Markov Decision Processes

Fundamentals of AI (AE1FAI)

Slides created by Dr Huan Jin

OUTLINE

- Markov decision processes
- Policy evaluation
- Value iteration

Markov Decision Process

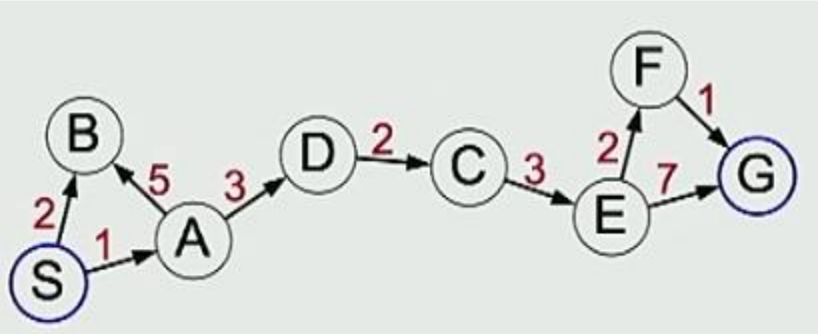
Question:

How would you get to Dongqian Lake on Saturday afternoon in the least amount of time?

- Bike
- Drive
- Didi
- Subway
- Fly

There is uncertainty in the nature!!

Review on search problem



Deterministic

State s, action a $\xrightarrow{\hspace{4em}}$ state $Succ(s, a)$

Uncertainty in the real word

State s'

State s, action a Random

State s''

Application:

- Robotics: decide where to move, but hit unseen obstacles, etc.
- Resource allocation: decide what to produce, but don't know the customer demand for different products
- Agriculture: decide what to plant, but don't know the weather and crop yield

Example 1: Dice game

For each round r=1,2,…

- You choose stay or quit.
- If quit, you get $10 and we end the game.
- If stay, you get $4 and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.

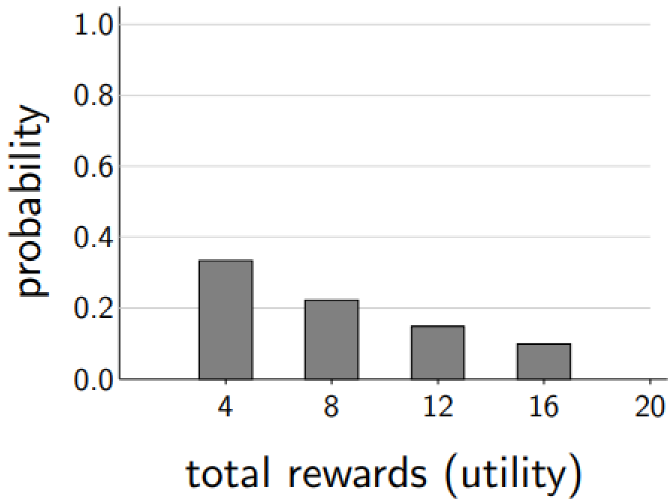
Dice:

Reward:

Policy

A policy is a choice of what action to choose at each state.

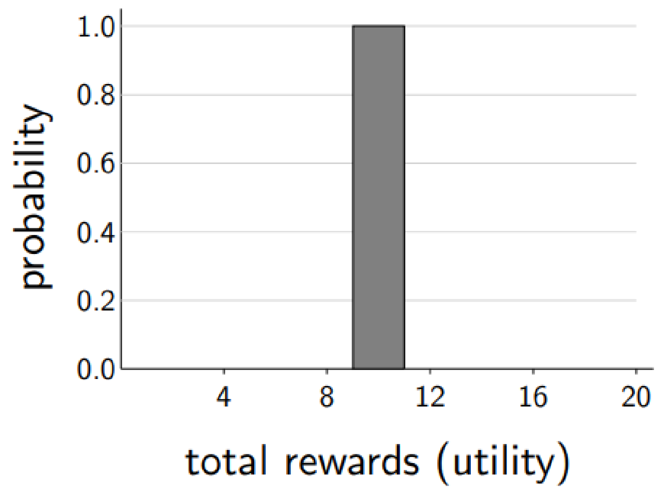
If follow policy “stay”:



Expected utility: $\frac{1}{3}(4) + \frac{2}{3} * \frac{1}{3}(8) + \frac{2}{3} * \frac{2}{3} * \frac{1}{3} * (12) + \cdots = ???$

Policy

If follow policy “quit”:



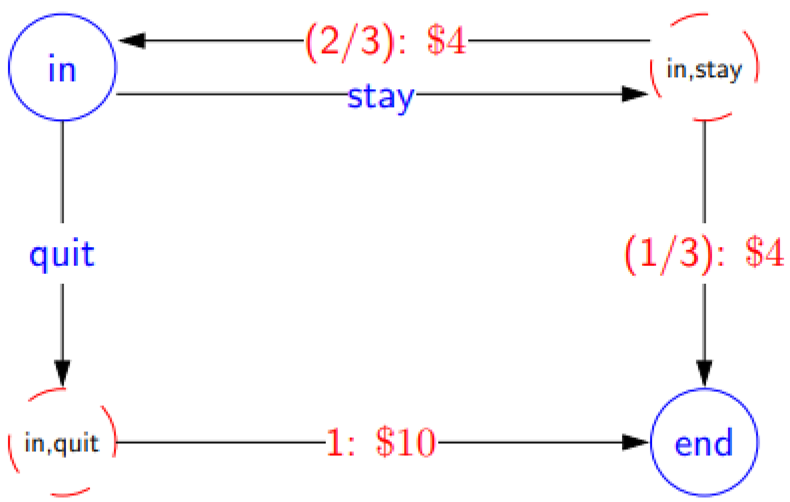
Expected utility:

1(10)=10

MDP for dice game

For each round r=1,2,…

- You choose stay or quit.
- If quit, you get $10 and we end the game.
- If stay, you get $4 and then I roll a 6-sided dice.
  - If the dice results in 1 or 2, we end the game.
  - Otherwise, continue to the next round.



Markov decision process

A MDP is defined by a tuple (*S*,*A*, *T*, *R*):

S: a set of states

A: a set of actions

T: a transition function,

- $T(s, a, s')$ where s ∈ S, a ∈ A, s' ∈ S, sometimes denoted as $P(s'|s, a)$

R: a reward function,

- $R(s, a, s')$ is reward for the transition (s, a, s')

Sometimes also have

- γ: discount factor, (0<= γ<=1)
- Terminal states: processes end after reaching these states, IsEnd(s)=True

In this example

Definition: Markov decision process

States: the set of states

$s_{\text{start}} \in$ States: starting state

Actions(s): possible actions from state s

$T(s, a, s')$: probability of s' if take action a in state s

Reward(s, a, s'): reward for the transition (s, a, s')

IsEnd(s): whether at end of game

$0 \leq \gamma \leq 1$: discount factor (default: 1)

Search Problem

Definition: search problem

States: the set of states

$s_{\text{start}} \in$ States: starting state

Actions(s): possible actions from state s

Succ(s, a): where we end up if take action a in state s

Cost(s, a): cost for taking action a in state s

IsEnd(s): whether at end

$$\text{Succ}(s, a) \Rightarrow T(s, a, s')$$

$$\text{Cost}(s, a) \Rightarrow \text{Reward}(s, a, s')$$

Transitions

The transition probabilities $T(s, a, s')$ specify the probability of ending up in state s' if taken action a in state s.

Example: transition probabilities

s	a	s'	$T(s,a,s')$
in	quit	end	1
in	stay	in	$2/3$
in	stay	end	$1/3$

Probabilities sum to 1

Example: transition probabilities

s	a	s'	$T(s,a,s')$
in	quit	end	1
in	stay	in	$2/3$
in	stay	end	$1/3$

For each state s and a :

$$\sum_{s' \in \text{States}} T(s,a,s') = 1$$

What is a solution?

Search problem: Path (sequence of actions)

MDP:

Definition: policy

A **policy** π is a mapping from each state $s \in$ States to an action $a \in$ Actions(s).

Evaluating a policy

Utility:

- Following a policy yields a random path.
- The utility of a policy is the (discounted) sum of the rewards on the path (also a random quantity).

Path	Utility
[in; stay, 4, end]	
[in; stay, 4, in; stay, 4, in; stay, 4, end]	
[in; stay, 4, in; stay, 4, end]	

Value (expected utility):

- The value of a policy is the expected utility.

Discounting

Path: $s_0, a_1 r_1 s_1, a_2 r_2 s_2, \ldots.$ (action, reward, new state)

The utility with discount γ is

$$u_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \ldots$$

Discount γ=1(save for the future):

[stay, stay, stay]: 4+4+4=12

Discount γ=0(live in the moment):

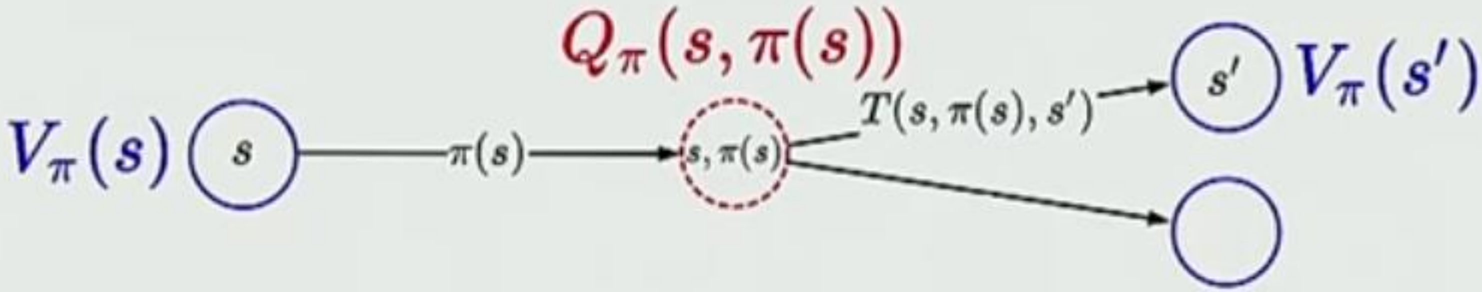
[stay, stay, stay]: 4+0+0=4

Discount γ=0.5 (balanced life):

[stay, stay, stay]: 4+0.5*4+0.5*0.5*4=7

Policy evaluation

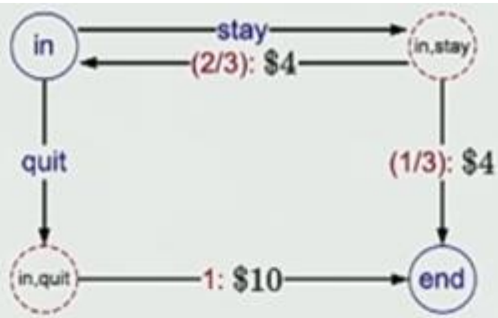
Plan: define recurrences relating value and Q-value



$$V_\pi(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_\pi(s, \pi(s)) & \text{otherwise.} \end{cases}$$

$$Q_\pi(s, a) = \sum_{s'} T(s, a, s')[\text{Reward}(s, a, s') + \gamma V_\pi(s')]$$

Dice game



(assume $\gamma = 1$)

Let π be the "stay" policy: $\pi(\text{in}) = \text{stay}$.

$$V_\pi(\text{end}) = 0$$

$$V_\pi(\text{in}) = \frac{1}{3}\left(4 + V_\pi(\text{end})\right) + \frac{2}{3}\left(4 + V_\pi(\text{in})\right)$$

In this case, can solve in closed form:

$$V_\pi(\text{in}) = \frac{1}{3}4 + \frac{2}{3}\left(4 + V_\pi(\text{in})\right)$$

Policy evaluation

Iterative algorithm:

Start with arbitrary policy values and repeatedly apply recurrences to converge to true values.

Algorithms:

Initialize $V_\pi^{(0)}(s) \leftarrow 0$ for all states s.

For iteration $t = 1, \dots, T_{PE}$:

For each state s:

$$V_\pi^{(0)}(s) \leftarrow \underbrace{\sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_\pi^{(t-1)}(s')]}_{Q^{(t-1)}(s, \pi(s))}$$

Repeat until : $max_{s \in S} |V_\pi^{(t)}(s) - V_\pi^{(t-1)}(s)| < \varepsilon$

Policy evaluation on dice game

Let π be the "stay" policy: $\pi(\text{in}) = \text{stay}$.

$$V_\pi^{(t)}(\text{end}) = 0$$

$$V_\pi^{(t)}(\text{in}) = \frac{1}{3}\left(4 + V_\pi^{(t-1)}(\text{end})\right) + \frac{2}{3}\left(4 + V_\pi^{(t-1)}(\text{in})\right)$$

s	end	in	
$V_\pi^{(t)}$	0.00	12.00	($t = 100$ iterations)

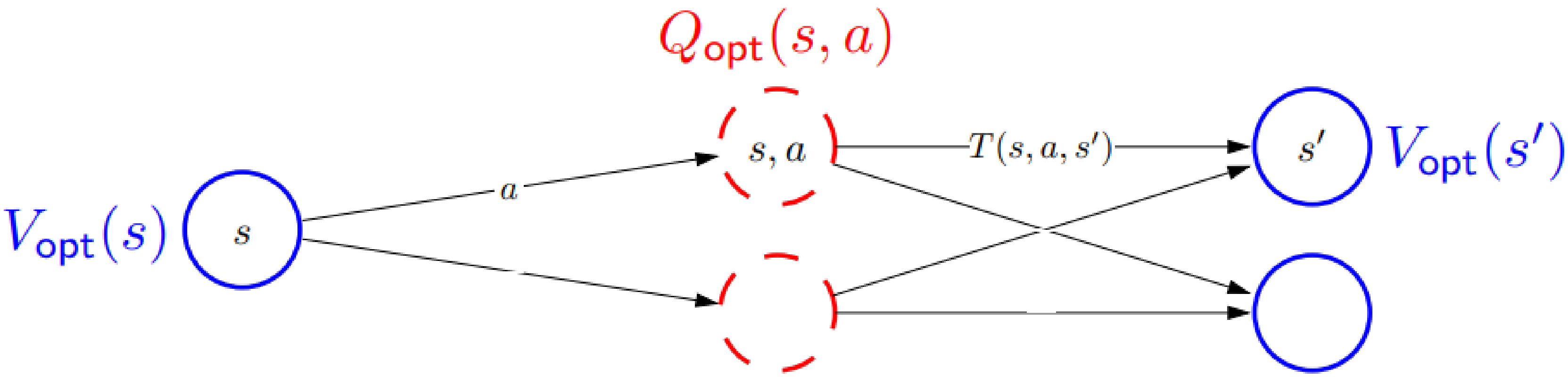
Converges to $V_\pi(\text{in}) = 12$.

Value iteration

Optimal value: $V_{opt}(s)$

The optimal value is the maximum value attained by any policy.

Optimal values and Q-values



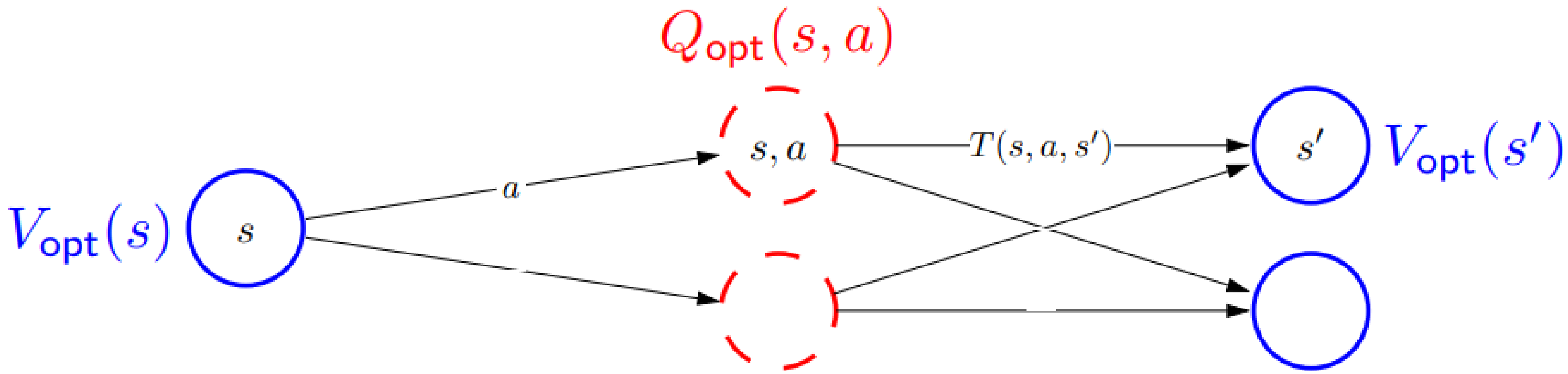
Optimal value if take action a in state s:

$$Q_{\text{opt}}(s,a) = \sum_{s'} T(s,a,s')[\text{Reward}(s,a,s') + \gamma V_{\text{opt}}(s')].$$

Optimal value from state s:

$$V_{\text{opt}}(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ \max_{a \in \text{Actions}(s)} Q_{\text{opt}}(s,a) & \text{otherwise.} \end{cases}$$

Optimal policy



Given Q_{opt}, read off the optimal policy:

$$\pi_{\text{opt}}(s) = \arg \max_{a \in \text{Actions}(s)} Q_{\text{opt}}(s, a)$$

Value iteration

Algorithms:

Initialize $V_{opt}^{(0)}(s) \leftarrow 0$ for all states s.

For iteration $t = 1, \dots, T_{VI}$:

For each state s:

$$V_{\pi}^{(0)}(s) \leftarrow \max_{a \in A} \underbrace{\sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_{opt}^{(t-1)}(s')]}_{Q_{opt}^{(t-1)}(s,a)}$$

Value iteration: dice game

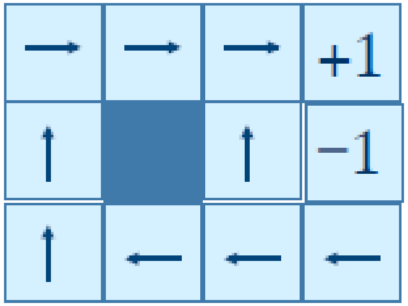
s	end	in	
$V_{\text{opt}}^{(t)}$	0.00	12.00	$(t = 100$ iterations$)$
$\pi_{\text{opt}}(s)$	-	stay	

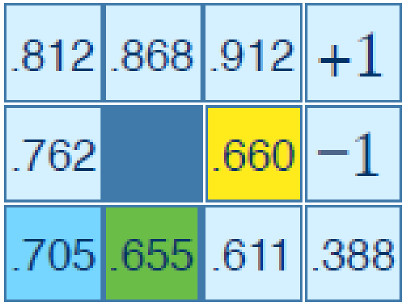
Example from textbook

Actions succeed with probability 0.8 and move at right angles!

with probability 0.1 (remain in the same position when"

there is a wall). Actions incur a small cost (0.04)."





Summary

MDPs cope with uncertainty.

Solutions are policies rather than paths.

Policy evaluation computes policy value (expected utility)

Value iteration computes optimal value (maximum expected utility) and optimal policy.