12 Bayes Networks

As discussed in Section 11, if you have a query of the form $P(X...|e...)$, the most straightforward step to answering this query is to convert it into **full joint distribution** consisting of the query variables $X...$, the evidence $e...$, and any remaining **hidden variables** Y, Z, W, ... etc. which form the full distribution. The problem is that this joint distribution could be *gigantic* if it involves a lot of variables (it's a high dimensional space) and variables of high complexity (like continuous distributions). It's not reasonable to require the computer to store this massive thing and perform queries over it. What we need to do is come up with a simpler approximation of the full joint which is usable in reality.

So far we've seen one such model: the Naive Bayes model. This model represents the full joint distribution by (1) assuming that there are no hidden Y variables — if hidden variables exist, they're just ignored — and (2) assuming that there are no relationships between the evidence variables except through their effect on the query variable X. These two huge assumptions allow the Naive Bayes model to compute the full joint distribution $P(X), E^{(1)}, E^{(2)}, ...$ as the product of a bunch of simpler (and far smaller) distributions, namely $P(X), P(E^{(1)}|X), P(E^{(2)}|X)$, and so on. This results in a lot of savings.

Just how much savings? Let's say that we had a full joint distribution of some $n+1$ discrete variables $P(X, E^{(1)}, E^{(2)}, ..., E^{(n)})$ and each of these variables — that is, the number of values a variable could take on —was of size k. The size of this distribution is equal to the number of possible combinations of these variables, so it's k^{n+1}. Yuck. But with Naive Bayes, we're storing these as $P(X), P(E^{(1)}|X), P(E^{(2)}|X), ...$ and so on and this is $k + k^2 + k^2... = k + nk^2$ in total size. That's *radically smaller*. Which is the whole point of Naive Bayes: we're willing to accept a bizarrely oversimplifying set of assumptions in order to store the full joint distribution in a small and manageable space.

The problem is that Naive Bayes makes far too simplifying a set of assumptions. In most cases the world just doesn't work like that. But perhaps we can borrow the general idea of breaking the full joint distributions into the product of a bunch of simpler distributions.

What we're going to do is build a model with much looser assumptions than Naive Bayes. Specifically, it permits hidden variables $Y...$, and it does *not* assume that all the variables are conditionally independent given the query variable. But it will make a certain assumption: that there exist quite a number of conditional independence relationships among the variables. Even if *none* of the variables exhibit conditional independence relationships, our model will identify the "most nearly" conditionally independent variables and simply assume they're conditionally independent.

This assumption allows us to once again break our full joint distribution into a bunch of much simpler distributions from which we can compute the full joint distribution. We call this model a **Bayes Network**. In some sense, a Bayes Network identifies and retains only the very *strongest* relationships found among the variables, and discards all of the *weaker* relationships. It will assume that those weaker relationships are conditionally independent given other variables and thus can be ignored.

Querying with the Full Joint Probability Distribution Let's extend our class example. We have a model that involves six boolean variables S, E, W, P, C, T:

- Whether a student **S**tudies hard for the exam

- Whether a student passes the **E**xam
- Whether a student **W**orks hard for on his project
- Whether a student passes the **P**roject
- Whether a student passes the **C**lass, partly as a result of his performance on the exam and project
- Whether a student gets offered an **I**nternship, partly based on the strength of his performance on project only.

Thus the joint probability distribution describing every possible situation is $P(S, E, W, P, C, I)$. Now let's say that a student studied hard for his exam, but didn't work hard on his project. We wish to know if he will or will not pass the class. That is, we want to know $P(C|S = \mathit{true}, W = \mathit{false})$, or simply $P(C|s_+, w_-)$. In this situation we have:

- *Query variables*: C
- *Evidence variables*: S, W
- *Hidden variables*: E, P, I

It's those hidden variables that are of interest to us. In the Naive Bayes model we didn't have hidden variables: but now we do, and we need to know how they impact on our result (or whether they do).

We begin by determining how to convert the query $P(C|s_+, w+-)$ into something which involves the full joint distribution. To start, recall that we can convert it to the joint distribution disregarding the hidden variables as: $P(C|s_+, w_-) = \alpha P(C, s_+, w_-)$. Now we have to roll in the hidden variables.

Marginalization Marginalization is a procedure we can use to add a hidden variable into a joint distribution (or to remove it). Let's say that we have a joint distribution of the form:

$$P(X... \,|\, e...) = \alpha P(X..., e...)$$

We're going to introduce a hidden variable Y into this equation.[95]

$$P(X..., e...) = \sum_{y \in Y} P(X..., y, e...)$$

[95] This looks like magic but it's not. Here's the intuition. Imagine that you want to know the probability of drawing a red card (there are three cards, red, blue, and green). That is, you want to know $P(\mathit{Draw})$ so you can compute $P(\mathit{draw}_{\mathit{red}})$. But you don't have this probability distribution. Instead you have the joint distribution of drawing cards *and* tripping and falling. That is, you have $P(\mathit{Draw, Trip})$. Don't ask why, just accept.

Let's define our distribution as $P(\mathit{Draw, Trip}) = \{P(\mathit{draw}_{\mathit{red}}, \mathit{trip}_+) = 0.1, P(\mathit{draw}_{\mathit{red}}, \mathit{trip}_-) = 0.2, P(\mathit{draw}_{\mathit{blue}}, \mathit{trip}_+) = 0.2, P(\mathit{draw}_{\mathit{blue}}, \mathit{trip}_-) = 0.0, P(\mathit{draw}_{\mathit{green}}, \mathit{trip}_+) = 0.4, P(\mathit{draw}_{\mathit{green}}, \mathit{trip}_-) = 0.1\}$. Now, for each card color there are two trip events: tripping or not tripping. The probability of drawing a card of a given color is simply the sum of these two events. For example, $P(\mathit{draw}_{\mathit{red}}) = P(\mathit{draw}_{\mathit{red}}, \mathit{trip}_+) + P(\mathit{draw}_{\mathit{red}}, \mathit{trip}_-)$. Thus $P(\mathit{Draw}) = \{P(\mathit{draw}_{\mathit{red}}) = 0.3, P(\mathit{draw}_{\mathit{blue}}) = 0.2, P(\mathit{draw}_{\mathit{green}}) = 0.4\}$.

What we did was: $P(\mathit{Draw}) = \sum_{\mathit{trip} \in \mathit{Trip}} P(\mathit{Draw, trip})$. That's marginalization. Essentially we're projecting away the joint space into subspaces where the trip event isn't considered — like summing across a table and writing the sum in the margin. Hence the name.

Using this we can get:

$$P(X... \mid e...) = \alpha P(X..., e...) = \alpha \sum_{y \in Y} P(X..., y, e...)$$

This gets us *one* hidden variable folded in. What if we have multiple hidden variables? Just repeat. Let's say we have another hidden variable called W. We'd have:

$$P(X... \mid e...) = \alpha P(X..., e...) = \alpha \sum_{w \in W} \sum_{y \in Y} P(X..., w, y, e...)$$

... and so on.

Conditioning A related technique, called **conditioning**, allows us to perform a marginalization-like procedure for conditional distributions rather than joint distributions. This is often useful:

$$P(X...) = \sum_{y \in Y} P(X... \mid y) P(y)$$

Or more generally,[96]

$$P(X... \mid e...) = \sum_{y \in Y} P(X... \mid y, e...) P(y \mid e...)$$

If you have multiple hidden variables, you can do the same trick in conditioning as you did in marginalization, along this pattern (I *think* this is right):

$$P(X... \mid e...) = \sum_{w \in W} \sum_{y \in Y} P(X... \mid w, y, e...) P(w \mid y, e...) P(y \mid e...)$$

[96]Oh, you wanted the proof? Let's start with the marginalization rule:

$$P(X..., e...) = \sum_{y \in Y} P(X..., e..., y)$$

Now we multiply both sides by $\frac{1}{P(e...)}$:

$$\frac{P(X..., e...)}{P(e...)} = \sum_{y \in Y} \frac{P(X..., e..., y)}{P(e...)}$$

...and multiply the right side by $\frac{P(y,e...)}{P(y,e...)}$ (which is 1):

$$\frac{P(X..., e...)}{P(e...)} = \sum_{y \in Y} \frac{P(X..., e..., y)}{P(e...)} \frac{P(y, e...)}{P(y, e...)}$$

Rearranging, we have

$$\frac{P(X..., e...)}{P(e...)} = \sum_{y \in Y} \frac{P(X..., y, e...)}{P(y, e...)} \frac{P(y, e...)}{P(e...)}$$

Now we know that $P(A..., b) = P(A... \mid b)\, P(b)$, or $\frac{P(A..., b)}{P(b)} = P(A... \mid b)$, and similarly we also know that $\frac{P(A..., b, c...)}{P(b, c...)} = P(A... \mid b, c...)$. So this allows us to convert some of these terms, and we get:

$$P(X... \mid e...) = \sum_{y \in Y} P(X... \mid y, e...) P(y \mid e...)$$

Computing the Solution with Marginalization We can now marginalize in the hidden variables E, P, T in our example as:

$$P(C|s_+, w_-) = \alpha P(C, s_+, w_-) = \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(C, s_+, w_-, e, p, i)$$

At this point we finally have an equation describing a query $P(X... \mid e....)$ in terms of elements drawn from the full joint distribution. If we had the full joint distribution on hand, we could then just look up $P(x..., y..., e...)$ in the full joint distribution for each permutation of $(x....)$, then run them through the equation.

For example, using our example (all boolean variables), we want to know the distribution $P(C)$, presumably because we want to know $P(c_+)$. So here's how we do it:

1. Compute $P(c_+|s_+, w_-) = \alpha P(c_+, s_+, w_-) = \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(c_+, s_+, w_-, e, p, i)$. Just ignore the α for now.
2. Likewise, compute $P(c_-|s_+, w_-) = \alpha P(c_-, s_+, w_-) = \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(c_-, s_+, w_-, e, p, i)$. Again, ignore the α for now.
3. Normalize these two probabilities (because of α) together. The set of these two probabilities $\{P(c_+|s_+, w_-), P(c_-|s_+, w_-)\}$ together form $P(C|s_+, w_-)$.
4. Return whichever of $P(c_+|s_+, w_-)$ or $P(c_-|s_+, w_-)$ we're interested in (presumably the first).

But that's a full joint distribution of six boolean values: it's a table of only *64 probability numbers.* That's microscopic. Most distributions of interest aren't like this: the full joint distributions are (theoretically) enormous numbers of probabilities, not to say the issues involved with distributions of continuous values. So we cannot assume that we will have the full joint distribution available to us. Instead we'll need to make a simplifying approximation. So back to the **Bayes Network**.

The Bayes Network Let's make some guesses about the nature of our space. If a student *studies for the exam*, this will likely effect his outcome on his exam. But if he *works on his project*, it won't affect his exam grade much (let's assume that studying and working on the project can be done independently — perhaps they're in different parts of the semester). Likewise studying for the exam doesn't affect the project outcome much. Furthermore, let's imagine that the exam and project both affect the class grade, but only the project is likely to affect whether the student is offered an internship.

Also consider that studying affects the course grade, but only because it affects the exam outcome. Thus studying and the course grade are *conditionally independent* given the exam outcome (if we know the exam outcome, whether the student studied tells us nothing about the course grade probability). And so on.

By making a number of these conditional independence and independence assumptions, we might be able to reduce the full joint distribution $P(S, E, W, P, C, I)$ into some equation based on some much simpler set of distributions that we can gather information on. In this case they'll be:

- $P(S)$ The probability that a student studied.
- $P(W)$ The probability that a student worked hard on his project.

- $P(E|S)$ The probability that a student passed his exam given that he studied (or didn't).
- $P(P|W)$ The probability that a student passed his project given that he worked hard on it (or didn't).
- $P(C|E,P)$ The probability that a student passed his the class given that he passed his exam and project (or didn't).
- $P(I|P)$ The probability that a student was offered an internship given that he passed his project (or didn't).

These seem like distributions we could reasonably build from sample data. So how is this a **network**? Because these particular distributions are traditionally shown like the graph structure at right.

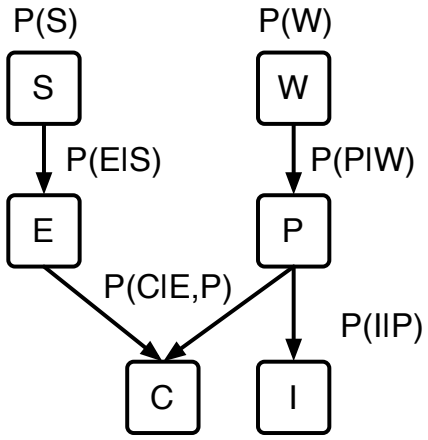

Figure 69 A Bayes Network

In a Bayes Network, the variables are nodes, and the edges between variables suggest the primary relationships of interest which involve those variables. We have discarded the other weaker or conditionally independent relationships and thus are left with a small number of distributions which follow these edges.

There are two kinds of distributions in a Bayes Network. First, there are the non-conditional distributions $P(S)$ and $P(W)$, which appear at the two roots of the network. They're not associated with edges, but directly with the nodes S and W. These are classically known in probability theory (not just Bayes Networks) as **prior probabilities**, and perhaps the diagram at right suggests why.

Second, there are the conditional distributions. In this example (and most commonly in general), for *all* the incoming edges to a certain node, we have exactly one conditional distribution. Thus $P(C|E,P)$ is associated with not one but *two* incoming edges. These distributions appear later than the prior distributions, which might help provide some intuition as to why conditional distributions are classically referred to as **posterior distributions**.

So how do we derive the equation using these distributions? Let's start by noting that, given the other variables S, E, W, and P, the two outcomes C and I are conditionally independent of each other. That is, if you know the settings of S, E, W, and P, knowing what C is doesn't tell you anything about I and vice versa. From this conditional independence, we can break this up into:

$$P(S,E,W,P,C,I) = P(C|S,E,W,P)\ P(I|S,E,W,P)\ P(S,E,W,P)$$

Furthermore, recall if variables A and B... are conditionally independent given D..., then $P(A|B...,D...) = P(A|D...)$, that is, the B... don't tell us anything useful. Now consider that whether a student gets an internship is entirely based on passing his project. Thus we can reduce this to:

$$P(S,E,W,P,C,I) = P(C|S,E,W,P)\ P(I|P)\ P(S,E,W,P)$$

Furthermore, whether a student passes his class is entirely based on the results of his project and exam regardless of how hard he worked on them or studied for them. So we can also say:

$$P(S,E,W,P,C,I) = P(C|E,P)\ P(I|P)\ P(S,E,W,P)$$

Also, recall that $P(A..., B...) = P(A...|B...)P(B...)$. From this we can break up the last term as so:

$$P(S, E, W, P, C, I) = P(C|E, P)\ P(I|P)\ P(E, P, |S, W)\ P(S, W)$$

Now passing one's exam and passing one's project are conditionally independent given the work put in and the studying: if we know how well someone studied and worked, knowing if someone passed his exam tells us nothing useful about whether he passed his project. This allows us to break $P(E, P|S, W)$ as:

$$P(S, E, W, P, C, I) = P(C|E, P)\ P(I|P)\ P(E|S, W)\ P(P|S, W)\ P(S, W)$$

Also consider that whether someone worked hard on his project has nothing to do with whether he passed his exam. Thus we can reduce $P(E|S, W)$ like this:

$$P(S, E, W, P, C, I) = P(C|E, P)\ P(I|P)\ P(E|S)\ P(P|S, W)\ P(S, W)$$

Similarly, whether someone studied hard for their exam has nothing to do with passing their project, thus:

$$P(S, E, W, P, C, I) = P(C|E, P)\ P(I|P)\ P(E|S)\ P(P|W)\ P(S, W)$$

Finally, we know that studying for the exam and working hard on the project are independent of one another. Thus we can say:

$$P(S, E, W, P, C, I) = P(C|E, P)\ P(I|P)\ P(E|S)\ P(P|W)\ P(S)\ P(W)$$

All these assumptions of conditional independence result in us reducing our joint distribution into...., well that's odd..., **it's just the product of all of our little distributions!** How convenient.

Computing with the Bayes Network So now we have an equation describing our full joint distribution in terms of our little distributions. If we want to compute $P(C|s_+, w_-)$ we can now do so as:

$$\begin{aligned}
P(C|s_+, w_-) =& \alpha P(C, s_+, w_-) \\
=& \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(C, s_+, w_-, e, p, i) \\
=& \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(C|e, p)\ P(i|p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-)
\end{aligned}$$

Revising to our computation procedure:

1. Compute $P(c_+|s_+, w_-) = \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(c_+|e, p)\ P(i|p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-)$. Just ignore the α for now.
2. Likewise, compute $P(c_-|s_+, w_-) = \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(c_-|e, p)\ P(i|p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-)$. Again, ignore the α for now.
3. Normalize these two probabilities (because of α) together. The set of these two probabilities $\{P(c_+|s_+, w_-), P(c_-|s_+, w_-)\}$ together form $P(C|s_+, w_-)$.
4. Return whichever of $P(c_+|s_+, w_-)$ or $P(c_-|s_+, w_-)$ we're interested in (presumably the first).

Efficiently Querying Bayes Networks Let's revisit the computation of our query, namely $P(C|s_+, w_-) = \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(C|e,p)\ P(i|p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-)$. Notice that, due to the three summations, we're re-computing a whole lot of probability values over and over again. Maybe we can pull a few probability values or distributions out of those sums for which they don't pertain:

$$\begin{aligned}
P(C|s_+, w_-) =& \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(C|e,p)\ P(i|p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-) \\
=& \alpha \sum_{e \in E} \sum_{p \in P} P(C|e,p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-) \sum_{i \in I} P(i|p) \\
=& \alpha \sum_{e \in E} P(e|s_+)\ P(s_+)\ P(w_-) \sum_{p \in P} P(C|e,p)\ P(p|w_-) \sum_{i \in I} P(i|p) \\
=& \alpha\ P(s_+)\ P(w_-) \sum_{e \in E} P(e|s_+) \sum_{p \in P} P(C|e,p)\ P(p|w_-) \sum_{i \in I} P(i|p)
\end{aligned}$$

Now a lot of probabilities, such as $P(s_+)$, aren't getting recomputed over and over again just because they're inside an unnecessary summation. Now notice that each variable is either associated with a prior distribution or is in the head (left hand side) of a posterior (conditional) distribution, and only once. For example, the "distribution for S" is $P(S)$, while the "distribution for C" is $P(C|E,P)$. If you view the distributions as the variables they're associated with, the four lines above may be viewed as just shuffling around the order of variables:

$$\begin{aligned}
C, I, E, P, S, W \longrightarrow& \alpha \sum_{e \in E} \sum_{p \in P} \sum_{i \in I} P(C|e,p)\ P(i|p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-) \\
C, E, P, S, W, I \longrightarrow& \alpha \sum_{e \in E} \sum_{p \in P} P(C|e,p)\ P(e|s_+)\ P(p|w_-)\ P(s_+)\ P(w_-) \sum_{i \in I} P(i|p) \\
E, S, W, C, P, I \longrightarrow& \alpha \sum_{e \in E} P(e|s_+)\ P(s_+)\ P(w_-) \sum_{p \in P} P(C|e,p)\ P(p|w_-) \sum_{i \in I} P(i|p) \\
S, W, E, C, P, I \longrightarrow& \alpha\ P(s_+)\ P(w_-) \sum_{e \in E} P(e|s_+) \sum_{p \in P} P(C|e,p)\ P(p|w_-) \sum_{i \in I} P(i|p)
\end{aligned}$$

One way to view this is to find the ordering of variables that results in the fewest total probability computations. But even if you figure this out, you're still looking at repeated computations: it can't be helped. For example, in our final arrangement, we're still computing $P(i|p)$ for all four possible settings of i and p *twice*, once for e_+ and once for e_-, because it's inside the $\sum_{e \in E}$ sum even though it doesn't have E as a variable.

A solution to this is to use a little dynamic programming. Let's say we've settled on S, W, E, C, P, I. What we'll do is work on each of these variables in reverse order, starting with P. The idea here is to cache intermediate results in tables called *factors*, which help us avoid repeated calculations.

- We first compute $\sum_{i \in I} P(i|P)$, that is, $\{\sum_{i \in I} P(i|p_+), \sum_{i \in I} P(i|p_-)\}$. Notice it's P and not p, because P is in an outer sum and can't be computed yet. We're going to call this table the *factor* for I, that is, $F(I)$.
- Using this factor table, we can then compute $\sum_{p \in P} P(C|E,p)\ P(p|w_-) \sum_{i \in I} P(i|p)$. Again, notice E rather than e. This is the factor $F(p)$. Just as was the case in the previous factor table,

after the p is summed out, this will have four values, one each for (c_+, e_+), (c_-, e_+), (c_+, e_-), and (c_-, e_-), since C is our query variable and E is in an outer sum.

- Using the factor table $F(P)$, we can compute the factor $F(E)$ as $\sum_{e \in E} P(e|s_+) \sum_{p \in P} P(C|e,p)\ P(p|w_-) \sum_{i \in I} P(i|p)$. This will only have two entries, since C is the only variable and there are no more summations.
- Finally, we can use $F(P)$ to compute $P(s_+)\ P(w_-) \sum_{e \in E} P(e|s_+) \sum_{p \in P} P(C|e,p)\ P(p|w_-) \sum_{i \in I} P(i|p)$. This gives us a table of two entries, which we then normalize (because of α), and we're finished.

This is the essence of a **dynamic programming** procedure called the **Variable Elimination Algorithm**. It is one way to (relatively) efficiently compute exact solutions to queries of a Bayes Network. This could have also been done forward rather than in reverse, using **memoization**, where we store the results of recursive function calls in a cache.

Computing exact solutions to Bayes Network queries, otherwise known as **exact inference**, can be extremely expensive. It's relatively cheap for so-called **polytrees**, that is, graph structures like ours where there is only one path from any node to any other node. But when this is not the case, the cost can become exponential in the number of nodes, even using dynamic programming.

Unfortunately, a great many "real-world" Bayes Network models are not polytrees. This means we need to apply some kind of **approximate inference** algorithm which converges to, well, an approximation of the query solution. These algorithms typically involve random sampling of events from the Bayes Network and computing their effect. The classic algorithm for this, which we will not discuss, is known as **Markov-Chain Monte Carlo**, or **MCMC**.