1 Variable Elimination Intuition

1.1 Inference

Given unlimited time, inference on a Bayes Net is very straight forward.

1. State the marginal probabilities that you need
2. Figure out the factors needed from the Bayes Net
3. Calculate and combine them under marginalization

For example, consider a case where you need $P(x_1|y_1, y_2)$. We know from Bayes Rule that

$$P(x_1|y_1, y_2) = \frac{P(x_1, y_1, y_2)}{P(y_1, y_2)}$$

Additionally, we know that $P(y_1, y_2)$ is a constant, and there fore need not be calculated. We can instead just **re-normalize** after computing $P(x_1, y_1, y_2)$. Note, **re-normalizing** is the process of finding a factor α such that your distribution * α sum's to 1, and then multiplying by it. In other words, the relative likelihood of each outcome with respect to the others stays the same, they all just get multiplied by a common factor to make sure they sum to 1, as every probability distribution must.

1.2 Why Not Use Inference?

Let's call a **hidden variable** to be a variable in the Bayes Net, but one that does not end up in your query. For example, consider a Bayes Net that contained $X_1, X_2, X_3, ..., X_n$, and we want to know $P(X_1|x_2)$. In this case, $X_3, X_4, ..., X_n$ are all hidden, and must be summed out using marginalization when performing inference.

But what if n is a large number, like 1000. Then we have to marginalize over 998 variables. Assuming binary variables, this has 2^{998} possibilities. We definitely don't want to compute that!

1.3 Variable Elimination

Variable elimination to the rescue! Variable elimination realizes that these hidden nodes may only depend on small subsets of other nodes. For this reason, they can be marginalized in small groups, rather than all at once. Each time we marginalize over one of these groups, we eliminate that variable we just marginalized over, hence the name.

Of course, variable elimination is not guaranteed to be more efficient that inference. For example: there may still be cases where every node depends on all other nodes. For this reason, it is NP Hard.

2 Variable Elimination Framework

Before jumping into an example, lets talk about a general framework for tackling variable elimination problems.

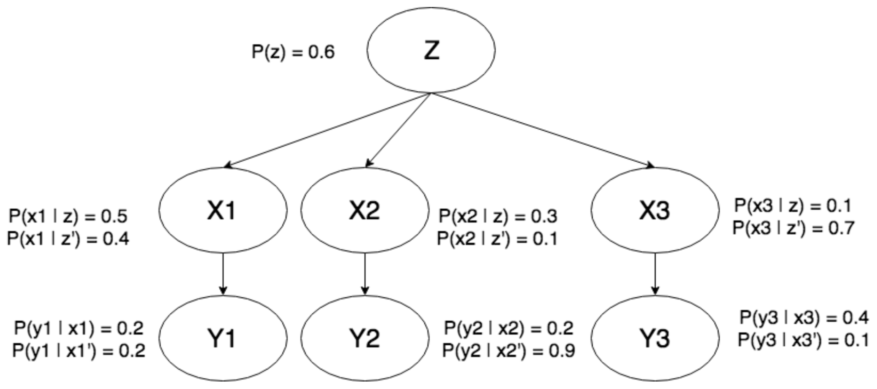
I will define two subprocesses, Join(X) and Eliminate(X)

Join(X): looks at all remaining factors in the problem that involve to variable X, and combines them into one joint table

Eliminate(X): marginalizes over variable X in a probability table, resulting in a distribution over all variables in that table other than X

Now using these, how to we arrive at a result? Well we start by including all factors that are present in the Bayes Net (all marginal probabilities). Then we look at the query, update our factors to reflect the evidence, and determine which variables from the Bayes Net are hidden (not present in the query). For each of these hidden variables, lets call them X, We perform repeated Join(X) followed by Eliminate(X), until all factors not related to the query have been marginalized over.

3 Variable Elimination Example



3.1 Factors

$$P(Z), P(X1|Z), P(X2|Z), P(X3|Z), P(Y1|X1), P(Y2|X2), P(Y3|X3)$$

3.2 Query

Let's assume we want to calculate $P(X3|y1, y2, y3)$

3.3 Calculation

The first step is to substitute all evidence into the original factors:

$$P(Z), P(X1|Z), P(X2|Z), P(X3|Z), P(y1|X1), P(y2|X2), P(y3|X3)$$

Now, we can start performing our join eliminate pairs. All hidden variables in this case are Z, $X1$, and $X2$.

Let start with $X1$. First we will join into a distribution called f_1':

$$f_1'(X1, Z, y1) = P(X1|Z)P(y1|X1)$$

Then we will eliminate $X1$ by marginalizing $X1$ out of f_1'. Will we call this resulting distribution f_1.

$$f_1(Z, y1) = \sum_{X1} f_1'(X1, Z, y1)$$

Now, we replace all original factors we just joined over factors with our new factor. Our new factors are now:

$$P(Z), P(X2|Z), P(X3|Z), P(y2|X2), P(y3|X3), f_1(Z, y1)$$

Next, lets join and eliminate $X2$:

$$f_2'(X2, Z, y2) = P(X2|Z)P(y2|X2)$$

$$f_2(Z, y2) = \sum_{X2} f_2'(X1, Z, y2)$$

Now our factors are:

$$P(Z), P(X3|Z), P(y3|X3), f_1(Z, y1), f_2(Z, y2)$$

Lastly, lets join and eliminate Z:

$$f_3'(Z, y1, y2, X3) = P(Z)P(X3|Z)f_1(Z, y1)f_2(Z, y2)$$

$$f_3(y1, y2, X3) = \sum_{Z} f_3'(Z, y1, y2, X3)$$

Now our factors are:

$$P(y3|X3), f_3(y1, y2, X3)$$

We have now gotten rid of all hidden variables, the last step is to join over the remaining terms into one distribution. The result is a distribution:

$$P(X3, y1, y2, y3)$$

Since we can re-normalize like we discussed before, we are done. Remember:

$$P(X3|y1, y2, y3) = \frac{P(X3, y1, y2, y3)}{P(y1, y2, y3)}$$

4 Variable Ordering

You may notice we skipped over how to select an ordering of variable to eliminate. While you may initially think order doesn't matter, in practice it does quite a bit. Consider the following as an example of order mattering. Instead of there only being three branches in the tree from the example above, consider a case where there are n branches. Additionally, instead of the query being $P(X3|y1, y2, y3)$, it becomes $P(X_n|y1, y2, ..., y_n)$

If we eliminate like we did before, first over all X_i for $i < n$, then each of our join's and eliminates marginalize over one unknown (X_i) and produce a distributions again over only 1 unknown (Z). This is perfect! The size of the tree didn't slow us down other than of course by the scale factor n.

But what if we were to join and eliminate over Z first. Well, there are n children that depend on Z (all of the X's). So such a marginalization would require producing a probability table with n unknowns. In other words, a probability table of size 2^n. This is dramatically worse than our previous ordering!

So how do we determine a good ordering?

4.1 Heuristics

The most obvious, and one that is quite good in practice is the **minimum degree** heuristic. Is states that you should always eliminate the variable that results in the smallest factor possible. That is what we in the example above.

Another common heuristic is is the **minimum fill** heuristic, which is beyond the scope of this class. There are many more, some have formals guarantees while others do not. Choosing the optimal ordering is NP complete (it can be reduced from Max Clique).

5 NumPy

NumPy is a package for scientific computing in Python. It provides multidimensional arrays, matrices, and fast implementations for common operations on these data structures. Many scientific Python based packages are migrating to using NumPy as their underlying data model, and learning how to use

NumPy is a very valuable skill.

Understanding the syntax for NumPy is quite intuitive. Attached are some resources about where to start:

https://docs.scipy.org/doc/numpy-1.13.0/user/whatisnumpy.html
http://cs231n.github.io/python-numpy-tutorial/numpy
https://www.tutorialspoint.com/numpy