Exact Inference: Variable Elimination
10708, Fall 2020
Pradeep Ravikumar

1 Introduction

The key utility of a probabilistic graphical model is that it enables probabilistic reasoning. As an example, suppose we have a joint distribution over thousands of symptom variables, and thousands of disease variables. A key question in medical diagnosis is: given some specific symptoms, what is the probability of some disease, say the flu? This can be cast as computing the probability of some set of variables given observed values of some other variables.

Let $X = (X_1, \ldots, X_p)$ denote the random vector associated with a DGM or UGM. We are interested in **probabilistic reasoning queries** that take the form:

$$P(X_F | X_E = x_e),$$

which can be seen to be a conditional marginal probability computation. This in turn can be seen to be equal to:

$$P(X_F | X_E = x_e) = \frac{P(X_F, X_E = x_e)}{P(X_E = x_e)},$$

so that the key technical quantity of interest is the marginal probability of some subset of variables $P(X_F)$.

2 Computational Complexity: Hardness Results

Unfortunately even though the representational complexity of PGMs is low, scaling exponentially in only the size of local factors, computing marginals requires "collating" all of this local information, and marginalizing out all but a specific subset. This turns out to be a "global" operation, and computationally hard in general. We will state these hardness results for DGMs, but since any DGM can be expressed as a UGM without increase in representational complexity: if inference with DGMs is hard, it entails that inference with UGMs is hard as well.

Theorem 1 *Consider the following decision problem: Given a discrete DGM P over $\mathcal{X}$, a variable $X \in \mathcal{X}$, and a value $x \in \mathrm{VAL}(X)$, decide whether $P(X = x) > 0$. This decision problem is NP-complete.*

It is common to state such hardness results in terms of decision problems, but one can extend this to the case of computing the probabilities themselves, since it amounts to computing a sum over possible configurations, which is an instance of so-called counting problems.

Theorem 2 *Consider the following counting problem: Given a discrete DGM P over $\mathcal{X}$, a variable $X \in \mathcal{X}$, and a value $x \in \mathrm{VAL}(X)$, compute $P(X = x)$. This counting problem is #P-complete.*

One might then think that at the very least one might be able to *approximately* compute these quantities, with a guaranteed approximation factor.

We say that an estimate $\widehat{P}(X_F)$ for $P(X_F)$ has relative error ϵ if:

$$\frac{\widehat{P}(X_F)}{1+\epsilon} \leq P(X_F) \leq \widehat{P}(X_F)\,(1+\epsilon).$$

Such a multiplicative approximation factor makes more sense for probabilities than an additive approximation factor, where we would have the guarantee that: $|\widehat{P}(X_F) - P(X_F)| \leq \epsilon$, since the conditional probabilities could be very small e.g. 10^{-3}, in which case an additive factor of even 10^{-3} is basically useless: we could just output zero for such small probabilities to obtain a small additive error. On the other hand, the estimate of zero would have a multiplicative approximation factor of infinity. Unfortunately, we have the following theorem.

Theorem 3 *Consider the following problem: Given a discrete DGM P over $\mathcal{X}$, a variable $X \in \mathcal{X}$, and a value $x \in \mathrm{VAL}(X)$, compute $P(X = x)$ upto a multiplicative approximation factor of ϵ. This problem is NP-hard.*

It turns out that computing marginal probabilities up to additive approximation factor is not necessarily NP-hard, and in particular, there is a randomized polynomial time algorithm to do so (as we will see when we discuss sampling based approaches to inference). But this does not yield additive approximation guarantees for conditional marginal probabilities, since these involve ratios of two marginals, and in general this ratio need not have an additive approximation guarantee. In particular, there does not always exist an $\epsilon > 0$ such that ϵ-additive approximation guarantees for marginals entail ϵ' additive factor guarantee for the conditional:

$$\begin{aligned}\frac{\widehat{P}(X_F)}{\widehat{P}(X_E = x_e)} &\leq \frac{P(X_F) + \epsilon}{P(X_E = x_e) - \epsilon} \\ &\neq \frac{P(X_F)}{P(X_E = x_e)} + \epsilon'.\end{aligned}$$

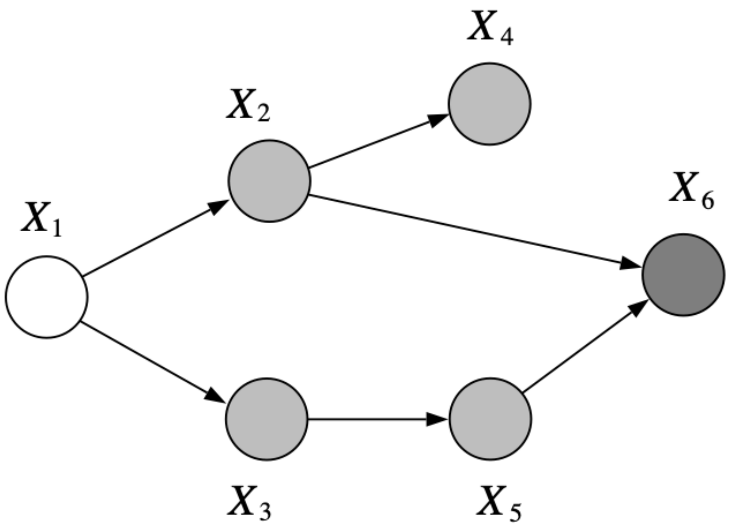
3 Warm Up

Consider the simple DGM over two variables X, Y with the DAG $X \to Y$. Then, if we wish to compute the marginal $P(X)$, this is already available as a CPD of the DGM. For $P(Y)$, we can compute:

$$P(Y) = \sum_x P(X = x)P(Y|X = x).$$

Similarly the conditional $P(Y|X = x)$ is already a CPD of the DGM. While $P(X|Y = y)$ can simply be computed via Bayes rule.

What we now need to do is scale this up to general graphs.



Consider the six node DAG in Figure 3. Consider a discrete RV $X = (X_1, \ldots, X_6)$ associated with the DAG. Suppose we wish to compute $P(X_1|X_6 = x_6) = P(X_1, X_6 = x_6)/P(X_6 = x_6)$. Let us focus on computing the numerator. We first note that:

$$P(x_1, x_6) = \sum_{x_2,x_3,x_4,x_5} P(x),$$

where we use the shorthand $\sum_{x_s} \phi(x)$ to denote $\sum_{x_s \in \mathcal{X}_s} \phi(x)$.

Brute Force Approach. One approach is to explicitly construct the probability table for $P(X)$. Suppose each variable takes K possible values. Then the probability table has size K^6. It can be seen that to compute the numerator above for all values of X_1, one would need to visit K^5 entries of the table. This is expensive, even more so if the number of variables were much larger. But the brute force approach does not really leverage the key promise of PGMs — local representations and hence small representational complexity — since the computations involved are global.

Distributive Property Consider a product of factors: $\phi(X) = \phi_1(X).\phi_2(X)$, and suppose we wish to compute: $\sum_{x_s} \phi$. Suppose $X_s \notin \text{scope}(\phi_1)$. Then,

$$\sum_{x_s} \phi_1.\phi_2 = \phi_1. \sum_{x_s} \phi_2.$$

Thus, the sum and the product terms can be exchanged. Note that after computing the sum, the residual functions do not depend on X_s i.e. the variable X_s is eliminated. It is thus also called sum-product elimination of variable X_s.

Let us consider the example in Figure 3, and apply this "sum product variable elimination" principle.

$$\begin{aligned}
P(x_1, x_6) &= \sum_{x_2,x_3,x_4,x_5} P(x) \\
&= \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} P(x_1)P(x_2 \,|\, x_1)P(x_3 \,|\, x_1)P(x_4 \,|\, x_2)P(x_5|x_3)P(x_6 \,|\, x_2, x_5) \\
&= P(x_1)\sum_{x_2} P(x_2 \,|\, x_1)\sum_{x_3} P(x_3 \,|\, x_1)\sum_{x_4} P(x_4 \,|\, x_2)\sum_{x_5} P(x_5|x_3)P(x_6 \,|\, x_2, x_5) \\
&= P(x_1)\sum_{x_2} P(x_2 \,|\, x_1)\sum_{x_3} P(x_3 \,|\, x_1)\sum_{x_4} P(x_4 \,|\, x_2)m_5(x_3, x_2, x_6) \\
&= P(x_1)\sum_{x_2} P(x_2 \,|\, x_1)\sum_{x_3} P(x_3 \,|\, x_1)m_5(x_3, x_2, x_6) \\
&= P(x_1)\sum_{x_2} P(x_2 \,|\, x_1)m_3(x_2, x_6) \\
&= P(x_1)m_2(x_1, x_6),
\end{aligned}$$

and

$$\begin{aligned}
P(x_6) &= \sum_{x_1} P(x_1, x_6) = \sum_{x_1} P(x_1)m_2(x_1, x_6) \\
&= m_1(x_6),
\end{aligned}$$

so that $P(x_1 \,|\, x_6) = P(x_1)m_2(x_1, x_6)/m_1(x_6)$.

4 Variable Elimination

Definition 4 (Factor Marginalization) *Let $\mathbf{X}$ be a set of variables, and lLet $\phi(\mathbf{X})$ be some factor over these variables. Suppose $Y \in \mathbf{X}$. We then define the factor marginalization of Y in $\phi(\cdot)$ as new factor ψ with scope $\mathbf{X} - \{Y\}$ s.t.:*

$$\psi(\mathbf{X} - \{Y\}) = \sum_{Y} \phi(\mathbf{X}).$$

This is the key operation we will require when performing variable elimination, as shown in Algorithm 1.

Algorithm 1 Variable Elimination

```
  Input: Set of factors $\Phi$, ordered set of variables to be eliminated $\mathbf{Z}$
  for $i = 1, \ldots, |\mathbf{Z}|$ do
     Find all factors in $\Phi$ that reference variable $\mathbf{Z}_i$, and remove them from $\Phi$
     Let $\Phi_i(T_i)$ denote the product of these factors, where $T_i = \text{scope}(\phi)$ (i.e. set of variables referenced by $\phi(\cdot)$
     Factor marginalize out variable $\mathbf{Z}_i$ from $\Phi_i$ and suppose $m_i(S_i)$ is the new factor. Add this new factor to $\Phi$
     Return $\prod_{\phi \in \Phi} \phi$
  end for
```

What is the output of this algorithm? The following theorem answers this generalizing the specific example we had considered earlier.

Theorem 5 *Let $\mathbf{X}$ be some set of variables, and let Φ be some set of factors over these variables, so that $\cup_{\phi \in \Phi} scope(\phi) \subseteq \mathbf{X}$. Let $\mathbf{Z} \subset \mathbf{X}$ be an ordered set of variables that we eliminate using the variable elimination algorithm in Algorithm 1, and get the output factor ϕ^*. Then, $scope(\phi^*) \subseteq \mathbf{X} - \mathbf{Z}$, and is given as:*

$$\phi^* = \sum_{\mathbf{Z}} \prod_{\phi \in \Phi} \phi.$$

Thus the variable elimination algorithm takes in a (product of) factors, and marginalizes out a given ordered subset of variables.

Suppose we wish to compute $P(X_F \,|\, X_E)$. We then first set Φ to be the product of all the factors involved in the PGM. Note that for UGMs, these factors are simply the clique potentials $\{\phi_C(X_C)\}$, while for DGMs, these are the node-conditional distributions $\{P(X_i | X_{\text{PA}_i})\}$. Since variable elimination only operates on factors, it provides a unified probabilistic inference procedure encompassing both UGMs and DGMs.

Let $X_R = \text{scope}(\Phi) - X_F - X_E$. We then set the variables to be eliminated to be $X_R \cup X_F$ choosing an ordering such that variables in X_F occur after variables in X_R. Let $m_R(X_F, X_E) = \sum_{X_R} \Phi(X_F, X_E, X_R)$ be the factor obtained by just eliminating variables in X_R, and let $m_F(X_E) = \sum_{X_F} m_R(X_F, X_E)$ be the factor obtained by further eliminating variables in X_F. The conditional probability $P(X_F \mid X_E)$ is then as:

$$P(X_F \mid X_E) = m_R(X_F, X_E)/m_F(x_E).$$

5 Computational Complexity: Preliminary Analysis

Suppose we eliminate p random variables, and we have m initial factors. Let τ_i be the factor that we obtain when we are about to eliminate variable X_i, and let ψ_i be the factors that we obtain after eliminating X_i. Suppose N_i is the number of entries in τ_i (i.e. its storage complexity), and let $N_{\max} = \max_{i \in [p]} N_i$. The total number of factors that are involved in the whole process is $m + p$: m initial factors, and n intermediate factors ψ_i that are generated when we eliminate each of the p variables. Each of these factors are multiplied once: because when we eliminate variable X_i, we multiply all factors involving X_i, and then remove these factors from further consideration. And the cost of multiplying a factor ϕ to produce ψ_i is atmost N_i. So that the total number of multiplication steps is $(m + p)N_{\max}$. And the task of marginalizing out X_i from ψ_i in turn only involves N_i addition steps. So that the total number of addition steps is $pN_{\max}$. Thus, the total number of work is $O((m+p)N_{\max})$. Note that the number of entries N_i in a factor $\psi_i$scales exponentially in the number of variables in the scope of the factor ψ_i. This could in general be as large as the number of variables, but is typically much lower. To get a handle on what this maximum size would be, it is instructive to view variable elimination via a purely graph-theoretic lens.

6 Graph-theoretic Viewpoint of Variable Elimination

Let Φ be a set of factors over $\{X_1, \ldots, X_p\}$, and let $\mathbf{Z}$ be an ordered sequence of variables to be eliminated. The induced graph $G_{\Phi,\mathbf{Z}}$ is an undirected graph over $\{X_1, \ldots, X_p\}$ where X_i and X_j are connected by an edge if they both appear in an intermediate factor generated by variable elimination. We have the following properties of the induced graph.

Proposition 6 *Let $G_{\Phi,\mathbf{Z}}$ be the induced graph given VE of $\mathbf{Z}$ over factors Φ over $\{X_1, \ldots, X_p\}$. Suppose $\{\psi_i\}_{i=1}^m$ are the intermediate factors generated by VE. Then:*

- *For all $i \in [m]$, scope(ψ_i) is a clique in $G_{\Phi,\mathbf{Z}}$.*
- *Every maximal clique in $G_{\Phi,\mathbf{Z}}$ is the scope of some intermediate factor ψ_i.*

Proof. The first statement follows by construction of the induced graph. For the second statement, consider a maximal clique over variables $Y_1, \ldots, Y_k$, and suppose Y_1 is the first to be eliminated. After eliminating Y_1 no further factors involving Y_1 will be multiplied together. Then, existence of edges from Y_1 to Y_i for $i \in \{2, \ldots, k\}$ means that during the VE step for Y_1, there are factors involving Y_1 and Y_i. And the VE step for Y_1 would then multiply them all together, so that there is a factor involving all of $Y_1, \ldots, Y_k$. But there cannot be other variables in this factor because then Y_1 would be connected to that variable, and the induced graph would in turn connect this other variable to Y_i for $i \in \{2, \ldots, k\}$. This would entail that $Y_1, \ldots, Y_k$ is not a maximal clique. $\square$

Thus there is a direct correspondence between maximal cliques in the induced graph $G_{\Phi,\mathbf{Z}}$ and the sizes of the intermediate factors generated in VE. Note that the induced graph does not depend on the specific functional form of the factors Φ beyond their scopes. Given a UG G, the factors Φ have scopes corresponding to the cliques in G. Similarly, for a DAG G, the factors G have scopes corresponding to nodes and their parents. For any UG or DAG G, we then let $G_{\mathbf{Z}}$ be the induced graph with respect to corresponding UG or DAG scoped factors.

Definition 7 (Treewidth) *The width width(G) of a graph G is the size of its largest clique minus one. We then define the tree-width of a graph G as:*

$$\textit{tree-width}(G) = \min_{\sigma \in \mathcal{S}_p} \textit{width}(G_{\mathbf{X}_\sigma}),$$

where $\mathbf{X}_\sigma$ is an specific ordering wrt permutation σ of all the variables $\mathbf{X} = (X_1, \ldots, X_p)$, and $\mathcal{S}_p$ is the set of all permutations of p elements.

Examples.

- The tree-width of a tree graph is equal to one. This can be seen by eliminating the graphs "bottom-up" from leaves on to the root. This will only create factors of size atmost two: a node and its parent; so that the tree-width is one.
- The tree-width of an $n \times n$ grid graph is n. This is because the size of the largest clique in the optimal induced graph is $n + 1$ which can be seen by eliminating variables row by row, or column by column.

Thus, the tree-width is the (one minus the) size of the largest clique in the VE induced graph with respect to the best possible ordering i.e. the ordering where the largest clique is as small as possible. We thus see that the computational complexity of VE is at least exponential in the tree-width. But in practice it could actually be worse, because we may not have the optimal ordering that specifies the tree-width. And unfortunately, finding this optimal ordering in general is NP-Hard.

Theorem 8 *Given a graph G, and some bound K, determining whether there exists an elimination ordering such that the induced graph has width $\leq K$ is NP-Hard.*

There is however a class of graphs for which computing the optimal ordering is simple. We first have the following proposition.

Proposition 9 *Every VE induced graph is chordal.*

Proof. Cosnider any cycle $X_1 - \ldots - X_k$ with $k \geq 4$. Suppose wlog X_1 is the first node to be eliminated. Then, no edge incident to X_1 is added after it is eliminated: so edges $X_1 - X_2$ and $X_1 - X_k$ exist when X_1 is about to be eliminated. But by the same argument as in an earlier proof, the only way for this to happen is for there to be a factor involving X_1, X_2, X_k at the time of elimination of X_1, so that the induced graph will have a chordal edge $X_2 - X_k$. $\square$

We next see that for a chordal graph, one can compute the optimal elimination ordering very simply, and which moreover does not introduce any additional fill edges in the induced graph.

Proposition 10 *Any chordal graph admits an elimination ordering σ that does not introduce any fill edges i.e. the induced graph is simply the skeleton of the original graph.*

Proof. We know that a chordal graph has a simplicial vertex v such that its neighbors are fully connected. Suppose we eliminate v first. Then this does not introduce any additional edges since the factor created right before elimination of v would simply have v and all its neighbors: and the neighbors of v are already fully connected. By induction, since the remaining graph $G[V - \{v\}]$ is chordal, it admits an elimination ordering that does not introduce fill edges. $\square$

When the underlying graph is non-chordal, then as noted earlier, finding the optimal elimination ordering is intractable in general. There are however a number of heuristics that seem to have good empirical performance. These heuristics are all based on picking the next vertex to eliminate greedily, based on some criterion. We discuss some popular criteria below:

- Min-degree: we pick the vertex with the smallest degree (so that we would fully connect a smaller number of nodes)
- Min-fill: we pick the vertex that would lead to the smallest number of additional "fill" edges