



人工智能原理与技术

4. 马尔科夫决策MDP

王翔

中国科学技术大学
数据科学实验室LDS



Question 问题

How would you get groceries on a Saturday afternoon in the least amount of time?

order grocery delivery

bike to the store

drive to the store

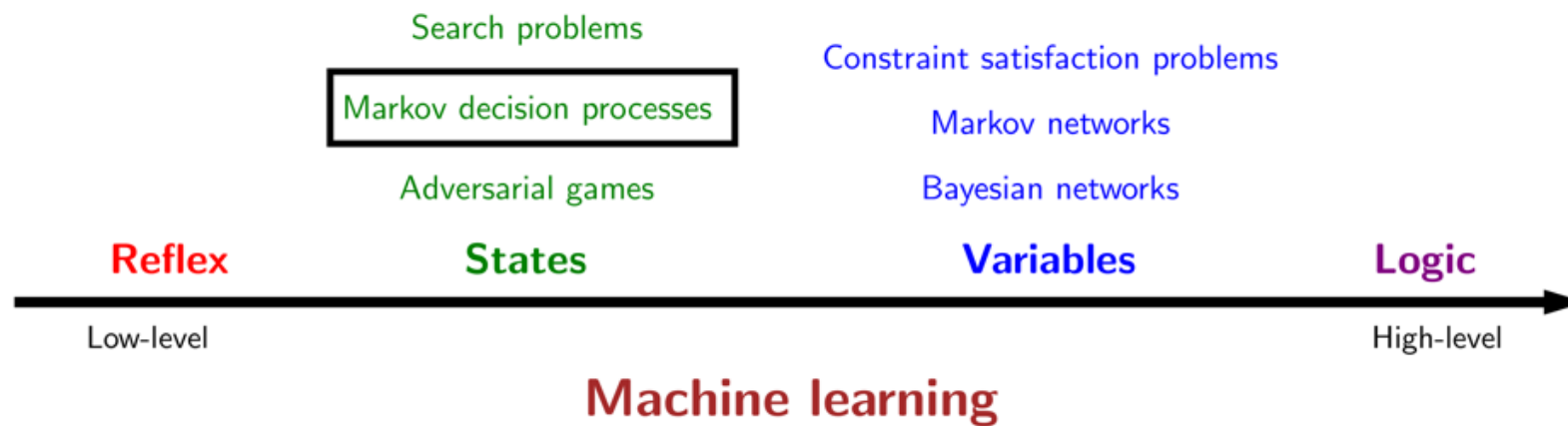
Uber/Lyft to the store

fly to the store



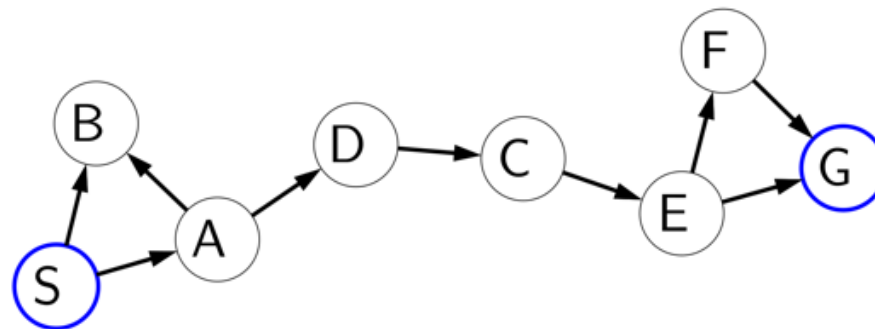
Course Plan

课程安排





So far: search problems 搜索问题回顾



确定性的
deterministic

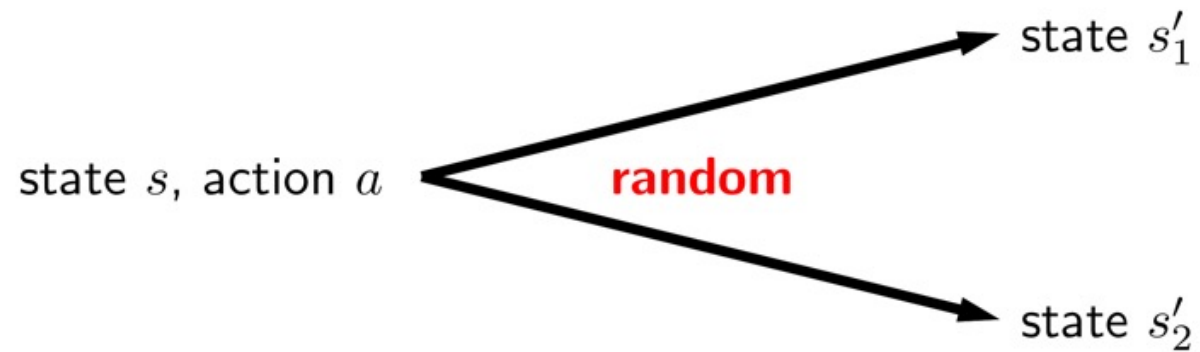
state s , action a \longrightarrow state $\text{Succ}(s, a)$





Uncertainty in the real world

现实世界中的不确定性





History 历史

- MDPs: Mathematical Model for decision making under uncertainty.
不确定性下决策的数学模型
- MDPs were first introduced in 1950s-60s.
- Ronald Howard's book on Dynamic Programming and Markov Processes
- The term 'Markov' refers to Andrey Markov as MDPs are extensions of Markov Chains, and they allow making decisions (taking actions or having choice).
mdp是马尔可夫链的延伸，它们允许做出决定（采取行动或有选择余地）



Applications

应用



Robotics: decide where to move, but actuators can fail, hit unseen obstacles, etc.
机器人：决定移动到哪里，但执行器可能会失败，碰到看不见的障碍等。



Resource allocation: decide what to produce, don't know the customer demand for various products
资源配置：决定生产什么，不知道客户对各种产品的需求



Agriculture: decide what to plant, but don't know weather and thus crop yield
农业：决定种植什么，但不知道天气和作物产量



Volcano crossing 穿越火山



```
// Model
moveReward = 0 // For every action you take
passReward = 20 // If get to far green
volcanoReward = -50 // If fall into volcano
slipProb = 0.3 // If slip, go in random direction
discount = 1 // How much to value the future

// CHANGE THIS to 10
numIters = 10 // # iterations of value iteration
```

1.4 ↓	←2.9	-50	20 ↑
1.9 ↓	1.1 ↓	-50	13.8 ↑
2 ↑	6.5 →	7.5 →	13.2 ↑

Value: 1.86

- 向东(E)/ 西(W)/ 南(S)/ 北(N) 移动一格
- 移动到红色或绿色游戏结束
- -50: 火山 2: 安全区 20: 美丽海岛
- 怎么移动使收益最大?

Run (or press ctrl-enter)

2024/3/21



Roadmap

路线图

Modeling

Modeling MDP Problems

Algorithms

Policy Evaluation

Value Iteration

Learning

Intro to Reinforcement Learning

Model-Based Monte Carlo

Model-Free Monte Carlo

SARSA

Q-learning

Epsilon Greedy

Function Approximation



MDPs: modeling 马尔可夫决策过程：建模

Modeling

Modeling MDP Problems

Algorithms

Policy Evaluation

Value Iteration

Learning

Intro to Reinforcement Learning

Model-Based Monte Carlo

Model-Free Monte Carlo

SARSA

Q-learning

Epsilon Greedy

Function Approximation



Dice game 骰子游戏



Example: dice game

在每一轮 $r = 1, 2, \dots$

- 你选择留下或退出。
- 如果你退出，你会得到10美元，我们结束游戏。
- 如果留下，你得到4美元然后我掷一个六面骰子。
 - 如果骰子的结果是1或2，我们就结束游戏。
 - 否则，继续进行下一轮。

Start

Stay

Quit

Dice:

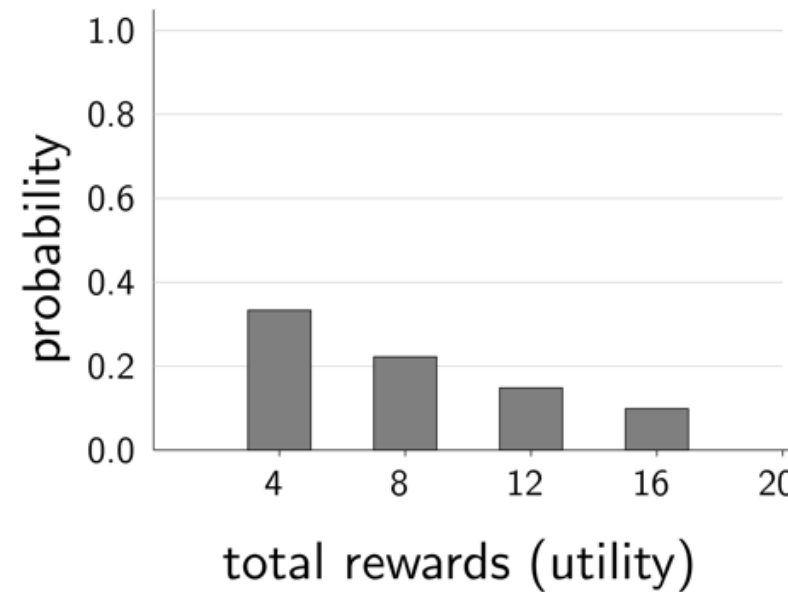
Rewards:

0



Rewards 奖励

If follow policy "stay":



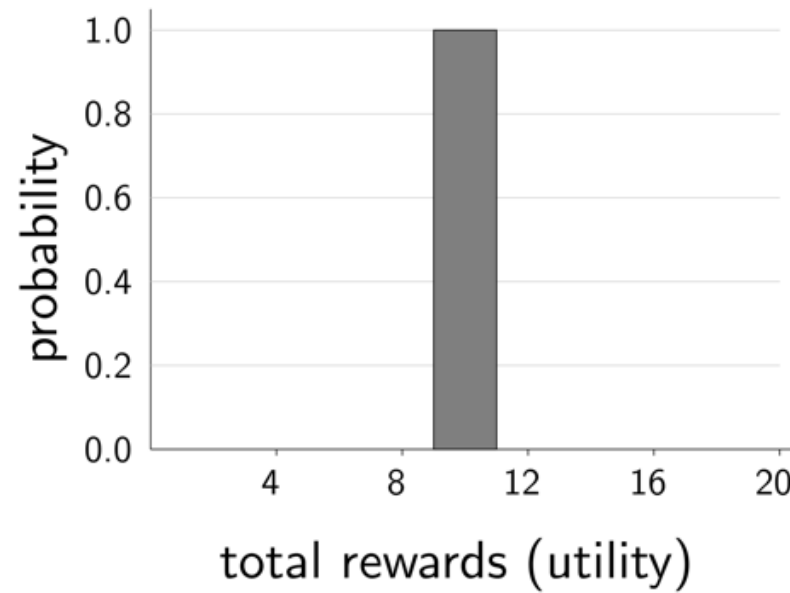
Expected utility:

$$\frac{1}{3}(4) + \frac{2}{3} \cdot \frac{1}{3}(8) + \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}(12) + \dots = 12$$



Rewards 奖励

If follow policy "quit":



Expected utility:

$$1(10) = 10$$



MDP for dice game

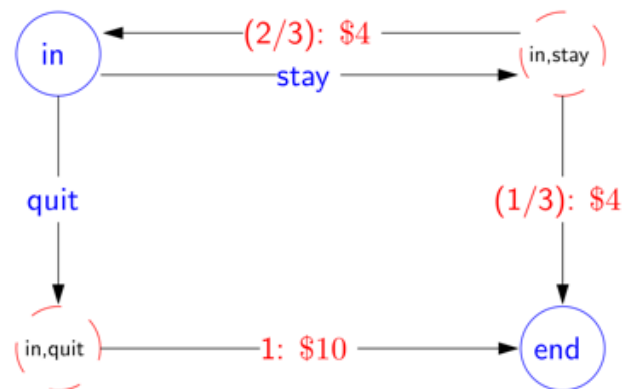
骰子游戏中的MDP



Example: dice game

在每一轮 $r = 1, 2, \dots$

- 你选择留下或退出。
- 如果你退出，你会得到10美元，我们结束游戏。
- 如果留下，你得到4美元然后我掷一个六面骰子。
 - 如果骰子的结果是1或2，我们就结束游戏。
 - 否则，继续进行下一轮。





Markov decision process

马尔可夫决策过程



Definition: Markov decision process

States: the set of states

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

$\text{Actions}(s)$: possible actions from state s

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

$\text{Reward}(s, a, s')$: reward for the transition (s, a, s')

$\text{IsEnd}(s)$: whether at end of game

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



Search problems

对比：搜索问题



Definition: search problem

States: the set of states

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

$\text{Actions}(s)$: possible actions from state s

$\text{Succ}(s, a)$: where we end up if take action a in state s

$\text{Cost}(s, a)$: cost for taking action a in state s

$\text{IsEnd}(s)$: whether at end

- $\text{Succ}(s, a) \Rightarrow T(s, a, s')$ $T(s, a, s')$: probability of s' if take action a in state s
- $\text{Cost}(s, a) \Rightarrow \text{Reward}(s, a, s')$ $\text{Reward}(s, a, s')$: reward for the transition (s, a, s')



Transitions

转移概率



Definition: transition probabilities

The **transition probabilities** $T(s, a, s')$ specify the probability of ending up in state s' if taken action a in state s . 从状态 s 采取行动 a 最终到 s' 的概率



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 one 转移概率之和为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 action a :

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

Successors: s' such that $T(s, a, s') > 0$



Transportation example

运输的例子



Example: transportation

街块编号1到 n 。

从 s 走到 $s+1$ 需要1分钟。

从 s 到 $2s$ 乘一辆神奇的电车需要2分钟。

如何在最短的时间内从1到 n ？

电车故障概率为0.5。



What is a solution? 解决方法是什么?

Search problem: path (sequence of actions)

MDP:



Definition: policy

从状态 s 到行动 a 的映射

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



Example: volcano crossing

s	$\pi(s)$
(1,1)	S
(2,1)	E
(3,1)	N
...	...



MDPs: policy evaluation

马尔可夫决策过程：策略评估

Modeling

Modeling MDP Problems

Algorithms

Policy Evaluation

Value Iteration

Learning

Intro to Reinforcement Learning

Model-Based Monte Carlo

Model-Free Monte Carlo

SARSA

Q-learning

Epsilon Greedy

Function Approximation



Evaluating a policy 评估一项政策

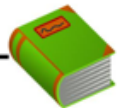


Definition: utility

Following a policy yields a **random path**.

The **utility** of a policy is the (discounted) sum of the rewards on the path (this is a random variable). 针对一条path而言，是随机变量

Path	Utility
[in; stay, 4, end]	4
[in; stay, 4, in; stay, 4, in; stay, 4, end]	12
[in; stay, 4, in; stay, 4, end]	8
[in; stay, 4, in; stay, 4, in; stay, 4, in; stay, 4, end]	16
...	...



Definition: value (expected utility)

The **value** of a policy at a state is the **expected** utility.



Evaluating a policy: volcano crossing

策略评估：穿越火山

```
// Model
moveReward = -0.1 // For every action you take
passReward = 40 // If get to far green
volcanoReward = -50 // If fall into volcano
slipProb = 0.3 // If slip, go in random direction
discount = 0.9 // How much to value the future

// Run algorithms on model
numIters = 10 // # iterations of value iteration
numEpisodes = 1 // # simulations
```

Run (or press ctrl-enter)

2.4	-0.5	-50	40	a	r	s
↓	↓					(2,1)
3.7 →	5	-50	31	E	-0.1	(2,2)
	↓		↑	S	-0.1	(2,1)
				E	-0.1	(2,2)
				S	-0.1	(3,2)
				E	-0.1	(3,3)
				E	-0.1	(3,4)
				N	-0.1	(3,4)
				N	-0.1	(3,3)
				E	-0.1	(3,4)
				N	-0.1	(2,4)
				N	39.9	(1,4)

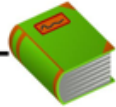
Value: 3.73

Utility: 13.26

➤ 通过程序模拟，随机生成路径，得到收敛后的value/utility



Discounting “折扣”



Definition: utility

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

The **utility** with discount γ is

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

Discount $\gamma = 1$ (save for the future):

$$[\text{stay}, \text{stay}, \text{stay}, \text{stay}]: 4 + 4 + 4 + 4 = 16$$

Discount $\gamma = 0$ (live in the moment):

$$[\text{stay}, \text{stay}, \text{stay}, \text{stay}]: 4 + 0 \cdot (4 + \dots) = 4$$

Discount $\gamma = 0.5$ (balanced life):

$$[\text{stay}, \text{stay}, \text{stay}, \text{stay}]: 4 + \frac{1}{2} \cdot 4 + \frac{1}{4} \cdot 4 + \frac{1}{8} \cdot 4 = 7.5$$



Policy evaluation 策略评估



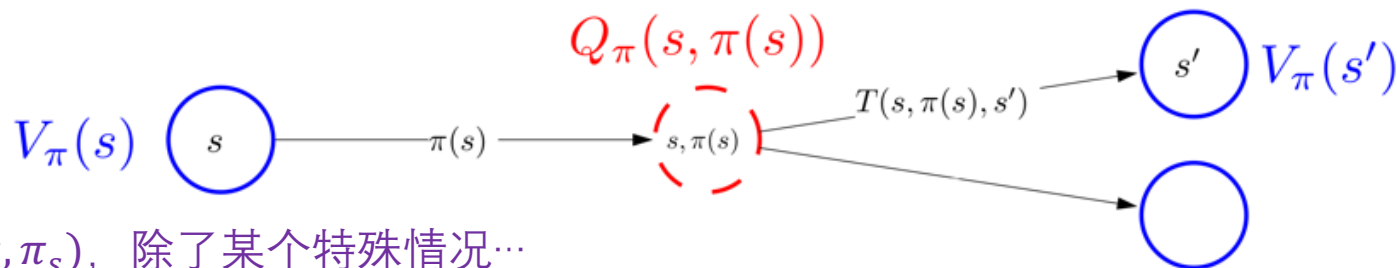
Definition: value of a policy

Let $V_\pi(s)$ be the expected utility received by following policy π from state s .



Definition: Q-value of a policy

Let $Q_\pi(s, a)$ be the expected utility of taking action a from state s , and then following policy π .

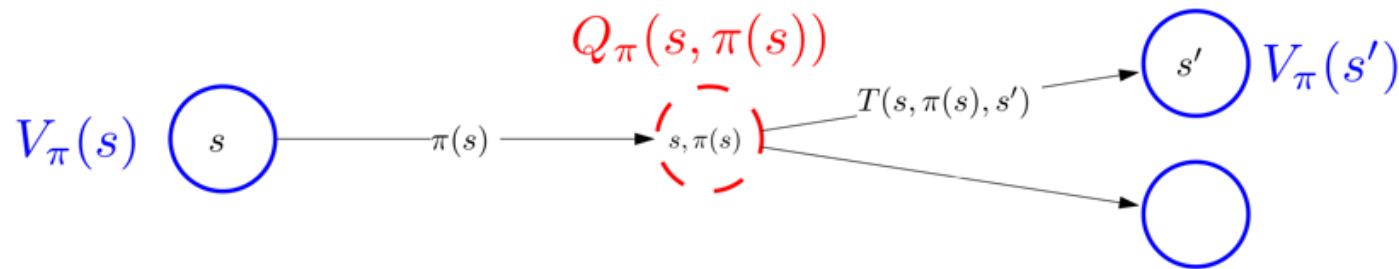


➤ $V_\pi(s) = Q_\pi(s, \pi_s)$, 除了某个特殊情况...



Policy evaluation 策略评估

Plan: define recurrences relating value and Q-value

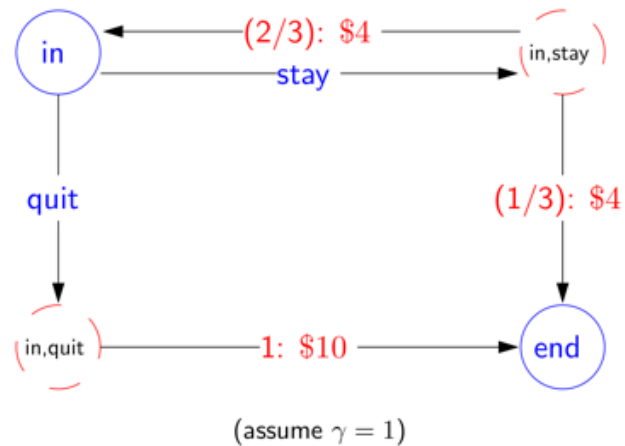


$$V_\pi(s) = \begin{cases} 0 & \text{if IsEnd}(s) \\ Q_\pi(s, \pi(s)) & \text{otherwise.} \end{cases}$$

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



Dice game 骰子游戏



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

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

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

In this case, can solve in closed form: 闭式解

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



Policy evaluation

策略评估



Key idea: iterative algorithm

Start with arbitrary policy values and repeatedly apply recurrences to converge to true values. 无闭式解时，通过迭代至收敛



Algorithm: policy evaluation

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

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

For each state s :

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



Policy evaluation implementation

策略评估的实现

How many iterations (t_{PE})? Repeat until values don't change much:

$$\max_{s \in \text{States}} |V_{\pi}^{(t)}(s) - V_{\pi}^{(t-1)}(s)| \leq \epsilon$$

Don't store $V_{\pi}^{(t)}$ for each iteration t , need only last two:

$$V_{\pi}^{(t)} \text{ and } V_{\pi}^{(t-1)}$$



Complexity 复杂度分析



Algorithm: policy evaluation

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

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

For each state s :

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

MDP complexity

S states

A actions per state

S' successors (number of s' with $T(s, a, s') > 0$)

Time: $O(t_{\text{PE}} S S')$



Policy evaluation on dice game 骰子游戏的策略评估

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

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

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

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

($t = 100$ iterations)

Converges to $V_{\pi}(\text{in}) = 12$. ➤ Try to implement it in Python.



Summary so far 总结

- **MDP**: graph with states, chance nodes, transition probabilities, rewards
具有状态、机会节点、转移概率、奖励的图
- **Policy**: mapping from state to action (solution to MDP)
从状态到动作的映射 (MDP的解)
- **Value of policy**: expected utility over random paths
随机路径上的期望效用
- **Policy evaluation**: iterative algorithm to compute value of policy
计算策略价值的迭代算法



MDPs: value iteration

马尔可夫决策过程：价值迭代

Modeling

Modeling MDP Problems

Algorithms

Policy Evaluation

Value Iteration

Learning

Intro to Reinforcement Learning

Model-Based Monte Carlo

Model-Free Monte Carlo

SARSA

Q-learning

Epsilon Greedy

Function Approximation



Optimal value and policy

最优价值与策略

Goal: try to get directly at maximum expected utility



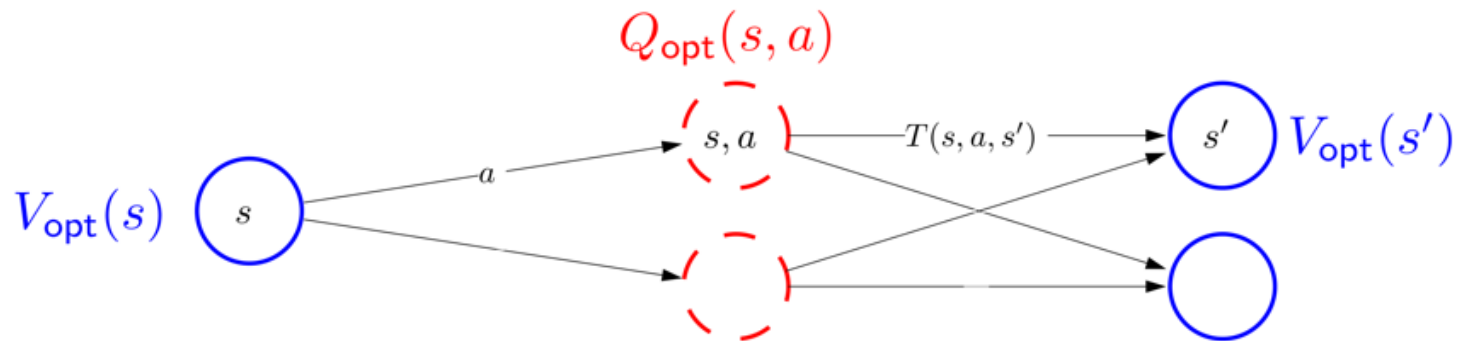
Definition: optimal value

The **optimal value** $V_{\text{opt}}(s)$ is the maximum value attained by any policy.



Optimal values and Q-values

最优值和Q-value



Optimal value if take action a in state s :

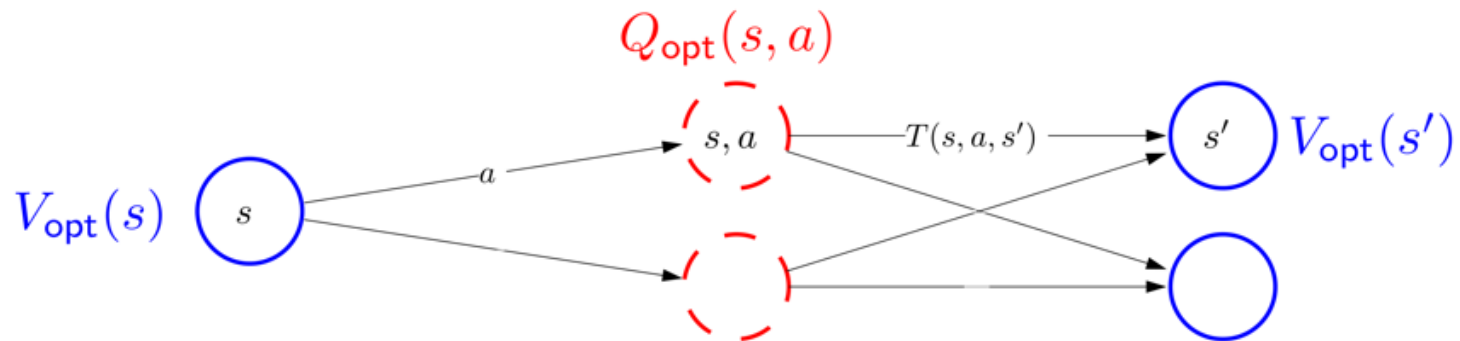
$$Q_{\text{opt}}(s, a) = \sum_{s'} T(s, a, s') [\text{Reward}(s, a, s') + \gamma V_{\text{opt}}(s')].$$

Optimal value from state s :

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



Optimal policies 最优策略



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 价值迭代



Algorithm: value iteration [Bellman, 1957]

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

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

For each state s :

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

Time: $O(t_{\text{VI}} S A S')$

[semi-live solution]



Value iteration: dice game

价值迭代：骰子游戏

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



Value iteration: volcano crossing 价值迭代：穿越火山

```
// Model  
moveReward = 0 // For every action you take  
passReward = 20 // If get to far green  
volcanoReward = -50 // If fall into volcano  
slipProb = 0.1 // If slip, go in random direction  
discount = 1 // How much to value the future  
  
// CHANGE THIS to 0, 1, 2, 3, ...  
numIters = 10 // # iterations of value iteration
```

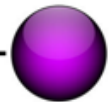
Run (or press ctrl-enter)

13.4 ↓	12.3 ↓	-50	20
13.7 →	14.1 ↓	-50	18.2 ↑
2 ↑	15.9 →	16.3 →	18.1 ↑

Value: 13.68



Convergence 收敛性



Theorem: convergence

Suppose either

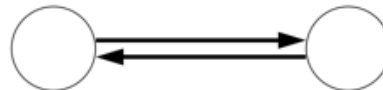
- discount $\gamma < 1$, or
- MDP graph is acyclic.

Then value iteration converges to the correct answer.



Example: non-convergence

discount $\gamma = 1$, zero rewards





Summary of algorithms

算法概述

- Policy evaluation: $(\text{MDP}, \pi) \rightarrow V_\pi$
- Value iteration: $\text{MDP} \rightarrow (Q_{\text{opt}}, \pi_{\text{opt}})$



Unifying idea 观点的统一

Algorithms:

- Search DP computes $\text{FutureCost}(s)$
- Policy evaluation computes policy value $V_{\pi}(s)$
- Value iteration computes optimal value $V_{\text{opt}}(s)$

Recipe:

- Write down recurrence (e.g., $V_{\pi}(s) = \dots V_{\pi}(s') \dots$)
- Turn into iterative algorithm (replace mathematical equality with assignment operator)