Access Code: **280562**

CMP9794M
Advanced Artificial Intelligence

Heriberto Cuayahuitl

UNIVERSITY OF
LINCOLN

School of Engineering and Physical Sciences

Last Week

- Discussion on structure learning

- Algorithms to induce the structure of Bayes nets
  - PC-Stable (step 1 in particular)
  - Hill Climbing
  - Min-Max Hill Climbing

- Multiple metrics to measure the performance of probabilistic models on test data

Today

- **Inference by stochastic simulation**

- Approximate inference in Bayesian nets (BNs)
  - Rejection Sampling
  - Likelihood Weighting
  - Gibbs Sampling

- Data discretisation for learning discrete BNs

- Questions and answers related to assignment

Acknowledgement: Content of slides adapted from Prof. Simon Parsons' slides.

Inference by Stochastic Simulation

- Basic idea:
  - Draw N samples from a sampling distribution S
  - Compute an approximate posterior probability $\hat{P}$
  - $\hat{P}$ converges to the true probability P as the number of samples tends to infinity.
- That can be used when exact inference is intractable (in the case of large Bayes nets or many hidden vars.)
- Let's look at an example of the basic idea.

Inference by Stochastic Simulation

- Imagine we have a stochastic process that generates numbers in the range: 0, 1, . . . 9
- How do we determine the probability of each value being generated?
  - One possibility is to generate lots of numbers (samples) and look at the sample average for each value.
  - If we generate N samples, and m of them are 9, then the sample average is: $\frac{m}{N}$
  - This is an estimate of the probability of 9 being generated.

Inference by Stochastic Simulation

- Overall process:
  - Draw N samples from a sampling distribution S. The "sampling distribution" is generated by the process.
  - Compute an approximate posterior probability $\hat{P}$. It is an estimate (what the hat means) because it is approximate.
  - $\hat{P}$ converges to the true probability P as the number of samples tends to infinity.
- But note that the more samples, the more accurate our estimates will be.

Inference by Stochastic Simulation

- This works for any process where we can just observe the samples:
  - Rain in Lincoln (in October).
  - How many students turn up to Advanced AI.
  - How many apples my apple tree produces.
  - How long it takes me to walk to work.

- However, for our BN we have to generate the samples.

- That is what we will look at next.

Today

- Inference by stochastic simulation

- Approximate inference in Bayesian nets (BNs)
  - **Rejection Sampling**
  - Likelihood Weighting
  - Gibbs Sampling

- Data discretisation for learning discrete BNs

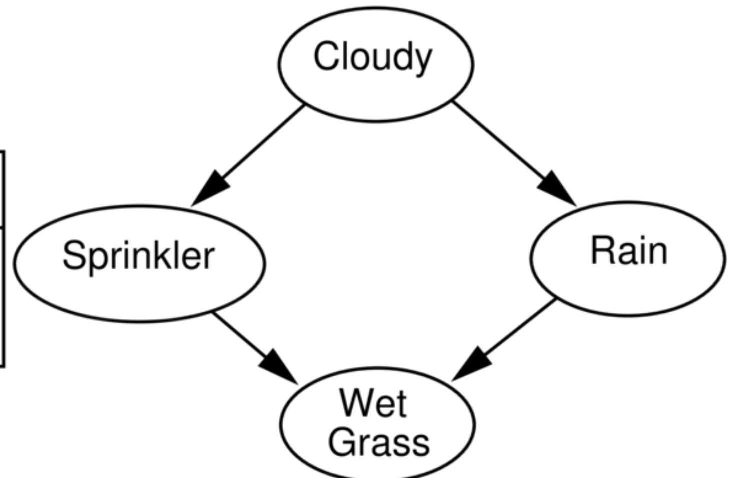
- Questions and answers related to assignment

Acknowledgement: Content of slides adapted from Prof. Simon Parsons' slides.

Prior Sampling

P(C)
.50

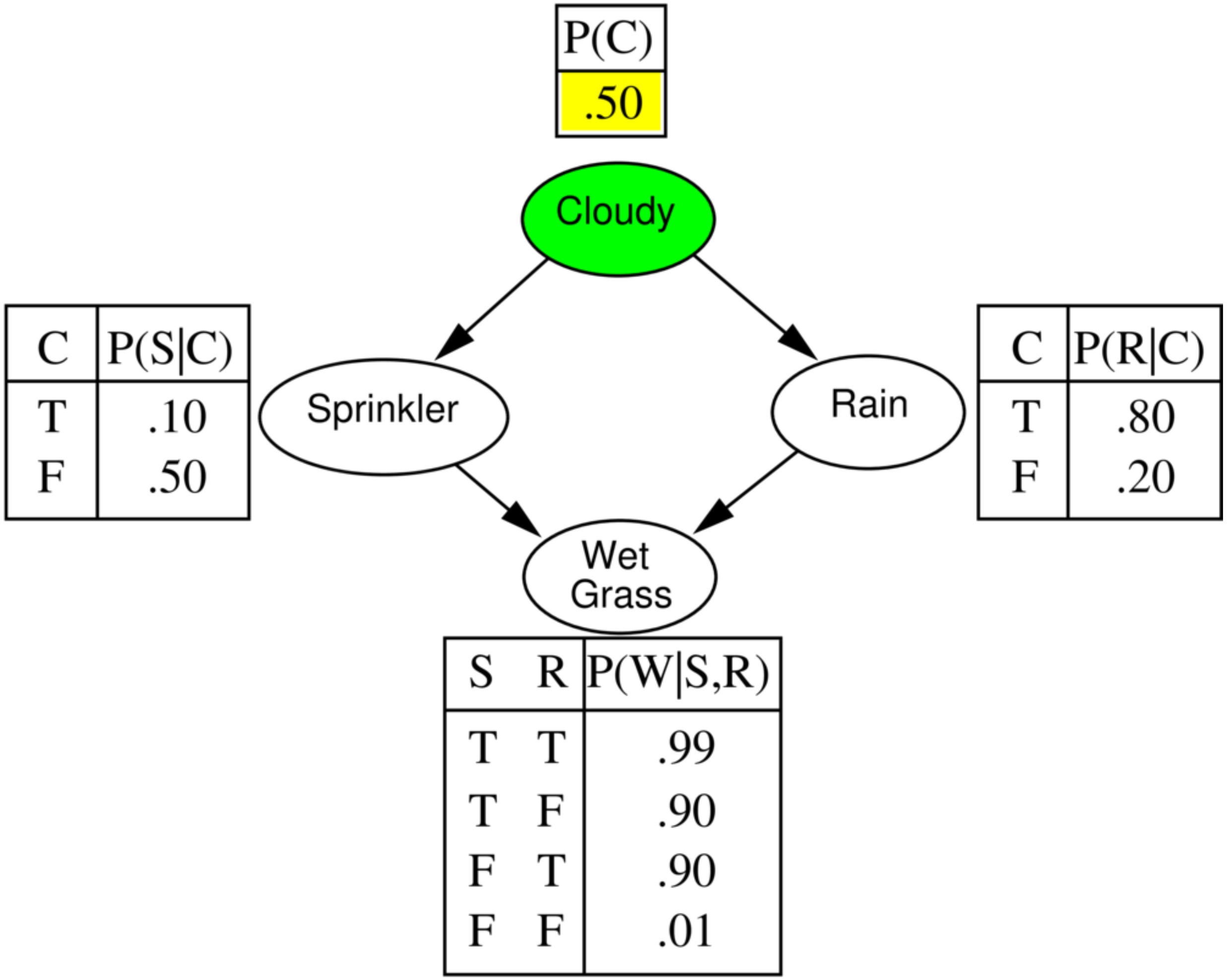
C	P(S|C)
T	.10
F	.50



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

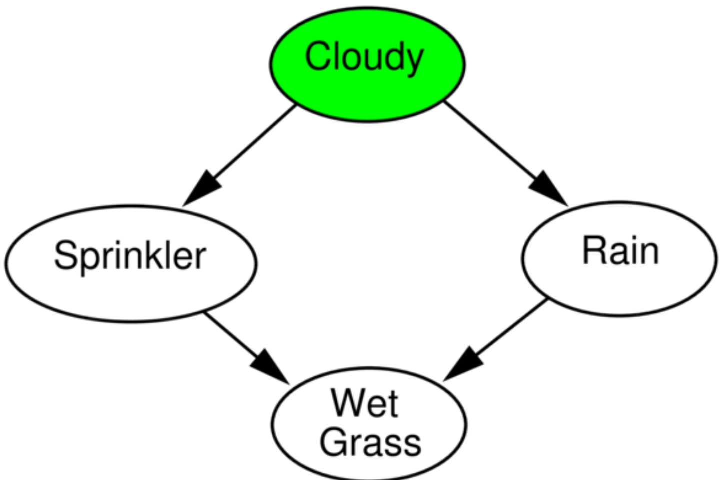
Prior Sampling



Cloudy = true

Prior Sampling

P(C)
.50



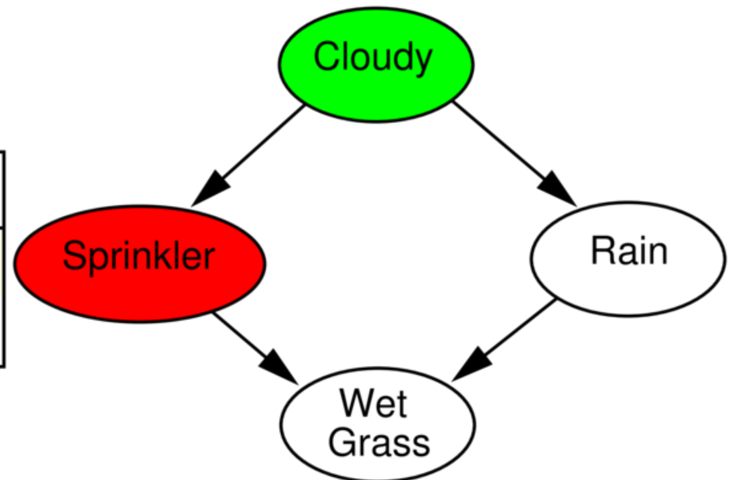
C	P(S|C)
T	.10
F	.50

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

P(C)
.50



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

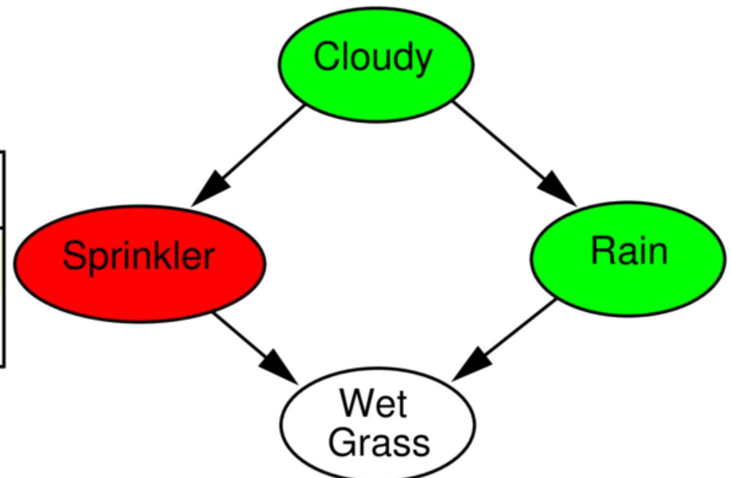
C	P(R|C)
T	.80
F	.20

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

Sprinkler = false

Prior Sampling

P(C)
.50



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

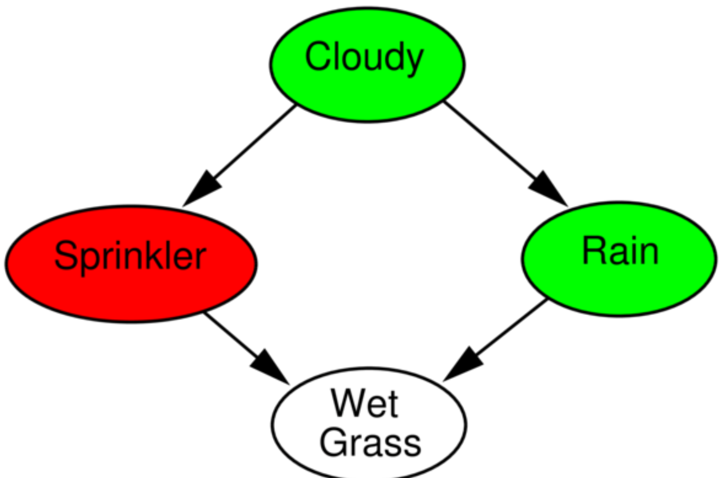
C	P(R|C)
T	.80
F	.20

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

Rain = true

Prior Sampling

P(C)
.50

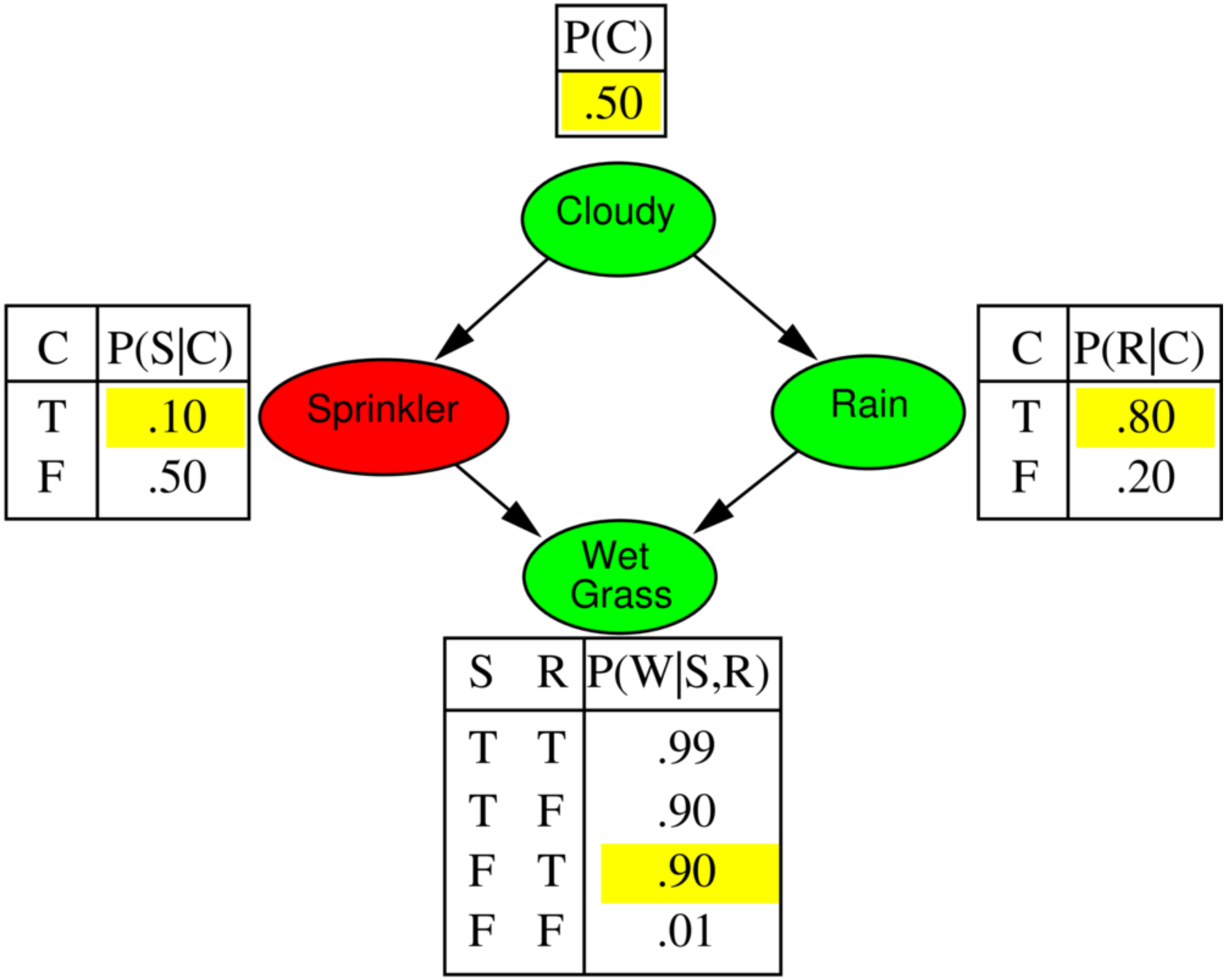


C	P(S|C)
T	.10
F	.50

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



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

WetGrass = true

Prior Sampling

- So, we get [$Cloudy = true,\ Sprinkler = false, Rain = true, WetGrass = true$]
- We write this as [$true, false, true, true$], assuming a specific order of random variables (hierarchical).

- If we repeat the process many times (N), we can count the number of times [$true, false, true, true$] is the result (m).
- The proportion $\frac{m}{N}$ gives us $P(c, \neg s, r, w)$
- The more runs, the more accurate the probability.

Prior Sampling: Question for you

- Similarly, the proportion $[false, true, true, true]$ gives us $P(\neg c, s, r, w)$. Thus, given:
  - [true, false, true, true]
  - [false, false, true, true]
  - [true, true, true, true]
  - [true, false, false, true]
  - [true, false, true, true]
  - [true, false, true, false]
- $P(c, \neg s, r, w) \approx ?$
- $P(\neg c, s, r, w) \approx ?$

Prior Sampling: Algorithm

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 $\mathbf{x}$

Rejection Sampling: Algorithm

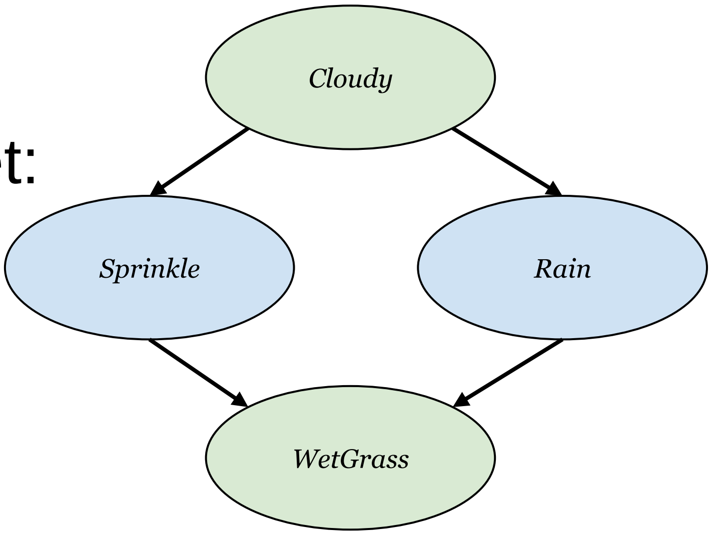
function REJECTION-SAMPLING(X, $\mathbf{e}$, bn, N) **returns** an estimate of $P(X|\mathbf{e})$
local variables: $\mathbf{N}$, a vector of counts over X, initially zero

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

Just calls prior samping, but only counts cases that are consistent with the evidence.

Rejection Sampling: Example

Consider the Sprinkler Bayes net:



Estimate $P(Rain \mid Sprinkler = true)$ from 100 samples

- 73 have $Sprinkler = false$, therefore 73 are rejected
- Out of the remaining samples (27) with $Sprinkler = true$, **8** have $Rain = true$ and **19** have $Rain = false$
- The approximate distribution is $\hat{P}(Rain|Sprinkler = true) = \alpha < 8{,}19 >=< 0.296, 0.704 >$

Rejection Sampling: Summary

- Rejection sampling uses (conditional) probabilities to sample events, reject those that do not match the evidence, and calculate posterior probabilities.
- For unlikely events, may have to wait a long time to get enough matching samples.
- That can be inefficient.
- So, use alternative algorithms.

Today

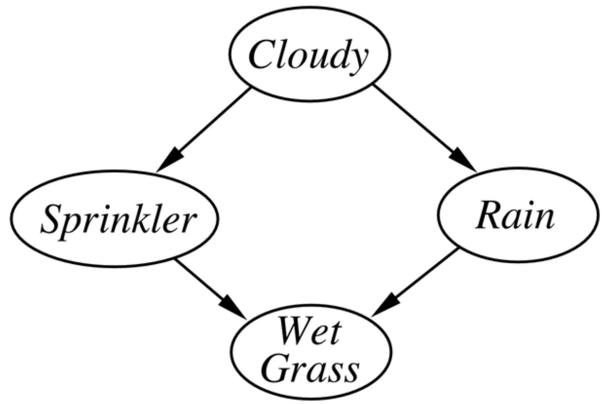
- Inference by stochastic simulation
- Approximate inference in Bayesian nets (BNs)
  - Rejection Sampling
  - **Likelihood Weighting**
  - Gibbs Sampling
- Data discretisation for learning discrete BNs
- Questions and answers related to assignment

Acknowledgement: Content of slides adapted from Prof. Simon Parsons' slides.

Likelihood Weighting: Example

Consider we have the Sprinkler network:

$P(C) = .5$



C	$P(S)$
t	.10
f	.50

C	$P(R)$
t	.80
f	.20

S	R	$P(W)$
t	t	.99
t	f	.90
f	t	.90
f	f	.01

Say we want to establish

$$P(Rain \mid Cloudy = true, WetGrass = true)$$

Likelihood Weighting: Example

- We pick a variable ordering as before, say: *Cloudy, Sprinkler, Rain, WetGrass*.
- Set the weight $w = 1$ and we start.
- Deal with each variable in order.

- $Cloudy$ is true, so:
  - $w = w * P(Cloudy = true) = 1 * 0.5 = 0.5$
- Event:
  - $[Cloudy = true, Sprinkler = ?, Rain = ?, WetGrass = ?]$

Likelihood Weighting: Example

- $Sprinkler$ is not an evidence variable, so we do not know whether it is true or false.
- Sample a value just as we did for prior sampling:
  - $P(Sprinkler|Cloudy = true) =< 0.1, 0.9 >$
- Let's assume this returns false.
- w remains the same.

Only update weights in the case of evidence variables.

- Event:
  - $[Cloudy = true, Sprinkler = false, Rain = ?, WetGrass = ?]$

Likelihood Weighting: Example

- $Rain$ is not an evidence variable, so we also do not know whether it is true or false.
- Sample a value just as we did for prior sampling:
  - $P(Rain|Cloudy = true) =< 0.8, 0.2 >$
- Let's assume this returns true.
- w remains the same.

- Event:
  - $[Cloudy = true, Sprinkler = false, Rain = true, WetGrass = ?]$

Likelihood Weighting: Example

- $WetGrass$ is an evidence variable, with value true, so we set:
  - $w = w * P(WetGrass = true | Sprinkler = false, Rain = true) = 0.5 * 0.90 = 0.45$

- Event:
  - $[Cloudy = true, Sprinkler = false, Rain = true, WetGrass = true]$

- So we end with the event [*true*, *false*, *true*, *true*] and weight $w = 0.45$.

Likelihood Weighting: Example

10 samples:

$Cloudy$	$Sprinkler$	$Rain$	$GrassWet$	w
$true$	$false$	$true$	$true$	0.45
$true$	$true$	$true$	$true$	0.475
$true$	$false$	$false$	$true$	0.05
$true$	$false$	$true$	$true$	0.45
$true$	$false$	$true$	$true$	0.45
$true$	$false$	$true$	$true$	0.45
$true$	$true$	$true$	$true$	0.475
$true$	$false$	$true$	$true$	0.45
$true$	$false$	$false$	$true$	0.05
$true$	$false$	$true$	$true$	0.45

To find $P(Rain = true | Cloudy = true, WetGrass = true)$ we add up the green-related weights and divide by the sum of all the weights.

Likelihood Weighting: Full Example

Query $P(R \mid c, g)$, assuming order [*Cloudy*, *Sprinkler*, *Rain*, *WetGrass*]

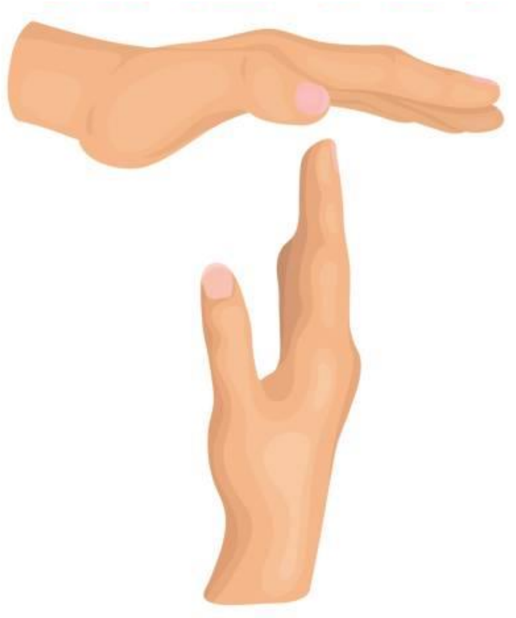
- $w \leftarrow 1$
- *Cloudy* is evidence, therefore $w = w \times P(c) = 1 \times 0.5 = 0.5$
- *Sprinkler* is not evidence, therefore sample from $P(S \mid c)$ − assume *false*
- *Rain* is not evidence, therefore sample from $P(R \mid c)$ − assume *true*
- *WetGrass* is evidence, therefore $w = w \times P(g \mid \neg s, r) = 0.5 \times 0.9 = 0.45$
- Return event $[true, false, true, true]$ with weight $w = 0.45$

Repeating that 10 times:

$$
\begin{aligned}
P(r|c,g) &= \frac{\sum_r w}{\sum w} \\
&= \frac{0.45 + 0.475 + 0.45 + 0.45 + 0.45 + 0.475 + 0.45 + 0.45}{0.45 + 0.475 + 0.05 + 0.45 + 0.45 + 0.45 + 0.475 + 0.45 + 0.05 + 0.45} \\
&= 0.9733
\end{aligned}
$$

$$
\begin{aligned}
P(\neg r|c,g) &= \frac{\sum_{\neg r} w}{\sum w} \\
&= \frac{0.05 + 0.05}{0.45 + 0.475 + 0.05 + 0.45 + 0.45 + 0.45 + 0.475 + 0.45 + 0.05 + 0.45} \\
&= 0.0267
\end{aligned}
$$

Break



Likelihood Weighting: Algorithm

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($\mathbf{W}[X]$)

Adds in a new weight for a value of X.

Likelihood Weighting: Algorithm

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

Generates a new sample.

Today

- Inference by stochastic simulation
- Approximate inference in Bayesian nets (BNs)
  - Rejection Sampling
  - Likelihood Weighting
  - **Gibbs Sampling**
- Data discretisation for learning discrete BNs
- Questions and answers related to assignment

Acknowledgement: Content of slides adapted from Prof. Simon Parsons.

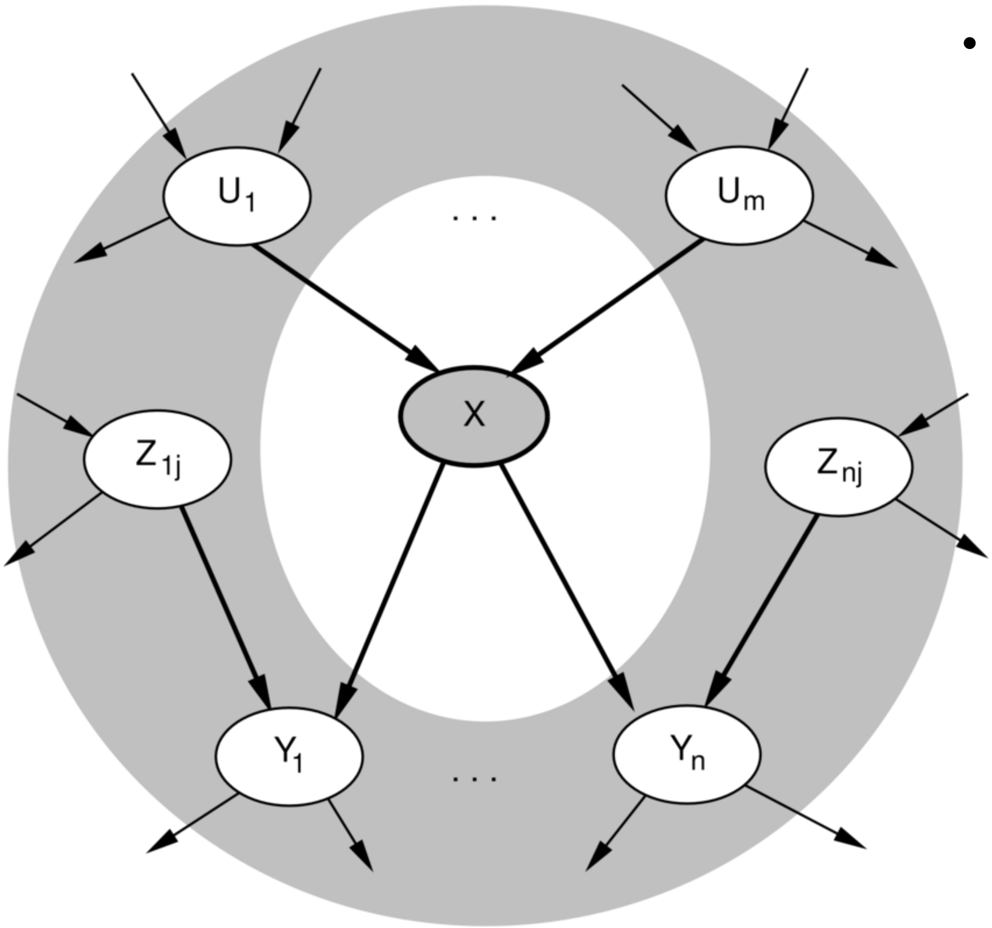
Gibbs Sampling

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

Gibbs Sampling

- Gibbs sampling starts with an arbitrary state.
- So, pick state with evidence variables fixed at observed values (if *Cloudy*=*true*, we pick that).
- Generate next state by randomly sampling from a non-evidence variable – but 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 evidence variables fixed”.

Markov Blanket



- The Markov blanket of X is:
  - Its parents (U_i);
  - Its children (Y_i); and
  - Its children's (other) parents (Z_i).

Gibbs Sampling: Example

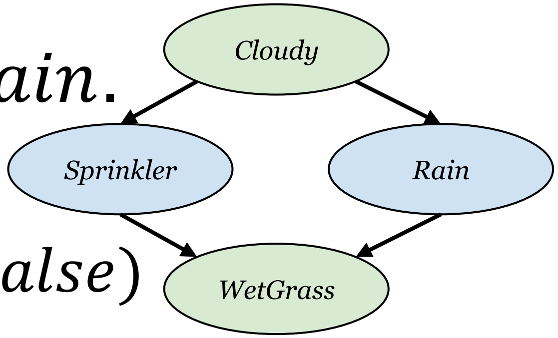
- Start with a query:
  - $P(Rain|Sprinkler = true, WetGrass = true)$
- Our initial state takes values of evidence variables:
  - $[Cloudy =?, Sprinkler = true, Rain =?, WetGrass = true]$

  and the other two are initialised randomly, e.g.:
  - $[Cloudy = true, Sprinkler = true, Rain = false, WetGrass = true]$
- Now we can start the main sampling process.

Gibbs Sampling: Example

- First we sample $Cloudy$ given the current state of its Markov blanket.

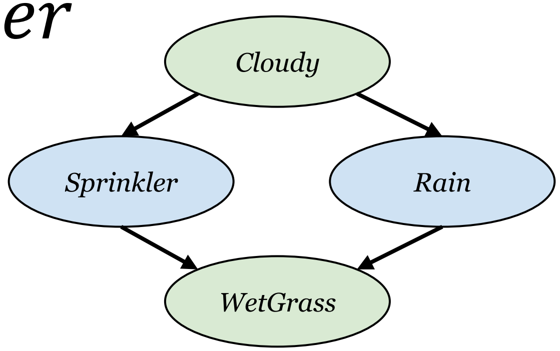
- Markov blanket is $Sprinkler$ and $Rain$.
- So, sample from:
  - $P(Cloudy|Sprinkler = true, Rain = false)$



- Suppose we get $Cloudy = false$, then the new state is:
  - $[Cloudy = false, Sprinkler = true, Rain = false, WetGrass = true]$

Gibbs Sampling: Example

- Next we sample $Rain$ given its Markov blanket.
- Markov blanket is $Cloudy$, $Sprinkler$ and $WetGrass$
- So, sample from:
  - $P(Rain|Cloudy = false, Sprinkler = true, WetGrass = false)$



- Suppose we get $Rain = true$, then new state is:
  - $[Cloudy = false, Sprinkler = true, Rain = true, WetGrass = true]$
- Then repeat the sampling.

Gibbs Sampling: Example

- Each state visited during this process contributes to our estimate for:
  - $P(Cloudy|Sprinkler = true, WetGrass = true)$
- Say the process visits 80 states:
  - In 20, $Cloudy = true$
  - In 60, $Cloudy = false$
- Then
  - $P(Cloudy|Sprinkler = true, WetGrass = true)$
  - $= Normalise(< 20,60 >)$
  - $=< 0.25,0.75 >$

Gibbs Sampling: Algorithm

function GIBBS-ASK(X, **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 **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: Markov Blanket

- 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:

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

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

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

Gibbs Sampling: Markov Blanket

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

- For the sprinker example, consider we want to compute:
  - $P(Cloudy|Sprinkler = true, Rain = false)$

- Then we need:

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

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

Gibbs Sampling: Markov Blanket

- Which gives us:

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

$$= \alpha P(C)P(s|C)P(\neg r|C)$$

$$= \alpha < 0.5, 0.5 >< 0.1, 0.5 >< 0.2, 0.8 >$$

$$= \alpha < 0.5 * 0.1 * 0.2, 0.5 * 0.5 * 0.8 >$$

$$= \alpha < 0.01, 0.2 >$$

$$=< 0.048, 0.952 >$$

- And then we sample as normal.

Today

- Inference by stochastic simulation

- Approximate inference in Bayesian nets (BNs)
  - Rejection Sampling
  - Likelihood Weighting
  - Gibbs Sampling

- **Data discretisation for learning discrete BNs**

- Questions and answers related to assignment

How to train discrete Bayes Nets with Continuous Random Variables?

1. Manual discretisation of continuous data—to be able to use the methods discussed so far.
   – Age (years/days/minutes): young, middle aged, old
   – Temperature: very cold, cold, mild, hot, very hot

2. Split continuous data into uniform intervals (e.g., 1, 2, 3,… intervals between $\min$ & $\max$).

3. Jointly discretising data and performing structure learning (*bnlearn* implements this*).

* Chen, Y., et al. "Learning Discrete Bayesian Networks from Continuous Data", JAIR, 2017

API for Data Discretisation

bnlearn.discretize(data, edges, continuous_columns, max_iterations, verbose**)**

Example code snippet (using cardiovascular_data):

```
data = pd.read_csv(TRAINING_DATA, encoding='latin')
edges = [('target', 'ï»¿age'),('target', 'gender'),('target', 'height'),('target', 'weight'),('target', 'ap_hi'),('target', 'ap_lo'),('target', 'cholesterol'),('target', 'gluc'),('target', 'smoke'),('target', 'alco'),('target', 'active')]
continuous_columns = ["ï»¿age", "height", "weight", "ap_hi", "ap_lo"]
df_discrete = bn.discretize(data, edges, continuous_columns, max_iterations=1, verbose=3)
```

Data Discretisation: Example Output

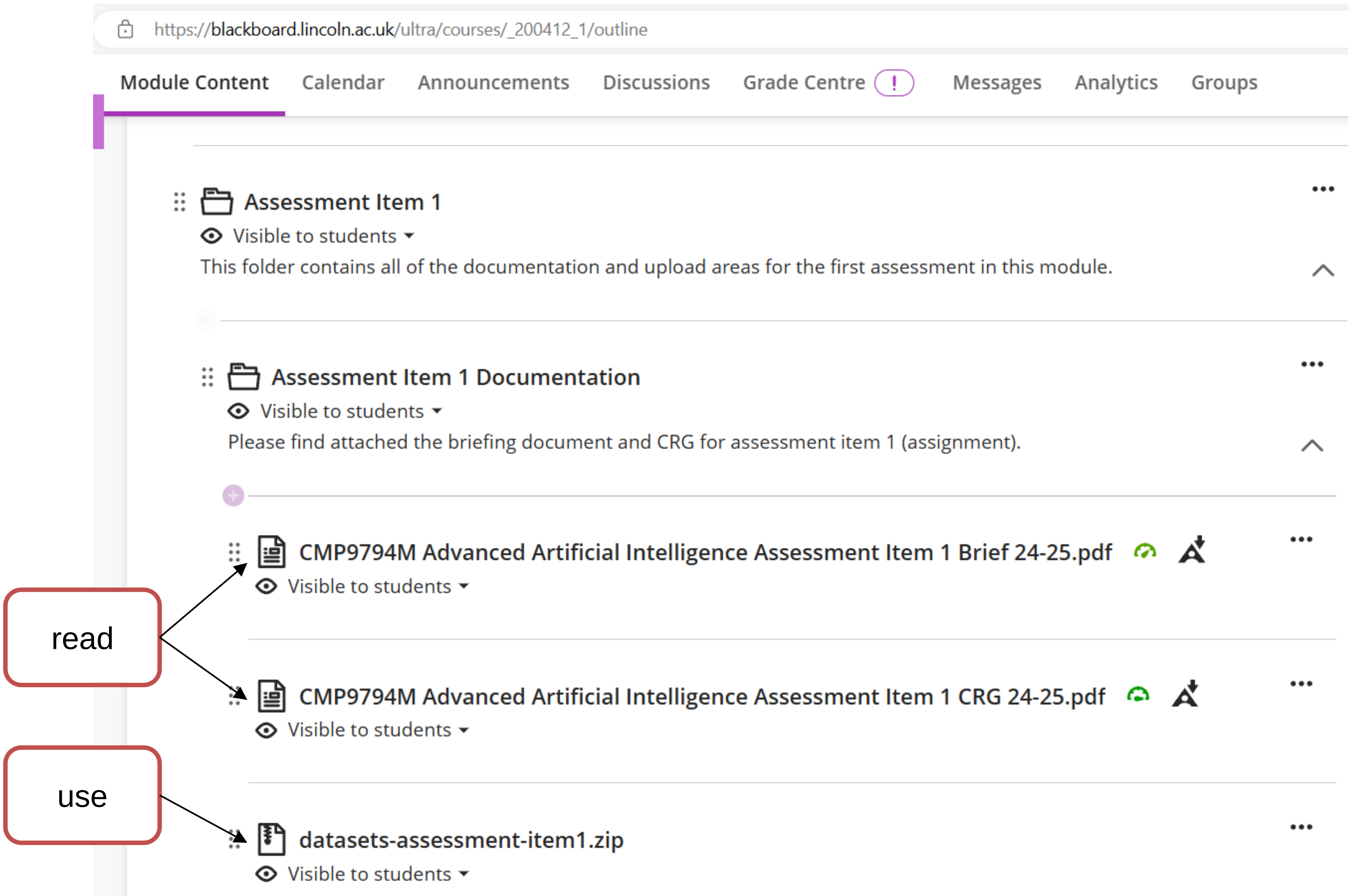
```
(env_bnlearn) C:\Lincoln\slides\CMP9794M-2024-25\aai-workshop-w4>python bnlearn_DataDiscretisation.py
DATA:
      ï»¿age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke  alco  active  target
0     17623       2     169      82    150    100            1     1      0     0       1       1
1     17474       1     156      56    100     60            1     1      0     0       0       0
2     14791       2     165      60    120     80            1     1      0     0       0       0
3     17482       1     154      68    100     70            1     1      0     0       0       0
4     21413       1     157      69    130     80            1     1      0     0       1       0
..      ...     ...     ...     ...    ...    ...          ...   ...    ...   ...     ...     ...
495   21111       1     160      71    140     90            1     2      0     0       1       0
496   19655       1     168      82    120     90            1     1      0     0       1       0
497   19594       2     174      54    120     80            2     2      1     0       1       0
498   19711       1     162     100    120     80            3     1      0     0       1       1
499   21242       2     168      79    130     90            1     1      0     0       1       0

[500 rows x 12 columns]
[bnlearn] >Discretizer for continuous values. Iteration [0].
                 ï»¿age  gender           height          weight                            ap_hi           ap_lo  cholest
0    (14213.54, 19994.0]       2  (140.45, 190.0]  (79.5, 151.5]                    (138.5, 563.0]  (88.5, 575.0]
1    (14213.54, 19994.0]       1  (140.45, 190.0]  (41.74, 79.5]  (2.0499999999999, 138.5]  (-11.0, 88.5]
2    (14213.54, 19994.0]       2  (140.45, 190.0]  (41.74, 79.5]  (2.0499999999999, 138.5]  (-11.0, 88.5]
3    (14213.54, 19994.0]       1  (140.45, 190.0]  (41.74, 79.5]  (2.0499999999999, 138.5]  (-11.0, 88.5]
4     (19994.0, 23653.0]       1  (140.45, 190.0]  (41.74, 79.5]  (2.0499999999999, 138.5]  (-11.0, 88.5]
..                   ...     ...              ...            ...                       ...            ...
495   (19994.0, 23653.0]       1  (140.45, 190.0]  (41.74, 79.5]            (138.5, 563.0]  (88.5, 575.0]
496  (14213.54, 19994.0]       1  (140.45, 190.0]  (79.5, 151.5]  (2.0499999999999, 138.5]  (88.5, 575.0]
497  (14213.54, 19994.0]       2  (140.45, 190.0]  (41.74, 79.5]  (2.0499999999999, 138.5]  (-11.0, 88.5]
498  (14213.54, 19994.0]       1  (140.45, 190.0]  (79.5, 151.5]  (2.0499999999999, 138.5]  (-11.0, 88.5]
499   (19994.0, 23653.0]       2  (140.45, 190.0]  (41.74, 79.5]  (2.0499999999999, 138.5]  (88.5, 575.0]
```

Those outputs can be used by `CPT_Generator.py`, `BayesNetInference.py`, and `ModelEvaluator.py`

Today

- Inference by stochastic simulation
- Approximate inference in Bayesian nets (BNs)
  - Rejection Sampling
  - Likelihood Weighting
  - Gibbs Sampling
- Data discretisation for learning discrete BNs
- **Questions and answers related to assignment**

Assessment Item 1: Assignment



First 4 Weeks of this Semester

We have asked and answered the following when training and evaluating Bayesian Networks (BNs):

1. Where do the probabilities come from?

   A: Usually from training data (MLE, not the only one)

2. Where does the structure come from?

   A: Prior knowledge or structure learning (preferred)

3. How to do probabilistic inference?

   A: Inference by enumeration, variable elimination, rejection sampling, likelihood weighting, Gibbs sampling

4. What code to use for training/evaluating BNs?

   A: Module's code and bnlearn code – you've got both.

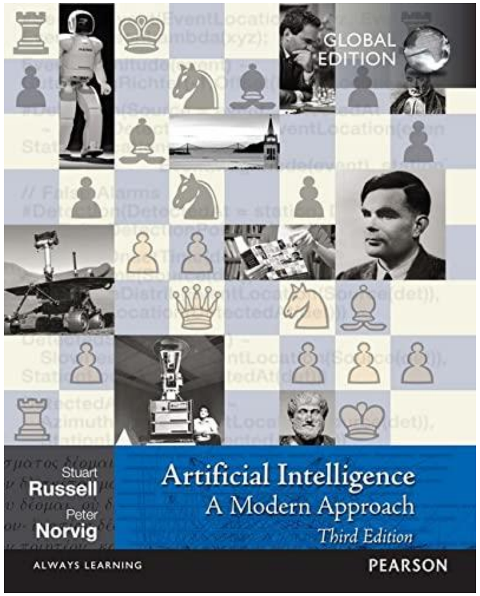
Today

- Approximate inference should be used if exact inference becomes unfeasible/impractical.

- We looked at algorithms for approximate inference in Bayes nets—based on sampling.
  - Rejection sampling
  - Likelihood weighting
  - Gibbs sampling

Readings:

Russell & Norvig 2016. [Chapter 14.5]



This and Next Week

Workshop (today):
Code for approximate probabilistic inference via Rejection Sampling, Gibbs Sampling, etc.

Lecture (next week):
Gaussian Bayesian Networks
Reading 1: Koller & Friedman 2009. Section 7.2
Reading 2: Bishop. C. 2006. Section 8.1.4

Any other questions?