Deep Reinforcement Learning
Lecture 1: Motivation + Overview + MDPs + Exact Solution Methods

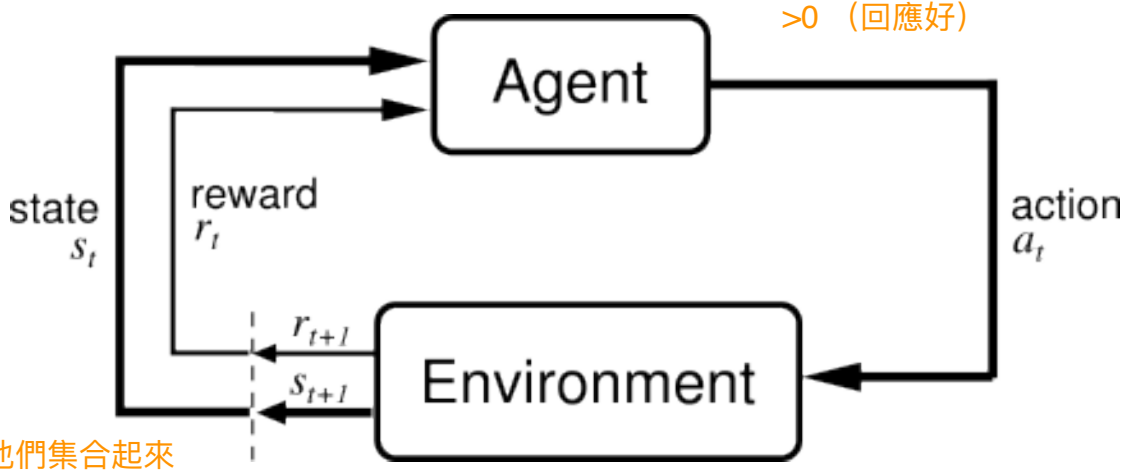
Pieter Abbeel

OpenAI / UC Berkeley / Gradescope

Many slides made with John Schulman, Yan (Rocky) Duan and Xi (Peter) Chen

Markov Decision Process

Agent 要跟環境互動，每次互動會有一個reward
reward有可能是0（內有回應）
<0（回應不好）
>0 （回應好）



最後有個collection把他們集合起來

有個環境，會有一個current state

Assumption: agent gets to observe the state

[Drawing from Sutton and Barto, Reinforcement Learning: An Introduction, 1998]

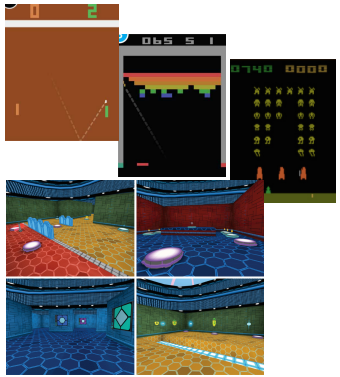
Some Reinforcement Learning Success Stories

Kohl and Stone, 2004

Ng et al, 2004

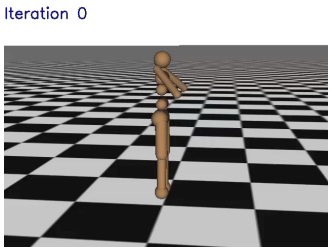
Tedrake et al, 2005

Kober and Peters, 2009

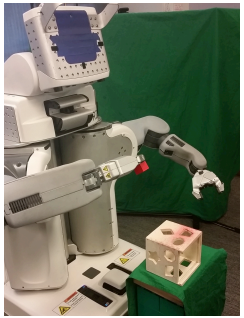


Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

Silver et al, 2014 (DPG)
Lillicrap et al, 2015 (DDPG)



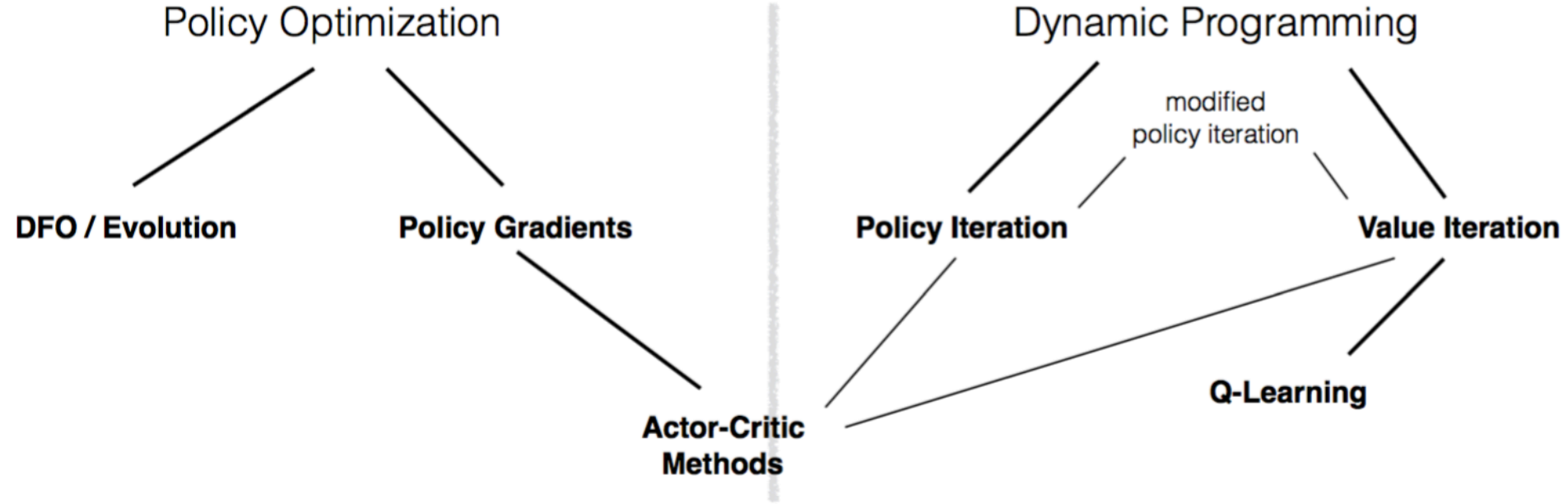
Schulman et al, 2016 (TRPO + GAE)



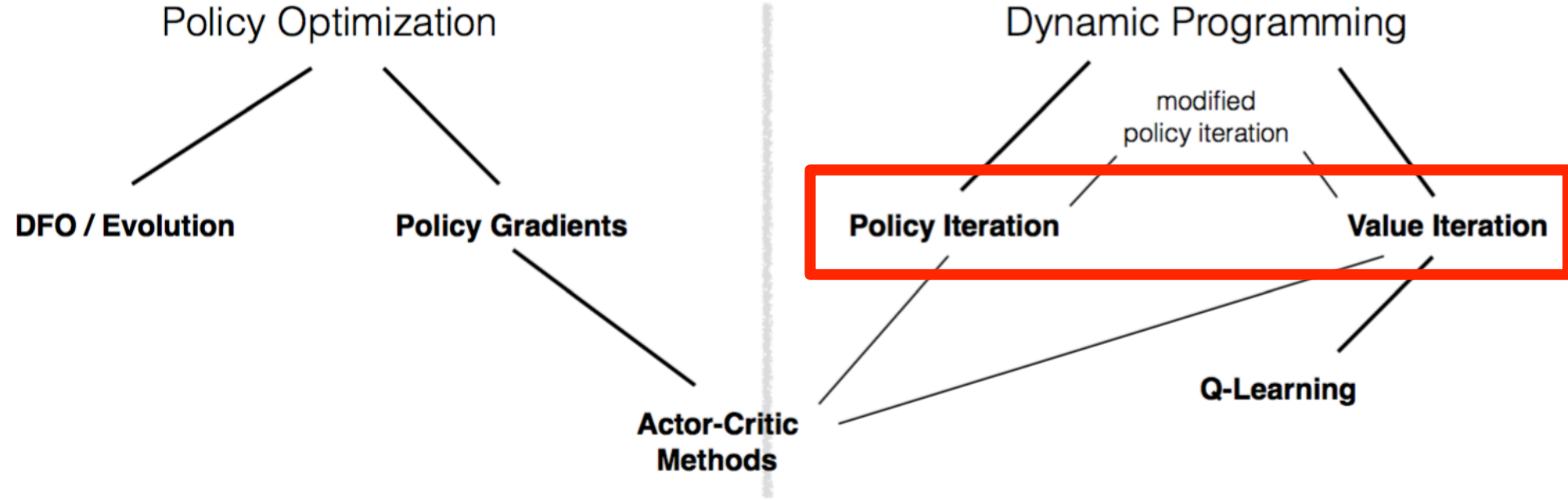
Levine*, Finn*, et al, 2016 (GPS)

Silver*, Huang*, et al, 2016 (AlphaGo)

RL Algorithms Landscape



RL Algorithms Landscape

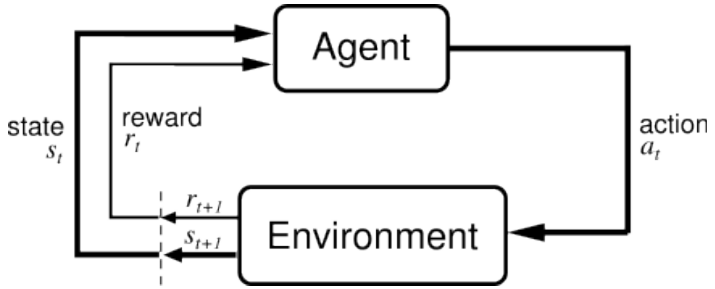


Markov Decision Process (MDP)

An MDP is defined by:

定義state space是什麼（apha go 就是棋譜）

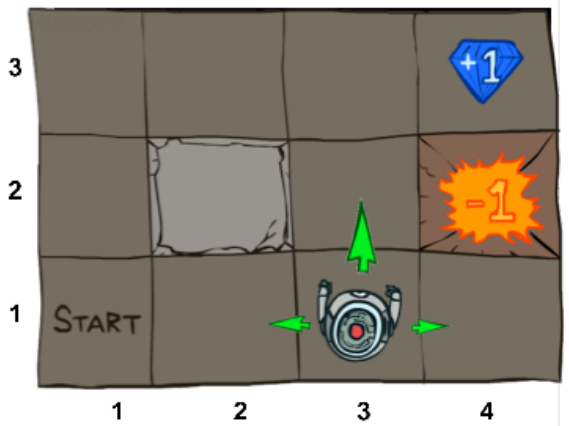
- Set of states S
- Set of actions A 可以做什麼動作（開車、下棋）
- Transition function $P(s' \mid s, a)$ 在現在環境state經過動作a會轉換到下一個環境state'
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ apply 在 reward 上面
- Horizon H 整個流程



Example MDP: Gridworld

An MDP is defined by:

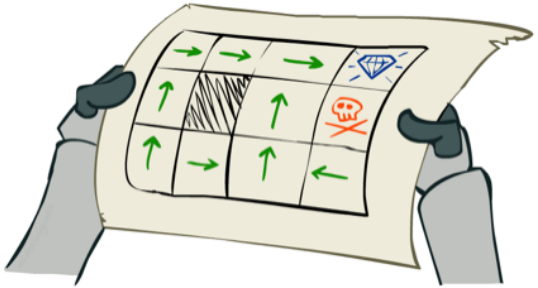
- Set of states S
- Set of actions A
- Transition function $P(s' \mid s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



funtion input是 state
reward 越大越好

Goal: $max_\pi \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$

π:



Outline

- Optimal Control

  =

  given an MDP (S, A, P, R, γ, H)

  find the optimal policy π^* to maximize goal

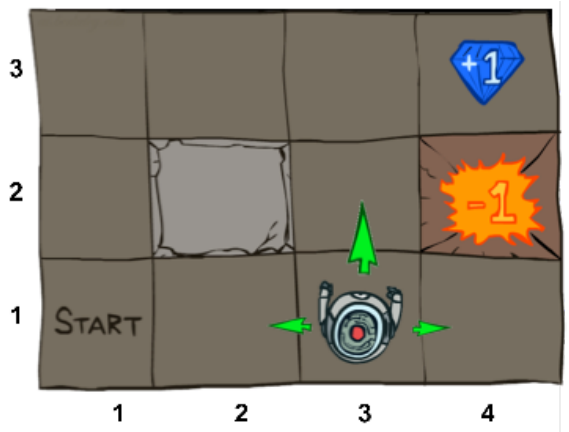
- Exact Methods:
  - ***Value Iteration***
  - Policy Iteration

For now: small discrete state-action spaces as they are simpler to get the main concepts across. We will consider large / continuous spaces later!

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally

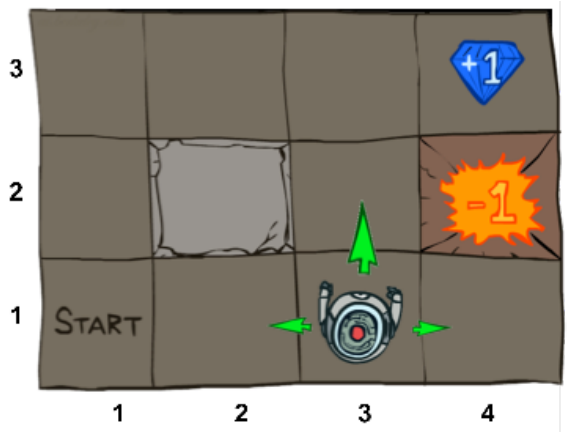


Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally

當gamma是1的時候，其實只要走到（4,3）就好，不管你從哪邊走，要走多久，只要不要踩到（4,2）都可以



Let's assume:

actions deterministically successful, gamma = 1, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

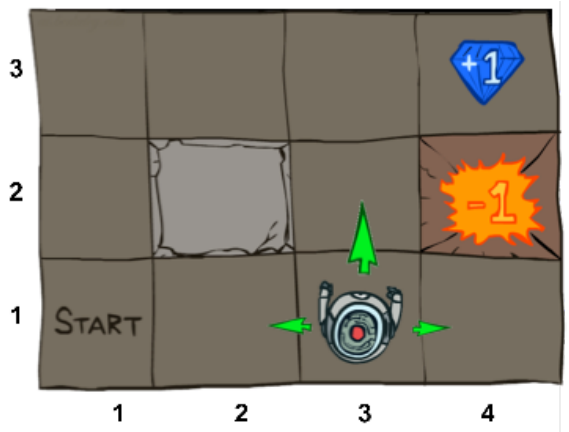
V*(1,1) =

V*(4,2) =

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume:

actions deterministically successful, gamma = 0.9, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

當gamma = 0.9的時候，走越久會有懲罰，所以會盡量走短

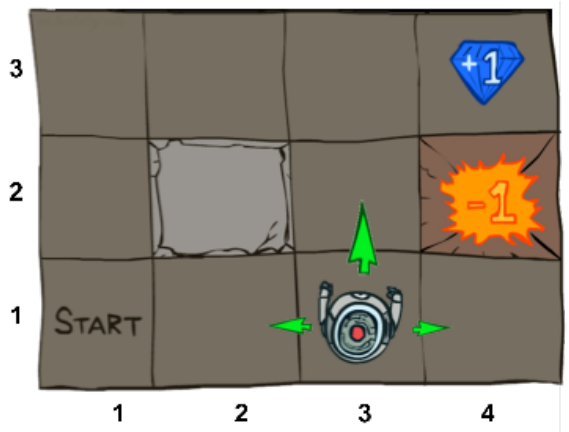
V*(1,1) = 1*0.9*0.9*0.9*0.9*0.9

V*(4,2) =

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume: 如果今天多加了一個機率，0.8會真的走向最好的地方

actions successful w/probability 0.8, gamma = 0.9, H = 100

V*(4,3) =

V*(3,3) =

V*(2,3) =

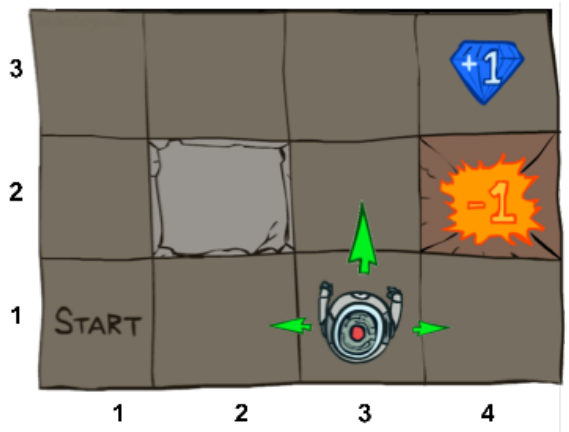
V*(1,1) =

V*(4,2) =

Optimal Value Function V*

$$V^*(s) = \max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) \mid \pi, s_0 = s\right]$$

= sum of discounted rewards when starting from state s and acting optimally



Let's assume:

actions successful w/probability 0.8, gamma = 0.9, H = 100

V*(4,3) = 1

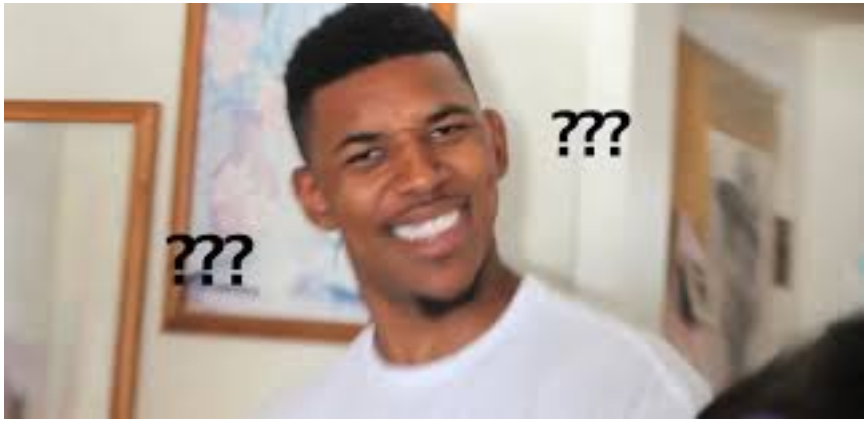
V*(3,3) = 0.8*0.9*V*(4,3)+0.1*0.9*V*(3,3)+0.1*0.9*V*(3,2)

所有可能的期望值

V*(2,3) =

V*(1,1) =

V*(4,2) =



David Silver to the Rescue

RL2-MDP.pdf

跳去MDP講義

Value Iteration

- $V_0^*(s)$ = optimal value for state s when H=0

要計算V*，經過一次一次的反覆計算來找到最好的V* （下標代表第幾次更新）

  - $V_0^*(s) = 0 \quad \forall s$

- $V_1^*(s)$ = optimal value for state s when H=1
  - $V_1^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0^*(s'))$
- $V_2^*(s)$ = optimal value for state s when H=2
  - $V_2^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when H = k
  - $V_k^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$

Value Iteration

Algorithm:

Start with $V_0^*(s) = 0$ for all s.

For k = 1, … , H:

For all states s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

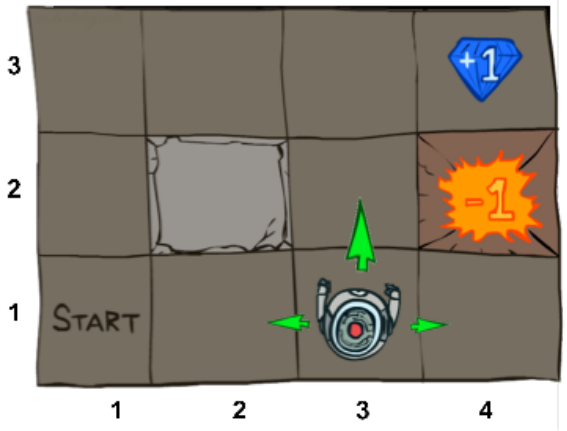
$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

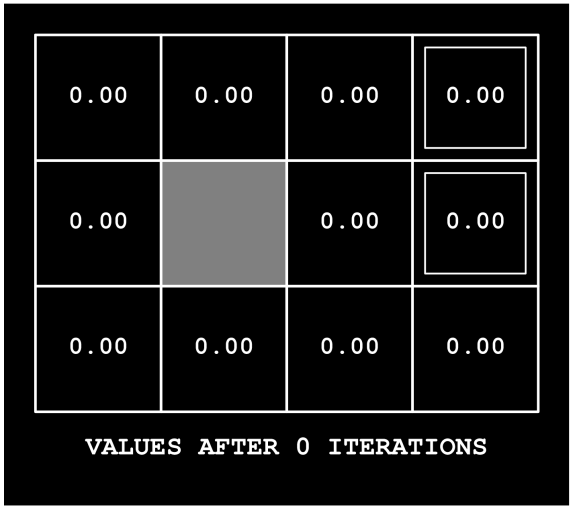
This is called a value update or Bellman update/back-up

Value Iteration

$$V_0(s) \leftarrow 0$$

k = 0

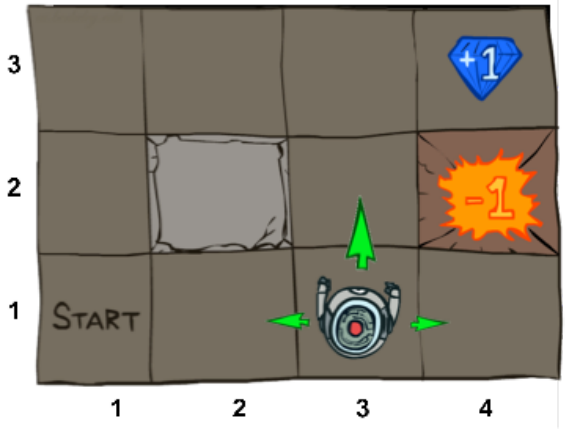




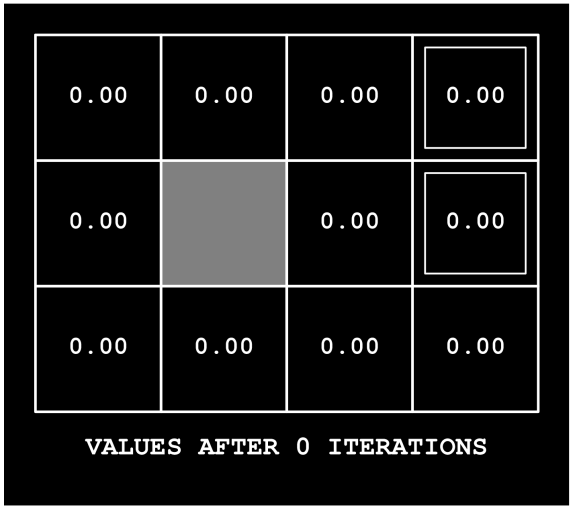
Noise = 0.2
Discount = 0.9

Value Iteration

$$V_1(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0(s'))$$



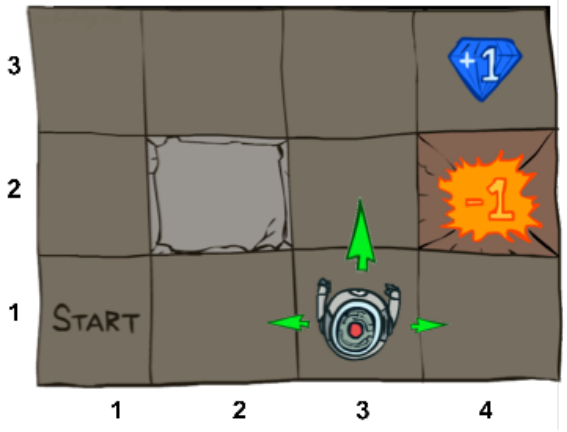
k = 0



Noise = 0.2
Discount = 0.9

Value Iteration

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$



k = 1

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_2(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1(s'))$$

k = 2

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 3

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 4

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 5

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 6

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 7

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 8

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 9

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 10

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 11

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 12

Noise = 0.2
Discount = 0.9

Value Iteration

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_k(s'))$$

k = 100

Noise = 0.2
Discount = 0.9

Value Iteration Convergence 收斂

Theorem. *Value iteration converges. At convergence, we have found the optimal value function V* for the discounted infinite horizon problem, which satisfies the Bellman equations*

$$\forall S \in S: \quad V^*(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

有個叫做Value Iteration 的方式可以幫助我們找到V*
V*可以幫助我們找到Q*
Q*可以幫助我們找到pie*(最佳的function)

- Now we know how to act for infinite horizon with discounted rewards!
  - Run value iteration till convergence.
  - This produces V*, which in turn tells us how to act, namely following:

$$\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

- Note: the infinite horizon optimal policy is stationary, i.e., the optimal action at a state s is the same action at all times. (Efficient to store!)

Convergence: Intuition

- $V^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for ∞ steps
- $V_H^*(s)$ = expected sum of rewards accumulated starting from state s, acting optimally for H steps

- Additional reward collected over time steps H+1, H+2, …

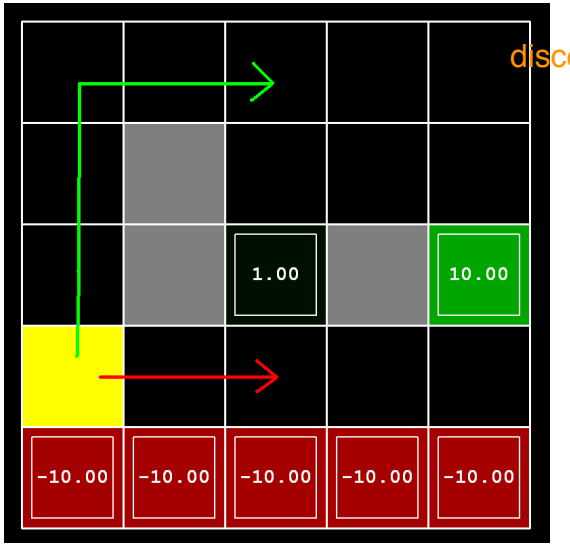
$$\gamma^{H+1}R(s_{H+1}) + \gamma^{H+2}R(s_{H+2}) + \ldots \leq \gamma^{H+1}R_{max} + \gamma^{H+2}R_{max} + \ldots = \frac{\gamma^{H+1}}{1-\gamma}R_{max}$$

goes to zero as H goes to infinity

Hence $V_H^* \xrightarrow{H\to\infty} V*$

For simplicity of notation in the above it was assumed that rewards are always greater than or equal to zero. If rewards can be negative, a similar argument holds, using max |R| and bounding from both sides.

Exercise 1: Effect of Discount and Noise



discount（gamma小）越大代表會越注重眼前，
越小會越注重長遠

(a) Prefer the close exit (+1), risking the cliff (-10)

(b) Prefer the close exit (+1), but avoiding the cliff (-10)

(c) Prefer the distant exit (+10), risking the cliff (-10)

(d) Prefer the distant exit (+10), avoiding the cliff (-10)

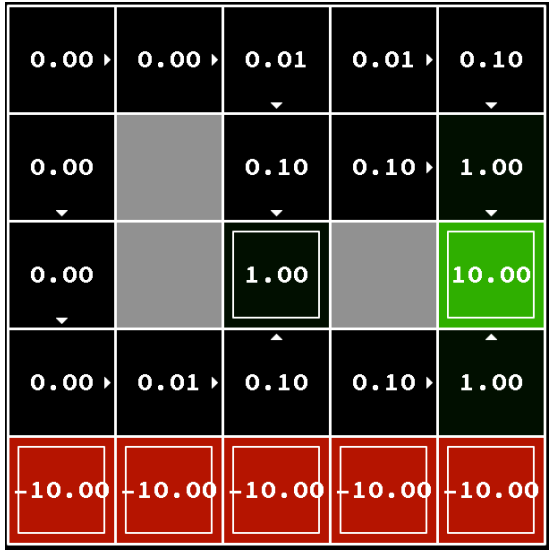
(1) γ = 0.1, noise = 0.5

(2) γ = 0.99, noise = 0

(3) γ = 0.99, noise = 0.5

(4) γ = 0.1, noise = 0

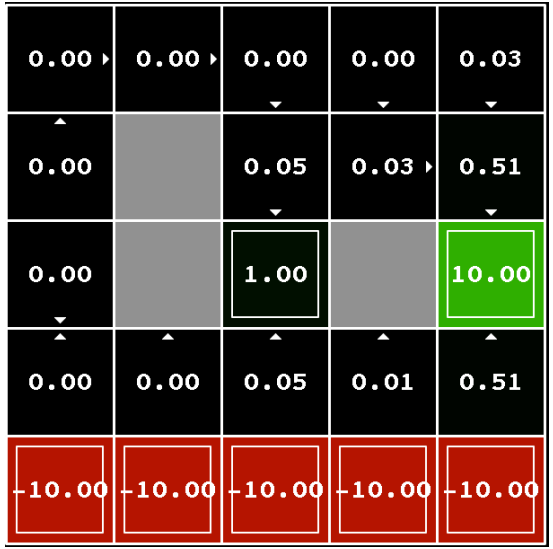
Exercise 1 Solution



(a) Prefer close exit (+1), risking the cliff (-10) --- (4) γ = 0.1, noise = 0

因為discount太大，走到10不值得

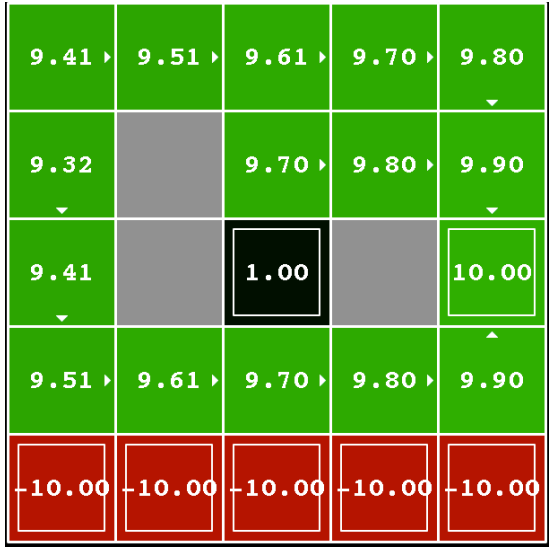
Exercise 1 Solution



(b) Prefer close exit (+1), avoiding the cliff (-10) --- (1) γ = 0.1, noise = 0.5

noise 太大，不敢走下面

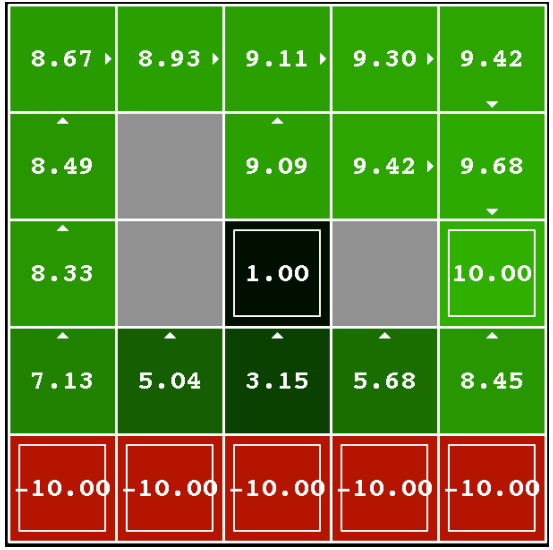
Exercise 1 Solution



(c) Prefer distant exit (+1), risking the cliff (-10) --- (2) γ = 0.99, noise = 0

因為沒有noise 所以會大膽地走在邊緣

Exercise 1 Solution



(d) Prefer distant exit (+1), avoid the cliff (-10) --- (3) γ = 0.99, noise = 0.5

Q-Values

$Q^*(s, a)$ = expected utility starting in s, taking action a, and (thereafter) acting optimally

Bellman Equation:

$$Q^*(s,a) = \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

Q-Value Iteration:

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

Q-Value Iteration

$$Q^*_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*_k(s',a'))$$

k = 100

Noise = 0.2

Discount = 0.9

Outline

- Optimal Control

  =

  given an MDP (S, A, P, R, γ, H)

  find the optimal policy π^*

- Exact Methods:
  - ***Value Iteration*** Bellman equations
  - Policy Iteration
    1. Policy Evaluation

For now: small discrete state-action spaces as they are simpler to get the main concepts across. We will consider large / continuous spaces later!

Policy Evaluation

- Recall value iteration:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V_{k-1}^*(s')\right)$$

- Policy evaluation for a given $\pi(s)$:

$$V_k^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V_{k-1}^\pi(s))$$

At convergence:

$$\forall s \quad V^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V^\pi(s))$$

Exercise 2

Consider a *stochastic* policy $\pi(a|s)$, where $\pi(a|s)$ is the probability of taking action a when in state s. Which of the following is the correct update to perform policy evaluation for this stochastic policy?

1. $V^{\pi}_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V^{\pi}_{k}(s')\right)$

Answer 2. $V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} \sum_a \pi(a|s) P(s'|s,a)\left(R(s,a,s') + \gamma V^{\pi}_{k}(s')\right)$

3. $V^{\pi}_{k+1}(s) \leftarrow \sum_a \pi(a|s) \max_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V^{\pi}_{k}(s')\right)$

Policy Iteration

One iteration of policy iteration:

- Policy evaluation for current policy π_k :
  - Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) \left[R(s, \pi(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

update policy

- Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

- Repeat until policy converges
- At convergence: optimal policy; and converges faster than value iteration under some conditions

Policy Evaluation Revisited

- *Idea 1:* modify Bellman updates

$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} P(s'|s,\pi(s)) \left[R(s,\pi(s),s') + \gamma V_i^\pi(s')\right]$$

- Idea 2: it is just a linear system, solve with Numpy (or whatever)

  variables: $V^\pi(s)$

  constants: P, R

$$\forall s \quad V^\pi(s) = \sum_{s'} P(s'|s,\pi(s)) \left[R(s,\pi(s),s') + \gamma V^\pi(s')\right]$$

Policy Iteration Guarantees

Policy Iteration iterates over:

- Policy evaluation for current policy π_k :
  - Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) \left[R(s, \pi(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

- Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

Theorem. Policy iteration is guaranteed to converge and at convergence, the current policy and its value function are the optimal policy and the optimal value function!

Proof sketch:

(1) Guarantee to converge: In every step the policy improves. This means that a given policy can be encountered at most once. This means that after we have iterated as many times as there are different policies, i.e., (number actions)$^{(\text{number states})}$, we must be done and hence have converged.

(2) Optimal at convergence: by definition of convergence, at convergence $\pi_{k+1}(s) = \pi_k(s)$ for all states s. This means $\forall s \;\; V^{\pi_k}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i^{\pi_k}(s')\right]$

Hence V^{π_k} satisfies the Bellman equation, which means V^{π_k} is equal to the optimal value function V*.

Outline

- Optimal Control

  =

  given an MDP (S, A, P, R, γ, H)

  find the optimal policy π^*

- Exact Methods:
  - ✓ ***Value Iteration***
  - ✓ ***Policy Iteration***

Limitations:

states 跟 action 要夠小，不然沒辦法 train （memory）也會不夠

- Iteration over / storage for all states and actions: requires small, discrete state-action space　space * action 如果太大，就做不動了
- Update equations require access to dynamics model