10 The Elimination Algorithm

Having now completed our introduction to the different types of graphical models (and some relationships among them) as well as learning parameters, we now turn to addressing how to use the structure of these graphs to efficiently carry out inference involving distributions represented by them.

10.1 Inference Tasks

As we mentioned in our introduction to the subject, there are generally two main inference tasks of interest:

1. the task of computing posterior beliefs
2. the task of computing most-probable configurations (MPC)

We will soon discuss the variety of inference tasks and the relationships among them more broadly, but for now consider the two tasks described above.

Calculating posterior beliefs. Let us start with the task of computing posterior beliefs. Specifically, we have a model $p_{\mathbf{x},\mathbf{y},\mathbf{z}}$ describing the relationship between three collections of variables: $\mathbf{x}$, which represents the latent variables of interest; $\mathbf{y}$, which represents observed variables; and $\mathbf{z}$, which represent latent variables not of direct interest. The prior belief on $\mathbf{x}$ is $p_{\mathbf{x}}$, which is a marginal of the model. We seek to calculate the *posterior belief*

$$p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y}) = \frac{\sum_{\mathbf{z}} p_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\mathbf{y},\mathbf{z})}{\sum_{\mathbf{x},\mathbf{z}} p_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\mathbf{y},\mathbf{z})} \propto \sum_{\mathbf{z}} p_{\mathbf{x},\mathbf{y},\mathbf{z}}(\mathbf{x},\mathbf{y},\mathbf{z}),$$

where to obtain the last expression we have used that since it is given, $\mathbf{y}$ is a constant. Evidently, this computation involves a marginalization and a normalization, the former of which typically dominates the complexity.

Calculating most-probable configurations. Here, we have a model $p_{\mathbf{x},\mathbf{y}}$ describing the relationship between two collections of variables: $\mathbf{x}$, which represents the variables of interest; and $\mathbf{y}$, which represents observed variables.[1] We seek to determine a most-probable configuration (MPC)

$$\hat{\mathbf{x}} \in \arg\max_{\mathbf{x}} p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y}) = \arg\max_{\mathbf{x}} p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y})$$

[1]Note that the two sets may share variables in common. Also, there may also be additional variables $\mathbf{z}$ not of interest, in which case marginalization is performed first as a preliminary step.

In the first half of these notes, we focus on the problem of posterior belief computation; we will address the problem of most-probable configuration computation in the second half of these notes.

10.2 Posterior Belief Computation with Undirected Graphs

We begin our development by examining posterior belief computation in the context of undirected graphical models. For concreteness, consider a collection of random variables $\mathsf{x}_1, \ldots, \mathsf{x}_N$, each taking on values from alphabet $\mathcal{X}$ and with their joint distribution represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = \{1, \ldots, N\}$ and factoring over the maximal cliques $\mathrm{cl}^*(\mathcal{G})$ as

$$p_{\mathsf{x}_\mathcal{V}}(x_\mathcal{V}) = \frac{1}{Z} \prod_{\mathcal{C} \in \mathrm{cl}^*(\mathcal{G})} \psi_\mathcal{C}(x_\mathcal{C}),$$

with Z, as usual, denoting the partition function.

The problem of posterior belief computation is ultimately one of marginalization. To emphasize this point through an example, suppose we are interested in learning about x_1 based on observing x_N, i.e., we want the posterior belief $p_{\mathsf{x}_1|\mathsf{x}_N}(\cdot|x_N)$. Calculating this posterior belief requires us to marginalize out $\mathsf{x}_2, \ldots, \mathsf{x}_{N-1}$, which can be computationally quite complex if structure in the joint distribution is not exploited carefully.

In the sequel we develop a procedure referred to as the *variable elimination algorithm* for implementing marginalization. This algorithm can be applied to all undirected graphs, although the computational complexity is better in some cases than others.

We first develop marginalization when there are no observations, and then discuss how observations are naturally incorporated into the algorithm to generate posterior beliefs.

10.3 The Variable Elimination Algorithm

We develop the basic approach via the example graph of Fig. 1, which expresses the structure in the distribution among a set of random variables $\mathsf{x}_1, \ldots, \mathsf{x}_5$, each drawn from alphabet $\mathcal{X}$. The corresponding probability distribution factors according to

$$p_{\mathsf{x}_1,\ldots,\mathsf{x}_5}(x_1, \ldots, x_5) \propto \psi_{12}(x_1, x_2)\, \psi_{13}(x_1, x_3)\, \psi_{25}(x_2, x_5)\, \psi_{345}(x_3, x_4, x_5), \qquad (1)$$

where the factors are the potential functions for the maximal cliques in the graph.

Suppose we want to compute the marginal $p_{\mathsf{x}_1}(\cdot)$. The naïve approach ignores the structure in the distribution and carries out the following computation: given a table describing the joint distribution over all the variables, "sum out" the variables

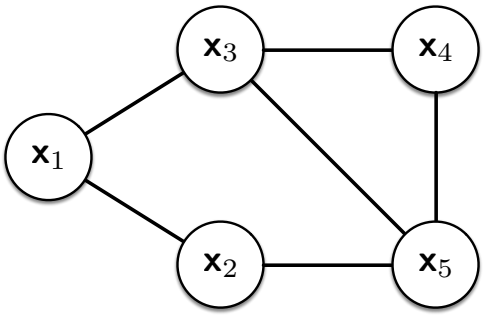


Figure 1: An example of an undirected graph expressing the struction in a distribution $p_{x_1,\ldots,x_5}(\cdot,\cdots,\cdot)$.

$x_2, \ldots, x_5$, i.e.,

$$p_{x_1}(x_1) = \sum_{x_2,x_3,x_4,x_5 \in \mathcal{X}} p_{x_1,\ldots,x_5}(x_1,\ldots,x_5), \tag{2}$$

which requires $O(|\mathcal{X}|^5)$ operations.

A better approach exploits the factorization structure (1). To see this, we start by substituting (1) into (2) to obtain

$$\begin{aligned} p_{x_1}(x_1) &= \sum_{x_2,x_3,x_4,x_5 \in \mathcal{X}} p_{x_1,\ldots,x_5}(x_1,\ldots,x_5) \\ &\propto \sum_{x_2,x_3,x_4,x_5 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,\psi_{13}(x_1,x_3)\,\psi_{25}(x_2,x_5)\,\psi_{345}(x_3,x_4,x_5). \end{aligned}$$

Next, we note that the order we choose to sum out variables can affect the computational complexity of marginalization. This order is referred to as the *elimination order*. For example, consider an elimination order $(5,4,3,2,1)$, whereby we sum out x_5 first, x_4 second, and so forth until we get to x_1, the sum for which provides the final normalization of the desired marginal $p_{x_1}(\cdot)$.

The key to achieving a reduction in complexity is repeatedly exploiting the distributive law of algebra to make each successive summation over as few factors as

possible. Applying this insight to our example, we obtain

$$
\begin{aligned}
p_{x_1}(x_1) &\propto \sum_{x_2,x_3,x_4,x_5 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,\psi_{13}(x_1,x_3)\,\psi_{25}(x_2,x_5)\,\psi_{345}(x_3,x_4,x_5) \\
&= \sum_{x_2,x_3,x_4 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,\psi_{13}(x_1,x_3) \underbrace{\sum_{x_5 \in \mathcal{X}} \psi_{25}(x_2,x_5)\,\psi_{345}(x_3,x_4,x_5)}_{\triangleq m_5(x_2,x_3,x_4)} \\
&= \sum_{x_2,x_3,x_4 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,\psi_{13}(x_1,x_3)\,m_5(x_2,x_3,x_4) \\
&= \sum_{x_2,x_3 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,\psi_{13}(x_1,x_3) \underbrace{\sum_{x_4 \in \mathcal{X}} m_5(x_2,x_3,x_4)}_{\triangleq m_4(x_2,x_3)} \\
&= \sum_{x_2,x_3 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,\psi_{13}(x_1,x_3)\,m_4(x_2,x_3) \\
&= \sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1,x_2) \underbrace{\sum_{x_3 \in \mathcal{X}} \psi_{13}(x_1,x_3)\,m_4(x_2,x_3)}_{\triangleq m_3(x_1,x_2)} \\
&= \sum_{x_2 \in \mathcal{X}} \psi_{12}(x_1,x_2)\,m_3(x_1,x_2) \\
&\triangleq m_2(x_1).
\end{aligned}
$$

Finally, we normalize m_2 to get $p_{x_1}(x_1)$, i.e.,

$$
p_{x_1}(x_1) = \frac{m_2(x_1)}{\sum_{x_1' \in \mathcal{X}} m_2(x_1')}.
$$

Evidently, this normalization is obtained by summing out the remaining variable in the elimination order, x_1. The procedure above is an instance of what is referred to as *variable elimination algorithm*, or simply the *elimination algorithm*.

Some aspects of the algorithm are worth emphasizing. First, as our development reflects, this process leads to the construction of intermediate tables m_5, m_4, m_3, and m_2. The sizes of these tables ultimately affects the computational (and storage) complexity of the procedure. In particular, we note that the most expensive summation carried out was in the construction of m_5, which required forming a table over four variables and summing out x_5. This incurs a cost of $O(|\mathcal{X}|^4)$ as $|\mathcal{X}| \to \infty$. Each of the other summations was cheaper, i.e., $o(|\mathcal{X}|^4)$, so the overall computational complexity is $O(|\mathcal{X}|^4)$, improving on the $O(|\mathcal{X}|^5)$ complexity of the naïve approach.

In general, versions of the elimination algorithm corresponding to different choices of elimination ordering have different computational complexities. For example, it is straightforward to check that a better order would be $(4, 5, 3, 2, 1)$. Of course, to find the best order would require checking all $4! = 24$ possibilities.

Finally, we comment that these tables can be interpreted as "messages" that nodes need to share with one another to carry out the marginalization. For example, $m_5(x_2, x_3, x_4)$ can be viewed as a summary of the information that node 5 has to provide to its neighbor nodes 2, 3, and 4. We will further develop this "message-passing" interpretation of distributed implementations of inference algorithms in a subsequent installment of the notes.

10.3.1 The General Case

As our example reveals, the elimination algorithm simply marginalizes out variables in some order, at each step summing only over the relevant potentials and messages.

In general, the key idea is to maintain a list of "active" potentials. Initially, this list includes all potentials associated with our graph, including possibly the Kronecker singleton potentials on observed nodes. Each time we eliminate a variable by summing it out, at least one potential function gets removed from the list of active potentials and a new message (which acts as a potential) gets added to the list. In the earlier example, upon eliminating x_5, potentials ψ_{25} and ψ_{345} were removed from the list of active potentials whereas m_5 was added to the list. With this bookkeeping in mind, we present the elimination algorithm:

Input: Potentials $\psi_{\mathcal{C}}$ over maximal cliques, subset $\mathcal{A}$ over which to compute marginal $p_{\mathsf{x}_\mathcal{A}}(\cdot)$, and an elimination ordering I
Output: Marginal $p_{\mathsf{x}_\mathcal{A}}(\cdot)$
Initialize active potentials Ψ to be the set of input potentials.
for *node i in I that is not in $\mathcal{A}$* **do**
  Let $\mathcal{S}_i$ be the set of all nodes (not including i and previously eliminated nodes) that share a potential with node i.
  Let Ψ_i be the set of potentials in Ψ involving x_i.
  Compute

$$m_i(x_{\mathcal{S}_i}) = \sum_{x_i} \prod_{\psi \in \Psi_i} \psi(x_{\{i\} \cup \mathcal{S}_i}).$$

  Remove elements of Ψ_i from Ψ.
  Add m_i to Ψ.
end
Normalize

$$p_{\mathsf{x}_\mathcal{A}}(x_\mathcal{A}) \propto \prod_{\psi \in \Psi} \psi(x_\mathcal{A}).$$

Algorithm 1: The Variable Elimination Algorithm

Let's analyze the algorithm's complexity. At each step i, we have a table of size $|\mathcal{X}|^{\mathcal{S}_i}$. To fill in each element, we sum $|\mathcal{X}|$ terms, each of which involves multiplying

$|\Psi_i|$ terms. This means that step i, which results in computing message m_i requires $O(|\mathcal{X}|^{|\mathcal{S}_i|}|\mathcal{X}|\,|\Psi_i|) = O(|\Psi_i|\,|\mathcal{X}|^{|S_i|+1})$ operations. Thus, the total complexity is

$$\sum_i O(|\Psi_i|\,|\mathcal{X}|^{|\mathcal{S}_i|+1}).$$

We can upper-bound $|\Psi_i|$ with the number C of maximal cliques in the graph, and $|\mathcal{S}_i|$ with $\max_i |\mathcal{S}_i|$ to obtain

$$\sum_i O(|\Psi_i|\,|\mathcal{X}|^{|\mathcal{S}_i|+1}) = \sum_i O(C\,|\mathcal{X}|^{\max_i |\mathcal{S}_i|+1}) = O(N\,C\,|\mathcal{X}|^{\max_i |\mathcal{S}_i|+1}),$$

recalling that $O(\cdot)$ is an upper bound. Note that the quantity $\max_i |\mathcal{S}_i|$ dictates how efficient the elimination algorithm will be. We next look a way to compute $\max_i |\mathcal{S}_i|$.

10.3.2 The Reconstituted Graph

To compute $\max_i |\mathcal{S}_i|$, we will look at what new edges we introduced during marginalization. Returning to our running example of Fig. 1 and using elimination ordering $(5, 4, 3, 2, 1)$, upon summing out x_5, we are left with potentials $\psi_{12}(x_1, x_2)$, $\psi_{13}(x_1, x_3)$, and $m_5(x_2, x_3, x_4)$, which can be viewed as a new graph where the m_5 term couples x_2, x_3, and x_4, i.e., effectively we have added edges $(2, 3)$ and $(2, 4)$.

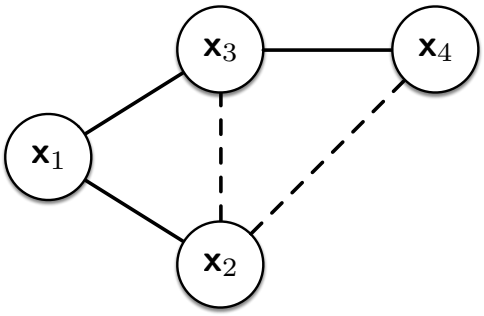
In general, when eliminating a node, the node "sends" a message that couples all its neighbors into a clique. With this analysis, in our running example:

- After eliminating x_5: make $\{2, 3, 4\}$ into a clique (add edges $(2, 3)$ and $(2, 4)$)
- After eliminating x_4: make $\{2, 3\}$ into a clique (already done, so no edge added)
- After eliminating x_3: make $\{1, 2\}$ into a clique (already done, so no edge added)
- After eliminating x_2: nothing left to do

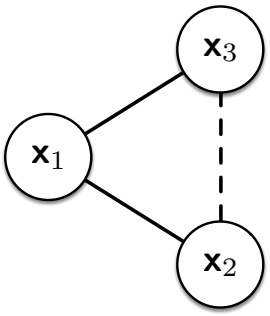
The progressively smaller graphs produced by this sequence of variable eliminations is depicted in Fig. 2. The dashed lines in each graph represent new edges effectively introduced into the graph via the elimination procedure via the intermediate tables produced. The original graph with these additional edges included is referred to as the *reconstituted graph*; that corresponding to our example is depicted in Fig. 3. The factorization corresponding to the reconstituted graph describes the structure in the original distribution that is ultimately exploited by the carrying out variable elimination with the given elimination order.

Here, we denote the added edges using dashed lines.

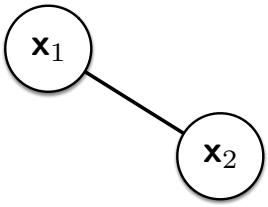
The reconstituted graph has two key properties. First, the largest clique size in the reconstituted graph is $\max_i |\mathcal{S}_i| + 1$ (note that $\mathcal{S}_i$ does not include node i) from our complexity analysis earlier. Second, the reconstituted graph is always chordal



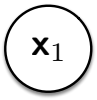
(a) After eliminating x_5.



(b) After eliminating x_4.



(c) After eliminating x_3.



(d) After eliminating x_2.

Figure 2: The sequence of graphs corresponding to eliminating variables in distribution coresponding to the graph of Fig. 1 according to the elimination order $(5, 4, 3, 2, 1)$ to produce marginal $p_{x_1}(\cdot)$. The dashed lines represent edges introduced by the elimination procedure.

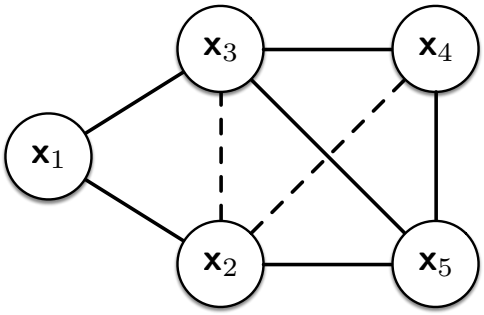


Figure 3: The reconstituted graph corresponding to carrying out variable elimination on the graph of Fig. 1 using the elimination order $(5, 4, 3, 2, 1)$.

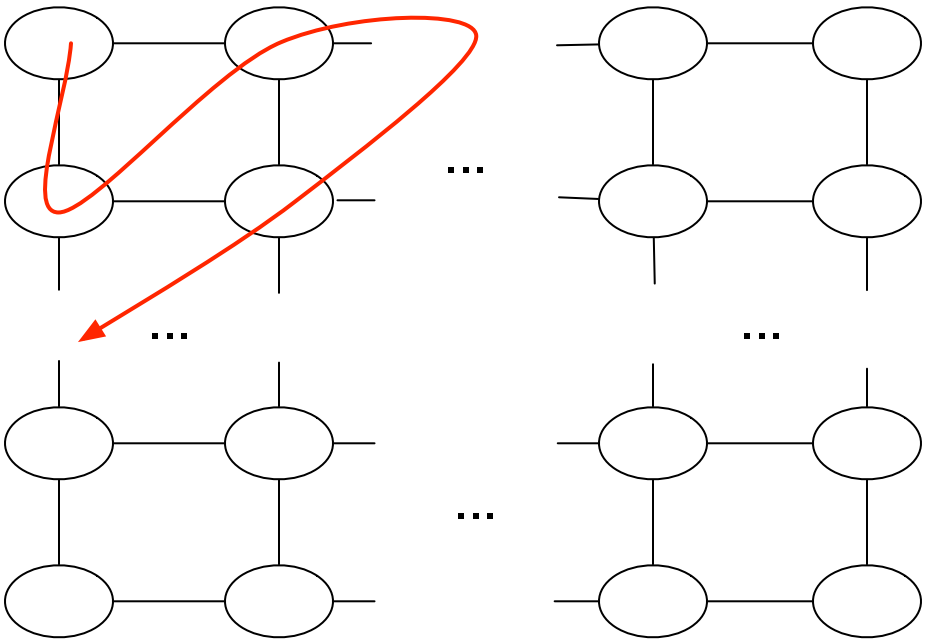


Figure 4: The grid graph

(i.e., every cycle of length at least 4 has edge connecting non-adjacent vertices in the cycle). So, if we want to make a graph chordal, we could just run the elimination algorithm and then construct the reconstituted graph. Note that the reconstituted graph depends on the elimination ordering. To see this, try the elimination ordering $(4, 5, 3, 2, 1)$. The minimum time complexity achieveable using the elimination algorithm for a graph is thus given by $\min_I \max_i |\mathcal{S}_i|$, and this quantity is referred to as the *treewidth* of the graph. As we will see later, in other scenarios, complexity scales with treewidth as well.

You may be wondering how to find a good elimination ordering, or equivalently, how to make a graph chordal with the smallest maximum clique size. For a graph of treewidth k, an algorithm of Eyal Amir (UAI 2001) achieves within a factor $O(\log k)$ of optimal. This was improved by Feige, Hajiaghayi, and Lee in 2005 to $O(\sqrt{\log k})$.

10.3.3 A Comment on Square Lattice Graphs

We end with a neat result, which will be stated without proof. Consider the grid graph over $N = n^2$ nodes as shown in Figure 4 below.

Brute-force marginalization requires $O(|\mathcal{X}|^N)$ operations. However, it is possible to achieve an elimination ordering where $\max_i |\mathcal{S}_i| = \sqrt{N}$, using the zig-zag pattern denoted by the arrow above. In fact, a famous result of Lipton and Tarjan (on a related "vertex separator" problem) implies that any planar graph has an elimination

order such that $\max_i |\mathcal{S}_i| = O(\sqrt{N})$.

10.4 Inferring posterior beliefs with observations

Recall that our ultimate inference goal is calculating posterior beliefs, which correspond to marginals of conditional distributions. Hence, to complete our development, we discuss how to marginalize when some of the variables are observed.

Returning once again to our example of Fig. 1, suppose we seek $p_{x_1|x_3}(x_1|x_3)$, the marginal for x_1 when $x_3 = x_3 = a$ is observed. Then, we can view a as a constant so

$$p_{x_1,x_2,x_4,x_5|x_3}(x_1, x_2, x_4, x_5|a) \propto p_{x_1,x_2,x_3,x_4,x_5}(x_1, x_2, a, x_4, x_5) \tag{3}$$

Thus it suffices to run the elimination algorithm, but with variable x_3 removed from the elimination ordering since it is fixed. Graphically, we depict the constraint that this observation imposes by shading node 3 in the graph, as depicted in Fig. 5(a).

We can equivalently express this modified procedure in terms of the associated *reduced graph*. In particular, we can view the problem of marginalizing over the graph with node 3 removed. To understand this, with a viewed as a constant, the factorization (1) can be expressed as

$$p_{x_1,x_2,x_3,x_4,x_5}(x_1, x_2, a, x_4, x_5) \propto \underbrace{\psi_{12}(x_1, x_2)\, \psi_{13}(x_1, a)}_{\triangleq \psi'_{12}(x_1,x_2)}\ \psi_{25}(x_2, x_5)\ \underbrace{\psi_{345}(a, x_4, x_5)}_{\triangleq \psi'_{45}(x_4,x_5)},$$

which corresponds to the reduced graph depicted in Fig. 5(b). In general, the reduced graph is generated by removing the observed node(s) and all edges incident on them.

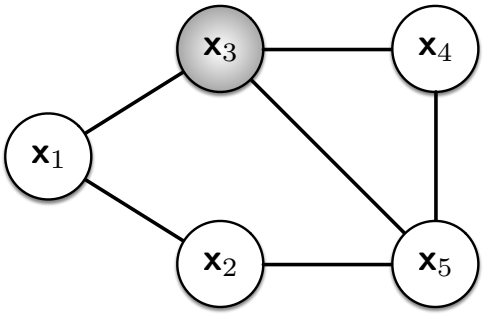
As a further example, Fig. 5(c) describes the scenario when we want to compute marginals conditioned on x_2 and x_4, and Fig. 5(d) is the associated reduced graph.

As an additional perspective, note that yet another equivalent way to have the elimination algorithm produce conditional marginals is to augment the joint distribution with Kronecker singleton potentials. For example, if we observe $x_3 = x3 = a$, then we can write

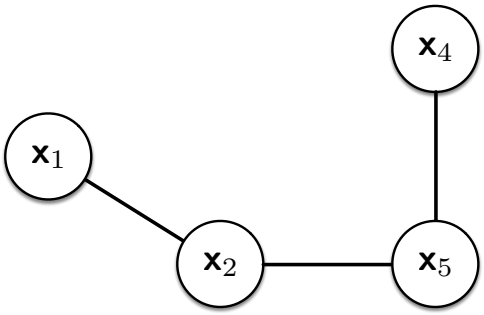
$$p_{x_1,x_2,x_4,x_5|x_3}(x_1, x_2, x_4, x_5|a) \propto p_{x_1,x_2,x_3,x_4,x_5}(x_1, x_2, x_3, x_4, x_5) \cdot \underbrace{\mathbb{1}_{x_3=a}}_{\triangleq \phi_3(x_3)},$$

so it suffices to augment one of the clique potentials involving x_3 in the original factorization—i.e., either $\psi_{13}(x_1, x_3)$ or $\psi_{345}(x_3, x_4, x_5)$—with the additional singleton potential $\phi_3(x_3) = \mathbb{1}_{x_3=a}$. This serves to impose the observation as a constraint in the graph.

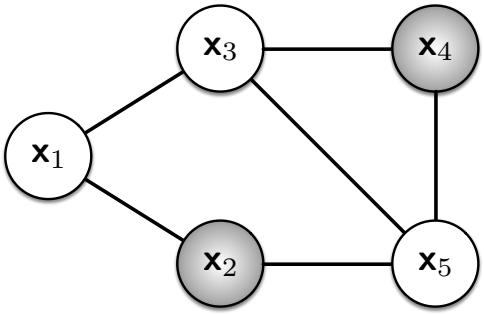
As a final comment, it is important to distinguish between the two types of smaller graphs developed in these notes: those produced by elimination steps as described in Section 10.3.2, and those produced by observations as described in this section. In particular, the former correspond to marginalization, and the latter to conditioning, and thus are generated in very different ways.



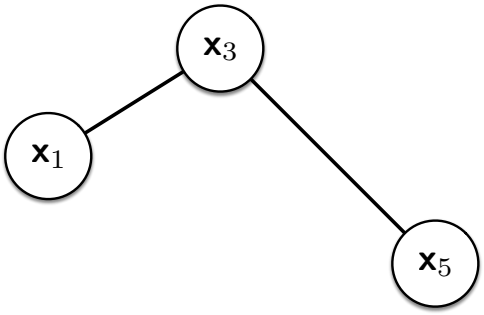
(a) Graph depicting x_3 observed.



(b) Equivalent reduced graph when x_3 observed.



(c) Graph depicting x_2, x_4 observed.



(d) Equivalent reduced graph when x_2, x_4 observed.

Figure 5: Expressing conditioning in the distribution corresponding to the graph of Fig. 1. The top graphs express conditioning on x_3, while the bottom express conditioning on x_2, x_4.

10.5 Maximum A Posteriori (MAP) Estimation

Our development of inference thus far has focused on the first of our main tasks—computing posterior beliefs—with respect to which marginalization plays a key role. We now turn out attention to the second of our main tasks, which is computing *maximum a posteriori (MAP) estimates*. In particular, if $\mathbf{x}$ represents our hidden variables, and $\mathbf{y}$ our observed variables, then recall that the MAP estimate is the mode of the posterior distribution, i.e.,

$$\hat{\mathbf{x}}(\mathbf{y}) \in \arg\max_{\mathbf{x}} p_{\mathbf{x}|\mathbf{y}}(\mathbf{x}|\mathbf{y}) = \arg\max_{\mathbf{x}} p_{\mathbf{x},\mathbf{y}}(\mathbf{x},\mathbf{y}). \tag{4}$$

We emphasize that in (4) we have written $\in$ instead of $=$ because in general the maximum may be attained by multiple values of $\mathbf{x}$, though typically we are interested in any one of them.

As in the case of posterior beliefs, we first reduce the problem to a simpler one without observations, since we can subsequently incorporate the effect of the latter.

10.6 Most Probable Configurations

The corresponding underlying problem without observations is one of computing the *most probable configuration (MPC)* of variables in a joint distribution, i.e., the mode of the distribution. Specifically, with random variables $\mathbf{x} = (x_1, \ldots, x_N) \in \mathcal{X}^N$ distributed according to $p_{\mathbf{x}}(\mathbf{x}) = p_{x_1,\ldots,x_N}(x_1, \ldots, x_N)$, our goal is to compute

$$\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_N) \quad \text{where} \quad \hat{\mathbf{x}} \in \arg\max_{\mathbf{x}\in\mathcal{X}^N} p_{\mathbf{x}}(\mathbf{x}). \tag{5}$$

At the outset, it should be stressed that the problem of MPC computation is fundamentally different from that of marginalization. Specifically, we cannot obtain an MPC via marginalization. In particular, one might wonder if the choice

$$(\hat{x}'_1, \ldots, \hat{x}'_N) \quad \text{with} \quad \hat{x}'_i \in \arg\max_{x_i\in\mathcal{X}} p_{x_i}(x_i)$$

corresponds to an MPC. However, as the following example makes clear, this is generally not the case.

Example 1. Consider the following joint distribution for variables x_1, x_2, x_3, each drawn from the binary alphabet $\mathcal{X} = \{0, 1\}$:

$$p_{x_1,x_2,x_3}(x_1, x_2, x_3) = \begin{cases} 3/10, & \text{if } (x_1, x_2, x_3) = (1, 0, 0) \\ 3/10, & \text{if } (x_1, x_2, x_3) = (0, 1, 0) \\ 2/5, & \text{if } (x_1, x_2, x_3) = (0, 0, 1) \\ 0, & \text{otherwise.} \end{cases}$$

Then the individually most probable values for x_1, x_2, and x_3 are

$$\arg\max_{x_1} p_{\mathsf{x}_1}(x_1) = \arg\max_{x_2} p_{\mathsf{x}_2}(x_2) = \arg\max_{x_2} p_{\mathsf{x}_3}(x_3) = 0,$$

but the most probable (joint) configuration is

$$(\hat{x}_1, \hat{x}_2, \hat{x}_3) = \arg\max_{(x_1,x_2,x_3)} p_{\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_3}(x_1, x_2, x_3) = (0, 0, 1).$$

In fact, the configuration $(0, 0, 0)$ occurs with zero probability!

10.7 The MPC Elimination Algorithm

As discussed in the introductionary notes, MPC computation generally requires an exhaustive search over all possible configurations, the complexity of which is $O(|\mathcal{X}|^N)$. However, as in the case of elimination, when there are conditional independencies in the joint distribution or it otherwise factors, the complexity of MPC computation can be reduced.

To explore the relationship between the factorization structure in the distribution and the complexity of this inference task, we shall focus on distributions represented by undirected graphical models.

10.7.1 An Illustrative Example

Let us start with the example model depicted in Fig. 6, which describes a distribution over a collection of variables $(\mathsf{x}_1, \ldots, \mathsf{x}_5)$ that factors according to

$$\begin{aligned} p_{\mathsf{x}_1,\ldots,\mathsf{x}_5}(x_1, \ldots, x_5) &\propto \exp(\theta_{12}x_1x_2 + \theta_{13}x_1x_3 + \theta_{24}x_2, x_4 + \theta_{34}x_3x_4 + \theta_{35}x_3x_5) \\ &= \exp(x_1x_2 - x_1x_3 - x_2x_4 + x_3x_4 + x_3x_5). \end{aligned} \tag{6}$$

In addition, the variables are binary-valued, i.e., $x_i \in \mathcal{X} = \{0, 1\}$, for $1, \ldots, N$.

In a manner analogous to how we performed marginalization, let us consider performing the required maximization over (6) one variable at a time, according to some elimination order. In particular, let us choose the elimination order $(5, 4, 3, 2, 1)$.

For the first stage, we eliminate x_5 via

$$\max_{\mathbf{x}} p_{\mathbf{x}}(\mathbf{x}) = \max_{x_1,\ldots,x_4} \exp(x_1x_2 - x_1x_3 - x_2x_4 + x_3x_4) \underbrace{\max_{x_5} \exp(x_3x_5)}_{\triangleq m_5(x_3)},$$

where we have grouped the terms involving x_5 and moved the maximization over x_5 past all of the terms that do not involve x_5. Note that the maximization in

$$m_5(x_3) = \max_{x_5} \exp(x_3\, x_5) \tag{7}$$

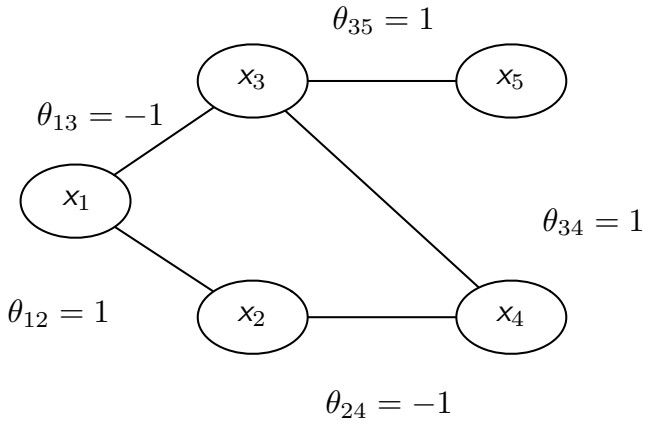


Figure 6: An example undirected graph with node potentials given by $\phi_i(x_i) \equiv 1$ and clique potentials given by $\ln \psi_{ij}(x_i, x_j) = \theta_{ij} x_i x_j$.

is attained by $x_5 = 1$ if $x_3 = 1$, and by either $x_5 = 0$ or $x_5 = 1$ if $x_3 = 0$. Hence, the message (7) can be written in the form

$$m_5(x_3) = e^{x_3}.$$

At this point, we further diverge from the elimination algorithm for marginalization by creating a companion message $x_5^*(x_3)$ that corresponds to a maximizing value of x_5 in (7). Since we are free to break ties arbitrarily, from the above discussion one valid companion message is

$$x_5^*(x_3) = 1.$$

The importance of companion messages will become clear shortly. The resulting graph after eliminating x_5 is depicted in Fig. 7.

For the second stage, we eliminate x_4 in the same way, grouping the terms involving x_4 and moving the maximization over x_4 past terms that do not involve it to obtain

$$\max_{\mathbf{x}} p_{\mathbf{x}}(\mathbf{x}) = \max_{x_1,x_2,x_3} \exp(x_1x_2 - x_1x_3)\, m_5(x_3) \underbrace{\max_{x_4} \exp(x_3x_4 - x_2x_4)}_{\triangleq m_4(x_2,x_3)},$$

from which we obtain our next message pair

$$\begin{aligned} m_4(x_2, x_3) &= \exp(x_3\big(1 - x_2\big)\big) \\ x_4^*(x_2, x_3) &= 1 - x_2. \end{aligned}$$

The resulting graph after eliminating x_4 is depicted in Fig. 8.

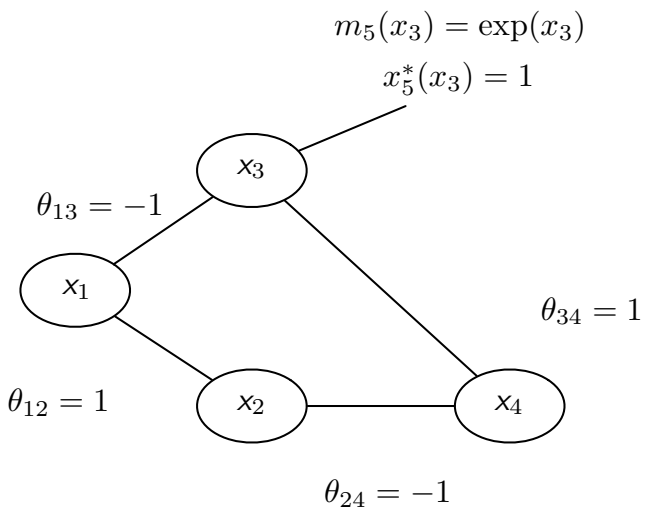


Figure 7: The undirected graph of Fig. 6 after eliminating x_5.

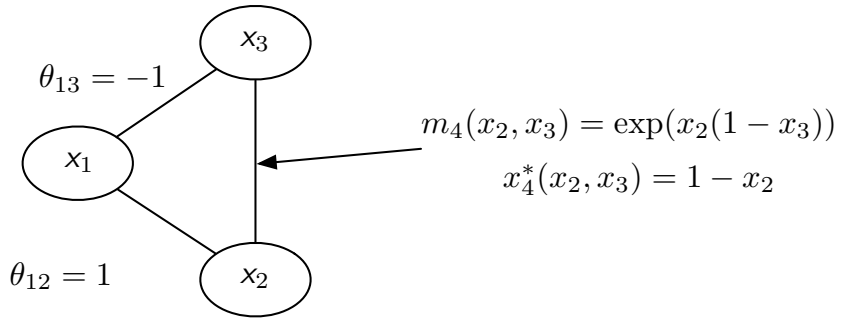


Figure 8: The undirected graph of Fig. 7 after eliminating x_4.

For the third stage, we eliminate x_3 via

$$\begin{aligned}\max_{\mathbf{x}} p_{\mathbf{x}}(\mathbf{x}) &= \max_{x_1,x_2} \exp(x_1x_2) \max_{x_3} \exp(-x_1x_3)\, m_5(x_3)\, m_4(x_2,x_3) \\ &= \max_{x_1,x_2} \exp(x_1x_2) \max_{x_3} \exp\big(-x_1x_3 + x_3 + x_3(1-x_2)\big) \\ &= \max_{x_1,x_2} \exp(x_1x_2) \underbrace{\max_{x_3} \exp\big(x_3(2-x_1-x_2)\big)}_{\triangleq m_3(x_1,x_2)},\end{aligned}$$

whence

$$\begin{aligned}m_3(x_1,x_2) &= \exp(2-x_1-x_2) \\ x_3^*(x_1,x_2) &= 1.\end{aligned}$$

For the fourth stage, we eliminate x_2 via

$$\begin{aligned}\max_{\mathbf{x}} p_{\mathbf{x}}(\mathbf{x}) &= \max_{x_1,x_2} \exp(x_1x_2)\, m_3(x_1,x_2) \\ &= \max_{x_1,x_2} \exp(x_1x_2) \exp(2-x_1-x_2) \\ &= \max_{x_1} \exp(2-x_1) \underbrace{\max_{x_2} \exp\big(x_2(x_1-1)\big)}_{\triangleq m_2(x_1)} \\ &= \max_{x_1} \exp(2-x_1)\, m_2(x_1) \\ &= \max_{x_1} \exp(2-x_1) \qquad (8) \\ &= e^2. \qquad (9)\end{aligned}$$

where to obtain (8) we have used that

$$\begin{aligned}m_2(x_1) &= 1 \qquad (10) \\ x_2^*(x_1) &= 0, \qquad (11)\end{aligned}$$

and where to obtain (9) we have used that (8) is maximized by the choice

$$x_1^* = 0. \qquad (12)$$

We now see that we can read out the most probable configuration by tracing back through the companion messages, which makes their role clear. Specifically, we have

$$\begin{aligned}\hat{x}_1 &= x_1^* = 0 \\ \hat{x}_2 &= x_2^*(x_1^*) = x_2^*(0) = 0 \\ \hat{x}_3 &= x_3^*(x_1^*, x_2^*) = x_3^*(0,0) = 1 \\ \hat{x}_4 &= x_4^*(x_2^*, x_3^*) = x_4^*(0,1) = 1 \\ \hat{x}_5 &= x_5^*(x_3^*) - x_5(1) = 1,\end{aligned}$$

i.e.,

$$\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{x}_5) = (0,0,1,1,1).$$

10.7.2 The General Case

From the preceding example, we see that the MPC elimination algorithm for general undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, corresponding to distributions that factor over the associated maximal cliques $\text{cl}^*(\mathcal{G})$ according to

$$p_{\mathsf{x}_\mathcal{V}}(x_\mathcal{V}) \propto \prod_{\mathcal{C} \in \text{cl}^*(\mathcal{G})} \psi_\mathcal{C}(x_\mathcal{C}),$$

is as follows.

Input: Potentials $\psi_\mathcal{C}$ over maximal cliques, subset $\mathcal{A}$ of nodes over which to compute MPC, and an elimination ordering I

Output: $\hat{x}_\mathcal{A}(x_{\mathcal{V}\setminus\mathcal{A}}) \in \arg\max_{x_\mathcal{A} \in \mathcal{X}^{|\mathcal{A}|}} p_{\mathsf{x}_\mathcal{A}, \mathsf{x}_{\mathcal{V}\setminus\mathcal{A}}}(x_\mathcal{A}, x_{\mathcal{V}\setminus\mathcal{A}})$

Initialize active potentials Ψ to be the set of input potentials.

for *node i in I that is not in $\mathcal{A}$* **do**

Let $\mathcal{S}_i$ be the set of all nodes (not including i and previously eliminated nodes) that share a potential with node i.

Let Ψ_i be the set of potentials in Ψ involving x_i.

Compute

$$m_i(x_{\mathcal{S}_i}) = \max_{x_i} \prod_{\psi \in \Psi_i} \psi(x_{\{i\} \cup \mathcal{S}_i})$$

$$x_i^*(x_{\mathcal{S}_i}) \in \arg\max_{x_i} \prod_{\psi \in \Psi_i} \psi(x_{\{i\} \cup \mathcal{S}_i})$$

where ties are broken arbitarily.

Remove elements of Ψ_i from Ψ.

Add m_i to Ψ.

end

Produce $\hat{x}_\mathcal{A}(x_{\mathcal{V}\setminus\mathcal{A}})$ by traversing I in a reverse order, specifically setting assignment for each $j \in \mathcal{A}$ as

$$\hat{x}_j = x_j^*(x_{j+1}^*, \ldots, x_N^*).$$

Algorithm 2: The MPC Elimination Algorithm

Note that by inclusion of the set $\mathcal{A}$, our general specification of the elimination algorithm includes the case of MAP estimation as a special case. In particular, $x_\mathcal{A}$ represent the latent variables whose most probable configuration we want as a function of $x_{\mathcal{V}\setminus\mathcal{A}}$, which thus represent observed quantities. Of course, we can equivalently fold the effects of these observables directly into the potential functions of the variables of interest, as we did in the case of marginalization, viewing the observables as constants.

We should note too that if we are interested in finding unconditional most probable *subconfigurations* of a subset of variables x_A, then we must first perform marginalization to obtain the relevant distribution before finding the MPC.

Finally, that our new elimination algorithm is so similar in some key respects to our original one for marginalization is not a coincidence. Indeed, while the elimination algorithm for marginalization exploited the distributive law of arithmetic, i.e., that[2]

$$ab + ac = a(b + c), \quad \text{for all } a, b, c, \tag{13}$$

our new elimination algorithm for most probable configuration computation exploits a different distributive law:[3]

$$\max(ab, ac) = a \max(b, c), \quad \text{for nonnegative } a, b, c. \tag{14}$$

From this perspective, anytime we have an inference task for which there is an associated distributive law, we can expect that an elimination algorithm can be developed.

10.7.3 Complexity

Because of the strong structural relationship, the complexity scaling of the elimination algorithm for MPC computation is identical to that for marginalization, i.e.,

$$\sum_i O(|\Psi_i|\,|\mathcal{X}|^{|\mathcal{S}_i|+1}) = \sum_i O(C\,|\mathcal{X}|^{\max_i |\mathcal{S}_i|+1}) = O(N\,C\,|\mathcal{X}|^{\max_i |\mathcal{S}_i|+1}),$$

where C is the number of maximal cliques in the graph. Analogously, the size of the largest clique in the graph strongly influences the complexity.

[2]Note that more elaborate forms of this law, such as, e.g.,

$$\sum_{u,v} f(u)\, g(u, v) = \sum_u f(u) \sum_v g(u, v),$$

immediately follow from (13).

[3]Likewise, more elaborate forms of this law, such as, e.g.,

$$\max_{u,v} f(u)\, g(u, v) = \max_u f(u) \max_v g(u, v), \quad f, g \geq 0,$$

immediately follow from (14).