50.021 -AI

Alex

Week 12: Probabilistic Graphical Models

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Probabilistic Graphical models – a way to efficiently capture probability distributions between large numbers of variables

Recap

Recap: How to use Bayesian nets

- you have a BN modeling $P(X_1, \ldots, X_n)$.
- You have observed values for evidence variables $X_{set2} = (E_1, \ldots, E_l) \subset (X_1, \ldots, X_n)$. You can condition on all values that you have observed.
- You are interested in a model for a joint distribution over Query variables $X_{set1} = Q_1, \ldots, Q_k \subset (X_1 \ldots, X_n)$ conditioned on the observed events
- Usually one then computes $P(Q_1, \ldots, Q_k | E_1, \ldots, E_l) = P(X_{set1} | X_{set2})$ - this involves summing out/marginalizing out all variables in $(X_1, \ldots, X_n) \setminus X_{set1 \cup set2}$, that is all variables which are not in X_{set1} or X_{set2}
- conditioning $P(\cdot | X_{set2})$ is inputting information about observed events.

In this class you should see – only for discrete variables

- a theoretical justification for empirical estimation and laplace smoothing.
- A quick side drop to maximum likelihood and maximum a posteriori.
- How to do efficient variable elimination in Bayes nets
- understanding what a factor is, how the size affects efficiency

How to fit a model - the simplest method

We have designed a Bayesian network. We have data. How to fit the parameters?

Suppose we have a node $P(c|s,b)$ - the probability of lung cancer c, conditioned on smoking (s) and asbestos exposure (a). Now we have a dataset of triples $x_i = (a_i, s_i, c_i), i = 1, \ldots, 10$.

What is a natural way to compute it? Empirical estimate

$$\hat{P}(c = 1|a = 1, b = 0) = \frac{\#\{(a,b,c)|a = 1, b = 0, c = 1\}}{\#\{(a,b,c)|a = 1, b = 0, c \in \{0,1\}\}}$$

Cannot be used to compute events such that conditioning has no data. That is: if $a = 1, b = 1$ has no samples, then this is not usable

Drawback: Unobserved data obtains zero probability.

This might be also undesirable, even when we can compute conditional probabilities:

Suppose one learns probabilities of words over a large vocabulary with many words. Unobserved words get zero probability. This is a kind of overfitting. It might be useful to have for every word a small non-zero probability, even if you never observed it, simply because your set of words might be too small.

Laplace[1] correction is one possible remedy: Suppose a random variable has K outcomes $p_1, \ldots, p_K$, then use

[1] `https://en.wikipedia.org/wiki/Pierre-Simon_Laplace`

$$\hat{p}_i = \frac{\text{\#observed outcomess of type } i + 1}{\text{total number of all observations} + K}$$

What happened here? One pretends to have observed for every outcome one more sample. A ghost sample. In total we have K additional observations, thats why we must divide by the number of original observations $+K$,

This can be generalized to have more smoothing s with s observed ghost samples for every outcome:

$$\hat{p}_i = \frac{\text{\#observed outcomess of type } i + s}{\text{total number of all observations} + s \cdot K}$$

As $s \to \infty$, the smoothed estimates converges towards the uniform distribution, where every outcome has same probability.

Laplace correction is theoretically justifiable as a maximum a posteriori estimate, that is maximum likelihood with a prior distribution on the variable to be modeled.

Laplace correction can be used with $s > 0$, even if $s > 0$ is a small **non-integer** such as 0.09 :).

Can we use this idea for $\hat{P}(c = 1|a = 1, b = 0)$ above ? Yes!

$$P(c = 1|a = 1, b = 0) = \frac{P(c = 1, a = 1, b = 0)}{P(a = 1, b = 0)}$$

Can use laplace correction for estimates of $P(c = 1, a = 1, b = 0)$ and $P(a = 1, b = 0)$. No zero prob failures anymore.

Coin probability estimation - the MLE way

Toss a coin n times. Outcome of i-th toss is a random variable V_i:

$$V_i = \begin{cases} 1 & \text{if in i-th toss the coin has heads} \\ 0 & \text{if in i-th toss the coin has tails} \end{cases}$$

Goal: estimate the probability θ that the coin will show head.
$\theta = p(V = 1|\theta)$

Idea1: can same be used to estimate $P(c = 1, a = 1, b = 0)$. We have n samples $(a_1, b_1, c_1), (a_2, b_2, c_2), \ldots, (a_n, b_n, c_n)$

The i-th variable has value 1: $V_i = 1$ (heads) if it is $c = 1, a = 1, b = 0$. the i-th variable has value 0: $V_i = 0$ (tails) if it is every other event regarding values of (a, b, c).

Idea2: can same be used to estimate $P(c = 1, a = 1|b = 0, d = 1)$. Consider only all samples with $b = 0, d = 1$, heads is $c = 1, a = 1$, tails is every other event among above restricted set of samples.

The maximum likelihood principle (discrete variables)

Given a model $P(V_1, \ldots, V_N|\theta)$ which depends on a parameter θ, choose θ such that the probability of the observed data is maximized.

$$\theta_{MLE} = \text{argmax}_\theta P(V_1 = v_1, \ldots, V_n = v_n|\theta)$$

MLE for coins

We make one assumption: knowing θ we have conditional independence of coin tosses

$$P(V_1 = v_1, \ldots, V_n = v_n|\theta) = P(V_1 = v_1|\theta)P(V_2 = v_2|\theta) \cdot \ldots \cdot P(V_n = v_n|\theta)$$

We know

$$P(V_i = v_i|\theta) = \begin{cases} \theta & \text{if } v_i = 1 \\ 1-\theta & \text{if } v_i = 0 \end{cases}$$
$$= \theta^{[v_i==1]}(1-\theta)^{[v_i==0]}$$
$$= \theta^{[v_i==1]}(1-\theta)^{1-[v_i==1]}$$

Therefore:

$$P(V_1 = v_1, \ldots, V_n = v_n|\theta) = \prod_{i=1}^{n} \theta^{[v_i==1]}(1-\theta)^{1-[v_i==1]}$$
$$= \theta^{\sum_{i=1}^{n}[v_i==1]}(1-\theta)^{\sum_{i=1}^{n}(1-[v_i==1])}$$
$$= \theta^{\sum_{i=1}^{n}[v_i==1]}(1-\theta)^{n-\sum_{i=1}^{n}[v_i==1]}$$

Let c_1 be the sum of heads $c_1 = \sum_{i=1}^{n}[v_i == 1]$, then

$$P(V_1 = v_1, \ldots, V_n = v_n|\theta) = \theta^{c_1}(1-\theta)^{n-c_1}$$

Which θ maximizes the term?

Lets compute its derivative

$$0 = c_1\theta^{c_1-1}(1-\theta)^{n-c_1} - (n-c_1)\theta^{c_1}(1-\theta)^{n-c_1-1} \quad | \cdot \frac{1}{\theta^{(c_1-1)}(1-\theta)^{(n-c_1-1)}}$$
$$0 = c_1(1-\theta) - (n-c_1)\theta = c_1 - c_1\theta - n\theta + c_1\theta$$
$$n\theta = c_1$$
$$\theta = \frac{c_1}{n}$$

so the MLE estimator for the probability of heads is just $\frac{c_1}{n}$ - the ratio of heads to total number of samples. This explains the reasoning in last lecture for using the empirical estimates

Coin probability estimation - the MAP way

Instead of the maximum likelihood principle one can use the maximum a posteriori estimator.

The idea is: If you have a guess for θ, where it should be, then you can express it by a probability distribution $P(\theta)$ over values of θ. Its called the prior distribution because it expresses our prior guess before seeing any data.

MAP principle: Given a model $P(V_1, \ldots, V_N|\theta)$ which depends on a parameter θ, choose θ such that the probability of the observed data, **weighted with the prior distribution** $p(\theta)$ is maximized.

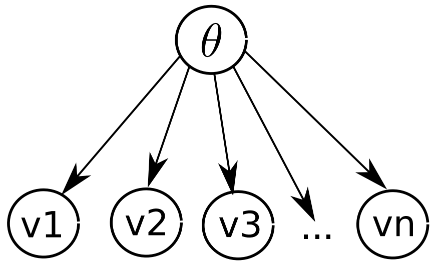
$$\theta_{MAP} = \text{argmax}_\theta P(V_1 = v_1, \dots, V_n = v_n|\theta)P(\theta)$$

Note that

$$P(V_1 = v_1, \dots, V_n = v_n|\theta)P(\theta) = P(V_1 = v_1, \dots, V_n = v_n, \theta)$$

This is a joint! Have Bayesian network for that?

If we assume that coin flips are statistically independent conditional on having observed θ, then



$$P(v_1, \dots, v_n, \theta) = P(\theta)\prod_{i=1}^{n} p(v_i|\theta)$$

So one can get the MAP approach in this case also by using a simple Bayesian network!

the MAP estimator as the argmax of the posterior

One more important aspect: $P(V_1 = v_1, \dots, V_n = v_n, \theta)$ is related to $P(\theta|V_1 = v_1, \dots, V_n = v_n)$ which is the probability of parameter θ after having observed the coin flips as evidence.

$$P(\theta|V_1 = v_1, \dots, V_n = v_n) = \frac{P(V_1 = v_1, \dots, V_n = v_n, \theta)}{P(V_1 = v_1, \dots, V_n = v_n)}$$

$$\text{but } P(v_1, \dots, v_n) = \int_\theta P(v_1, \dots, v_n, \theta)d\theta = \text{ a real number that does not contain } \theta$$

So what does it mean?

We get the same solution θ_{MAP}, no matter we maximize $P(\theta)$ or $3 \cdot P(\theta)$ or $P(\theta)/15$!

Since the normalizer $P(V_1 = v_1, \dots, V_n = v_n)$ is a positive real number that does not contain θ,

$$\begin{aligned}\theta_{MAP} &= \text{argmax}_\theta P(V_1 = v_1, \dots, V_n = v_n|\theta)P(\theta)\\ &= \text{argmax}_\theta \frac{P(V_1 = v_1, \dots, V_n = v_n|\theta)P(\theta)}{P(V_1 = v_1, \dots, V_n = v_n)}\\ &= \text{argmax}_\theta P(\theta|V_1 = v_1, \dots, V_n = v_n)\end{aligned}$$

So, the MAP estimator is the single parameter which maximizes the probability of θ after having observed the coin flips as evidence.

From MAP to Laplace smoothing

We will show here that Laplace smoothing is the argmax solution, when we use $P(\theta)$ as the Beta distribution

The Beta distribution:

$$P(\theta) = P(\theta|\alpha, \beta) = \frac{1}{c}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

where $z = z(\alpha, \beta)$ is a normalizer, that wont play a role here (wikipedia lookup, a quotient of gamma functions).

Then:

$$\begin{aligned}P(v_1, \dots, v_n, \theta) &= \frac{1}{c}\theta^{\alpha-1}(1-\theta)^{\beta-1}\theta^{c_1}(1-\theta)^{n-c_1}\\ &= \frac{1}{z}\theta^{c_1+\alpha-1}(1-\theta)^{n-c_1+\beta-1}\end{aligned}$$

We this has same shape as the MLE problem above!

Maximizing this for θ yields

$$\begin{aligned}0 &= (c_1 + \alpha - 1)(1-\theta) - (n - c_1 + \beta - 1)\theta\\ \theta(c_1 + \alpha - 1 + n - c_1 + \beta - 1) &= c_1 + \alpha - 1\\ \theta &= \frac{c_1 + \alpha - 1}{n + \alpha + \beta - 2}\end{aligned}$$

Now set $\alpha = s + 1, \beta = s + 1$

$$\theta = \frac{c_1 + s}{n + 2s}$$

This is laplace smoothing with parameter s, and the standard solution if $s = 0$! As claimed above, it has a theoretical foundation via a MAP solution with a special prior.

For the extra knowledge: we also know: $P(\theta|v_1, \dots, v_n)$ is equal to $P(v_1, \dots, v_n, \theta)$ times some real number (which ensures that $P(\theta|v_1, \dots, v_n)$ integrates up to 1 over θ). This gives us :

$$P(\theta|v_1,\ldots,v_n) = \frac{1}{z^*} \qquad \theta^{(c_1+\alpha)-1}(1-\theta)^{(n-c_1+\beta)-1}$$
$$\text{Beta-Dist: } P(\theta) = \frac{1}{z} \qquad \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

We can see that $P(\theta|v_1,\ldots,v_n)$ is a beta distribution itself with parameters $\alpha^* = c_1+\alpha, \beta^* = n-c_1+\beta$

$$P(\theta|v_1,\ldots,v_n) = \frac{1}{z^*}\theta^{\overbrace{c_1+\alpha}^{}_{\alpha^*}-1}(1-\theta)^{\overbrace{n-c_1+\beta}^{}_{\beta^*}-1}$$

Choosing a prior which fits a likelihood such that the posterior distribution has same distribution as the prior is called conjugate prior. They are computationally simple, but restrict your modelling.

General Inference

- you have a BN modeling $P(X_1,\ldots,X_n)$.
- You have observed values for evidence variables $(E_1,\ldots,E_l) \subset (X_1,\ldots,X_n)$. You can condition on all values that you have observed.
- You are interested in a model for a joint distribution over Query variables $Q_1,\ldots,Q_k \subset (X_1\ldots,X_n)$ conditioned on the observed events
- **Goal:** Want to compute $P(Q_1,\ldots,Q_k|E_1=e_1,\ldots,E_l=e_l)$
- this involves summing out/marginalizing out all variables which are not in E or Q
- denote $H_1,\ldots,H_r$ all variables which are not in E or Q

  How to compute $P(Q_1,\ldots,Q_k|E_1=e_1,\ldots,E_l=e_l)$ **using the BN?**

The general variable elimination algorithm

Typically three steps:

1⊙ For all variables E_i with evidence, set them to the observed values $E_i = e_i$

2⊙ Sum out all variables H_i

3a⊙ then we have obtained the marginal $P(Q_1,\ldots,Q_k,E_1=e_1,\ldots,E_l=e_l)$

3b⊙ since

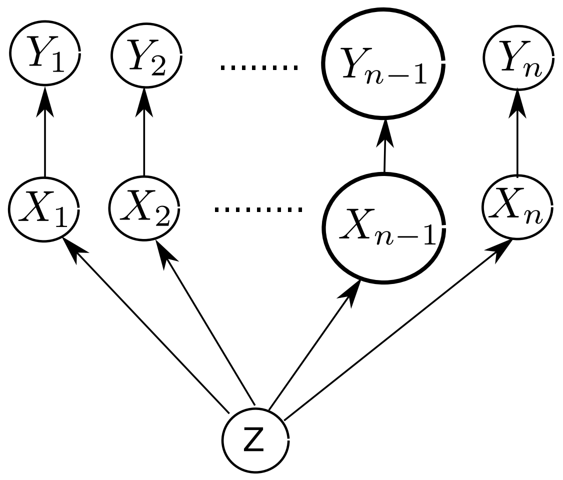
$$P(Q_1,\dots,Q_k|E_1=e_1,\dots,E_l=e_l)=\frac{P(Q_1,\dots,Q_k,E_1=e_1,\dots,E_l=e_l)}{P(E_1=e_1,\dots,E_l=e_l)}$$

, we must compute the normalizer

$$P(E_1=e_1,\dots,E_l=e_l)=\sum_{Q_1,\dots,Q_k}P(Q_1,\dots,Q_k,E_1=e_1,\dots,E_l=e_l)$$

A bad example of variable elimination - the naive way

Consider the following problem:



$Z\to X_i\to Y_i\ \forall i$

Goal: $P(X_n,Y_1=y_1,\dots,Y_n=y_n)$

The direct/naive approach is to construct the joint first, then sum out over the hidden variables:

$$P(X_n,Y_1=y_1,\dots,Y_n=y_n)=\sum_z\sum_{x_1}\sum_{x_2}\dots\sum_{x_{n-1}}P(Z,X_1,\dots,X_{n-1},X_n,Y_1=y_1,\dots,Y_n=y_n)$$
$$\sum_z\sum_{x_1}\sum_{x_2}\dots\sum_{x_{n-1}}P(Z)\prod_i P(Y_i|X_i)P(X_i|Z)$$

We will see: this is computationally very expensive:

$2\cdot 2^{n-1}$ summands,
each summand is
$P(Z,X_1,\dots,X_{n-1},X_n,Y_1=y_1,\dots,Y_n=y_n)=P(Z)\prod_i P(Y_i|X_i)P(X_i|Z)$

so each summand has 1 (from $P(Z)\cdot$) $+2n$ (from $\prod_i P(Y_i|X_i)P(X_i|Z)$
) multiplications,

so: $2^n - 1$ add, $2^n(1+2n)$ mul ops!

For this graph we can do faster!

Lets look at a simple example to see the computational impact of summing early versus summing late:

$$\begin{aligned}
&(\sum_i a_i)(\sum_k b_k)(\sum_l c_l) = \\
&= (a_1 + a_2)(b_1 + b_2)(c_1 + c_2) \ \rightarrow \ \text{2 mult, 3 additions} \\
&= a_1 b_1 c_1 + a_1 b_1 c_2 + a_1 b_2 c_1 + \ldots \\
&= \sum_i \sum_k \sum_l a_i b_k c_l \ \rightarrow 2^3 \text{ times 2 mult, } 2^3 - 1 \text{ add}
\end{aligned}$$

Lets apply this idea and consider and alternative shape:

A smart example of variable elimination - same bayesian network

$$P(X_n, Y_1 = y_1, \ldots, Y_n = y_n) = \sum_z \sum_{x_1} \sum_{x_2} \ldots \sum_{x_{n-1}} P(Z, X_1, \ldots, X_{n-1}, X_n, Y_1 = y_1, \ldots, Y_n = y_n)$$

$$\sum_z P(z) \cdot P(Y_n = y_n | X_n) P(X_n | Z) \cdot \left(\left(\sum_{x_1} P(Y_1|X_1) P(X_1|Z) \right) \left(\sum_{x_2} \right) \ldots \left(\sum_{x_{n-1}} P(Y_{n-1}|X_{n-1}) P(X_{n-1}|Z) \right) \right)$$

All I did was putting sums inside.

How much operations are involved in computing the marginal this way?

$\left(\sum_{x_{n-1}} P(Y_{n-1} = y_{n-1}|X_{n-1}) P(X_{n-1}|Z)\right)$ is:
1 add, 2 mul times possible values of the unfixed variables - which is only Z, so times 2, and so:
2 add, 4 mul

have this $n-1$ times: $\left(\sum_{x_1}\right) \left(\sum_{x_2}\right) \ldots \left(\sum_{x_{n-1}}\right)$

$\sum_{x_i} P(Y_i = y_i|X_i) P(X_i|Z)$ total for all $i \leq n-1$ has cost
$2(n-1)$ add, $4(n-1)$ mul

lets count the multiplications in between: $\left(\sum_{x_1}\right)_1 \left(\sum_{x_2}\right)_2 \left(\sum_{x_2}\right)_3 \cdots$

$\left(\sum_{x_1}\right)_1 \left(\sum_{x_2}\right)_2$ has 1 mul in between, so in total $n-2$ mul additionally, but for both possible values of Z it is $2(n-2)$ mul.

Note in all terms we use either $Z = 0$ or $Z = 1$ at the same time. We never have to multiply one term that has $Z = 0$ with another term that has $Z = 1$. thats why it is $2(n-2)$ mul only.

for now: $2(n-1)$ add, $6(n-1)-2$ mul

$\sum_z P(z) \cdot P(Y_n = y_n|X_n) \cdot P(X_n|Z) \cdot ()_z$ is 1 add times both possible values of X_n, 3 mul times both possible values of X_n and both possible values of z, so 12 mul

total: $2(n-1)+2$ add, $6(n-1)-2+12 = 6n+10$ mul.

compare against above:

$$\begin{aligned} naive\ add &: 2^n - 1 \\ naive\ mul &: 2^n(1+2n) \\ smart\ add &: 2n \\ smart\ mul &: 6n+10 \end{aligned}$$

Result: putting summing early inside – when done in the right order– can make computation much faster! This is the core idea of variable elimination.[2]

[2] but you cannot always do that successfully

Side note: if you would sum over Z first, then you would have much more ops!

analysis tool: factor

We have encountered factors like

$$f(Z) = \sum_{x_{n-1}} P(Y_{n-1}|X_{n-1})P(X_{n-1}|Z)$$

on the way to compute the marginal. A **factor** is any combination of (products/sums) of terms of the BN which come up when ... computing the marginal that one wants.

In particular, if you would describe factor by a generating grammar:

- any conditional probability term $P(X_i|parents(X_i))$ is a factor $f(\{X_i\} \cup parents(X_i))$
- a product of factors is a factor (like $f(X_{n-1}, Z) = P(Y_{n-1} = y_{n-1}|X_{n-1})P(X_{n-1}|Z))$
- a sum of a factors over (the hidden) variables is a factor.

the factor notation:

We denote for a factor $f(A, B, C)$ to express: A,B,C are variables which appear in the terms of the factor and which are not fixed by evidence, for example

$$f(A, B, C) = P(A|B)P(B)P(C)$$

We do not mention fixed evidence variables in the arguments, so

$$P(Y = y|B) = f(B)$$

The we call A, B, C in $f(ABC)$ **the free variables** in a factor

factors are like n-dimensional matrices

examples:

$$f(A) = P(A)$$

A	P(A)
0	0.2
1	0.8

$= \begin{pmatrix} 0.2 & 0.8 \end{pmatrix}$ like a one-dim matrix

$$f(AB) = P(A)P(B)$$
$$f(AB) = P(A|B)P(B)$$

A	B	f(A,B)
0	0	0.1
1	0	0.2
0	1	0.65
1	1	0.05

$= \begin{pmatrix} 0.1 & 0.65 \\ 0.2 & 0.05 \end{pmatrix}$ - like a two-dim matrix

$$f(ABC) = P(A|B)P(B)P(C)$$

A	B	C	f(A,B,C)
0	0	0	0.05
1	0	0	0.1
0	1	0	0.3
1	1	0	0.05
0	0	1	0.1
1	0	1	0.2
0	1	1	0.05
1	1	1	0.15

$$f(A,B,C=0) = \begin{pmatrix} 0.05 & 0.3 \\ 0.1 & 0.05 \end{pmatrix}$$
$$f(A,B,C=1) = \begin{pmatrix} 0.1 & 0.05 \\ 0.2 & 0.15 \end{pmatrix}$$
$$\text{Stack } \begin{pmatrix} 0.05 & 0.3 \\ 0.1 & 0.05 \end{pmatrix} \text{ and } \begin{pmatrix} 0.1 & 0.05 \\ 0.2 & 0.15 \end{pmatrix} \text{ into a 3-dimensional matrix of } 2 \times 2 \times 2!$$

How to do variable elimination

There are two parts to efficient variable elimination:

- sum out early
- choose a good order of variables

What is a **good order**?

a good order means: the factors created when summing out hidden variables should have only a few free variables. to see this, by example ;) :

- $P(Z, X_1, \ldots, X_{n-1}, X_n, Y_1 = y_1, \ldots, Y_n = y_n)$ is a factor with many unset variables - all X_i plus Z!

  it needs 2^n real numbers to store it.

- the factor $\left(\sum_{x_{n-1}} P(Y_{n-1}|X_{n-1})P(X_{n-1}|Z)\right)_{n-1}$ has only one unset variable, because $Y_i = y_i$ is fixed by evidence, and X_i is summed out. So it is only a function of Z!

  - it needs only 2 real numbers to store it!

- $\left(\left(\sum_{x_1}\right)_1 \left(\sum_{x_2}\right)_2 \cdots \left(\sum_{x_{n-1}} P(Y_{n-1}|X_{n-1})P(X_{n-1}|Z)\right)_{n-1}\right)_z$ looks big, but it is a product of $n-1$ functions which only depends on Z. and we only multiply terms with the same z-value

  So it is again just a function of Z! - it needs only 2 real numbers to store it!

- there is not always an efficient solution to the variable order problem, as some problems are equivalent to $3-SAT$ Problems, which are NP-hard. so there should not exist a solution which can keep factors small for such problems, unless $P = NP$ which seems not to be true.

We will explain below why the size of a factor matters.

Example: How to do variable elimination

Lets execute above algorithm on this BN:

$A \rightarrow B \rightarrow C \rightarrow D$ for $P(A, D = 1)$

A	P(A)
0	0.3

A	B	P(B|A)
0	0	0.5
1	0	0.2

B	C	P(C|B)
0	0	0.1
1	0	0.8

C	D	P(D|C)
0	0	0.3
1	0	0.6

$$\begin{aligned}
P(A, D = 1) &=? \\
P(A, B, C, D) &= P(D|C)P(C|B)P(B|A)P(A) \\
P(A, B, C, D = 1) &= P(D = 1|C)P(C|B)P(B|A)P(A) \\
P(A, D = 1) &= \sum_B \sum_C P(D = 1|C)P(C|B)P(B|A)P(A) \\
&= \sum_C P(D = 1|C) \sum_B P(C|B)P(B|A)P(A) \\
&= \sum_C P(D = 1|C) \underbrace{\sum_B P(C|B)P(B|A)}_{=f_1(AC)} P(A)
\end{aligned}$$

Here we took two entries from the BN $P(C|B), P(B|A)$, multiplied them, and summed out a variable, B, in them. The result is a new factor $f_1(AC)$ of variables C and A . Note that this function does not depend anymore on the summed variable B.

$$f_1(AC) = \sum_B P(C|B)P(B|A)$$

A	C	$f_1(AC)$
0	0	?
1	0	?
0	1	?
1	1	?

this function $f_1(AC)$ is then multiplied with other terms:

$$\begin{aligned} P(A,B,C,D=1) &= \sum_C P(D=1|C) \underbrace{\sum_B P(C|B)P(B|A)}_{=f_1(AC)} P(A) \\ &= \sum_C P(D=1|C) f_1(AC) P(A) \\ &= \underbrace{\sum_C P(D=1|C) f_1(AC)}_{=f_2(A)} P(A) \\ &= f_2(A)P(A) \end{aligned}$$

Here we introduced another factor $f_2(A)$ - which is a function of only one variable, because the value for D is already fixed, because it is an evidence variable

A	$f_2(A)$
0	?
1	?

We see that two operations are involved: multiplying factors[3], and summing them[4]

[3] remember: any term $P(X_i|parents)$ is by definition also a factor

[4] Side note: in theory one can have two factors with the same signature $f(A)$ but different BN terms in it. For finding a good order that does not matter. For avoiding confusion I added a subscript to distinguish factor not only by their variable signature (e.g. $f(A), f(ABC), f(ACE)$) but also by a subscript.

Variable elimination algorithm with factors

1⊙ For all variables E_i with evidence, set them to the observed values $E_i = e_i$. Create an initial list of factors: the set of all $p(x_i|parents(X_i))$

2a⊙ Pick a variable ordering, see below how

2b⊙ pick a hidden variable H_i, create a new factor which is a product of all existing factors which contain H_i. Sum out H_i. Put the resulting factor into the list.

Finish when no H_i left :)

3⊙ Now we have obtained the marginal $P(Q_1, \ldots, Q_k, E_1 = e_1, \ldots, E_l = e_l)$

since

$$P(Q_1, \ldots, Q_k | E_1 = e_1, \ldots, E_l = e_l) = \frac{P(Q_1, \ldots, Q_k, E_1 = e_1, \ldots, E_l = e_l)}{P(E_1 = e_1, \ldots, E_l = e_l)}$$

, we must compute the normalizer

$$P(E_1 = e_1, \ldots, E_l = e_l) = \sum_{Q_1, \ldots, Q_k} P(Q_1, \ldots, Q_k, E_1 = e_1, \ldots, E_l = e_l)$$

and divide, of course, to get our conditional *the size of a factor*

Lets see now how factor sizes affect computational cost when marginalizing. There are two operations involved:

- multiplying factors
- summing factors

what comes out when we multiply 2 factors ? $f(set1) \cdot f(set2)$

$$f(AB) = P(A)P(B),\ f(AC) = P(A|C),\ f(AB) \cdot f(AC) = f(?)$$
$$f(ABC) = P(A|B,C)P(B|C)P(C), f(BCEF) = P(B|E,F)P(C|F)$$
$$f(ABC) \cdot f(BCEF) = f(?)$$

- rule is $f(set1) \cdot f(set2) = f(set1 \cup set2)$ - the new factor contains all the variables from both factors
- size of a factor $f(set)$ is (if all variables have 2 values): $2^{|set|}$
- what happens if you sum out a variable?
  $\sum_v f(set) = f(set \setminus \{v\})$ – remember in a factor a variable can appear on both sides

We will see: computational costs are related to the size of a factor

the computational cost of multiplying two factors

$$f(AB)f(B) = f(AB)$$

2^2 mul

Why?: you multiply $f(A=0,B=0)$ with $f(B=0)$, $f(A=0,B=1)$ with $f(B=1)$, $f(A=1,B=0)$ with $f(B=0)$ and $f(A=1,B=1)$ with $f(B=1)$.

$$f(A)f(B) = f(AB)$$

2^2 mul, too

$$f(ABC)f(B) = f(AB)$$

2^3 mul

$$f(ABV)f(AC) = f(ABCV)$$

we need to do 2^4 multiplications, one for every value of each variable

the computational cost of summing a factor

the factor inside was multiplied before, so we look only at the number of terms that need to be summed.

examples:

$$\sum_{v\in\{0,1\}} f(ABV) = \sum_{v\in\{0,1\}} P(A|V)P(V|B)$$

for every possible value of A and B we have to sum once, so 4

$$\sum_{v\in\{0,1\}} P(A,C|V)P(V|B)$$

for every possible value of A and B and C we have to sum once, so 8

$$\sum_{v\in\{0,1\}} P(A|V)P(V|B,C)$$

all the same, whether a variable appears on the left, or on the right, or in multiple terms

$$\sum_{v\in\{0,1\}} P(A|V,C)P(V|B,C)$$

If a variable appears in multiple terms, then we compute a product only of terms which have same value for that variable! We never multiply $P(A|C=1)$ with $P(A|C=0)$

as a consequence, the **computational cost** is:

2^N, N is the number of variables in the factor which are not evidence and which are not summed over

$$\sum_{v\in\{0,1\}} f(set) = f(set \setminus \{v\}) \to \mathcal{O}(2^{|set|-1}) = \mathcal{O}(2^{|set|})$$

Conclusion: the cost of summing a factor $f(ABCDE)$ is related to the number of free variables ($ABCDE$) in it! Same holds for multiplication of factors

If we have discreta variables, and d values in every variables domain, then it is of the order $d^{|set|}$.

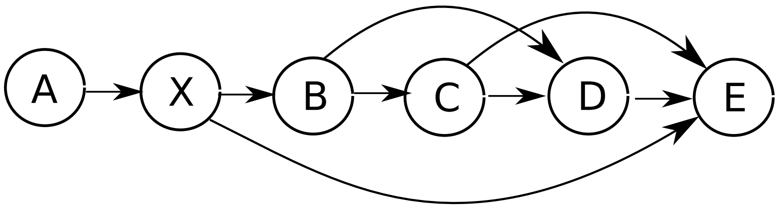
Takeaway

Consequence: choose a variable order, so that all factors encountered on the way have as small number of variables as possible!

Now we can explain, why computing the joint at first, like in
One more trick:

Can remove at the start "empty heads", that is $P(H_i|parents(H_i))$ such that H_i appears in no term on the right hand side as conditioning argument. When you remove such empty head, recheck: another variable in $parents(H_i)$ could now be an empty head.

Example: You want $P(B|A=0, C=1)$ in:



Initally you cannot remove $P(D|\cdot)$ because it is needed in $P(E|\cdot)$. However, once you remove $P(E|\cdot)$ due to $\sum_e P(E=e|\cdot)=1$, in the next step $P(D|\cdot)$ becomes removable, too. So you can start to do variable elimination with the remaining network $A \to X \to B \to C$

Side Question:

What joint probability does that one define?

Limits and Strengths of Variable Elimination

Variable Elimination can be hard: can encode 3-SAT formulas[5] in a BN

5 https://en.wikipedia.org/wiki/Boolean_satisfiability_problem#3-satisfiability

$$(x1 \vee \neg x2 \vee \neg x3) \wedge (\neg x1 \vee x4 \vee x5) \dots$$
$$P(X_i = 0) = 0.5$$
$$P(Y_{123}|x1, x2, x3) = \begin{cases} 1 & \text{if } (x1 \vee \neg x2 \vee \neg x3) \text{ eval to TRUE} \\ 0 & \text{otherwise} \end{cases}$$
$$A \wedge B \to P(Y_{AB}|A, B) = 1 \text{ if } A = 1, B = 1$$

If we can answer $P(output\ variable|X) = 1$, then we can answer if the 3-SAT has a solution - which is NP-Hard. So there is not general fast solution, but one can experience factor blowup to huge factors.

Variable Elimination is efficient in polytrees:

BN is a polytree if between two nodes there exists only one path connecting them. The diamond is not a polytree!

Algorithm:

- consider the undirected graph G created from BN, pick one node as root in the undirected G. Enumerate nodes in BN in the inverse topological order coming from the undirected G.
- eliminate nodes in BN in the inverse topological order obtained from the undirected graph above
- factor size during elimination never increases over the size of the largest $p(X_i|parent(X_i))$ because there is between two nodes always only one path connecting them! Optimally one picks as a root the largest factor, so that one treats it as the last.
- Linear in size of representation.

Further reading:

Belief Propagation! - Calculates the marginal distribution for each non-evidence node in parallel, conditioned on all evidence nodes.
`https://en.wikipedia.org/wiki/Belief_propagation`