Last Time:

- ☐ decision-making
- ☐ MDPs

Lecture 3

HRI, Fall '24

Andrea Bajcsy

Today:

- ☐ values & Q-values
- ☐ hands-on exercise

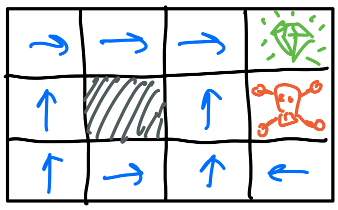
- Today: MDPs
- Thurs: Probability
- Next Tues: Paper Reading (Fundamentals)
- Next Thurs: —"— (Math. Human Models)

RECAP: We left off having defined a Markov Decision Process:

- set of states S
- set of actions A
- transition function $P(s'|s,a)$ $\leftarrow$ {or $T(s',a,s)$}
- reward function $r(s,a)$ // sometimes $r(s)$, $r(s,a,s')$, $r(s,a)$
- discount factor $\gamma \in [0,1]$

MDP quantities we have seen so far:

- policy [π]: choice of action for each state



- cumulative reward: sum of (discounted) rewards

Our goal is to find a policy π that maximizes the discounted sum of rewards:

$$\overset{1}{\gamma^0} r(s^0,a^0) + \gamma^1 r(s^1,a^1) + \gamma^2 r(s^2,a^2) + \dots$$

this trades off short & long-horizon reward

VALUE FUNCTION

lets quantify the best cumulative reward we could ever get starting from a state s. This will precisely be our value function: $V: S \to \mathbb{R}$

Simpler case: Deterministic Transition.

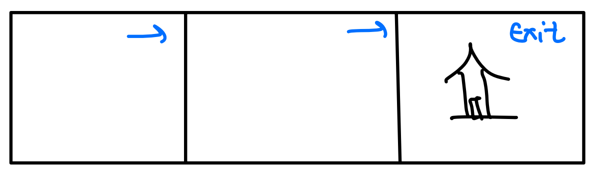
$$V(s) = \underbrace{\max_{\pi}}_{\text{② try every possible policy and get the maximum possible value}} \underbrace{\left[\sum_{t=0}^{\infty} \gamma^t r(s^t, a^t) \right]}_{\text{① discounted sum of rewards}}$$

It turns out that we can rewrite this equation recursively because of Bellman's "principle of optimality." It says that you can decompose a complex problem into smaller sub-problems. Mathematically, it lets us redefine our value:

$$V(s) = \underbrace{\max_{a \in \mathcal{A}} \Big[\underbrace{r(s,a)}_{\text{reward right now!}} + \gamma \cdot V(s') \Big]}_{\text{maximum possible discounted value of next state plus current reward.}}$$

s' is the next state you get to after taking a from state s

ex.



s= 0 1 2

$\mathcal{A} = \{ \rightarrow, \text{Exit} \}$

you can take Exit @ home (s=2)

deterministic $P(s' \mid s, a) = \begin{cases} 1 & \text{if } s' = s + a \\ 0 & \text{else} \end{cases}$

$r(s,a) = 0$ UNLESS you $r(s=2, a=\text{Exit}) = +1$

① $V(s=2) = r(s=2, a=\text{Exit}) = \boxed{+1}$

② $V(s=1) = \underbrace{r(s=1, a=\rightarrow)}_{=0} + \gamma \cdot V(s=2) = \boxed{\gamma}$

③ $V(s=0) = \underbrace{r(s=0, a=\rightarrow)}_{=0} + \gamma \cdot V(s=1) = \boxed{\gamma^2}$

Stochastic Transitions

$$V(s) = \max_{a \in \mathcal{A}} \Big[r(s,a) + \gamma \underbrace{\sum_{s' \in S} P(s'|s,a) V(s')}_{} \Big]$$

when transitions are stochastic, we get expected value across next states we may visit.

$\boxed{\text{Q-function}}$

One more helpful quantity is the Q-function (Q value). You'll see this alot in papers :). Expected value of taking some action a from state s, and then acting optimally from there on.

$\boxed{f(s,a) = s'}$

Determistic: $Q(s,a) = r(s,a) + \gamma \cdot V(s')$ ← taking a

Stochastic: $Q(s,a) = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s')$

Intuition: How good is action a in the short & long-term?

Optimal policy: $\pi(s) = \arg\max_{a \in A} Q(s,a)$

But, we were staring at our value function, we need to deal w/ the recursion!

$$V(s) = \max_{a \in \mathcal{A}} \Big[r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s') \Big]$$

$$V(s') = \max_{a \in \mathcal{A}} \Big[r(s',a) + \gamma \sum_{s'' \in S} P(s''|s',a) V(s'') \Big]$$

recursive ("dynamic programming")

VALUE ITERATION

$V[s] \leftarrow 0$ // current estimate of value (zero $\forall s \in S$!)

$V'[s] \leftarrow 0$ // next / updated estimate of value

$\delta \leftarrow 0$ // maximum change btwn. V and V'

while True:

$V \leftarrow V'$ and $\delta \leftarrow 0$ // we will update V' so remember prior values

for each state $s \in S$:

$$V'[s] \leftarrow \max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V[s'] \right]$$

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

$\delta \leftarrow |V'[s] - V[s]|$

if $\delta \leq \varepsilon$
return $V'[s]$

// stopping condition: ε = max error
if we have less error than ε, return!

Exercise [from CS188 Berkeley]

Pacman is using MDPs to maximize his expected utility. In each environment:

- deterministic transition ↑ ← ↓ →
- Pacman has the standard actions {North, East, South, West} unless blocked by an outer wall
- There is a reward of 1 point when eating the dot (for example, in the grid below, $R(C, South, F) = 1$)
  i.e. transition to state F
- The game ends when the dot is eaten
  i.e. zero reward at F state

(a) Consider a the following grid where there is a single food pellet in the bottom right corner (F). The **discount** factor is 0.5. There is no living reward. The states are simply the grid locations.

i.e. you only get reward at the dot

A	B	C
D	E	F ○

(i) What is the optimal policy for each state?

(ii) What is the optimal value for the state of being in the upper left corner (A)? Reminder: the discount factor is 0.5.

Run 4 iterations of value iteration by filling out the table:

iter	V(A)	V(B)	V(C)	V(D)	V(E)	V(F)
0	0	0	0	0	0	0
1						
2						
3						
4						

Solution:

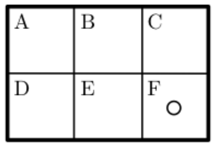
Pacman is using MDPs to maximize his expected utility. In each environment:

- Pacman has the standard actions {North, East, South, West} unless blocked by an outer wall
- There is a reward of 1 point when eating the dot (for example, in the grid below, $R(C, South, F) = 1$)
- The game ends when the dot is eaten

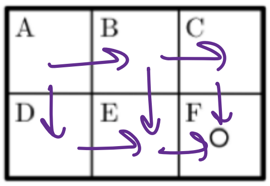
i.e. transition to state F

(a) Consider a the following grid where there is a single food pellet in the bottom right corner (F). The **discount** factor is 0.5. There is no living reward. The states are simply the grid locations.

i.e. you only get reward at the dot



(i) What is the optimal policy for each state?



(ii) What is the optimal value for the state of being in the upper left corner (A)? Reminder: the discount factor is 0.5.

$V^*(A) = 0.25$

k	V(A)	V(B)	V(C)	V(D)	V(E)	V(F)
0	0	0	0	0	0	0
1	0	0	1	0	1	0
2	0	0.5	1	0.5	1	0
3	0.25	0.5	1	0.5	1	0
4	0.25	0.5	1	0.5	1	0

$$k=1: \; V(E) = \max\big[r(E,\rightarrow) + 0.5\, V(F),\; r(E,\uparrow) + 0.5\, V(B),\; r(E,\leftarrow) + 0.5\, V(D) \big] = \max[1, 0, 0] = \boxed{1}$$

$$k=2: \; V(B) = \max\big[r(B,\rightarrow) + 0.5\, V(C),\; r(B,\downarrow) + 0.5\, V(E),\; r(B,\leftarrow) + 0.5\, V(A) \big] = \max[0.5, 0.5, 0] = \boxed{0.5}$$

$$k=3: \; V(A) = \max\big[r(A,\rightarrow) + 0.5\, V(B),\; r(A,\downarrow) + 0.5\, V(D) \big] = \max[0.25, 0.25] = \boxed{0.25}$$

If I give you $v^*(s)$ optimal value, then I can extract the optimal policy via a ONESTEP DECISION-MAKING PROBLEM!

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a)\, v^*(s') \right]$$