Variable elimination

Next, we turn our attention to the problem of *inference* in graphical models. Given a probabilistic model (such as a Bayes net or an MRF), we are interested in using it to answer useful questions, e.g., determining the probability that a given email is spam. More formally, we will be focusing on two types of questions:

> *Marginal inference*: what is the probability of a given variable in our model after we sum everything else out (e.g. probability of spam vs non-spam)?

$$p(y = 1) = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_n} p(y = 1, x_1, x_2, \ldots, x_n).$$

> *Maximum a posteriori (MAP) inference*: what is the most likely assignment to the variables in the model (possibly conditioned on evidence).

$$\max_{x_1, \ldots, x_n} p(y = 1, x_1, \ldots, x_n)$$

It turns out that inference is a challenging task. For many probabilities of interest, it will be NP-hard to answer any of these questions. Crucially, whether inference is tractable will depend on the structure of the graph that describes that probability. If a problem is intractable, we will still be able to obtain useful answers via approximate inference methods.

This chapter covers the first exact inference algorithm, *variable elimination*. We will discuss approximate inference in later chapters.

An illustrative example

Consider first the problem of marginal inference. Suppose for simplicity that we are given a chain Bayesian network, i.e. a probability of the form

$$p(x_1, \ldots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}).$$

We are interested in computing the marginal probability $p(x_n)$. We will assume for the rest of the chapter that the x_i are discrete variables taking d possible values each[1].

The naive way of performing this is to sum the probability over all the d^{n-1} assignments to $x_1, \ldots, x_{n-1}$:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1, \ldots, x_n).$$

However, we can do much better by leveraging the factorization of our probability distribution. We may rewrite the sum in a way that ``pushes in'' certain variables deeper into the product.

$$\begin{aligned} p(x_n) &= \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}) \\ &= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1) p(x_1). \end{aligned}$$

We perform this summation by first summing the inner terms, starting from x_1, and ending with x_{n-1}. More concretely, we start by computing an intermediary *factor* $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$ by summing out x_1. This takes $O(d^2)$ time because we must sum over x_1 for each assignment to x_2. The resulting factor $\tau(x_2)$ can be thought of as a table of values (though not necessarily probabilities), with one entry for each assignment to x_2 (just as factor $p(x_1)$ can be represented as a table. We may then rewrite the marginal probability using τ as

$$p(x_n) = \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2) \tau(x_2).$$

Note that this has the same form as the initial expression, except that we are summing over one fewer variable[2]. We may therefore compute another factor $\tau(x_3) = \sum_{x_2} p(x_3 \mid x_2)\tau(x_2)$, and repeat the process until we are only left with x_n. Since each step takes $O(d^2)$ time, and we perform $O(n)$ steps, inference now takes $O(nd^2)$ time, which is much better than our naive $O(d^n)$ solution.

Also, at each time, we are *eliminating* a variable, which gives the algorithm its name.

Eliminating Variables

Having established some intuitions, with a special case, we will now introduce the variable elimination algorithm in its most general form.

Factors

We will assume that we are given a graphical model as a product of factors

$$p(x_1, .., x_n) = \prod_{c \in C} \phi_c(x_c).$$

Recall that we can view a factor as a multi-dimensional table assigning a value to each assignment of a set of variables x_c. In the context of a Bayesian network, the factors correspond to conditional probability distributions; however, this definition also makes our algorithm equally applicable to Markov Random Fields. In this latter case, the factors encode an unnormalized distribution; to compute marginals, we first calculate the partition function (also using variable elimination), then we compute marginals using the unnormalized distribution, and finally we divide the result by the partition constant to construct a valid marginal probability.

Factor Operations

The variable elimination algorithm will repeatedly perform two factor operations: product and marginalization. We have been implicitly performing these operations in our chain example.

The factor product operation simply defines the product $\phi_3 := \phi_1 \times \phi_2$ of two factors ϕ_1, ϕ_2 as

$$\phi_3(x_c) = \phi_1(x_c^{(1)}) \times \phi_2(x_c^{(2)}).$$

The scope of ϕ_3 is defined as the union of the variables in the scopes of ϕ_1, ϕ_2; also $x_c^{(i)}$ denotes an assignment to the variables in the scope of ϕ_i defined by the restriction of x_c to that scope. For example, we define $\phi_3(a, b, c) := \phi_1(a, b) \times \phi_2(b, c)$.

Next, the marginalization operation "locally" eliminates a set of variables from a factor. If we have a factor $\phi(X, Y)$ over two sets of variables X, Y, marginalizing Y produces a new factor

$$\tau(x) = \sum_y \phi(x, y),$$

where the sum is over all joint assignments to the set of variables Y.$\bigoplus$

We use τ to refer to the marginalized factor. It is important to understand that this factor does not necessarily correspond to a probability distribution, even if ϕ was a CPD.

Orderings

Finally, the variable elimination algorithm requires an ordering over the variables according to which variables will be "eliminated". In our chain example, we took the ordering implied by the DAG. It is important note that:

> Different ordering dramatically alter the running time of the variable elimination algorithm.
>
> It is NP-hard to find the best ordering.

We will come back to these complications later, but for now let the ordering be fixed.

The variable elimination algorithm

We are now ready to formally define the variable elimination (VE) algorithm. Essentially, we loop over the variables as ordered by O and eliminate them in that ordering. Intuitively, this corresponds to choosing a sum and "pushing it in" as far as possible inside the product of the factors, as we did in the chain example.

More formally, for each variable X_i (ordered according to O),

1. Multiply all factors Φ_i containing X_i
2. Marginalize out X_i to obtain new factor τ
3. Replace the factors in Φ_i by τ

A former CS228 student has created an interactive web simulation for visualizing the variable elimination algorithm. Feel free to play around with it and, if you do, please submit any feedback or bugs through the Feedback button on the web app.

Examples

Let's try to understand what these steps correspond to in our chain example. In that case, the chosen ordering was $x_1, x_2, \ldots, x_{n-1}$. Starting with x_1, we collected all the factors involving x_1, which were $p(x_1)$ and $p(x_2 \mid x_1)$. We then used them to construct a new factor $\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1)p(x_1)$. This can be seen as the results of steps 2 and 3 of the VE algorithm: first we form a large factor $\sigma(x_2, x_1) = p(x_2 \mid x_1)p(x_1)$; then we eliminate x_1 from that factor to produce τ. Then, we repeat the same procedure for x_2, except that the factors are now $p(x_3 \mid x_2), \tau(x_2)$.

For a slightly more complex example, recall the graphical model of a student's grade that we introduced earlier.$\oplus$ The probability specified by the model is of the form

$$p(l, g, i, d, s) = p(l \mid g)p(s \mid i)p(i)p(g \mid i, d)p(d).$$

Let's suppose that we are computing $p(l)$ and are eliminating variables in their topological ordering in the graph. First, we eliminate d, which corresponds to creating a new factor $\tau_1(g,i) = \sum_d p(g \mid i,d)p(d)$. Next, we eliminate i to produce a factor $\tau_2(g,s) = \sum_i \tau_1(g,i)p(i)p(s \mid i)$; then we eliminate s yielding $\tau_3(g) = \sum_s \tau_2(g,s)$, and so forth. Note that these operations are equivalent to summing out the factored probability distribution as follows:

$$p(l) = \sum_g p(l \mid g) \sum_s \sum_i p(s \mid i)p(i) \sum_d p(g \mid i,d)p(d).$$

Note that this example requires computing at most d^3 operations per step, since each factor is at most over 2 variables, and one variable is summed out at each step (the dimensionality d in this example is either 2 or 3).

Introducing evidence

A closely related and equally important problem is computing conditional probabilities of the form

$$P(Y \mid E = e) = \frac{P(Y, E = e)}{P(E = e)},$$

where $P(X, Y, E)$ is a probability distribution, over sets of query variables Y, observed evidence variables E, and unobserved variables X.

We can compute this probability by performing variable elimination once on $P(Y, E = e)$ and then once more on $P(E = e)$.

To compute $P(Y, E = e)$, we simply take every factor $\phi(X', Y', E')$ which has scope over variables $E' \subseteq E$ that are also found in E, and we set their values as specified by e. Then we perform standard variable elimination over X to obtain a factor over only Y.

Running Time of Variable Elimination

It is very important to understand that the running time of Variable Elimination depends heavily on the structure of the graph.

In the previous example, suppose we eliminated g first. Then, we would have had to transform the factors $p(g \mid i, d), \phi(l \mid g)$ into a big factor $\tau(d, i, l)$ over 3 variables, which would require $O(d^4)$ time to compute. If we had a factor $S \to G$, then we would have had to eliminate $p(g \mid s)$ as well, producing a single giant factor $\tau(d, i, l, s)$ in $O(d^5)$ time. Then, eliminating any variable from this factor would require almost as much work as if we had started with the original distribution, since all the variable have become coupled.

Clearly some ordering are much more efficient than others. In fact, the running time of Variable Elimination will equal $O(md^M)$, where M is the maximum size of any factor during the elimination process and m is the number of variables.

Choosing variable elimination orderings

Unfortunately, choosing the optimal VE ordering is an NP-hard problem. However, in practice, we may resort to the following heuristics:

> *Min-neighbors*: Choose a variable with the fewest dependent variables.
>
> *Min-weight*: Choose variables to minimize the product of the cardinalities of its dependent variables.
>
> *Min-fill*: Choose vertices to minimize the size of the factor that will be added to the graph.

In practice, these methods often result in reasonably good performance in many interesting settings.