Chap. 16

decision-theoretic agent = utility theory + prob. theory
continuous measure of outcome quality

* maximize expected utility

agent may not know current state

RESULT(a) ≡ random variable
values: possible outcome states

$P(\text{RESULT}(a) = s' \mid a, \bar{e}) \equiv$ prob of outcome s' given $\bar{e}$

$= \sum_s P(\text{RESULT}(s,a) = s' \mid a) P(s_0 = s \mid \bar{e})$

utility function $U(s)$ desirability of state

expected utility $EU(a \mid \bar{e}) = \sum_{s'} P(\text{RESULT}(a) = s' \mid a, e) U(s')$
eqn (16.1)

principle of maximum expected utility (MEU)

action $= \underset{a}{\operatorname{argmax}}\ EU(a \mid \bar{e})$

if utility captures performance measure

then agent performs well

constraints on rational preferences

$A \succ B$ prefers A over B
$A \sim B$ indifferent between A + B
$A \succsim B$ indifferent or prefers A over B

consider set of outcomes for each action as a lottery
L: outcomes $S_1, \ldots S_n$ with probs $p_1, \ldots p_n$

$$L = [p_1, S_1; p_2, S_2; \ldots p_n, S_n]$$

goal: understand how preferences between lotteries are related to " " underlying states

constraints of preference relation (axioms of utility theory)

- Orderability $(A \succ B) \vee (B \succ A) \vee (A \sim B)$
- Transitivity $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$
- Continuity $A \succ B \succ C \Rightarrow \exists p \; [p, A; 1-p, C] \sim B$
- Substitutability $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$
- Monotonicity $A \succ B \Rightarrow [p > q \Leftrightarrow [p, A; 1-p, B] \succ [q, A; 1-q, B]]$
- Decomposability $[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$

if violated, then agent not rational.

Preferences to Utility

- **Existence of Utility Function** (not unique)

If agent's preferences obey axioms of utility, then
$\exists\, U \ni U(A) > U(B)$ iff $A \succ B$
$U(A) = U(B)$ iff $A \sim B$

- **Expected Utility of a lottery**

the utility of a lottery is the sum of the prob. of each outcome times the utility of that outcome

$$U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i U(S_i)$$

value function or ordinal utility function : preference ranking on states

utility : lotteries $\rightarrow \mathbb{R}$

preference elicitation : determine agent's utility function

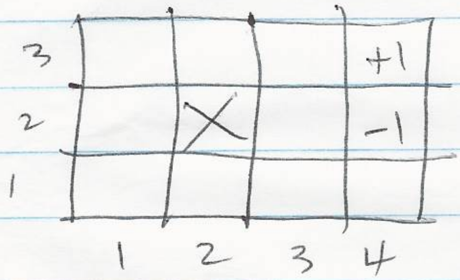
normalized scale $U(S) = u_T$ top, best $= 1$
$U(S) = u_\perp$ bottom, worst $= 0$

assess utility of state S : prob. $p \ni$ agent is indifferent in choice between S & standard lottery $[p, u_T; (1-p)\, u_\perp]$

Chapter 17

sequential decision making in stochastic environment

- utilities
- uncertainty
- sensing
-- search + planning



to make things easier ↓

(1,1) start state
(4,2) (4,3) final states
} fully observable

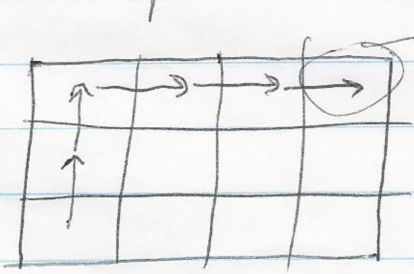
Actions(s) = A(s) = {Up, Down, Left, Right}

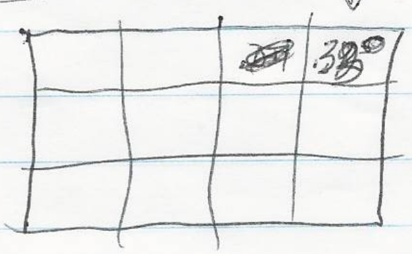
$R((4,2)) = -1$ $R((4,3)) = 1$ $R(s) = -0.04$ $s \neq (4,2) \vee (4,3)$

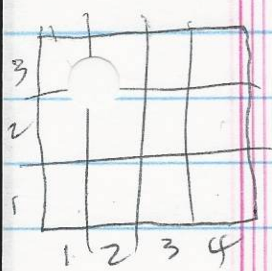
Nondeterministic action result

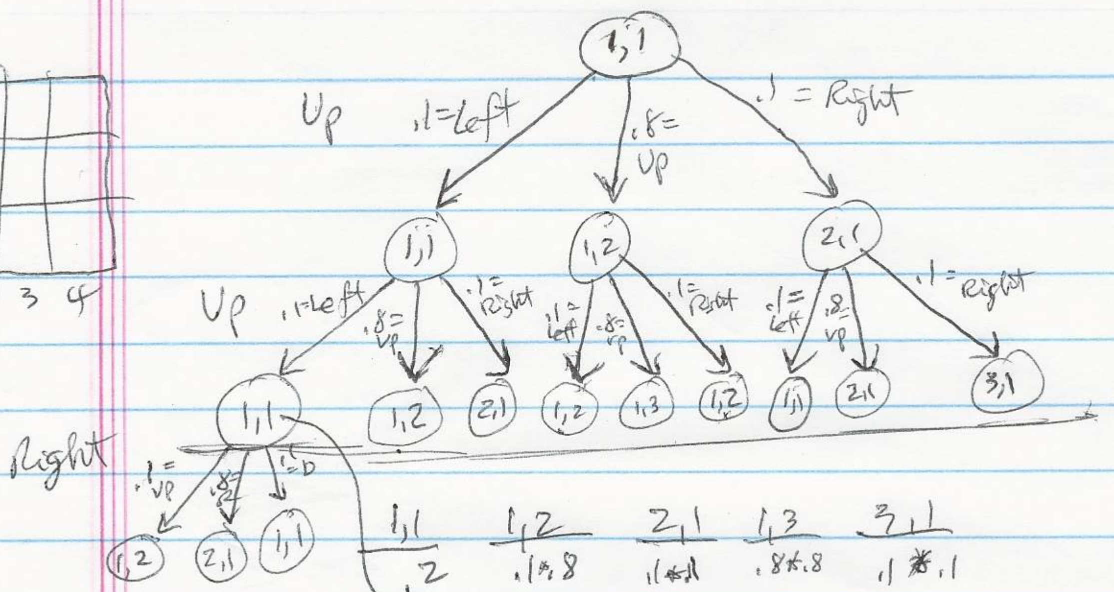
\+ .8 prob direction of selected action
.1 prob ⊥ orthog " " " "
.1 prob other " " " "

$.8^5 = .3277$









.08
.08
.08

1,1	1,2	2,1	1,3	3,1
2	.1*.8	.1*.1	.8*.8	.1*.1
.1	.8*.1	.1*.8		
	.8*.1			
0.02	0.24	0.09	0.64	0.01
0.02	0.24	0.09	0.64	0.01

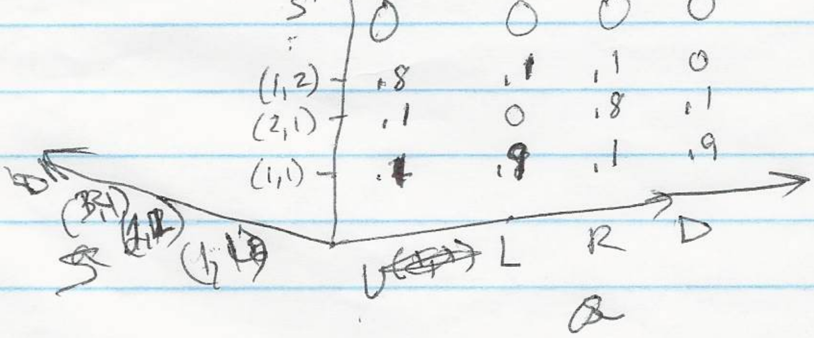
= Σ = 1

64
24
9
1
2
0

0.64			
0.24			
0.02	0.09	0.01	

Transition model : outcome of each action in each state

P(s' | s, a) (Markovian : only depends on s)



utility function

in each state s
agent receives a reward $R(s)$ (pos or neg)

e.g., -0.04 in all states except terminal
$R((4,3)) = +1$ $\quad$ $R((4,2)) = -1$

utility of environment history is sum of rewards

Markov Decision Process (MDP):

a sequential decision problem
- fully observable
stochastic environment
Markovian transition model
additive rewards

consists of:

set of states ($s_0 \equiv$ initial state)
set of actions $ACTIONS(s)$
transition model $P(s' \mid s, a)$
reward function $R(s)$ [could be $R(s, a, s')$]

<u>solution</u> : specify what agent should do in any state
called a <u>policy</u> π

action from π is $\pi(s)$
<u>complete policy</u> : knows what to do in any state

<u>policy quality</u> : measured by <u>expected utility</u>

<u>optimal policy</u> (π^*) : highest expected utility

→	→	→	+1
↑	X	↑	−1
↑	↑	←	←

optimal policy
(for $R(s) = -0.04$)

(Note action in (4,1) has 0.1 prob of going to −1
" " " " (3,2) " " " " " "

Is that best? Yes, for balance of risk + reward!

if $R(s) \leq -1.628$ then go to nearest exit
⋮
> 0 stay on board

Need an algorithm!

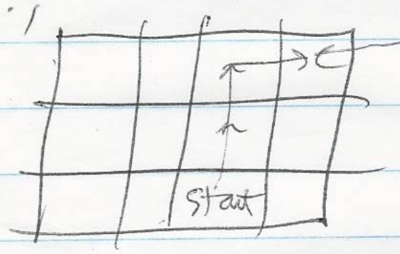
Finite horizon or infinite?

game over in fixed time, N

$$U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$$

history

E.g.,



any policy + $N = 3$

$N = 100 \Rightarrow$ go around

nonstationary: policy may change given more or less finite time

stationary: no fixed deadline

Assume: preferences between state sequences are stationary

if 2 state sequences $[s_0, s_1, \ldots]$, $[s_0, s_1', \ldots]$ have same first state, then their preference is same as $[s_1, s_2, \ldots]$ & $[s_1', s_2', \ldots]$

Then $\exists$ 2 ways to assign utilities to sequences:

1. Additive Rewards $U_h([s_0, s_1, \ldots]) = R(s_0) + R(s_1) + \ldots$

2. Discounted Rewards: $U_h([s_0, s_1, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$

where $0 < \gamma < 1$ (discount factor) if can have infinite sequence, then undiscounted rewards will be infinite.

To avoid:

1. Discounted:

$$U_h([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{(1-\gamma)}$$

2. If guaranteed to go to terminal state, then ok
3. average reward per time step

We use 1.

Compare policies by comparing expected utilities.

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]$$

$$\pi_s^* = \underset{\pi}{\operatorname{argmax}}\ U^{\pi}(s)$$ is a policy for every state independent of start state

utility of a state is: $U^{\pi^*}(s) \overset{\text{write as}}{\equiv} U(s)$

0.812	0.868	0.918	+1
0.762	X	0.660	-1
0.705	0.655	0.611	0.388

$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)\, U(s')$$

11|7

Value Iteration

Bellman equation for utilities

$$U(s) = R(s) + \gamma \underbrace{\max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)\, U(s')}_{\text{discounted utility of next state}}$$

9	10	11	12
5	6	7	8
1,1	2	3	4

use known utilities ~~to~~
show Up is best action

$$\overset{=1}{\downarrow}\ 0.7456 =$$

$$U(1,1) = -0.04 + \gamma \max \Big[0.8\, \overset{0.762}{U(1,2)} + 0.1\, \overset{0.655}{U(2,1)} + 0.1\, \overset{0.705}{U(1,1)}, \quad \text{Up}$$

$$0.7107 \quad 0.9\, \overset{0.705}{U(1,1)} + 0.1\, \overset{0.762}{U(1,2)}, \quad \text{Left}$$

$$0.700 \quad 0.9\, \overset{0.705}{U(1,1)} + 0.1\, \overset{0.655}{U(2,1)}, \quad \text{Down}$$

$$0.6707 \quad 0.8\, \overset{0.655}{U(2,1)} + 0.1\, \overset{0.762}{U(1,2)} + 0.1\, \overset{0.705}{U(1,1)} \Big] \quad \text{Right}$$

Algorithm

n equations + n unknowns
non linear equations (max)

Use iterative method (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a)\, U_i(s')$$

17|8

function Value-Iteration(mdp, ϵ) returns a utility function

inputs: mdp : S, A, P, R, γ

ϵ max error

local vars U, U' initially 0

δ max change in utility of any state in iteration

repeat

$U \leftarrow U'$; $\delta \leftarrow 0$

for each state s in S do

$$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$$

if $|U'[s] - U[s]| > \delta$

then $\delta \leftarrow |U'[s] - U[s]|$

until $\delta < \epsilon(1-\gamma)/\gamma$

return U

[S, A, R, P, U, Ut] = CS6380_run_value_iteration(0.999999, 3 50);

→	→	→	↑
↑	X	↑	↑
↑	←	←	← ↙

initialized to Up & don't change

p = CS6380_MDP_policy(S, A, P, U)

<u>World</u>

	1	2
1		1

$S = \{[1;1], [2;1]\} \equiv \{1, 2\}$

$A = \{\text{Up, Left, Down, Right}\} \equiv \{1, 2, 3, 4\}$ order matters!

$P =$ standard .8, .1, .1, 0

$\gamma = 0.999999$

$R = -0.04$ at $[1;1]$

$\varepsilon = 0.1$

$$
P \equiv \begin{array}{c} \\ A \end{array}
\begin{array}{cc} & s'=1 \\ \begin{array}{c} U\ 1 \\ L\ 2 \\ D\ 3 \\ R\ 4 \end{array} & \begin{bmatrix} .9 & .1 \\ 1 & .8 \\ .9 & .1 \\ .2 & 0 \end{bmatrix} \\ & \begin{array}{cc} 1 & 2 \end{array} \\ & S \end{array}
\quad
\begin{array}{cc} & s'=2 \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{bmatrix} .1 & .9 \\ 0 & .2 \\ .1 & .9 \\ .8 & 1 \end{bmatrix} \\ & \begin{array}{cc} 1 & 2 \end{array} \\ & S \end{array}
$$

$$
P \equiv \begin{array}{cc} & \begin{array}{c} A \\ U\ \ L\ \ D\ \ R \end{array} \\ \begin{array}{c} s\ 1 \\ 2 \end{array} & \begin{bmatrix} .9 & 1 & .9 & .2 \\ .1 & .8 & .1 & 0 \end{bmatrix} \\ & s'=1 \end{array}
\begin{array}{c} \\ \begin{bmatrix} .1 & 0 & .1 & .8 \\ .9 & .2 & .9 & 1 \end{bmatrix} \\ s'=2 \end{array}
$$

$U \leftarrow 0$

$U' \leftarrow 0$

$\delta \leftarrow 0$

<u>repeat</u>

$U \leftarrow U'$

$\delta \leftarrow 0$

~~* s loop~~

$s = 1$

17/8a 2

$$U'(1) = -0.04 + \underbrace{0.999999 \max_{a \in A(s)} \sum_{s'} P(s'|s,a)\,U(s')}$$

* a loop

a = 1

* s' loop

$s' = 1$

$v_1 = 0 = \underbrace{P(1|1,1)}_{.9}\,\underbrace{U(1)}_{=0}$

$s' = 2$

$v_2 = .1 = \underbrace{P(2|1,1)}_{=.1}\,\underbrace{U(2)}_{=1}$

end s' loop

a1_val = .1

a = 2

* s' loop

$s' = 1$

$v_1 = P(1|1,2)\,\underbrace{U(1)}_{=0} = 0$

$s' = 2$

$v_2 = \underbrace{P(2|1,2)}_{=0}\,\underbrace{U(2)}_{=1} = 0$

end s' loop

a2_val = 0

a = 3

s' loop

$s' = 1$

$v_1 = P(1|1,3)\,\underbrace{U(1)}_{=0} = 0$

$v_2 = \underbrace{P(2|1,3)}_{=.1}\,\underbrace{U(2)}_{=1} = .1$

a3_val = .1

a = 4

$v_1 = P(1|1,4)\,U(1) = 0$

$v_2 = \underbrace{P(2|1,4)}_{=.8}\,\underbrace{U(1)}_{=1} = .8$

a4_val = .8

17/8a 3

$U'(1) \leftarrow .76$

if $|U'(1) - U(1)| > \delta \equiv \quad 0.76 > 0$

$\delta = 0.76$

end

until $\delta < \epsilon(1-\gamma)/\gamma \qquad 0.76 < 10^{-7}$

loop again

$U(1) \leftarrow 0.76$

~~$\delta \leftarrow 0$ finds $U'(1) = 0.76$~~

~~+ ends~~

$s = 1$

$$U'(1) \leftarrow -0.04 + 0.999999 \underbrace{\max_{a \in A} \sum_{s'} P(s'|s,a)\,U(s')}$$

$a = 1$

$s' = 1$

$V_1 = P(1|1,1) * 0.76 \qquad 0.684$
$\quad .9 * .76$

$s' = 2$

$V_2 = P(2|1,1) * 0.76 \qquad = 0.076 \quad \Big\} \; 0.76$
$\quad .1$

$a = 2$

$s' = 1$

$V_1 = P(1|1,2) * .76 \qquad = 0.76 \quad \Big\} \; 0.76$
$\quad 1 * .76$

$s' = 2$

$V_2 = P(2|1,2) * 0.76 \qquad = 0$
$\quad = 0$

$a = 3$

$s' = 1$

$V_1 = P(1|1,3) * .76 \qquad 0.684 \quad \Big\} \; 0.76$
$\quad .9 * 0.76$

$V_2 = P(2|1,3) * 1 \qquad .076$

$a=4$

$s'=1$

$V_1 = P(1|1,4) * 0.76$ $\quad 0.152$

$.2 * 0.76$

$s'=2$

$V_2 = P(2|1,4) * \cancel{0.76}\ 1 = .8$

$.8$

$\left.\begin{matrix} \\ \\ \end{matrix}\right\} .952$

$U'(1) = -0.04 + 0.952 = 0.912$

⋮

- Policy selection given utilities & transition probabilities

$$\underbrace{\pi^*(s)} = \underbrace{\underset{a \in A(s)}{\operatorname{argmax}}} \left\{ \underbrace{\sum_{s'}} \underbrace{\left[P(s'|s,a)\, U(s') \right]} \right\}$$

pi_star

1 [ ]
⋮
n [ ]

for every state s

best_action = 0
best_val = -Inf

for every action a

a_sum ← 0

for every state s'

add in ← P(s'|s,a) U(s')

a_sum = a_sum +

end

pi_star(s) = ?   best action

17.8b.2

$U = \begin{bmatrix} 0.95 \\ 1.0 \end{bmatrix}$ P as before

s = 1
best-action = 0
best_val = -Inf

<u>a loop</u>

U a = 1
a_sum = 0
<u>s' loop</u>
s' = 1
a_sum = $P(1|1,1)U(1) = .9 * 0.95 = 0.855$
s' = 2
a_sum = $0.855 + P(2|1,1)1 = \overset{0.855+}{.1} * 1 = .1 = 0.955$

L <u>a = 2</u>
$P(1|1,2)U(1) + P(2|1,2) \cdot 1 = 0.095$
$.1 * 0.95 + 0 =$

D <u>a = 3</u>
$P(1|1,3)U(1) + P(2|1,3) \cdot 1 = 0.955$

R <u>a = 4</u>
$P(1|1,4)0.95 + P(2|1,4) \cdot 1$
$.2 * 0.95 + .8 * 1 = 0.99$

17\9

Policy Iteration

given an initial policy π_0
alternate between:

- Policy evaluation:

  given a policy π_i, calculate $U_i = U^{\pi_i}$

- Policy improvement

  calculate a new MEU policy π_{i+1}
  using one step look-ahead based on U_i

until no improvement
There are only finitely many policies for finite state space

Policy evaluation — simpler than standard Bellman because no ~~max~~: given policy fixes action

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) \, U_i(s')$$

linear set of equations

or iterate k times

★ Use this

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum P(s' \mid s, \pi_i(s)) \, U_i(s')$$

17) 10

p. 657

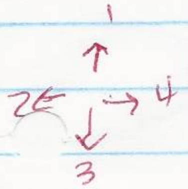
function Policy-iteration (mdp) returns a policy
inputs : mdp: S, A, P
local var's : U utilities, initially 0
π a policy, initially random

repeat

$U \leftarrow$ Policy-evaluation (π, U, mdp)
unchanged $\leftarrow$ true;
for each state s in S do
if $\max_{a \in A(s)} \sum_{s'} P(s' | s, a)\, U[s'] > \sum_{s'} P(s' | s, \pi[s])\, U[s']$

then do
$\pi[s] \leftarrow \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' | s, a)\, U[s']$
unchanged $\leftarrow$ false
endif
end for
until unchanged == true
return π

p_pi = cs6380_MDP_policy_iteration (S, A, P, R, k, γ)



R = R + 10

↓	←	←	1000
↓	X	←	-100
↓	←	←	↓