Approximate inference (Ch. 14)

Bayesian Network: Efficiency

Last time we talked about how exact inference is fine if you have a polytree

Otherwise, exact inference is exponential $O(2^n)$ and not really feasible

Instead we use an approximate approach, specifically we will look at Monte Carlo approaches that utilize sampling
(this let's use balance runtime with accuracy)

Sampling

Sampling can mean different things:

(1) Sample an unknown distribution

- Much like running an experiment

Tickle friend's nose while asleep

... see how many times they react

(2) Sample from a known distribution

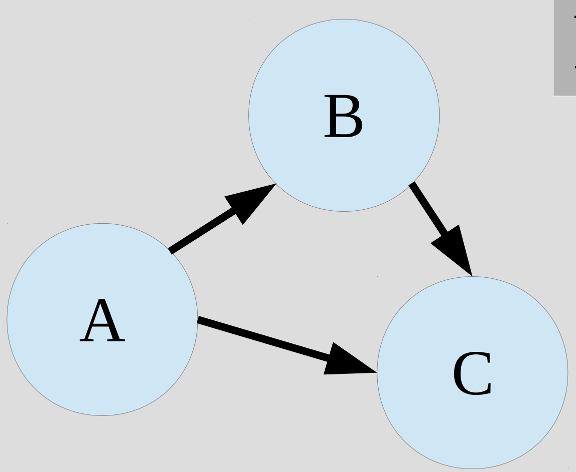
- Might also call this "simulation"
- Generate a random number to decide outcome of an event

we will use this way

Direct Sampling

The first method is called direct sampling, which is basically just running a simulation and tallying the results

Today we will use this simple Bay-net(work):



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

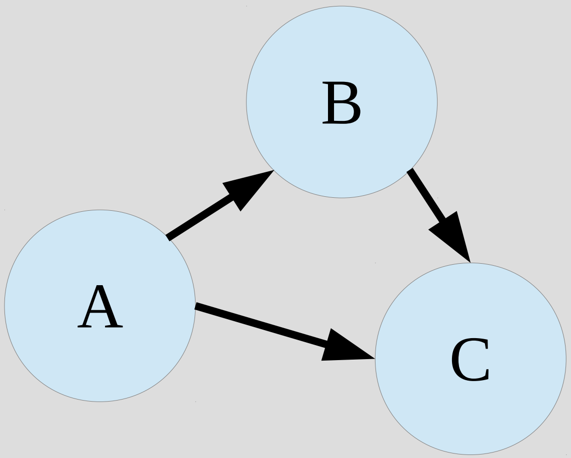
P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

Direct Sampling

Direct Sampling algorithm:

- Loop this a lot (N times)
  - Repeat until all nodes have values:
    - (1) Find any node with all parents having been given a value already value
    - (2) Generate a random number (0 to 1)
    - (3) Assign value to node based off of P(node | Parents(node))
- Calculate statistics

Direct Sampling



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

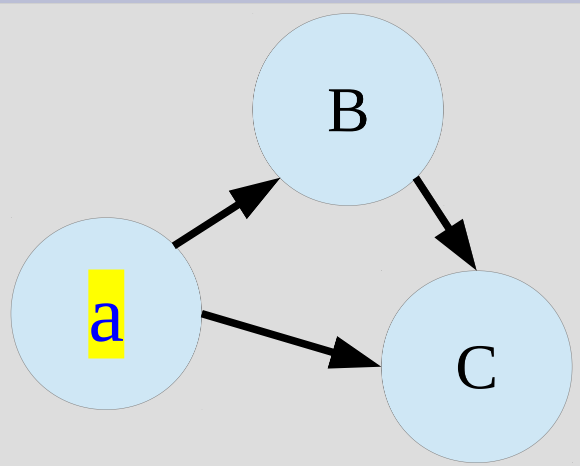
P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

(1) Only node who has all parents with values is node A (as it has no parents)

(2) Pretend random value is: 0.183712

(3) Since $0.183712 \leq 0.2$, set node A to a (true)

Direct Sampling



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

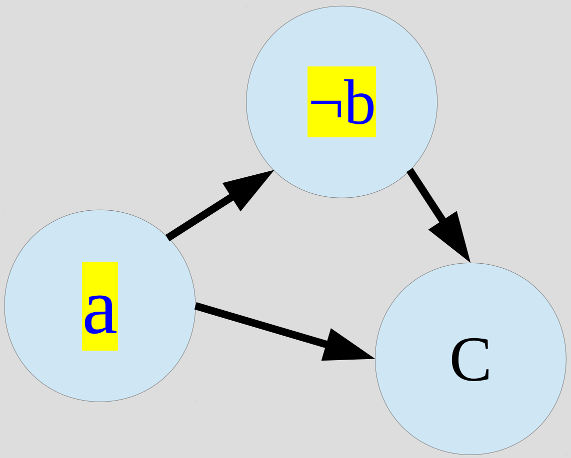
(1) Only node who has all parents with values is node B (as only A has a value)

(2) Pretend random value is: 0.910184

P(b|a), as A is a (i.e. a=true)

(3) Since 0.910184 > 0.4, set node B to be ¬b

Direct Sampling



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

(1) Only node left is C (has both parents)

(2) Pretend random value is: 0.634523

(3) Since $0.634523 \leq 0.7$, set node C to c

Direct Sampling

After running the inner loop once, we have a sample of (in format [A,B,C]):

[a, ¬b, c]

... we would then repeat this process N times (outer loop) to get a bunch of these

Pretend you got the results on the next slide

Direct Sampling

1. [a, ¬b, c]
2. [a, b, c]
3. [¬a, b, c]
4. [¬a, ¬b, ¬c]
5. [¬a, ¬b, ¬c]
6. [¬a, ¬b, ¬c]
7. [¬a, ¬b, ¬c]
8. [¬a, ¬b, ¬c]
9. [¬a, ¬b, ¬c]
10. [¬a, ¬b, ¬c]

From here we can calculate statistics of anything...

For example:

$$P(\neg c) = \frac{\text{count of "}\neg c\text{"}}{\text{total possible times}} = \frac{7}{10}$$

Direct Sampling

1. [a, ¬b, c]
2. [a, b, c]
3. [¬a, b, c]
4. [¬a, ¬b, ¬c]
5. [¬a, ¬b, ¬c]
6. [¬a, ¬b, ¬c]
7. [¬a, ¬b, ¬c]
8. [¬a, ¬b, ¬c]
9. [¬a, ¬b, ¬c]
10. [¬a, ¬b, ¬c]

In fact, you can estimate P(a,b,c) from this:

$$P(a, b, c) = 0.1$$
$$P(a, b, \neg c) = 0$$
$$P(a, \neg b, c) = 0.1$$
$$P(a, \neg b, \neg c) = 0$$
$$P(\neg a, b, c) = 0.1$$
$$P(\neg a, b, \neg c) = 0$$
$$P(\neg a, \neg b, c) = 0$$
$$P(\neg a, \neg b, \neg c) = 0.7$$

Rejection Sampling

1. [a, ¬b, c]
2. [a, b, c]
3. [¬a, b, c]
4. [¬a, ¬b, ¬c]
5. [¬a, ¬b, ¬c]
6. [¬a, ¬b, ¬c]
7. [¬a, ¬b, ¬c]
8. [¬a, ¬b, ¬c]
9. [¬a, ¬b, ¬c]
10. [¬a, ¬b, ¬c]

How would you compute:

$$P(a|b)$$

Rejection Sampling

1. [a, ¬b, c]
2. [a, b, c]
3. [¬a, b, c]
4. [¬a, ¬b, ¬c]
5. [¬a, ¬b, ¬c]
6. [¬a, ¬b, ¬c]
7. [¬a, ¬b, ¬c]
8. [¬a, ¬b, ¬c]
9. [¬a, ¬b, ¬c]
10. [¬a, ¬b, ¬c]

How would you compute:

$$P(a|b)$$

You do the same counting, but only look at entries with "b" being true

... thus P(a|b) = 0.5

Rejection Sampling

This technique is called <u>rejection sampling</u>, as you reject/ignore any samples that do not have the given conditional information

For direct sampling, with N samples:

$$P(a,b,c) = P(a)P(b|a)P(c|b,a) = \lim_{N \to \infty} \frac{count(a,b,c)}{N}$$

... or more generally...

$$P(x_1, \cdots x_n) = \prod_{i=1}^{n} P(x_i|Parents(X_i)) = \lim_{N \to \infty} \frac{count(x_1, \cdots x_n)}{N}$$

Let us call the right hand side: $N_{PS}(a,b,c)$

Rejection Sampling

From here it is fairly easy to prove that that rejection sampling is also finding the correct probability (assuming many samples):

$$P(a|b) = N_{PS}(a,b)/N_{PS}(b) = \lim_{N\to\infty} \frac{(\frac{count(a,b)}{N})}{(\frac{count(b)}{N})} = \frac{count(a,b)}{count(b)}$$

... or let "**x**" be what we want to find and "**e**" be the given info (here "**e**" = {b}, but both "**x**" and "**e**" could be multiple variables, like "**e**" = {b,c})

$$P(x|e) = \lim_{N\to\infty} \frac{(\frac{count(x,e)}{N})}{(\frac{count(e)}{N})} = \frac{count(x,e)}{count(e)}$$

Rejection Sampling

As number of samples, N, grows our accuracy of approximating probabilities gets better

Using the Law of Large Numbers, we can find that the standard deviation for our estimates is: $\frac{\sigma}{\sqrt{N}} \approx \frac{1}{\sqrt{N}}$

as we using probabilities, the mean & std dev in [0,1] (small)

in rejection sampling,
N = number non-rejected samples

So when we found P(a|b) = 0.5 (with 2 samples), we are 68.2% confident that P(a|b) is within: $[0.5 - \frac{0.5}{\sqrt{2}}, 0.5 + \frac{0.5}{\sqrt{2}}] = [0.15, 0.85]$

:(

Good Sampling?

What is the general issue(s) with direct and/or rejection sampling? (When is it good?)

Good Sampling?

What is the general issue(s) with direct and/or rejection sampling? (When is it good?)

These sampling techniques are pretty good for finding non-conditional probability: P(a,b,c)

However, if the given information is restrictive many samples will be rejected... leading to poor approximations of the probabilities

Good Sampling?

The given information(also called “evidence”) can be restrictive because:

(1) the tables have low probabilities

(2) many variables have to be satisfied

You will need exponentially more samples as you increase number of given variables

If $P(y) = P(z) = 0.5$, this table shows number of samples for same accuracy

	N
P(x)	100
P(x|y)	200
P(x|y,z)	400

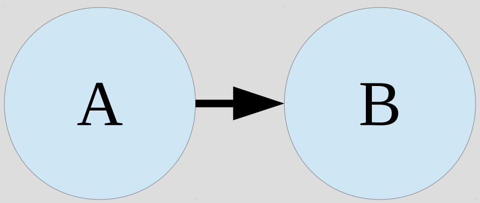
Likelihood Weighting

There a way to not waste time generating "rejected" samples called likelihood weighting

As mentioned before, direct sampling is decent at finding non-conditional probabilities

So for likelihood weighting we will assume we want to find a conditional probability

Likelihood Weighting



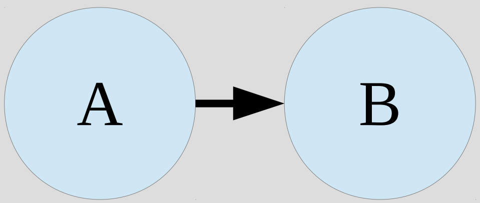
P(a)	0.5

P(b|a)	1
P(b|¬a)	0.2

Likelihood weighting, will weight samples:
For P(a|b): [a,b] w = 1, [¬a,b] w=0.2

If we did rejection sampling, we need about 5 ¬a to actually get a ‘b’, so in 10 samples:
[a,b], [a,b], [a,b], [a,b], [a,b],
[¬a,b], [¬a,¬b], [¬a,¬b], [¬a,¬b], [¬a,¬b]

Likelihood Weighting



P(a)	0.5

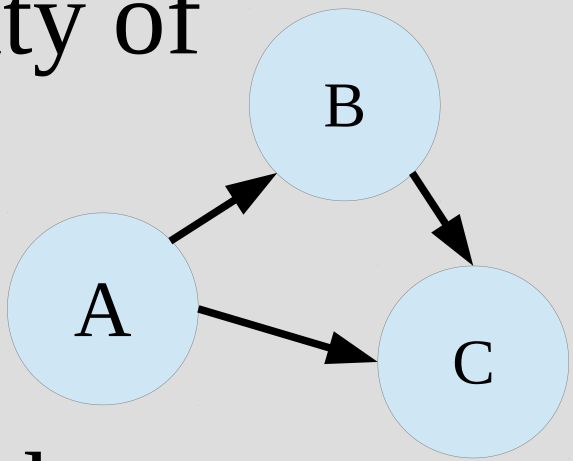
P(b|a)	1
P(b|¬a)	0.2

Since we normalize, all we care about is the ratio between [a,b] and [¬a,b]

In likelihood weighting, the weights create the correct ratio as “[¬a,b] : w=0.2” represents that you would actually need 5 of these to get a “true” sample

Likelihood Weighting

We will use a bit of notation here:
x = things we want the probability of
e = "evidence" or given info
y = anything else



So in our original sample network:
P(a|b) : x={a}, e={b}, y={c}
P(a|b,c) : x={a}, e={b,c}, y={}
P(a,b|c) : x={a,b}, e={c}, y={}

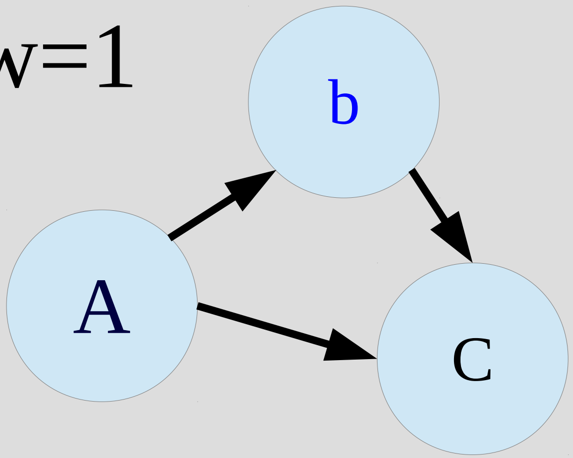
must be non-empty   assume non-empty for this alg

Likelihood Weighting

Likelihood weighting algorithm:
-Assign all given variables into network
-w = 1 // our "weight"
-Do once for every node:
  (1) Find a node where all parents have values
  (2a) If node given info (in set "e"):
    w = w * P(given | Parents(given))
  (2b) Else (in sets "x" or "y")
    Generate random number to determine T/F
- Repeat above a lot and calculate statistics

Likelihood Weighting

w=1



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

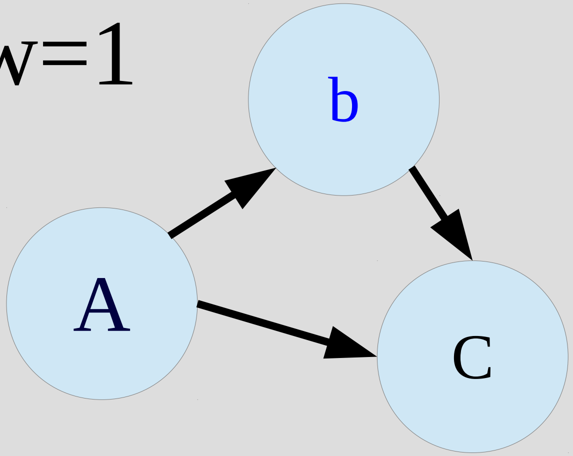
P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

Since we are finding P(a|b), we initially set b=true in the network (and start w=1)

From here we need to loop through all three nodes, finding any node that has all of its parents with values

Likelihood Weighting

w=1



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

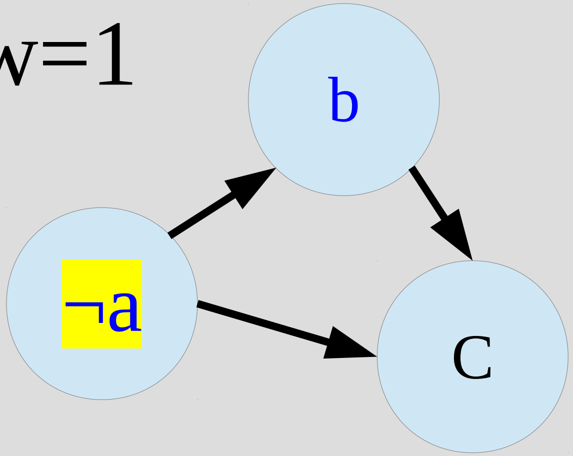
P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

(1) A is only one with all parents having values, so pick A to look at

(2a) A is not given information, so we generate a random number: 0.746949
0.746949 > 0.2, so we set A to ¬a

Likelihood Weighting

w=1



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

(1) Here we could pick ‘b’ or ‘C’ as ‘b’ has its parent and C has values for ‘a’ and ‘b’ ... let’s pick B

as A sampled to be ¬a this time

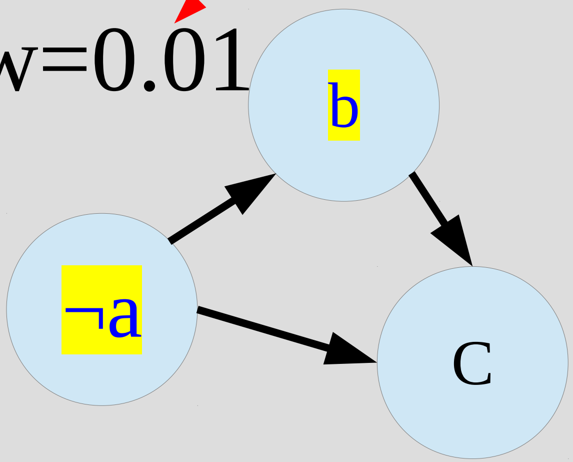
(2b) B is given information, so we simply multiply “w” by the probability P(b|¬a)

w = w*P(b|¬a) = 1*0.01 = <u>0.01</u>

Likelihood Weighting

if multiple given variables, w = product of all (multiple times)

w=0.01



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

(1) C is only node left... pick that

(2a) C is not given information, so generate random number to sample/simulate: 0.987924 > 0.3, so set C to ¬c

P(¬a,b)

Likelihood Weighting

Now we have a single sample:
[¬a, ¬c] : w=0.01

We would then repeat this process, say N times (make sure to reset w=1 every time)

Afterwards we would have a bunch of weighted samples where b=true always
... pretend they turned out as the next slide

Likelihood Weighting

tells us P(a,c|b)

1. [a, c] : w=0.4
2. [a, c] : w=0.4
3. [¬a, c] : w=0.01
4. [¬a, c] : w=0.01
5. [¬a, ¬c] : w=0.01
6. [¬a, ¬c] : w=0.01
7. [¬a, ¬c] : w=0.01
8. [¬a, ¬c] : w=0.01
9. [¬a, ¬c] : w=0.01
10. [¬a, ¬c] : w=0.01

Rather than doing a direct tally, we sum the weights... so:

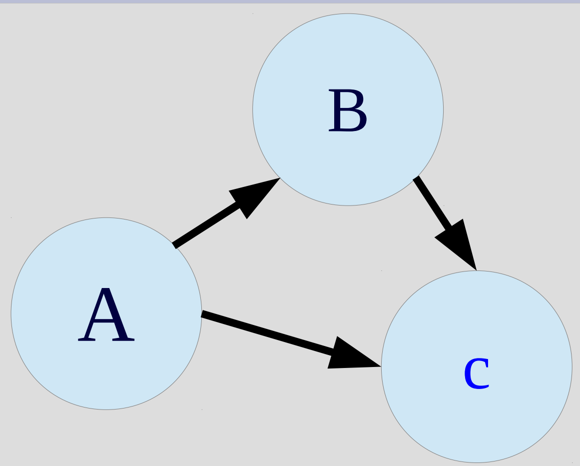
$$P(a|b) = \frac{0.4+0.4}{\underbrace{0.4+0.4+0.01+0.01+0.01+0.01+0.01+0.01+0.01+0.01}_{\text{all 10 of them}}}$$

P(a|b) = 0.8/0.88
= 0.909

This is also just our normalization trick...

$$P(a|b) = \alpha 0.8, P(\neg a|b) = \alpha 0.08$$

Likelihood Weighting



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

You try it! Calculate P(a|c) using this alg. and using these random numbers (20 of them):

0.784	0.859	0.934	0.760	0.543
0.532	0.967	0.229	0.781	0.002
0.168	0.439	0.873	0.415	0.471
0.053	0.646	0.694	0.325	0.368

use left to right, top to bottom

Likelihood Weighting

1. [¬a, ¬b] : w=0
2. [¬a, ¬b] : w=0
3. [¬a, ¬b] : w=0
4. [¬a, ¬b] : w=0
5. [¬a, b] : w=0.3
6. [a, ¬b] : w=0.7
7. [¬a, ¬b] : w=0
8. [¬a, ¬b] : w=0
9. [¬a, ¬b] : w=0
10. [¬a, ¬b] : w=0

You should get these samples from the random simulation

Thus:

$P(a|c) = \alpha\ 0.7$

$P(\neg a|c) = \alpha\ 0.3$

So, $P(a|c) = 0.7$

Likelihood Weighting

Any issues with this?

Likelihood Weighting

Any issues with this?

When w=0, this is basically like rejection sampling...

This happens because you do not consider the children when generating samples

In our example, A=true dominated the total weight(0.8 of 0.88), leading to accuracy issues

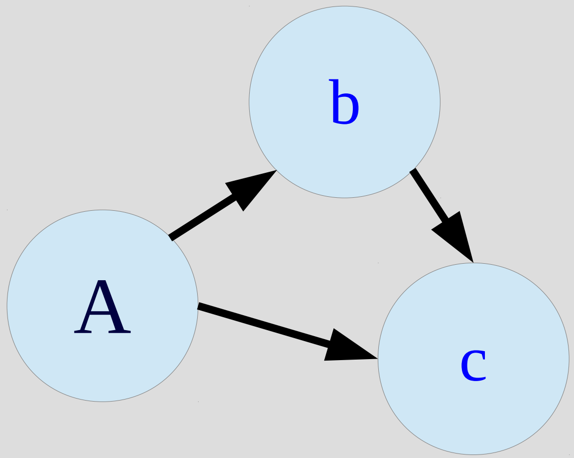
Likelihood Weighting

Why does this weight trick work? In our prob:

$$
\begin{aligned}
P(a|c) &= \alpha TrueA(wFor(inTable)) \\
&= \alpha P(a = trueInTable) \cdot wFor(inTable) \\
&= \alpha \sum_b P(a,b) \cdot wFor(inTable) \\
&= \alpha \sum_b P(a)P(b|a) \cdot wFor(inTable) \\
&= \alpha \sum_b \underbrace{P(a)P(b|a)}_{\text{percent in table table}} \cdot \underbrace{P(c|a,b)}_{\text{weight of sample}} \\
&= \alpha \sum_b P(a,b,c) \\
&= \alpha P(a,c)
\end{aligned}
$$

normalize trick:
P(a|c) = αP(a,c)

Likelihood Weighting



P(a)	0.2

P(b|a)	0.4
P(b|¬a)	0.01

P(c|a,b)	1
P(c|a,¬b)	0.7
P(c|¬a,b)	0.3
P(c|¬a,¬b)	0

I mentioned this in the algorithm, but did not do an example: weight's product is cumulative

So if we want to find P(a|b,c), say 3 samples:
[a] : w = 0.4, [a] w= 0.4*1 = 0.4
[¬a] : w = 0.01*0.3 = 0.003