Approximate inference (Ch. 14)
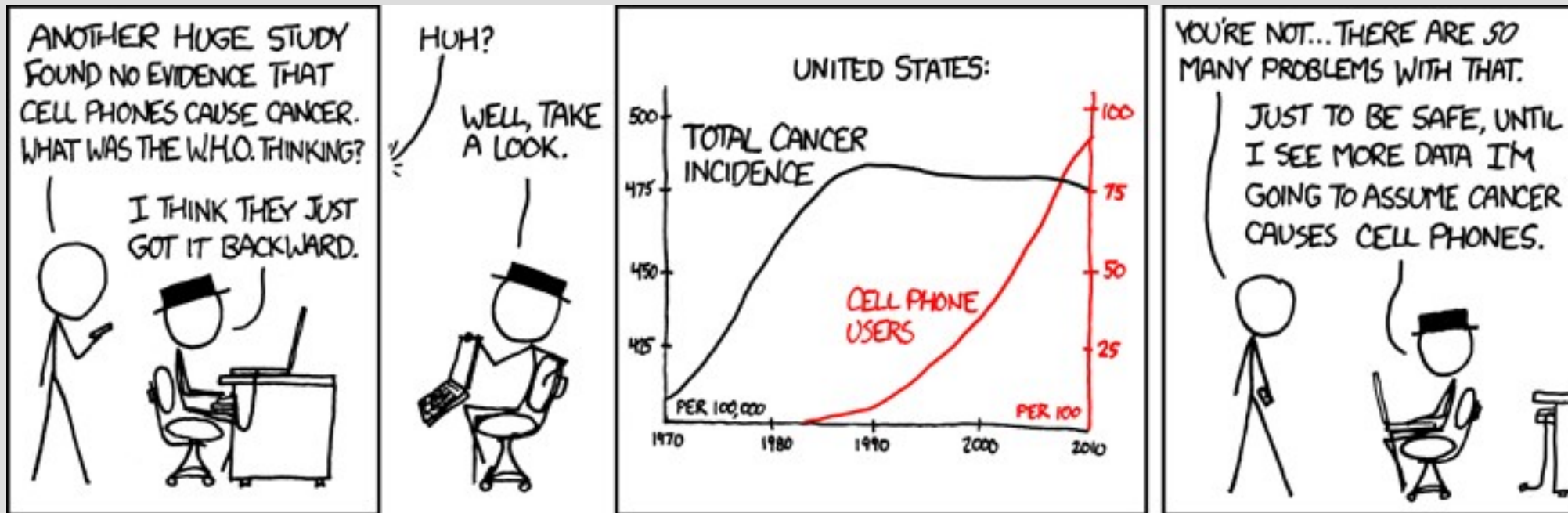
Likelihood Weighting
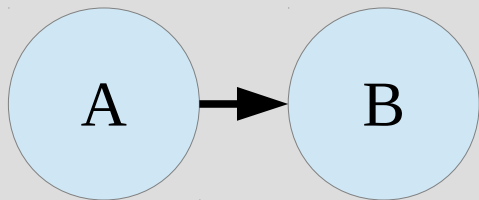
P(a)	0.5

P(b|a)	1
P(b|¬a)	0.2

In LW, say we generated 2 samples:
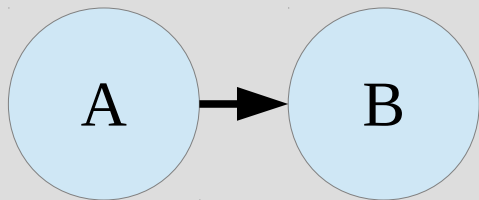[a] : w = 1, [¬a], w=0.2

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

Likelihood Weighting

A → B

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
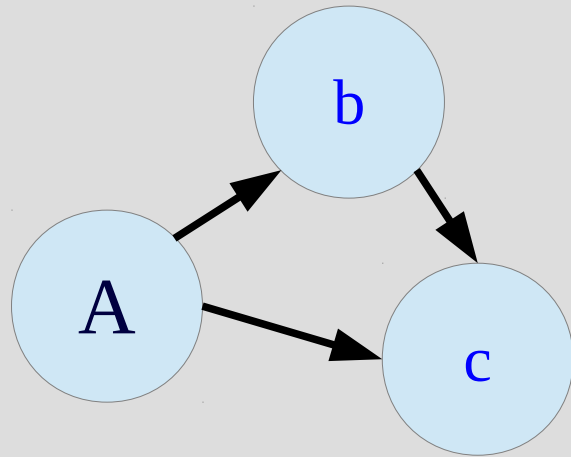
P(a)	0.2

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

Markov Chain

Today we will take a slightly different approach called <u>Gibbs sampling</u>

In likelihood weighting: if we wanted P(a,b|c), we would generate both 'a' and 'b' in loop

For Gibbs sampling: when finding P(a,b|c), we will only change 'a' or 'b' individually (rather than both at the same time)

Markov Chain

Gibbs sampling uses a <u>Markov chain</u> (since we use random numbers to generate samples, we call it Monte-Carlo Markov chain)

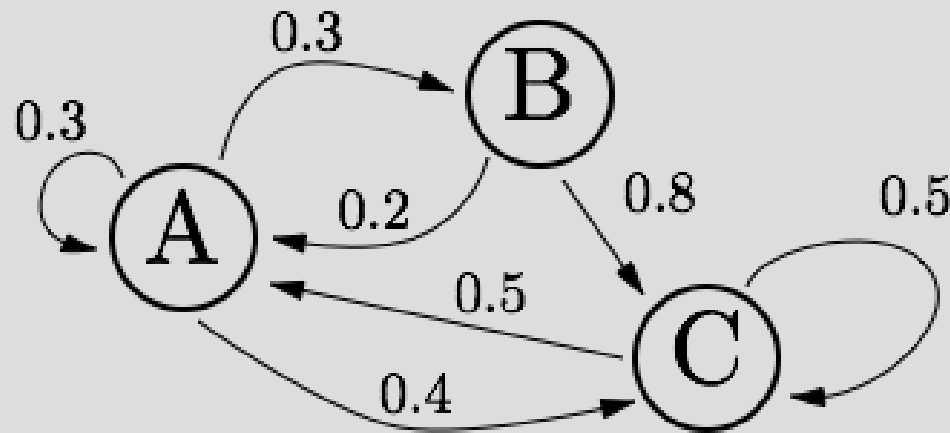A Markov chain can be thought of as a transition between states:
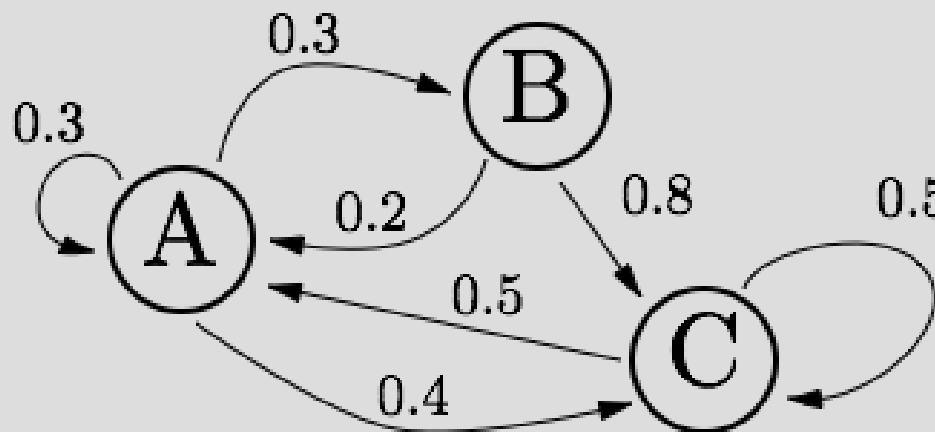
This transition says if you are in 'C' you have a 50% chance to stay in 'C' next time

Markov Chain

More generally, anything that is "memoryless" is a type of Markov chain:

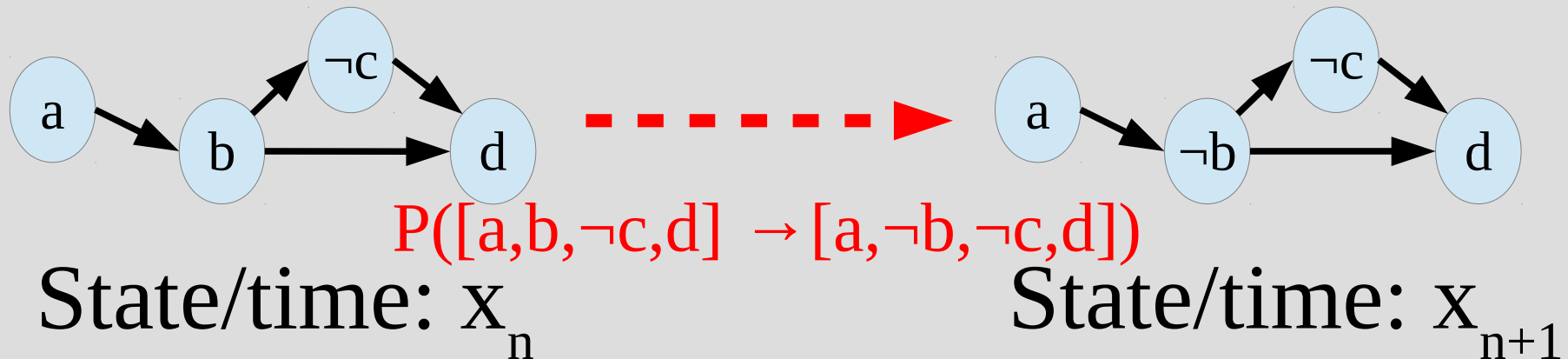$$P(x_{n+1}|x_n, x_{n-1}, x_{n-2} \cdots x_0) = P(x_{n+1}|x_n)$$

This property is simply: "Where you end up next only depends on where you currently are"

This is P(C→C)=0.5

Is Markov because only uses current state (C) not more previous states (like (B,C))

0.3

0.3

0.2

0.8

0.5

0.5

0.4

B

A

C

Markov Chain

We are going to change one value in the Bay net at a time to make a Markov chain:

$$P([a,b,\neg c,d] \rightarrow [a,\neg b,\neg c,d])$$

State/time: x_n State/time: x_{n+1}

After making a long Markov chain by having one variable change per step, we will average the states to find the probability we want

Gibbs sampling

Gibbs sampling algorithm:
- Set evidence variables (i.e. b=true if P(a|b))
- Randomly initialize everything else
- Loop a lot:
  (1) Pick a random non-evidence variable
  (2) Generate random number to determine
       if T/F (based on Markov blanket)
   - Record tally/count of resulting state
- Calculate statistics

Gibbs sampling
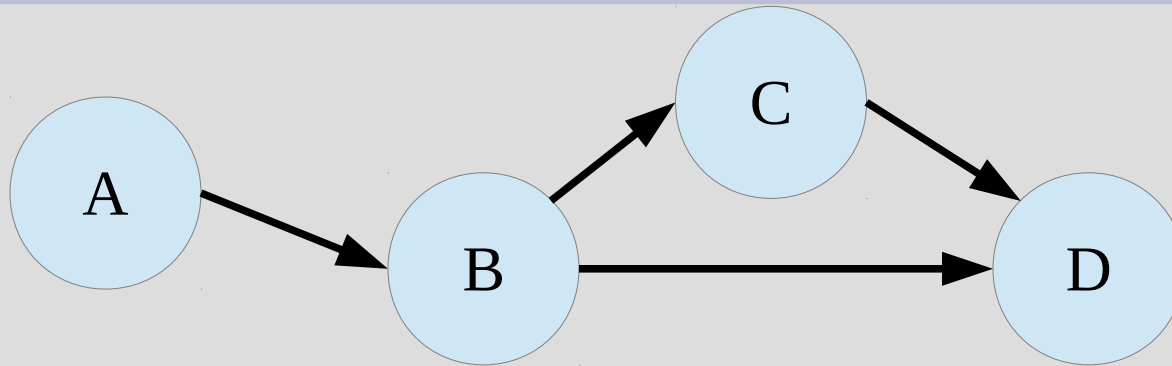
P(a)	0.1

P(b|a)	0.2
P(b|¬a)	0.3

P(c|b)	0.4
P(c|¬b)	0.5

P(d|b,c)	0.25
P(d|b,¬c)	1.0
P(d|¬b,c)	0.15
P(d|¬b,¬c)	0.05

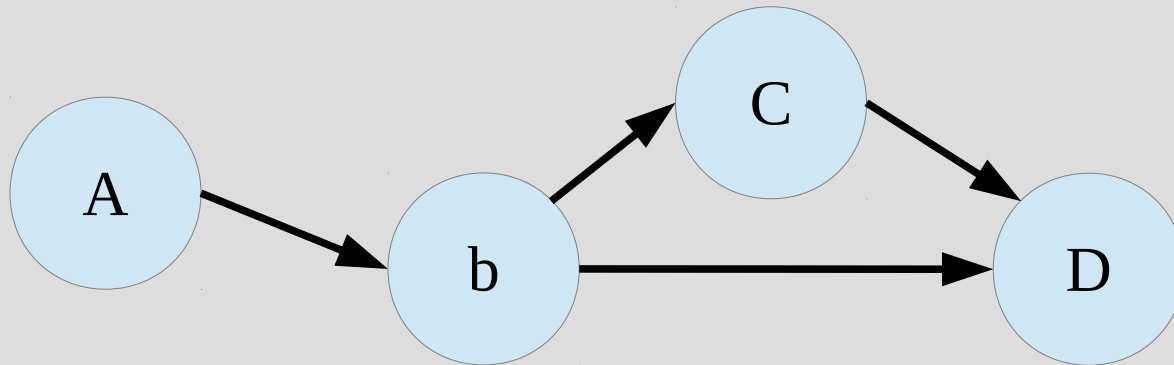Let's use the Bayesian network above to find:

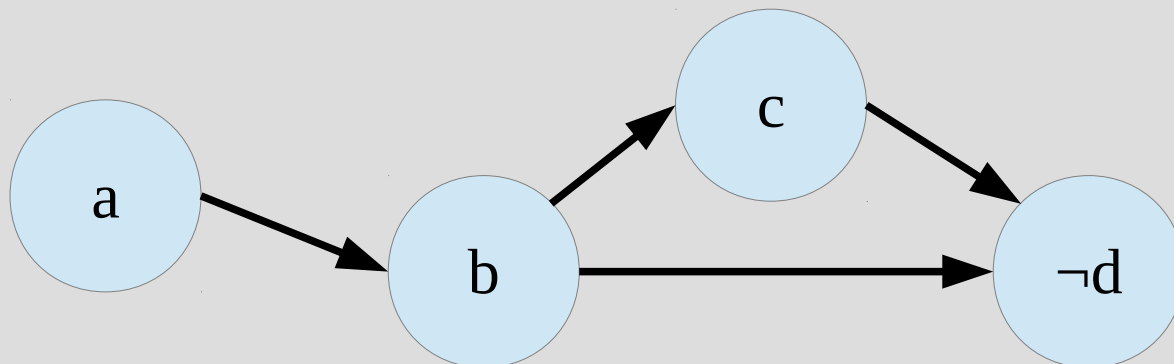$P(A,C,D|b)$ A   D   A   C   C

Using rand: 0.225, 0.108, 0.628, 0.781, 0.117

Gibbs sampling

Have to set evidence (b=true), but then randomly set a, c and d to [true, true, false]
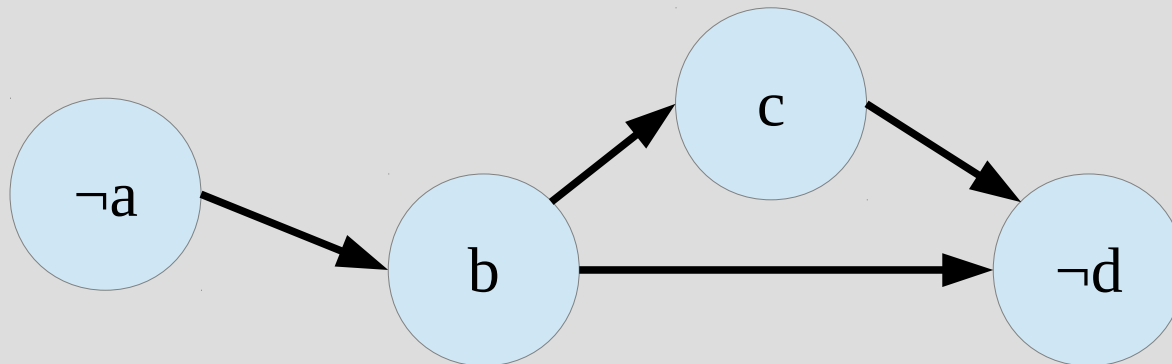
Gibbs sampling

(1) Pick a random non-evidence variable
    (i.e. anything other than 'b')
... let's randomly pick A

(2) Randomly change A based off Markov Blanket:

$$P(a|b, c, \neg d) = P(a|b) = \frac{P(b|a)P(a)}{P(b|a)P(a)+P(b|\neg a)P(\neg a)} = \frac{0.2(0.1)}{0.2(0.1)+0.3(0.9)} = 0.069$$

Rand = 0.225
set a=false as
0.225 > 0.069

Gibbs sampling

[¬a,c,¬d]

(1) Pick a random non-evidence variable
    (i.e. anything other than 'b')
... let's randomly pick A

Keep tally

(2) Randomly change A based off Markov Blanket:

$$P(a|b, c, \neg d) = P(a|b) = \frac{P(b|a)P(a)}{P(b|a)P(a)+P(b|\neg a)P(\neg a)} = \frac{0.2(0.1)}{0.2(0.1)+0.3(0.9)} = 0.069$$
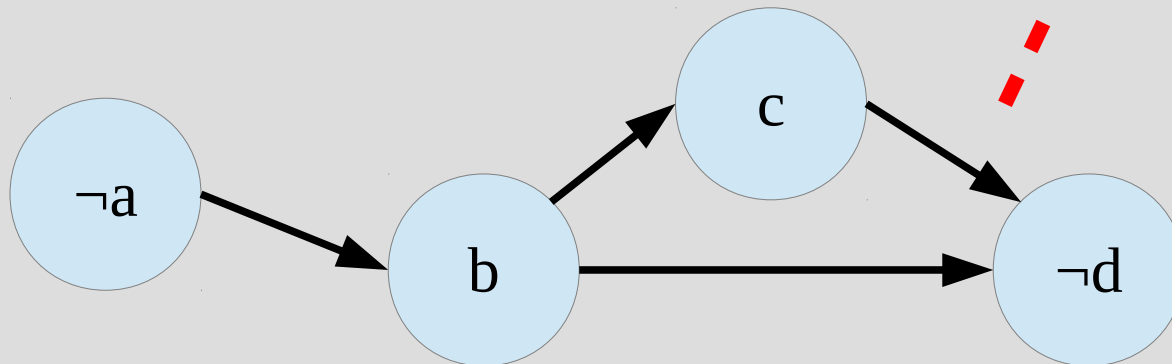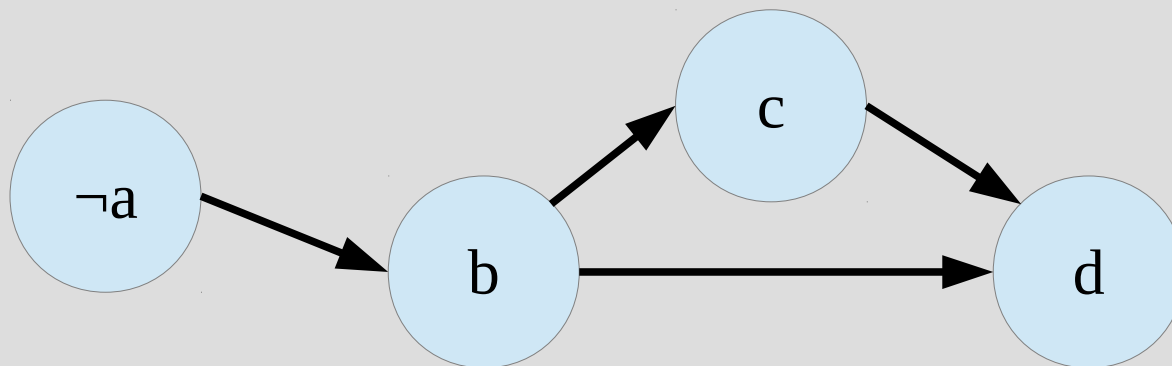
Rand = 0.225
set a=false as
0.225 > 0.069

Gibbs sampling

(1) Randomly pick D (from A, B, D)

(2) Randomly change D based off Markov Blanket:

$P(d|\neg a, b, c) = P(d|b, c) = 0.25$

Rand = 0.108
set d=true as
$0.108 \leq 0.25$

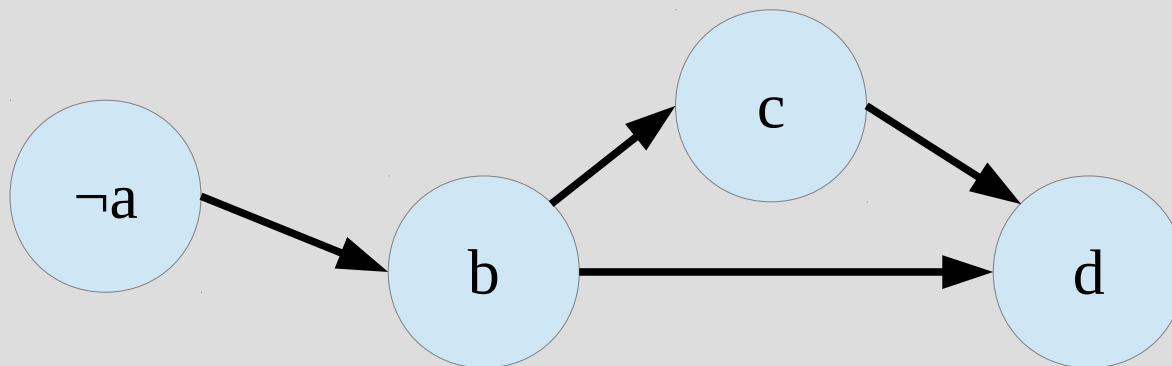# Gibbs sampling

[¬a,c,¬d]
[¬a,c,d]
[¬a,c,d]

(1) Randomly pick A (from A, B, D)

(2) Randomly change A based off Markov Blanket:

$$P(a|b,c,d) = P(a|b) = \frac{P(b|a)P(a)}{P(b|a)P(a)+P(b|\neg a)P(\neg a)} = \frac{0.2(0.1)}{0.2(0.1)+0.3(0.9)} = 0.069$$

Rand = 0.628
set a=false as
$0.628 \leq 0.069$

Gibbs sampling

[¬a,c,¬d]
[¬a,c,d]
[¬a,c,d]
[¬a,¬c,d]

(1) Randomly pick C (from A, B, D)

$$P(x|MarkovBlanket(x)) = \alpha P(x|Parents(x)) \prod_{y \in Children(x)} P(y|Parents(y))$$

(2) Randomly change C based off Markov
Blanket: $P(c|\neg a, b, d) = P(c|b,d) = \frac{P(b,c,d)}{P(b,d)} = \frac{P(d|b,c)P(b,c)}{P(b,d)}$
$= \frac{P(d|b,c)P(c|b)P(b)}{P(b,d)} = \alpha P(d|b,c)P(c|b)$

<P(c), P(¬c)> = <α 0.25(0.4), α 1(0.6)>
=<0.143, 0.857>

Rand = 0.781
set c=false as
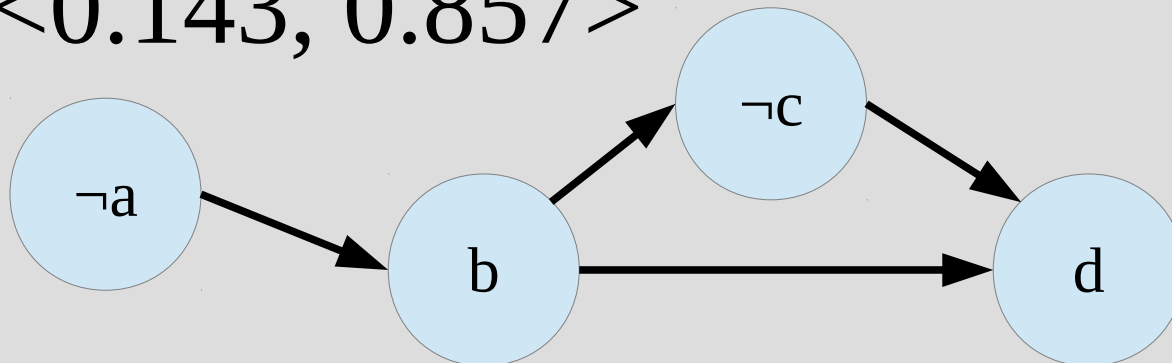0.781 > 0.143

Gibbs sampling

[¬a,c,¬d]
[¬a,c,d]
[¬a,c,d]
[¬a,¬c,d]
[¬a,c,d]

(1) Randomly pick C (from A, B, D)

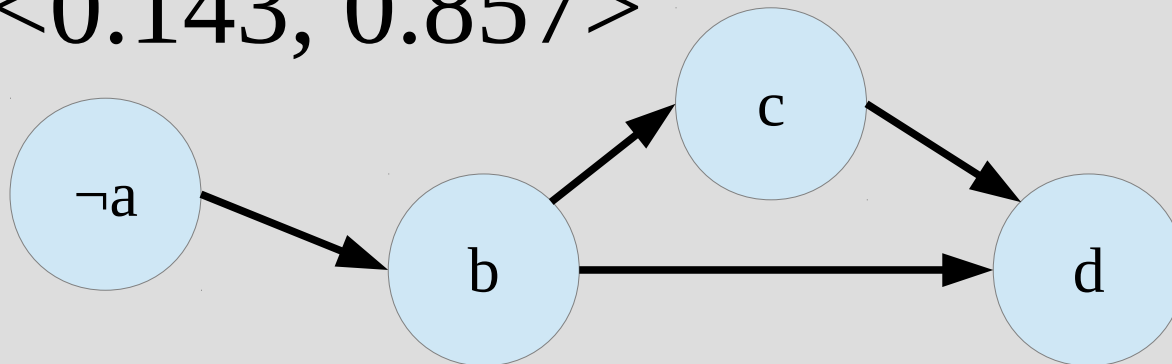$$P(x|MarkovBlanket(x)) = \alpha P(x|Parents(x)) \prod_{y \in Children(x)} P(y|Parents(y))$$

(2) Randomly change C based off Markov

Blanket: $P(c|\neg a, b, d) = P(c|b, d) = \frac{P(b,c,d)}{P(b,d)} = \frac{P(d|b,c)P(b,c)}{P(b,d)}$

$= \frac{P(d|b,c)P(c|b)P(b)}{P(b,d)} = \alpha P(d|b,c)P(c|b)$

$<P(c), P(\neg c)> = <\alpha\ 0.25(0.4),\ \alpha\ 1(0.6)>$
$=<0.143, 0.857>$

Rand = 0.117
set c=true as
$0.117 \leq 0.143$

Gibbs sampling

Now we have our five samples... ↗

We would just compute P(a,c,d|b) as: count(a,c,d)/totalSamples, so:

$$P(\ a,\ c,\ d|b) = 0$$

$$P(\ a,\ c,\neg d|b) = 0$$

$$P(\ a,\neg c,\ d|b) = 0$$

$$P(\ a,\neg c,\neg d|b) = 0$$

$$P(\neg a,\ c,\ d|b) = 0.6$$

$$P(\neg a,\ c,\neg d|b) = 0.2$$

$$P(\neg a,\neg c,\ d|b) = 0.2$$

$$P(\neg a,\neg c,\neg d|b) = 0$$
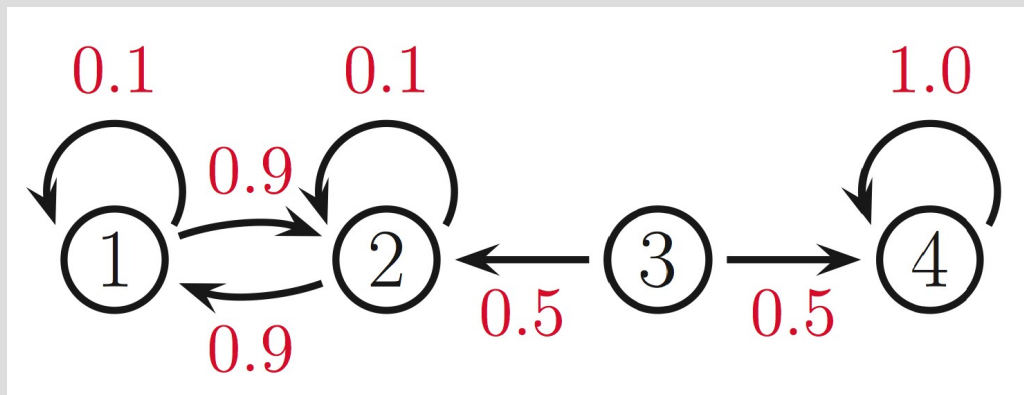
Obviously we should loop more than 5 times, but this should converge as long as the Markov chain doesn't have two properties...

Gibbs sampling

For Gibbs sampling to work we need:

(1) <u>Irreducibility</u>: Every state reachable from any other state in a finite number of steps

The above is not irreducible as if we start in state 3 and go to state 4, we cannot ever leave

Gibbs sampling

For Gibbs sampling to work we need:

(2) <u>Aperiodically</u>: Cannot have a "periodic" movement (always transition)
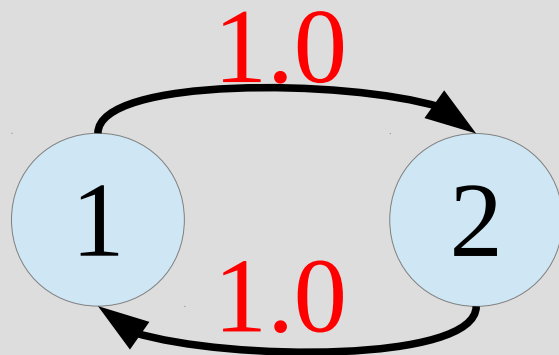
Formally:

at state i

$$\exists t \; \forall i, t' \geq t: \; P(x_{t'} = i | x_0 = i) > 0$$

time

In the above Markov chain we will spend half the time in state 1, it will always leave in the next step

Gibbs sampling
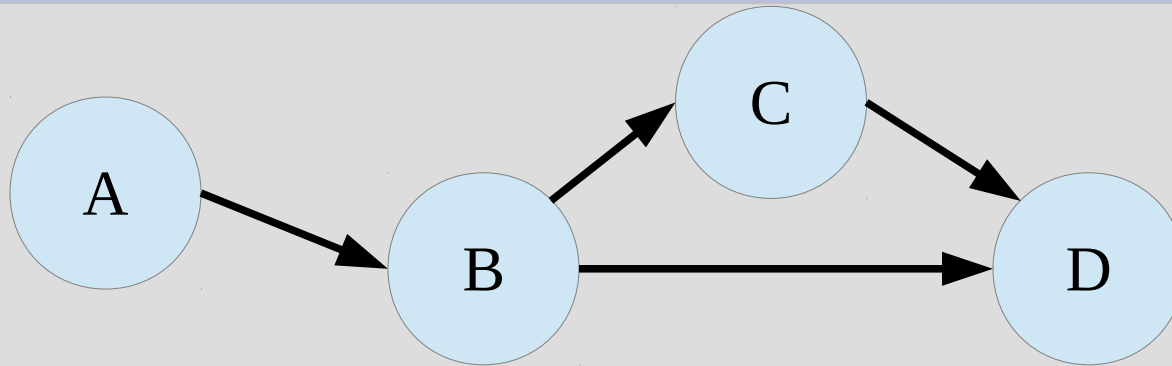
P(a)	0.1

P(b|a)	0.2
P(b|¬a)	0.3

P(c|b)	0.4
P(c|¬b)	0.5

P(d|b,c)	0.25
P(d|b,¬c)	1.0
P(d|¬b,c)	0.15
P(d|¬b,¬c)	0.05

You try! Find: $P(B, C | a, \neg d)$ (initial=¬b,c)

Random node: B   C   B   C   C

Using rand: 0.081, 0.476, 0.134, 0.095, 0.875

Gibbs sampling

Always [a, ¬d]
1. Pick B, P(b|a,c,¬d)=0.15>0.081, [b,c]
2. Pick C, P(c|a,b,¬d)=0.370<0.476, [b,¬c]
3. Pick B, P(b|a,¬c,¬d)=0<0.134, [¬b,¬c]
4. Pick C, P(c|a,¬b,¬d)=0.472>0.095, [¬b,c]
5. Pick C, P(c|a,¬b,¬d)=0.472<0.875, [¬b,¬c]

So   P(b,c|a,¬d) = 0.2   P(b,¬c|a,¬d) = 0.2
     P(¬b,c|a,¬d) = 0.2   P(¬b,¬c|a,¬d) = 0.4
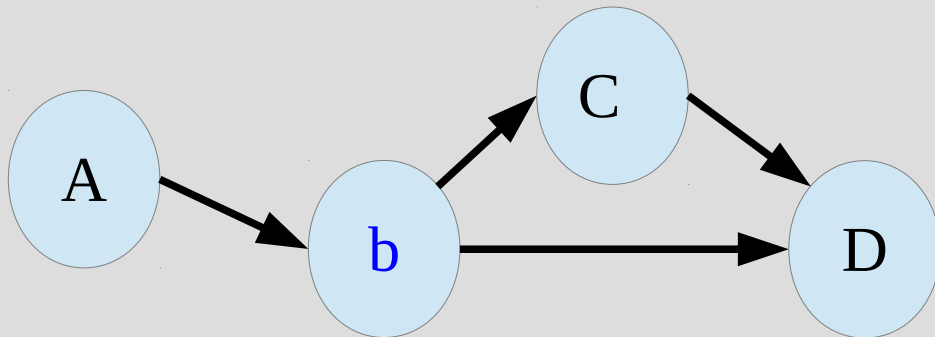
Why Gibbs works

Notation:

$\pi(x)$ = probability being in state x

e = "evidence", thus we finding P(x|e)

$\overline{x}$ = all non-evidence except x

line/bar over x

Example: Find P(a,c,d | b)

e = 'b' always

if x = {a}, $\overline{x}$ = {b,c}

if x = {b}, $\overline{x}$ = {a,c}

Why Gibbs works

To understand why Gibbs sampling works, we first need a bit more on Markov chains:

$$\pi_{t+1}(x') = \sum_x \pi_t(x) \cdot P(x \to x')$$

prob change states
(you just did this)
(e.g. $[\neg a, b, c] \to [a, b, c]$)

prob to get next
state (e.g. $[a,b,c]$)

prob in a state
(e.g. $[\neg a, b, c]$)

With the properties of irreducibility and aperiodicity, we will converge to a <u>stationary distribution</u> (i.e. stop changing)

$$\pi_{t+1}(x) = \pi_t(x)$$ (I will stop writing t's)

Why Gibbs works

Thus we get:

$$\pi(x') = \sum_x \pi(x) \cdot P(x \to x')$$

If you think about probabilities as "flows" then the flow into x' is the sum of partial (depending on P(x→x')) flow from all other x

But the flow from x' is also outgoing to other states... so the stationary distribution has equal "flow" on all of the probabilities

Why Gibbs works

One way way to satisfy in-flow=out-flow is to simply say you must have equal flow between pairs of nodes

$$\pi(x)P(x \to x') = \pi(x')P(x' \to x)$$

From here it is enough to show that if you set:
π(x) = P(a,c,d|b), where x = {a,c,d}
P(x→x') = P(x|MarkovBlanket(x))

... you will satisfy the stationary requirement

Why Gibbs works

In our P(a,c,d|b) example:

$$\pi(a, c, d)P([a, c, d] \rightarrow [\neg a, c, d])$$

$$= P(a, c, d|b)P(\neg a|b, c, d)$$

$$= P(a|b, c, d)P(c, d|b)P(\neg a|b, c, d)$$

$$= P(a|b, c, d)P(\neg a, c, d|b)$$

$$= P([\neg a, c, d] \rightarrow [a, c, d])\pi(\neg a, c, d)$$

Thus we have our required property:

$$\pi(x)P(x \rightarrow x') = \pi(x')P(x' \rightarrow x)$$

Why Gibbs works

In general:

$$\pi(x)P(x \to x')$$
$$= P(x|e)P(x_i'|\bar{x}_i, e)$$
$$= P(x_i, \bar{x}_i|e)P(x_i|\bar{x}_i, e)$$
$$= P(x_i|\bar{x}_i, e)P(\bar{x}_i|e)P(x_i'|\bar{x}_i, e)$$
$$= P(x_i|\bar{x}_i, e)P(x_i'|\bar{x}_i, e)$$
$$= P(x' \to x)\pi(x')$$

Note:
Technically, when finding P(x→x') we have all variables as given, but we only use the Markov blanket as the other variables are conditionally independent

Gibbs vs. Likelihood Weight

What are the differences (good and bad) between this method (Gibbs) and the one from last time (Likelihood Weighting)?

Gibbs vs. Likelihood Weight

Good:
- Will not ever generate a 0 weight sample (as uses all evidence: P(c|a,b,d) not just parents in LW: P(c|b))

Bad:
- Hard to tell when "converges" (no Law of Large Numbers to help bound error)
- Transition more unlikely if large blanket (as more probabilities multiplied = more variance)

Zzzzz...

The rest of the chapter both:
- Gives real-ish world examples to use algs.
- Shows other ways of solving that (in general) not as good as using Bayesian networks

This is kinda boring so I will skip all except the last part on "Fuzzy logic"

Fuzzy Logic

So far we have been saying things like:
A=true ... or ... OverAte=true

Fuzzy logic moves away from true/false and instead makes these continuous variables, so:
OverAte=0.4 is possible

This is <u>not</u> a 40% chance you overate,
it is more like your stomach is 40% full
(a known fact, not a thing of chance)

Fuzzy Logic

You can define basic logic operators in Fuzzy logic as well:
(A or B) = max(A,B)
(A and B) = min(A,B)
(¬A) = 1-(A)

... So if OverAte=0.4 and Desert=0.2
(OverAte or Desert) = 0.4

However, (Desert or ¬Desert)=0.8