Lecture 3: Efficient probabilistic reasoning

Frederik Mallmann-Trenn

Department of Informatics
King's College London

(Version 2.0)

Today

- Introduction
- Probabilistic Reasoning I
- Probabilistic Reasoning II
- Sequential Decision Making
- Temporal Probabilistic Reasoning
- Game Theory
- Argumentation I
- Argumentation II
- (A peek at) Machine Learning
- AI & Ethics

Recap

- In Lecture 2 we talked about using probability theory to represent uncertainty in an agent's knowledge of the world.
- Probability distribution gives values for all possible values of a random variable:

$$\mathbf{P}(Weather) = \langle 0.72, 0.1, 0.08, 0.1 \rangle$$

- Product rule relates joint probabilities to priors:

$$P(a \wedge b) = P(a|b)P(b) = P(b|a)P(a)$$

- Bayes rule relates conditional probabilities:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

Recap

- Chain rule comes from applying the product rule mulitple times:

$$\mathbf{P}(X_1, \ldots, X_n) = \prod_{i=1}^{n} \mathbf{P}(X_i | X_1, \ldots, X_{i-1})$$

- In terms of a more concrete example:

$$P(a, b, c) = P(a)P(b|a)P(c|b, a)$$

Recap

- Important step in probabilistic inference is establishing joint probabililties over all variables:

$$P(a, b, c)$$
$$P(\neg a, b, c)$$
$$\vdots$$

- With a full joint probability distribution over all the state variables
  - which we can either measure directly

$$P(toothache, cavity, \neg catch)$$

  or we can compute from conditionals

$$P(catch | toothache, cavity)$$

we can compute any specific values we want.

Recap

- Typically, we want the joint distribution of the query variables $\mathbf{Y}$ given specific values $\mathbf{e}$ for the evidence variables $\mathbf{E}$

$$\begin{aligned}\mathbf{P}(\mathbf{Y}|\mathbf{E}=\mathbf{e}) &= \alpha\mathbf{P}(\mathbf{Y},\mathbf{E}=\mathbf{e}) \\ &= \alpha\sum_{\mathbf{h}}\mathbf{P}(\mathbf{Y},\mathbf{E}=\mathbf{e},\mathbf{H}=\mathbf{h})\end{aligned}$$

- The $\mathbf{H}$ are hidden variables, the ones we don't care about for this query.

Recap

- For

$$
\begin{aligned}
\mathbf{P}(\mathbf{Y}|\mathbf{E}=\mathbf{e}) &= \alpha\mathbf{P}(\mathbf{Y},\mathbf{E}=\mathbf{e}) \\
&= \alpha\sum_{\mathbf{h}}\mathbf{P}(\mathbf{Y},\mathbf{E}=\mathbf{e},\mathbf{H}=\mathbf{h})
\end{aligned}
$$

- The **H** are hidden variables, the ones we don't care about for this query.

Recap

- For example, we have:

$$\mathbf{P}(Cavity, Toothache, Catch)$$

- We want to know the probability of having a cavity, given we have a toothache, i.e.,

$$Pr(cavity \mid toothache)$$

- $Toothache = toothache$ is the evidence variable.
- $\mathbf{P}(Cavity)$ is the query
- $Catch$ is the hidden variable.

Recap

- For example, we have:

$$\mathbf{P}(Cavity, Toothache, Catch)$$

- We want to know the probability of having a cavity, given we have a toothache.
- $Toothache = toothache$ is the evidence variable.
- $\mathbf{P}(Cavity)$ is the query
- $Catch$ is the hidden variable.

$$\begin{aligned}\mathbf{P}(Cavity|toothache) = {} & \mathbf{P}(Cavity, catch|toothache) \\ & + \mathbf{P}(Cavity, \neg catch|toothache)\end{aligned}$$

Recap

- Computationally this is awkward.
  - n binary variables requires 2^n probabilities.
- Conditional indpendence is the mechanism we exploit to do reduce this number.
- *Catch* is conditionally independent of *Toothache* given *Cavity*

$$\mathbf{P}(Catch|Toothache, Cavity) = \mathbf{P}(Catch|Cavity)$$

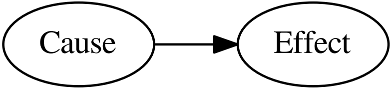
Catch does not depend on *Toothache* if we know that *Cavity* is true.

Recap

- Often easier to assess causal probabilities.

$$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$

- Can visualise this as:



- Bayes rule allows us to use these easier probabilities.

Bayes' Rule & conditional independence

- So, in our running dentist example

$$\begin{aligned}
&\mathbf{P}(\mathit{Cavity} \mid \mathit{toothache} \wedge \mathit{catch}) \\
&\quad = \alpha\, \mathbf{P}(\mathit{toothache} \wedge \mathit{catch} \mid \mathit{Cavity})\mathbf{P}(\mathit{Cavity}) \\
&\quad = \alpha\, \mathbf{P}(\mathit{toothache} \mid \mathit{Cavity})\mathbf{P}(\mathit{catch} \mid \mathit{Cavity})\mathbf{P}(\mathit{Cavity}) \\
&\quad = \alpha\, \mathbf{P}(\mathit{Cavity})\mathbf{P}(\mathit{toothache} \mid \mathit{Cavity})\mathbf{P}(\mathit{catch} \mid \mathit{Cavity})
\end{aligned}$$

- Conditional independence is an example of a naive Bayes model, in which one assumes that

$$\mathbf{P}(\mathit{Cause}, \mathit{Effect}_1, \ldots, \mathit{Effect}_n) = \mathbf{P}(\mathit{Cause}) \prod_i \mathbf{P}(\mathit{Effect}_i \mid \mathit{Cause})$$

Naive Bayes

- Models n conditionally independent effects

$$\mathbf{P}(\mathit{Cause}, \mathit{Effect}_1, \ldots, \mathit{Effect}_n) = \mathbf{P}(\mathit{Cause}) \prod_i \mathbf{P}(\mathit{Effect}_i | \mathit{Cause})$$

- Total number of parameters is linear in n
- It is called 'naive', because it is oversimplifying: in many cases the 'effect' variables aren't actually conditionally independent given the cause variable. Example:
    - *Cause*: it rained yesterday
    - Effect_1: the streets are wet this morning
    - Effect_2: I'm late for my class
    - If the streets were still wet, then an accident was more likely to happen and the caused traffic jam could be the reason for being late

Naive Bayes

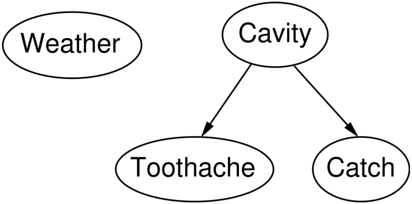
- Models two conditionally independent effects

$$\mathbf{P}(Cause, Effect_1, \ldots, Effect_n) = \mathbf{P}(Cause) \prod_i \mathbf{P}(Effect_i | Cause)$$

- Visualise as:

Bayesian networks

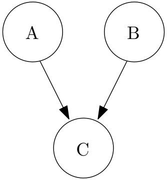
- A simple, graphical notation for conditional independence assertions and hence for compact specification of full joint distributions
- Topology of network encodes conditional independence assertions:



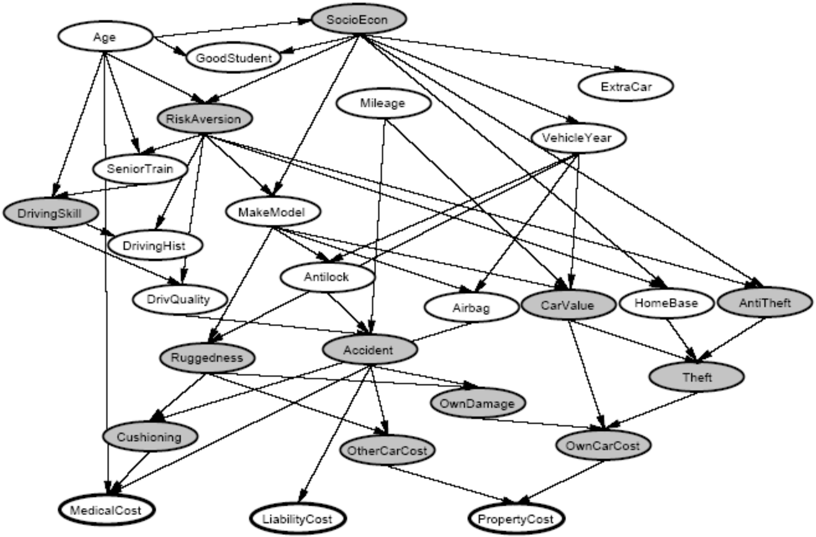
- *Weather* is independent of the other variables
- *Toothache* and *Catch* are conditionally independent given *Cavity*

Bayesian Networks

- Bayesian networks are a way to represent these dependencies:
  - Each node corresponds to a random variable (which may be discrete or continuous)
  - A directed edge (also called link or arrow) from node u to node v means that u is the parent of v.
  - The graph has no directed cycles (and hence is a directed acyclic graph, or DAG).
  - Each node v has a conditional probability $Pr(u \mid Parents(u))$ that quantifies the effect of the parent nodes
- Example: C depends on A and B, and A and B are independent.



Bayesian networks



(http://www.igi.tugraz.at)

Bayesian networks

- How can represent the knowledge about the probabilities?
- Conditional distribution represented as a conditional probability table (CPT) giving the distribution over u for each combination of parent values

A	B	$P(C \mid A, B)$
T	T	0.2
T	F	0.123
F	T	0.9
F	F	0.51

Bayesian networks

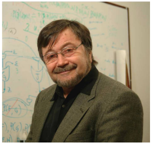
- Bayesian networks $\neq$ Naive Bayes
- These are somewhat orthogonal. Naive Bayes might be used in Bayesian networks.
- Also don't confuse them with Bayes' rule

Bayesian networks

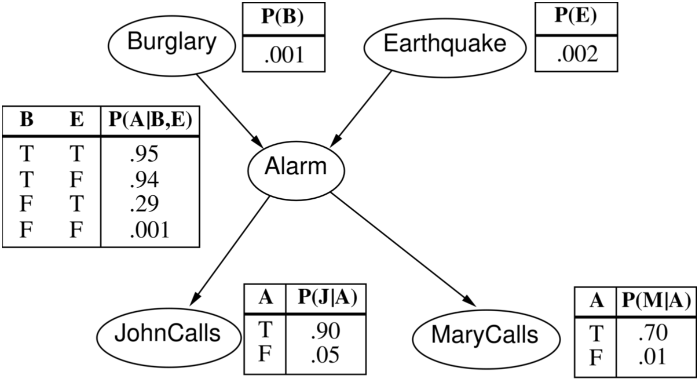
- An example (from California):

  I'm at work, neighbor John calls to say my alarm is ringing, but neighbor Mary doesn't call. Sometimes it's set off by minor earthquakes. Is there a burglar?

- Variables: *Burglar*, *Earthquake*, *Alarm*, *JohnCalls*, *MaryCalls*
- Network topology reflects "causal" knowledge:
  - A burglar can set the alarm off
  - An earthquake can set the alarm off
  - The alarm can cause Mary to call
  - The alarm can cause John to call



Bayesian networks



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

A note on CPTs

- The CPTs in the previous slide appear to be missing some values:

A	$P(J\|A)$
T	0.90
F	0.05

  has two values rather then the four which would completely specify the relation between J and A.
- The table tells us that:

$$P(J = T|A = T) = 0.9$$

  which means:

$$P(J = F|A = T) = 0.1$$

  because $P(J = T|A = T) + P(J = F|A = T) = 1$

KING'S College LONDON

A note on CPTs

- Or, writing the values of J and A the other way:

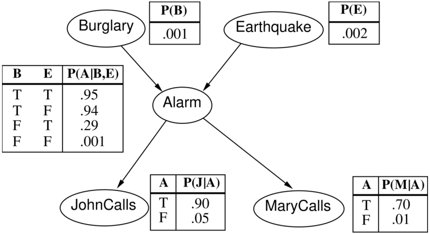
$$P(j|a) = 0.9$$
$$P(\neg j|a) = 0.1$$

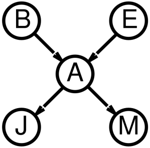
because $P(j|a) + P(\neg j|a) = 1$

Global semantics

- We can calculate the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$



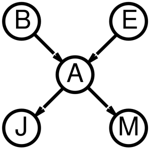


- Compute: $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$
- `https://pollev.com/frederikm106`

Global semantics

- We can calculate the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$



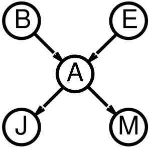
- $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$\begin{aligned} &= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \\ &\approx 0.00063 \end{aligned}$$

- Application of the chain rule.

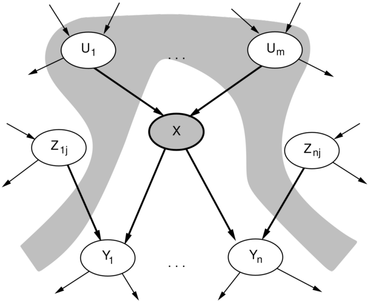
Compactness

- A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values
- Each row requires one number p for $X_i = true$ (the number for $X_i = false$ is just $1 - p$)
- If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers
- Grows linearly with n, vs. $O(2^n)$ for the full joint distribution
- For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)



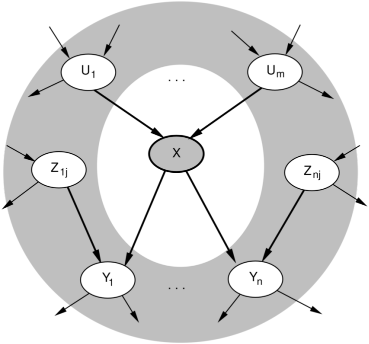
Local semantics

- A node X is conditionally independent of its non-descendants (e.g., the $Z_{i,j}$s) given its parents (the U_is shown in the gray area).



Markov blanket

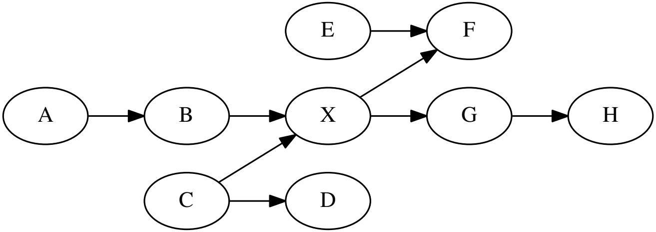
- Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



Andrey Markov

Markov blanket

- Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



- Markov blanket of X?

Markov blanket

`https://pollev.com/frederikm106`

Last year's success rate: 0.72

Answer: B,C,E,F,G

Constructing Bayesian networks

- Build Bayesian networks like any other form of knowledge representation.
- First figure out the variables that describe the world.
- Then decide how they are connected.
  Conditional independence.
- Then work out the values in the CPTs.

Kathy Laskey

Compact conditional distributions

- (This slide talks about more ways of reducing the required space to store the table(s))
- CPT grows exponentially with number of parents
  - Use distributions that are defined compactly
- Deterministic nodes are the simplest case.
- $X = f(Parents(X))$ for some function f
  - Boolean functions:

$$NorthAmerican \Leftrightarrow Canadian \vee US \vee Mexican$$

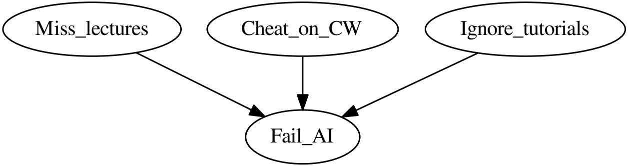
  - Numerical relationships among continuous variables

$$\frac{\partial Level}{\partial t} = \text{inflow + precipitation - outflow - evaporation}$$

KING'S College LONDON

Noisy-OR

- Say we have this network



- The noisy-OR model allows for uncertainty about the ability of each parent to cause the child to be true—the causal relationship between parent and child may be inhibited: a student could miss a lecture, but might not fail the class.
- This is different from conventional logic
- Compare Noisy-OR to Naive Bayes
- We would like to compute things such as:
  $P(fail\_AI \mid miss\_lectures, ignore\_tutorials, \neg cheat\_on\_CW)$

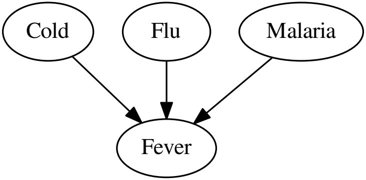
Compact conditional distributions

- Noisy-OR distributions model multiple noninteracting causes

1. Parents $U_1 \ldots U_k$ include all causes
   (Can add leak node that covers "miscellaneous causes.")
2. It assumes that inhibition of each parent is independent of inhibition of any other parents. Formally,
   $\Rightarrow\ P(X|U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \prod_{i=1}^{j} q_i,$

where $q_i = P(\neg X \mid U_i)$

Example



- $$\begin{aligned} q_{cold} &:= P(\neg fever \mid cold) &= 0.6 \\ q_{flu} &:= P(\neg fever \mid flu) &= 0.2 \\ q_{malaria} &:= P(\neg fever \mid malaria) &= 0.1 \end{aligned}$$
- In this model, we then get $P(\neg fever \mid cold, flu, \neg malaria) = q_{cold} \cdot q_{flu} = 0.12$
- Hence, $P(fever \mid cold, flu, \neg malaria) = q_{cold} \cdot q_{flu} = 1 - 0.12 = 0.88$
- Given that you have the flu and a cold, the only way you cannot have fever is if neither of them gave you fever. (Think of all possible outcomes)

More examples

$$
\begin{aligned}
q_{cold} &:= P(\neg fever \mid cold) &= 0.6\\
q_{flu} &:= P(\neg fever \mid flu) &= 0.2\\
q_{malaria} &:= P(\neg fever \mid malaria) &= 0.1
\end{aligned}
$$

Cold	*Flu*	*Malaria*	$P(Fever)$	$P(\neg Fever)$
F	F	F	**0.0**	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

- Number of parameters linear in number of parents

KING'S College LONDON

Noisy-OR

- The required space is small!

- Okay, so what can we do with Bayesian Networks?

- Okay, so what can we do with Bayesian Networks?
- They are useful for inference (a conclusion reached on the basis of evidence and reasoning)

Inference tasks

- Simple queries: compute posterior marginal $\mathbf{P}(X_i|\mathbf{E}=\mathbf{e})$

  $\mathbf{P}(Brexit|PM = Boris, Economy\ =\ not\_great)$

- Conjunctive queries

  $\mathbf{P}(X_i, X_j|\mathbf{E}=\mathbf{e})\ =\ \mathbf{P}(X_i|\mathbf{E}=\mathbf{e})\mathbf{P}(X_j|X_i, \mathbf{E}=\mathbf{e})$

- Optimal decisions: decision networks include utility information; probabilistic inference required for $P(outcome|action, evidence)$
- Value of information: which evidence to seek next?
- Sensitivity analysis: which probability values are most critical?
- Explanation: why do I need a new starter motor?

Inference tasks

- We will focus on simple queries:
- Compute posterior marginal

  $$\mathbf{P}(X_i|\mathbf{E}=\mathbf{e})$$

  $$\mathbf{P}(Brexit|PM=Boris, Economy\ =\ not\_great)$$

- We will look a several ways of doing this.
  1. Enumeration
  2. Rejection sampling (using prior sampling)
  3. Likelihood weighting
  4. Gibbs sampling

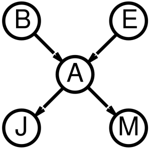
Inference by enumeration

- Simplest approach to evaluating the network is to do just as we did for the dentist example.
- Difference is that we use the structure of the network to tell us which sets of joint probabilities to use.
  - Thanks Professor Markov
- Gives us a slightly intelligent way to sum out variables from the joint without actually constructing its explicit representation

Inference by enumeration

- Simple query on the burglary network.

$$
\begin{aligned}
\mathbf{P}(B|j,m) &= \frac{\mathbf{P}(B,j,m)}{P(j,m)} \\
&= \alpha\mathbf{P}(B,j,m) \\
&= \alpha \sum_{e} \sum_{a} \mathbf{P}(B,e,a,j,m)
\end{aligned}
$$



- Rewrite full joint entries taking network into account:

$$
\begin{aligned}
\mathbf{P}(B|j,m) &= \alpha \sum_{e} \sum_{a} \mathbf{P}(B)P(e)\mathbf{P}(a|B,e)P(j|a)P(m|a) \\
&= \alpha\mathbf{P}(B) \sum_{e} P(e) \sum_{a} \mathbf{P}(a|B,e)P(j|a)P(m|a)
\end{aligned}
$$

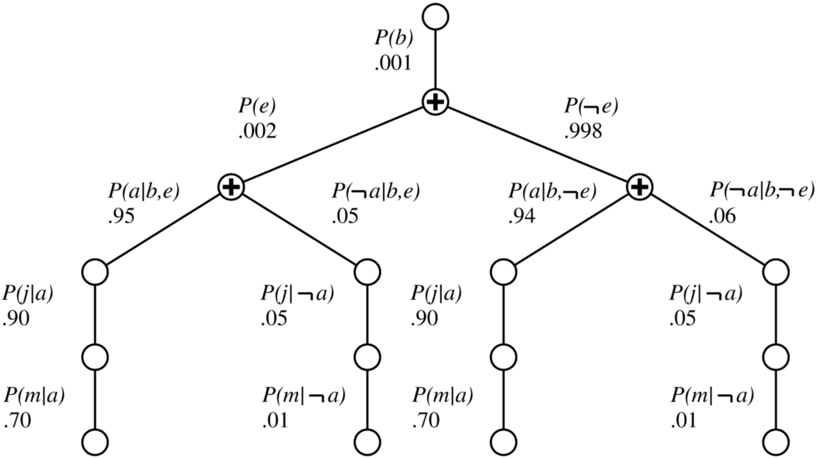
Inference by enumeration

- We evaluate this expression

$$\mathbf{P}(B|j,m) = \alpha\mathbf{P}(B)\sum_{e} P(e)\sum_{a}\mathbf{P}(a|B,e)P(j|a)P(m|a)$$

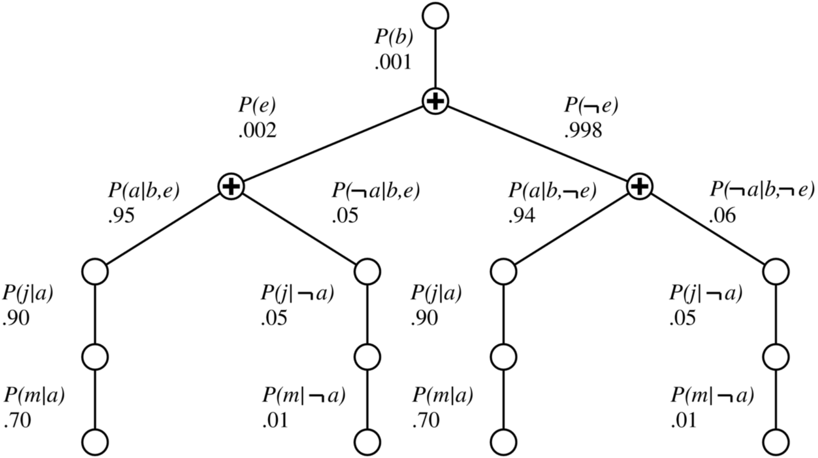
  by going through the variables in order, multiplying CPT entries along the way.
- At each point, we need to loop through the possible values of the variable.
- Involves a lot of repeated calculations.

Evaluation tree



$$\mathbf{P}(B|j,m) = \alpha \mathbf{P}(B) \sum_e P(e) \sum_a \mathbf{P}(a|B,e) P(j|a) P(m|a)$$

Evaluation tree



Inefficient: computes $P(j|a)P(m|a)$ for each value of e

Enumeration algorithm

function ENUMERATION-ASK($X, \mathbf{e}, bn$) **returns** a distribution over X

inputs: X, the query variable

$\mathbf{e}$, observed values for variables $\mathbf{E}$

bn, a Bayesian network $\{X\} \cup \mathbf{E} \cup \mathbf{Y}$

$\mathbf{Q}(X) \leftarrow$ a distribution over X, initially empty

for each value x_i of X **do**

extend $\mathbf{e}$ with value x_i for X

$\mathbf{Q}(x_i) \leftarrow$ ENUMERATE-ALL(VARS[bn], $\mathbf{e}$)

return NORMALIZE($\mathbf{Q}(X)$)

Enumeration algorithm

function ENUMERATE-ALL($vars$, $\mathbf{e}$) **returns** a real number
  if EMPTY?($vars$) **then return** 1.0
  $Y \leftarrow$ FIRST($vars$)
  if Y has value y in $\mathbf{e}$
    then return $P(y \mid Pa(Y))$
      $\times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}$)
    else return $\sum_y P(y \mid Pa(Y))$
      $\times$ ENUMERATE-ALL(REST($vars$), $\mathbf{e}_y$)
     where $\mathbf{e}_y$ is $\mathbf{e}$ extended with $Y = y$

Other exact approaches

- We can improve on enumeration.
- Variable elimination evaluates the enumeration tree bottom up, remembering intermediate values.
  - Simple and efficient for single queries
- Clustering algorithms can be more efficient for multiple queries
  - Group variables together strategically.
- However, *all* exact inference can be computationally intractable.

Complexity of exact inference

- Singly connected networks (or polytrees)
- Any two nodes are connected by at most one (undirected) path
- Time and space cost of variable elimination are:

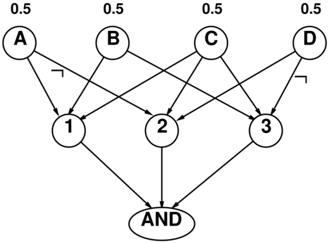
$$O(d^k n)$$

for k parents, d values.

Complexity of exact inference

- Multiply connected networks.
- Exponential time and space complexity, even when number of parents of a node is bounded.
- Inference is NP-hard.
- In fact, #P-complete.

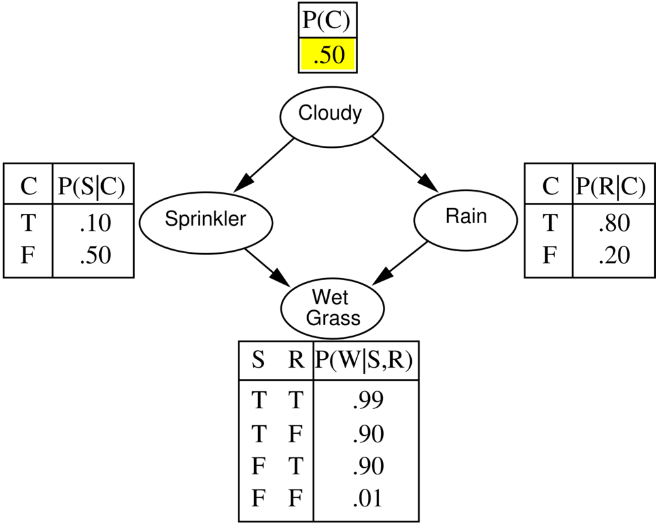
1. A v B v C
2. C v D v ¬A
3. B v C v ¬D



Prior sampling

- Let’s use stochastic sampling instead!

Prior sampling



P(C)
.50

C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

- How would you estimate $P(c, \neg s, r, w)$?

Take a step back

- How would you estimate $P(\text{die shows } 7)$?

Stochastic simulation

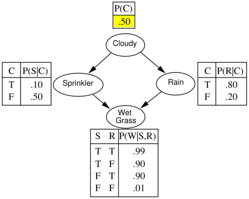
- How would you estimate $P(\text{die shows 7})$?
- Simple: you take n random samples from the network
- Let X_i be the binary r.v. that is 1 if the event sampled in the ith run is 7
- Simply output

$$\hat{P}(7) = \frac{\sum_{i=1}^{n} X_i}{n}$$

- Law of large numbers says that $\lim_{n \to \infty} \hat{P} = P$.

Stochastic simulation

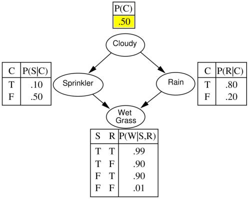
- Back to our Bayesian network with cloudy, sprinkler, rain, and wet grass.



- How would you estimate $P(c, \neg s, r, w)$?

Stochastic simulation

- Back to our Bayesian network with cloudy, sprinkler, rain, and wet grass.

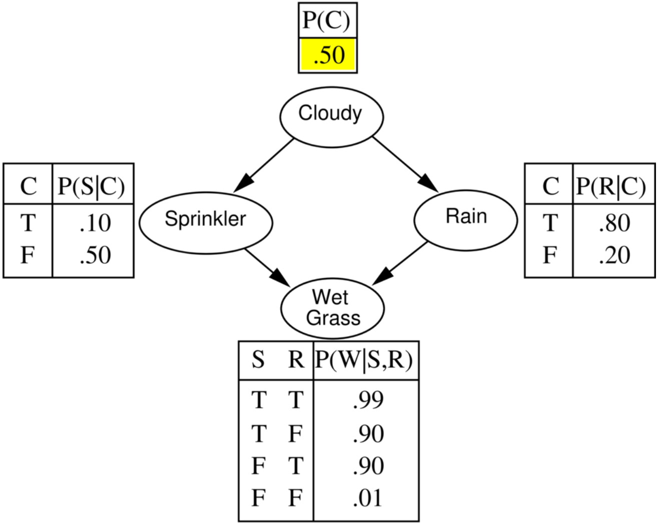


- How would you estimate $P(c, \neg s, r, w)$?
- Simple you just take say n random samples from the network
- Let X_i be the binary r.v. that is 1 if the event sampled in the ith run is $c, \neg s, r, w$
- Simply output

$$\hat{P}(c, \neg s, r, w) = \frac{\sum_{i=1}^{n} X_i}{n}$$

- Law of large numbers says that $\lim_{n \to \infty} \hat{P} = P$.

Prior sampling. Let's generate a random sample!



P(C)
.50

C	P(S|C)
T	.10
F	.50

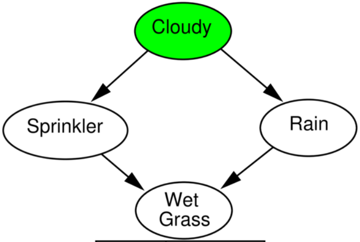
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Prior sampling.

- List of randomly generated numbers: 0.4, 0.2, 0.71, 0.2 for this run.
- We have 0.4 small than $P(C)$, so $Cloudy = true$

P(C)
.50

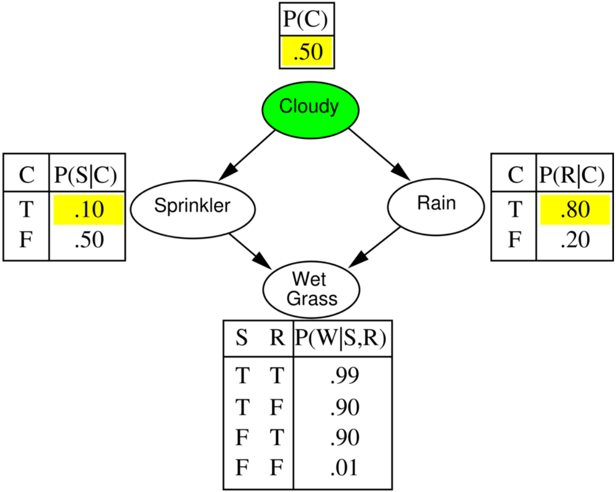


C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

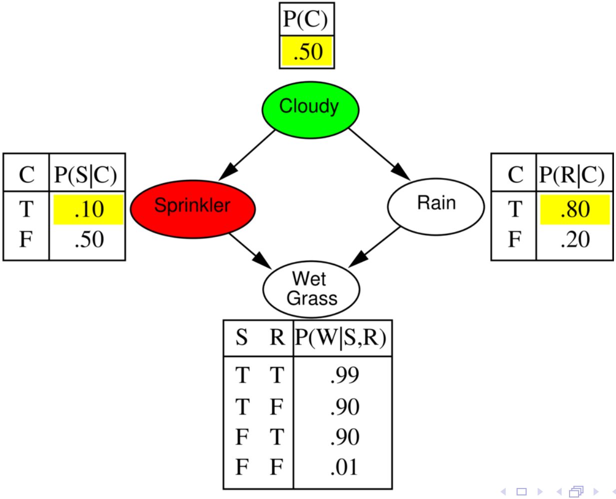
S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Prior sampling



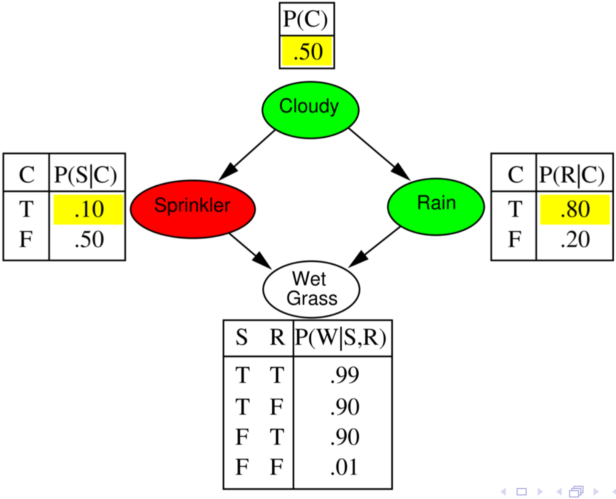
Prior sampling

- List of randomly generated numbers: 0.4, 0.2, 0.71, 0.2 for this run
- We have $0.2 \geqslant P(S \mid C)$ and hence, $\mathit{Sprinkler} = \mathit{false}$.

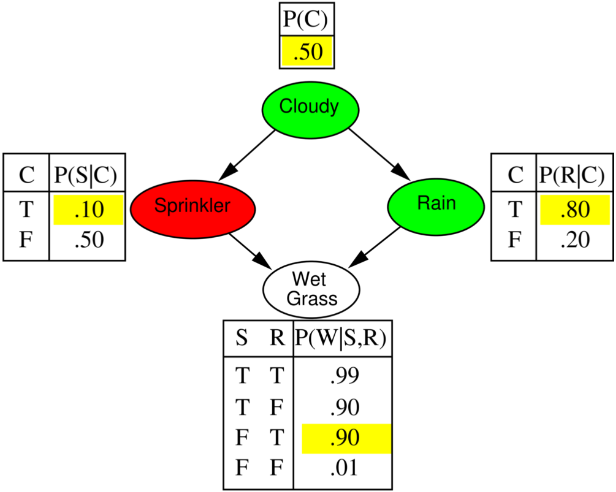


Prior sampling

- List of randomly generated numbers: 0.4, 0.2, 0.71, 0.2 for this run.
- We have $0.71 < P(R \mid C)$ and hence, $\mathit{Rain} = \mathit{true}$



Prior sampling



P(C)
.50

C	P(S|C)
T	.10
F	.50

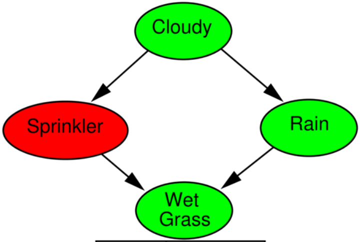
C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Prior sampling

- List of randomly generated numbers: 0.4, 0.2, 0.71, 0.2 for this run.
- We have $0.2 < P(W \mid S, R)$ and hence, *WetGrass* $=$ *true*

P(C)
.50



C	P(S|C)
T	.10
F	.50

C	P(R|C)
T	.80
F	.20

S	R	P(W|S,R)
T	T	.99
T	F	.90
F	T	.90
F	F	.01

Prior sampling

- So, this time we get the event

$$[Cloudy = true, Sprinkler = false, Rain = true, WetGrass = true]$$

- Will write this:

$$[true, false, true, true]$$

Prior sampling

- If we repeat the process many times, we can count the number of times $[true, false, true, true]$ is the result.
- The proportion of this to the total number of runs is:

$$P(c, \neg s, r, w)$$

- The more runs, the more accurate the probability.
- Similarly for other joint probabilities.

Prior sampling

function PRIOR-SAMPLE(*bn*) **returns** an event sampled from *bn*
  inputs: *bn*, a belief network specifying joint distribution $\mathbf{P}(X_1, \ldots, X_n)$

  $\mathbf{x} \leftarrow$ an event with n elements
  for $i = 1$ **to** n **do**
    $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
      given the values of $Parents(X_i)$ in $\mathbf{x}$
  return $\mathbf{x}$

Prior sampling limitation

- How would you get the following marginal distribution?

$$\mathbf{P}(X|\mathbf{e})$$

Rejection sampling

- Use Rejection sampling.
- Generate samples as before (using prior sampling)
- If the sample is such that **e** holds use it to build an estimate
- Otherwise, ignore it

Rejection sampling

function REJECTION-SAMPLING(X, **e**, bn, N) **returns** an estimate of $P(X|\mathbf{e})$
  local variables: **N**, a vector of counts over X, initially zero

  for $j = 1$ to N **do**
    $\mathbf{x} \leftarrow$ PRIOR-SAMPLE(bn)
    if **x** is consistent with **e** **then**
      $\mathbf{N}[x] \leftarrow \mathbf{N}[x]+1$ where x is the value of X in **x**
  return NORMALIZE($\mathbf{N}[X]$)

Rejection sampling

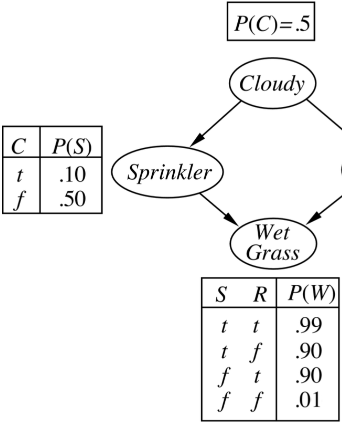
- More efficient than prior sampling, but
- For unlikely events, may have to wait a long time to get enough matching samples.
- Still inefficient.
- So, use likelihood weighting

Likelihood weighting

- Version of importance sampling.
- Fix evidence variable to *true*, so just sample relevant events.
- Have to weight them with the likelihood that they fit the evidence.
- Use the probabilities we know to weight the samples.

Likelihood weighting

- Consider we have the following network:



- Say we want to establish
  $\mathbf{P}(Rain | Cloudy = true, WetGrass = true)$

Likelihood weighting

- We want $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$
- We pick a variable ordering, say:

  $Cloudy, Sprinkler, Rain, WetGrass.$

  as before.
- Set the weight $w = 1$ and we start.
- Deal with each variable in order.

Likelihood weighting

- Remember, we want
  $\mathbf{P}(Rain|Cloudy = true, WetGrass = true)$
- *Cloudy* is true, so:

$$w \leftarrow w \times P(Cloudy = true)$$
$$w \leftarrow 0.5$$

- $Cloudy = true$, $Sprinkler = ?$, $Rain = ?$, $WetGrass = ?$.
- $w = 0.5$

Likelihood weighting

- *Sprinkler* is not an evidence variable, so we don't know whether it is true or false.
- Sample a value just as we did for prior sampling:

$$\mathbf{P}(Sprinkler|Cloudy = true) = \langle 0.1, 0.9 \rangle$$

- Let's assume this returns *false*.
- w remains the same.
- $Cloudy = true$, $Sprinkler = false$, $Rain = ?$, $WetGrass = ?$.
- $w = 0.5$

Likelihood weighting

- *Rain* is not an evidence variable, so we don't know whether it is true or false.
- Sample a value just as we did for prior sampling:

$$\mathbf{P}(Rain|Cloudy = true) = \langle 0.8, 0.2 \rangle$$

- Let's assume this returns *true*.
- w remains the same.
- $Cloudy = true$, $Sprinkler = false$, $Rain = true$, $WetGrass = ?$.
- $w = 0.5$

Likelihood weighting

- *WetGrass* is an evidence variable with value *true*, so we set:

  $$w \leftarrow w \times P(WetGrass = true | Sprinkler = false, Rain = true)$$
  $$w \leftarrow 0.45$$

- $Cloudy = true$, $Sprinkler = false$, $Rain = true$, $WetGrass = true$.
- $w = 0.45$

Likelihood weighting

- So we end with the event $[true, false, true, true]$ and weight 0.45.
- To find a probability we tally up all the relevant events, weighted with their weights.
- The one we just calculated would tally under

$$Rain = true$$

- As before, more samples means more accuracy.

Likelihood weighting

function LIKELIHOOD-WEIGHTING(X, $\mathbf{e}$, bn, N) **returns** an estimate of $P(X|\mathbf{e})$

local variables: $\mathbf{W}$, a vector of weighted counts over X, initially zero

for $j = 1$ to N **do**
  $\mathbf{x}, w \leftarrow$ WEIGHTED-SAMPLE(bn)
  $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in $\mathbf{x}$
return NORMALIZE($\mathbf{W}[X]$)

Likelihood weighting

function WEIGHTED-SAMPLE(bn, $\mathbf{e}$) **returns** an event and a weight

> $\mathbf{x} \leftarrow$ an event with n elements; $w \leftarrow 1$
> **for** $i = 1$ **to** n **do**
>   **if** X_i has a value x_i in $\mathbf{e}$
>   **then** $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
>   **else** $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
> **return** $\mathbf{x}$, w

Gibbs sampling

- A rather different approach to sampling.
- Part of the Markov Chain Monte Carlo (MCMC) family of algorithms.
- Don't generate each sample from scratch.
- Generate samples by making a random change to the previous sample.

Gibbs sampling

- Gibbs sampling for Bayesian networks starts with an arbitrary state.
- So pick state with evidence variables fixed at observed values.

  (If we know *Cloudy* $=$ *true*, we pick that.)
- Generate next state by randomly sampling from a non-evidence variable.
- Do this sampling conditional on the current values of the Markov blanket.
- “The algorithm therefore wanders randomly around the state space . . . flipping one variable at a time, but keeping the evidence variables fixed”.

- Consider the query:

$$\mathbf{P}(Cloudy \mid Sprinker = true, WetGrass = true)$$

- The evidence variables are fixed to their observed values.
- The non-evidence variables are initialised randomly.

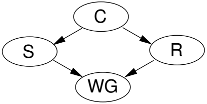
$$Cloudy = true$$
$$Rain = false$$

- State is thus:

$$[Cloudy = true, Sprinkler = true,$$
$$Rain = false, WetGrass = true].$$

Gibbs sampling

- First we sample *Cloudy* given the current state of its Markov blanket.
- Markov blanket is *Sprinkler* and *Rain*.
- So, sample from:



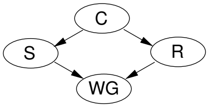
$$\mathbf{P}(Cloudy | Sprinkler = true, Rain = false)$$

- Suppose we get $Cloudy = false$, then new state is:

$$[Cloudy = false, Sprinkler = true,$$
$$Rain = false, WetGrass = true].$$

Gibbs sampling

- Next we sample *Rain* given the current state of its Markov blanket.
- Markov blanket is *Cloudy*, *Sprinkler* and *WetGrass*.
- So, sample from:



$$\mathbf{P}(Rain|Cloudy = false, Sprinkler = true, WetGrass = false)$$

- Suppose we get $Rain = true$, then new state is:

$$[Cloudy = false, Sprinkler = true,$$
$$Rain = true, WetGrass = true].$$

Gibbs sampling

- Each state visited during this process contributes to our estimate for:

$$\mathbf{P}(Cloudy|Sprinkler = true, WetGrass = true)$$

- Say the process visits 80 states.
- In 20, $Cloudy = true$
- In 60, $Cloudy = false$
- Then

$$\begin{aligned}\mathbf{P}(Cloudy|Sprinkler = true, WetGrass = true) \\ &= Normalize(\langle 20, 60\rangle) \\ &= \langle 0.25, 0.75\rangle\end{aligned}$$

Gibbs sampling

function GIBBS-ASK(X, $\mathbf{e}$, bn, N) **returns** an estimate of $P(X|\mathbf{e})$
local variables: $\mathbf{N}[X]$, a vector of counts over X, initially zero
$\mathbf{Z}$, the nonevidence variables in bn
$\mathbf{x}$, the current state of the network, initially copied from $\mathbf{e}$

initialize $\mathbf{x}$ with random values for the variables in $\mathbf{Z}$
for $j = 1$ to N **do**
    for each Z_i in $\mathbf{Z}$ **do**
        sample the value of Z_i in $\mathbf{x}$ from $\mathbf{P}(Z_i|mb(Z_i))$
            given the values of $MB(Z_i)$ in $\mathbf{x}$
        $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in $\mathbf{x}$
return NORMALIZE($\mathbf{N}[X]$)

Gibbs sampling

- All of this begs the question:

  "How do we sample a variable given the state of its Markov blanket?"

- For a value x of a variable X:

$$\mathbf{P}(X|mb(X)) = \alpha \mathbf{P}(X|parents(X)) \prod_{Y \in Children(X)} \mathbf{P}(Y|parents(Y))$$

  where $mb(X)$ is the Markov blanket of X.

- Given $\mathbf{P}(X|mb(X))$, we can sample from it just as we have before.

To summarize

- Bayesian networks exploit conditional independence to create a more compact set of information.
- Reasonably efficient computation for some problems.
- Five approaches to inference in Bayesian networks.
  - Exact: Inference by enumeration.
  - Approximate: Prior sampling
  - Approximate: Rejection sampling
  - Approximate: Importance sampling/likelihood weighting
  - Approximate: Gibbs sampling
- Can answer a simple query for any BN.

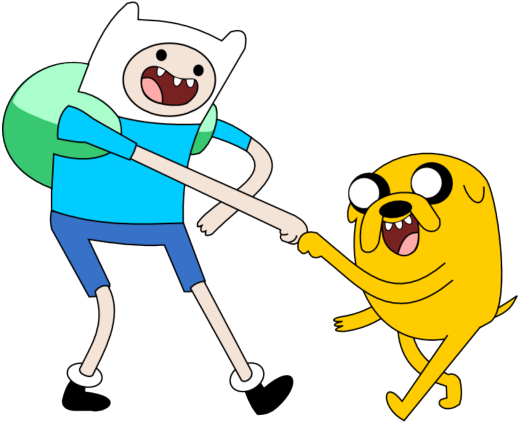
From probability to decision making

- What we have covered allows us to compute probabilities of interesting events.
- But beliefs alone are not so interesting to us.
- In Pacman don't care so much if there is a ghost in (2, 2), so much as we care whether we should go WEST or SOUTH.
- This is complicated in an uncertain world.
    - Don't know the outcome of actions.
    - Non-deterministic as well as partially observable

From probability to decision making

- Next time!

Mathematical!



(Pendleton Ward/Cartoon Network)

Summary

- This lecture started with exact probabilistic inference in Bayesian networks.
  - Inference by enumeration.
- Efficiency prompted us to look at approximate techniques based on sampling:
  - Prior sampling
  - Rejection sampling
  - Likelihood weighting (Importance sampling)
  - Gibbs sampling