_5 \mid x_3)\cdot p(\bar{x}_6 \mid x_2, x_5) \\
&\quad \boxed{\textit{Note, we must choose some ordering of summations. This affects the intermediate terms and, eventually, the complexity of the algorithm we design.}} \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\sum_{x_5} p(x_5 \mid x_3)\cdot p(\bar{x}_6 \mid x_2, x_5) \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\sum_{x_4} p(x_4 \mid x_2)\cdot m_5(x_2, x_3) \\
&\quad \text{where we define } m_5(x_2, x_3) \stackrel{\triangle}{=} \textstyle\sum_{x_5} p(x_5 \mid x_3)\cdot p(\bar{x}_6 \mid x_2, x_5). \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\sum_{x_3} p(x_3 \mid x_1)\cdot m_5(x_2, x_3)\overbrace{\sum_{x_4} p(x_4 \mid x_2)}^{m_4(x_2)} \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\cdot m_4(x_2)\sum_{x_3} p(x_3 \mid x_1)\cdot m_5(x_2, x_3) \\
&= p(x_1)\sum_{x_2} p(x_2 \mid x_1)\cdot m_4(x_2)\cdot m_3(x_1, x_2) \\
&= p(x_1)\cdot m_2(x_1). \\
&\quad \boxed{\textit{In essence, } m_2(x_1) \textit{ summarizes what's relevant to } x_1 \textit{ from the other nodes.}}
\end{aligned}$$

From this result we can also obtain the probability $p(\bar{x}_6)$ by taking an additional sum over x_1:

$$p(\bar{x}_6) = \sum_{x_1} p(x_1)\cdot m_2(x_1),$$

and the desired conditional can be obtained as follows:

$$p(x_1 \mid \bar{x}_6) = \frac{p(x_1)\cdot m_2(x_1)}{\sum_{x_1} p(x_1)\cdot m_2(x_1)}.$$

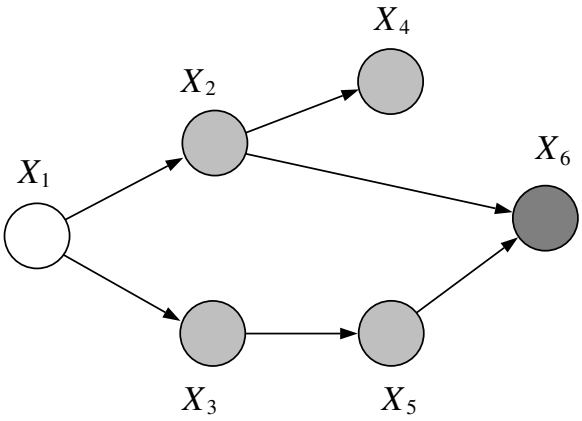


Figure 2: The dark shaded node X_6 is the node on which we condition, the lightly shaded nodes $\{X_2, X_3, X_4, X_5\}$ are nodes that are marginalized over, and the unshaded node X_1 is the node for which we wish to calculate conditional probabilities. Thus, for this example, we have $E = \{6\}$, $F = \{1\}$, and $R = \{2, 3, 4, 5\}$.

3.2 Evidence Potential

The evidence potential is a function on nodes and is a trick we use to avoid worrying about the bar on $\bar{x}_i$ or conditioning. It is related to the Kronecker delta function and is defined as follows: for all $i \in E$,

$$\delta(x_i, \bar{x}_i) \triangleq \begin{cases} 1 & \text{if } x_i = \bar{x}_i \\ 0 & \text{otherwise.} \end{cases}$$

Extending this to multiple variables, we have:

$$\delta(x_E, \bar{x}_E) \triangleq \begin{cases} 1 & \text{if } x_E = \bar{x}_E \\ 0 & \text{otherwise} \end{cases}$$

where E is the set of nodes whose values are to be conditioned on. This is equivalent to:

$$\delta(x_E, \bar{x}_E) \triangleq \prod_{i \in E} \delta(x_i, \bar{x}_i).$$

With this notation, we define:

$$p^E(x) \triangleq p(x) \cdot \delta(x_E, \bar{x}_E).$$

In essence, $p^E(x)$ is an unnormalized version of the conditional probability. Now we have:

$$p(x_F, \bar{x}_E) = \sum_{x_{E \cup R}} p^E(x)$$

In our example, we have

$$p(x_1, \bar{x}_6) = \sum_{x_2 \ldots x_6} p^E(x) = \sum_{x_2 \ldots x_6} p(x) \cdot \delta(x_6, \bar{x}_6) = \sum_{x_2 \ldots x_5} p(x_1, \ldots, x_5, \bar{x}_6)$$

where we treat evaluation as a summation. Essentially, we are using the sifting property of the δ function:

$$\sum_{x_i} g(x_i) \cdot \delta(x_i, \bar{x}_i) = g(\bar{x}_i)$$

so that we do not have to worry about what to sum over.

ELIMINATE($\mathcal{G}, E, F$)
    INITIALIZE($\mathcal{G}, F$)
    EVIDENCE(E)
    UPDATE($\mathcal{G}$)
    NORMALIZE(F)

INITIALIZE($\mathcal{G}, F$)
    choose an ordering I such that F appears last
    for each node X_i in $\mathcal{V}$
        place $p(x_i \mid x_{\pi_i})$ on the active list
    end

EVIDENCE(E)
    for each i in E
        place $\delta(x_i, \bar{x}_i)$ on the active list
    end

UPDATE($\mathcal{G}$)
    for each i in I
        find all potentials from the active list that reference x_i and remove them from the active list
        let $\phi_i(x_{T_i})$ denote the product of these potentials
        let $m_i(x_{S_i}) = \sum_{x_i} \phi_i(x_{T_i})$
        place $m_i(x_{S_i})$ on the active list
    end

NORMALIZE(F)
    $p(x_F \mid \bar{x}_E) \leftarrow \phi_F(x_F) / \sum_{x_F} \phi_F(x_F)$

Figure 3: The ELIMINATE algorithm for probabilistic inference on directed graphs.

3.3 Example Revisited

Let us now return to our earlier example where we want to compute $p(x_1 \mid \bar{x}_6)$. In Figure 3, we give the ELIMINATE algorithm. Following the state of ELIMINATE to compute $p(x_1 \mid \bar{x}_6)$ using the elimination ordering $\{6, 5, 4, 3, 2, 1\}$, we have:

- Active List: $p(x_1), p(x_2 \mid \bar{x}_1), \ldots, p(x_6 \mid x_2, x_5), \delta(x_6, \bar{x}_6)$

  Eliminating 6: $\phi(x_{T_6}) = p(x_6 \mid x_2, x_5) \cdot \delta(x_6, \bar{x}_6)$ $\quad T_6 = \{2, 5, 6\}$
  $m_6(x_{S_6}) = \sum_{x_6} p(x_6 \mid x_2, x_5) \cdot \delta(x_6, \bar{x}_6)$ $\quad S_5 = \{2, 5\}$

- Active List: $p(x_1), p(x_2 \mid \bar{x}_1), p(x_3 \mid \bar{x}_1), p(x_4 \mid \bar{x}_2), p(x_5 \mid x_3), m_6(x_2, x_5)$

  Eliminating 5: $\phi(x_{T_5}) = p(x_5 \mid x_3) \cdot m_6(x_2, x_5)$ $\quad T_5 = \{2, 3, 5\}$
  $m_5(x_{S_5}) = \sum_{x_5} p(x_5 \mid x_3) \cdot m_6(x_2, x_5)$ $\quad S_5 = \{2, 3\}$

- Active List: $p(x_1), p(x_2 \mid \bar{x}_1), p(x_3 \mid \bar{x}_1), p(x_4 \mid \bar{x}_2), m_5(x_2, x_3)$

$\vdots$

- (End) Active List: $p(x_1), m_2(x_1)$ *just like before*

Now we just need to normalize the output using NORMALIZE in Figure 3.

4 Undirected Graphs

4.1 Elimination Algorithm

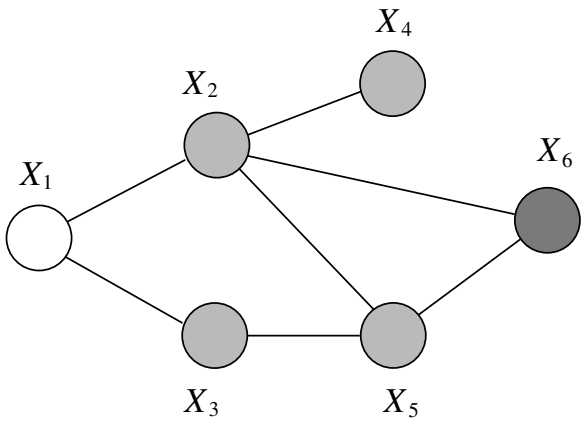


Figure 4: The dark shaded node, X_6, is the nodes on which we condition, the lightly shaded nodes, $\{X_2, X_3, X_4, X_5\}$, are the nodes that are marginalized over, and the unshaded node, X_1, is the node for which we wish to calculate conditional probabilities.

The elimination algorithm applies to undirected graphs with almost no change. Let us consider Figure 4, an example which is analogous to the previous example for directed graphs. We calculate the probability $p(x_1 \mid \bar{x}_6)$, representing the joint probability on the graph via potential functions on the cliques $\{X_1, X_2\}$ $\{X_1, X_3\}$, $\{X_2, X_4\}$, $\{X_3, X_5\}$, and $\{X_2, X_5, X_6\}$.

We first calculate the probability $p(x_1, \bar{x}_6)$. To simplify the presentation we drop the subscript in the $\psi_{X_C}(x_C)$ notation, relying on the argument to the function to make it clear which potential function is being referred to. Also we again make use of the notation $m_i(x_{S_i})$ to denote the intermediate factor that results from the summation over x_i.

$$
\begin{aligned}
p(x_1, \bar{x}_6) &= \frac{1}{Z} \sum_{x_2} \sum_{x_3} \sum_{x_4} \sum_{x_5} \sum_{x_6} \psi(x_1, x_2)\psi(x_1, x_3)\psi(x_2, x_4)\psi(x_3, x_5)\psi(x_2, x_5, x_6)\delta(x_6, \bar{x}_6) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) \sum_{x_6} \psi(x_2, x_5, x_6)\delta(x_6, \bar{x}_6) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) \sum_{x_4} \psi(x_2, x_4) \sum_{x_5} \psi(x_3, x_5) m_6(x_2, x_5) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) \sum_{x_4} \psi(x_2, x_4) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) \sum_{x_3} \psi(x_1, x_3) m_5(x_2, x_3) m_4(x_2) \\
&= \frac{1}{Z} \sum_{x_2} \psi(x_1, x_2) m_4(x_2) m_3(x_1, x_2) \\
&= \frac{1}{Z} m_2(x_1).
\end{aligned}
$$

Marginalizing further over x_1 yields:

$$p(\bar{x}_6) = \frac{1}{Z} \sum_{x_1} m_2(x_1),$$

and we calculate the desired conditional as:

$$p(x_1 \mid \bar{x}_6) = \frac{m_2(x_1)}{\sum_{x_1} m_2(x_1)},$$

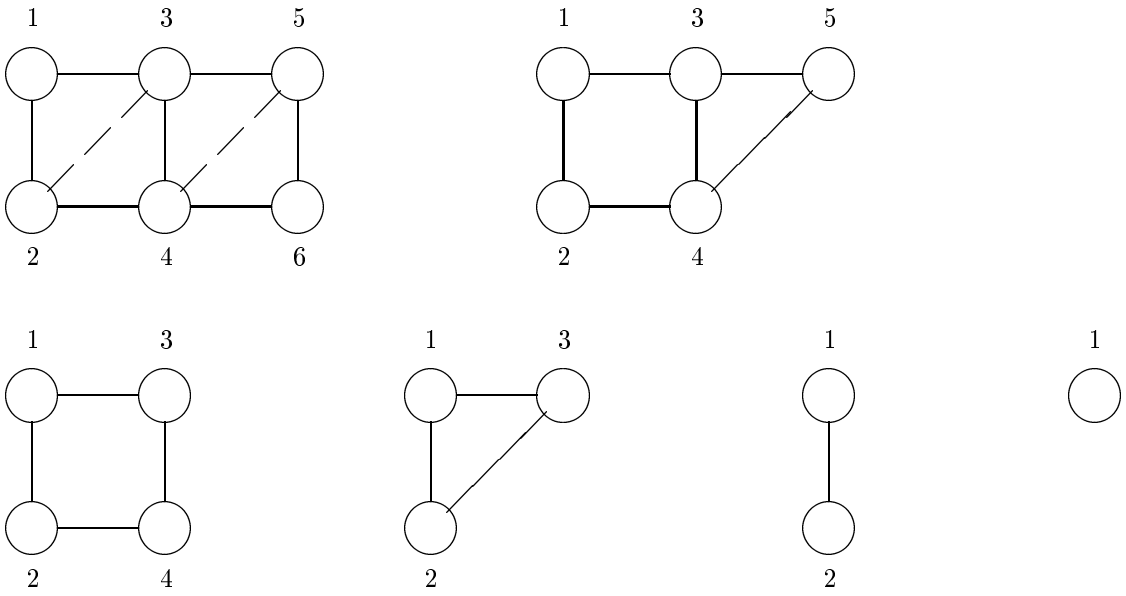
where the normalization factor Z cancels.

4.2 Computational Complexity of ELIMINATE

```
UNDIRECTEDGRAPHELIMINATE(G, I)
    for each node X_i in I
        connect all of the remaining neighbors of X_i
        remove X_i from the graph
    end
```

Figure 5: A simple greedy algorithm for eliminating nodes in an undirected graph $\mathcal{G}$.

We begin by defining an algorithm UNDIRECTEDGRAPHELIMINATE, which is presented in Figure 5. Below, we show a progression of UNDIRECTEDGRAPHELIMINATE on an example graph using the elimination ordering $\{6, 5, 4, 3, 2, 1\}$:



In the above example, the dotted lines indicate edges that were added during the execution of UndirectedGraphEliminate. Note in Eliminate on this graph using the same elimination order, we would have the term:

$$m_6(x_4, x_5) = \sum_{x_6} \psi(x_5, x_6)\psi(x_4, x_6).$$

Here, marginalization creates a new dependency between the nodes X_4 and X_5 that is not present in the original graph. We call this an *induced dependency* between the nodes, which we can see corresponds exactly to a dotted line above. The complexity of the Eliminate algorithm hinges upon computing expressions such as:

$$\sum_{x_i} \phi(x_{T_i}).$$

Assuming no special algebraic structure in the summand (which we could exploit), the complexity of the algorithm is thus exponential in the size of the maximal *induced* clique (*i.e.* the largest elimination clique created by UndirectedGraphEliminate).

This quantity turns out to be a well-studied problem in graph theory. The *treewidth* of a graph is equal to the minimum of the maximal induced clique size minus one.