Goals and Preferences

Alice . . . went on "Would you please tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
"I don't much care where —" said Alice.
"Then it doesn't matter which way you go," said the Cat.

Lewis Carroll, 1832–1898
Alice's Adventures in Wonderland, 1865
Chapter 6

Review

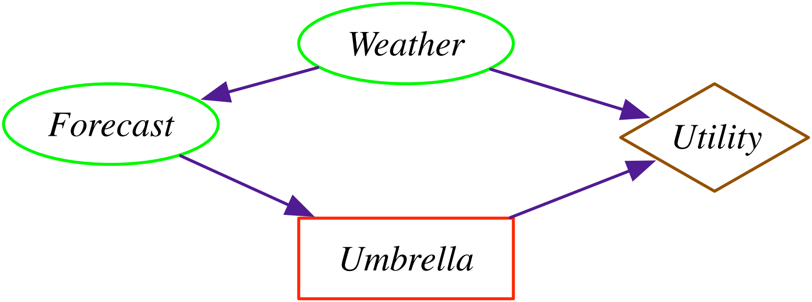
Decision network:

- DAG with three sorts of nodes: decision (rectangle), random (ellipse), utility (diamond)
- Domain for each decision and random node (no domain for utility)
- Factor for each random node and the utility (no initial factor for decision nodes)
- A decision function maps assignments to the parents of a decision node to a value in the domain of the decision node.
- A policy assigns a decision function to each decision node.

Variable Elimination to Find an Optimal Policy

- Create a factor for each conditional probability table and a factor for the utility.
- Repeat:
  - Sum out random variables that are not parents of a remaining decision node.
  - Select the decision variable D that is only in a factor f with (some of) its parents.
  - Eliminate D by maximizing. This returns:
    - an optimal decision function for D: $\arg\max_D f$
    - a new factor: $\max_D f$
- until there are no more decision nodes.
- Sum out the remaining random variables. Multiply the factors: this is the expected utility of an optimal policy.

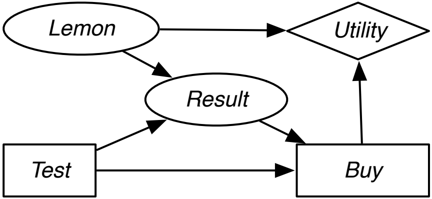
Umbrella Decision Network



What happens if we add an arc from *Weather* to *Umbrella*?

Clicker Question

For the decision network:



Which random variable/s is/are eliminated (summed out) initially (before first maximization)

A *Lemon*, *Result*

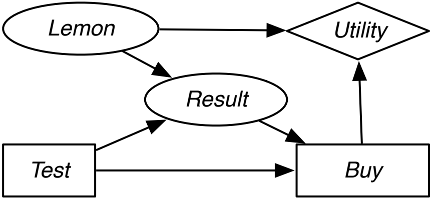
B *Result*

C *Lemon*

D *Utility*, *Buy*, *Test*

E No random variable is summed

Clicker Question



When eliminating *Lemon*, what factors are multiplied?

A $P(Lemon)$, $P(Result \mid Lemon, Test)$

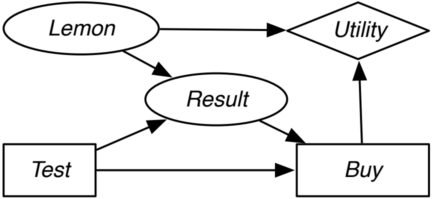
B $P(Result \mid Lemon, Test)$, $u(Lemon, Buy)$

C $P(Lemon)$, $P(Result \mid Lemon, Test)$, $u(Lemon, Buy)$

D $P(Lemon)$, $P(Result \mid Lemon, Test)$, $u(Lemon, Buy)$, $P(Test)$

E No factors are multiplied

Clicker Question



After eliminating *Lemon* a factor on what variable(s) is created?

A *Result*

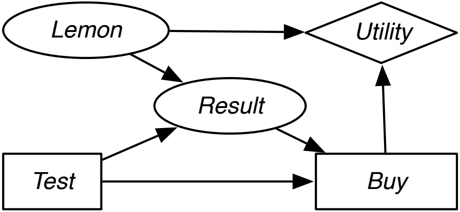
B *Result*, *Buy*

C *Result*, *Buy*, *Test*

D *Utility*, *Result*, *Buy*, *Test*

E No factor is created

Clicker Question



Which of the decision variables is eliminated first

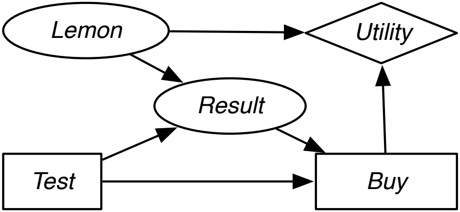
A *Buy*

B *Test*

C Neither

D Both

Clicker Question



After eliminating *Buy* (by maximization), a factor on what variable(s) remains

A *Buy*

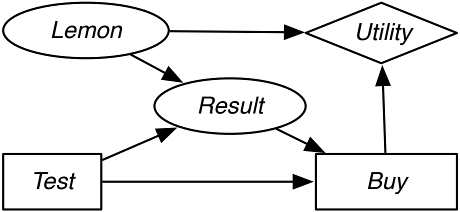
B *Test*

C *Result*

D *Test*, *Result*

E no factors remain

Clicker Question



What is eliminated next after eliminating *Buy*

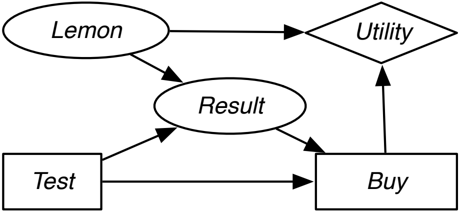
A *Result* is summed out

B *Test* is maximized

C Either *Result* or *Test*, it doesn't matter

D There is nothing more to eliminate

Clicker Question



What does a policy specify?

A just a value for *Test*

B just a value for *Buy*

C a value for *Test* and a value for *Buy*

D a value for *Test* and a value for *Buy* for each value of *Test* and *Result*

E just a value for *Buy* for each value of *Test* and *Result*

Learning Objectives

At the end of the class you should be able to:

- predict which of rejection sampling, importance sampling and particle filtering work best for a problem.

Stochastic Simulation

- Idea: probabilities $\leftrightarrow$ samples
- Get probabilities from samples:

X	$count$
x_1	n_1
$\vdots$	$\vdots$
x_k	n_k
$total$	m

$\leftrightarrow$

X	$probability$
x_1	n_1/m
$\vdots$	$\vdots$
x_k	n_k/m

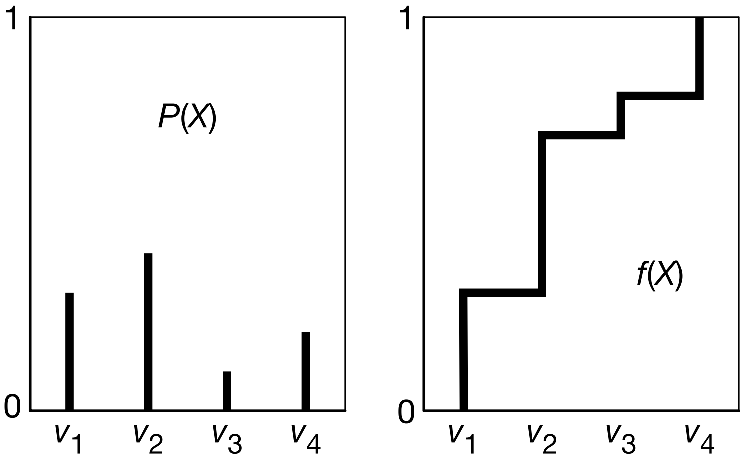
- If we could sample from a variable's (posterior) probability, we could estimate its (posterior) probability.

Generating samples from a distribution

For a variable X with a discrete domain or a (one-dimensional) real domain:

- Totally order the values of the domain of X.
- Generate the cumulative probability distribution: $f(x) = P(X \leq x)$.
- Select a value y uniformly in the range $[0, 1]$.
- Select the x such that $f(x) = y$.

Cumulative Distribution



Hoeffding's inequality

Theorem (Hoeffding): Suppose p is the true probability, and s is the sample average from n independent samples; then

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}.$$

Guarantees a probably approximately correct estimate of probability.
If you are willing to have an error greater than ϵ in less than δ of the cases, solve $2e^{-2n\epsilon^2} < \delta$ for n, which gives

$$n > \frac{-\ln \frac{\delta}{2}}{2\epsilon^2}.$$

ϵ	δ	n
0.1	0.05	185
0.01	0.05	18,445
0.1	0.01	265

Forward sampling in a belief network

- Sample the variables one at a time; sample parents of X before sampling X.
- Given values for the parents of X, sample from the probability of X given its parents.

Clicker Question

Suppose we have a belief network where A is the only parent of B, with the following probabilities specified for the belief network:

$$P(a) = 0.6$$
$$P(b \mid a) = 0.3$$
$$P(b \mid \neg a) = 0.8$$

Suppose a sample has $A = true$ (which is written as a), which of the following is true in that sample:

- A B should be assigned *true* with probability 0.5
- B B should be assigned *true* with probability 0.3
- C B should be assigned *true* with probability $0.6 * 0.3 + (1 - 0.6) * 0.8$
- D B should be assigned *true* with probability 0.8
- E None of the above

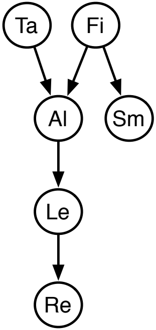
Rejection Sampling

- To estimate a posterior probability given evidence $Y_1{=}v_1 \wedge \ldots \wedge Y_j{=}v_j$:
- Reject any sample that assigns Y_i to a value other than v_i.
- The non-rejected samples are distributed according to the posterior probability:

$$P(\alpha \mid e) \approx \frac{\sum_{s:\alpha \wedge e \text{ is true in s}} 1}{\sum_{s:e \text{ is true in s}} 1}$$

where we are summing over the samples s that are consistent with the evidence e.

Rejection Sampling Example: $P(ta \mid sm, re)$



Observe $Sm = true, Re = true$

	Ta	Fi	Al	Sm	Le	Re	
s_1	false	true	false	true	false	false	✘
s_2	false	true	true	true	true	true	✔
s_3	true	false	true	false	—	—	✘
s_4	true	true	true	true	true	true	✔
...							
s_{1000}	false	false	false	false	—	—	✘

$P(sm) = 0.02$
$P(re \mid sm) = 0.32$
There are 1000 samples.
How many are used (not rejected), on average?

Doesn't work well when evidence is unlikely.

Importance Sampling

- Samples have weights: a real number associated with each sample that takes the evidence into account.
- Probability of a proposition is weighted average of samples:

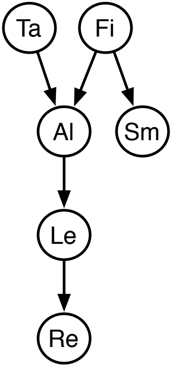
$$P(\alpha \mid evidence) \approx \frac{\sum_{sample:\alpha \text{ is true in sample}} weight(sample)}{\sum_{sample} weight(sample)}$$

- Mix exact inference with sampling: don't sample all of the variables, but weight each sample according to $P(evidence \mid sample)$.

Importance Sampling (Likelihood Weighting)

```
procedure likelihood_weighting(Bn, e, H, n):
    # Approximate P(H | e) in belief network Bn using n samples.
    # H has domain {0, 1}
    mass := 0
    hmass := 0
    repeat n times:
        weight := 1
        for each variable X_i in order:
            if X_i = o_i is observed
                weight := weight × P(X_i = o_i | parents(X_i))
            else assign X_i a random sample of P(X_i | parents(X_i))
        mass := mass + weight
        hmass := hmass + weight ∗ (value of H in current assignment)
    return hmass/mass
```

Importance Sampling Example: $P(ta \mid sm, re)$



	Ta	Fi	Al	Le	Weight
s_1	true	false	true	false	0.01×0.01
s_2	false	true	false	false	0.9×0.01
s_3	false	true	true	true	0.9×0.75
s_4	true	true	true	true	0.9×0.75
...					
s_{1000}	false	false	true	true	0.01×0.75

$P(sm \mid fi) = 0.9$
$P(sm \mid \neg fi) = 0.01$
$P(re \mid le) = 0.75$
$P(re \mid \neg le) = 0.01$

Particle Filtering

Importance sampling can be seen as:

> *for each particle:*
>
> > *for each variable:*
> >
> > > *sample / absorb evidence / update query*

where particle is one of the samples.
Instead we could do:

> *for each variable:*
>
> > *for each particle:*
> >
> > > *sample / absorb evidence / update query*

Why?

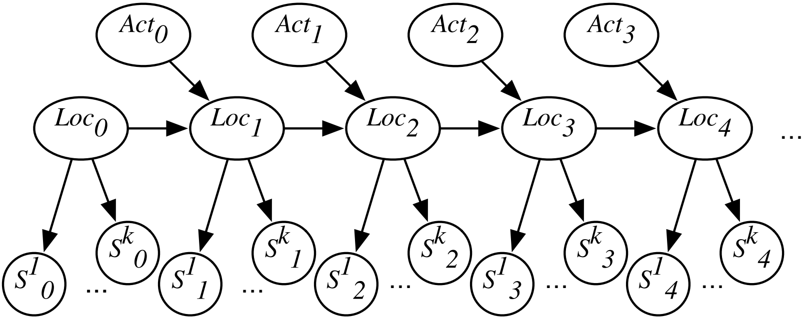
- We can have a new operation of resampling
- It works with infinitely many variables (e.g., HMM)

Particle Filtering for HMMs

- Start with random chosen particles (say 1000)
- Each particle represents a history.
- Initially, sample states in proportion to their probability.
- Repeat:
  - Absorb evidence: weight each particle by the probability of the evidence given the state of the particle.
  - Resample: select each particle at random, in proportion to the weight of the particle.
    Some particles may be duplicated, some may be removed. All new particles have same weight.
  - Transition: sample the next state for each particle according to the transition probabilities.

To answer a query about the current state, use the set of particles as data.

Example: Localization (revisited)



Loc consists of (x, y, θ) – position and orientation

$k = 24$ sonar sensors (all very noisy)

See Sebastian Thrun's video on Monte Carlo Localization.

sca80a0.avi from `http://robots.stanford.edu/videos.html`