Deep Learning
Lecture-16:
Reinforcement Learning

Datalab

Outline

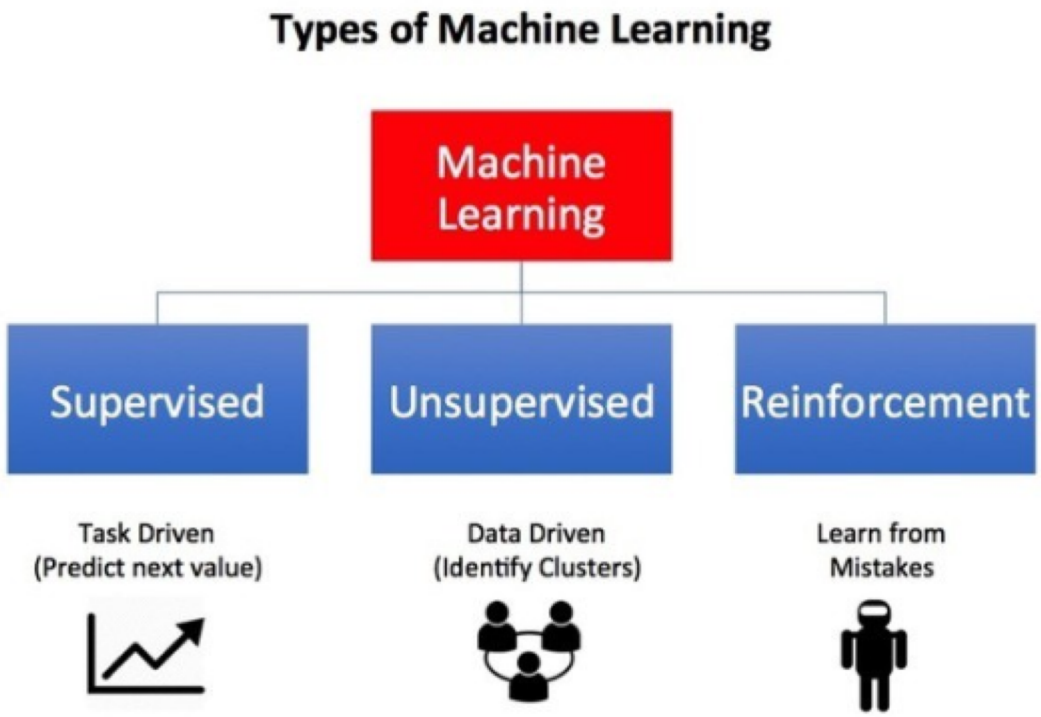
- Introduction of RL
- Markov Decision Process(MDP)
- Value Iteration
- Policy Iteration

Outline

- Introduction of RL
- Markov Decision Process(MDP)
- Value Iteration
- Policy Iteration

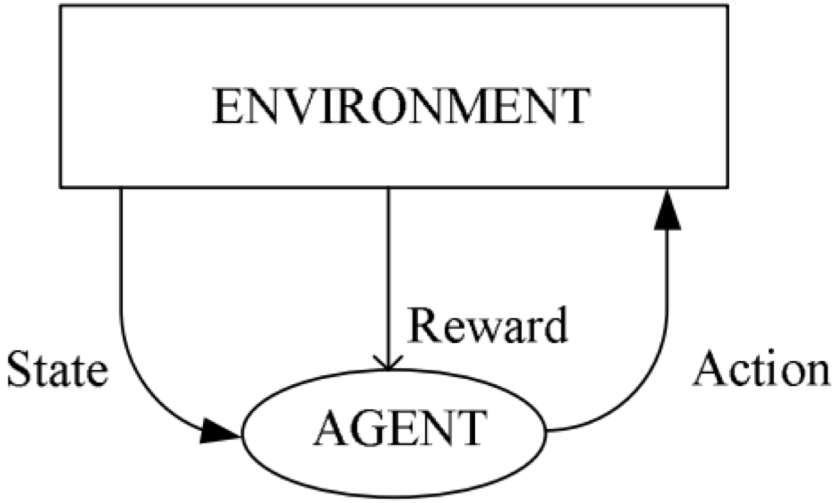
Introduction of RL

- Types of Machine Learning



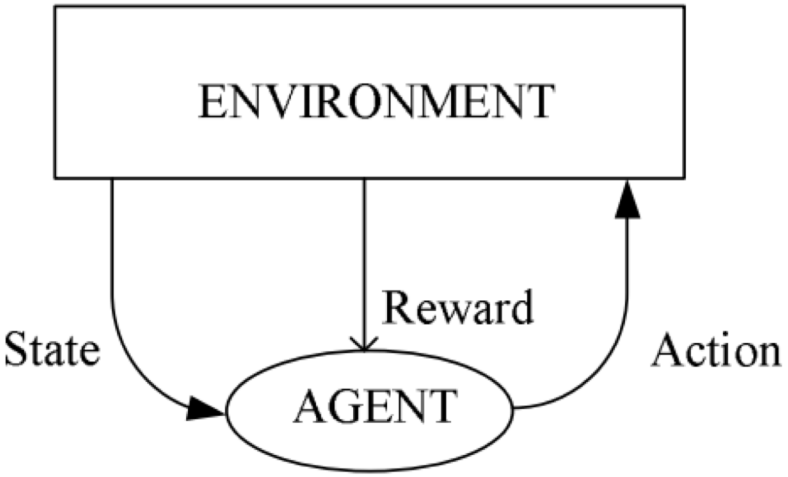
Introduction of RL

- An agent sees ***states*** $s^{(t)}$'s of an environment, takes ***actions*** $a^{(t)}$'s, and receives ***rewards*** $R^{(t)}$'s (or penalties)
  - Environment does ***not*** change over time
  - The state of the environment may change due to an action
  - Reward $R^{(t)}$ may depend on $s^{(t+1)}, s^{(t)}, \cdots$ or $a^{(t)}, a^{(t-1)}, \cdots$



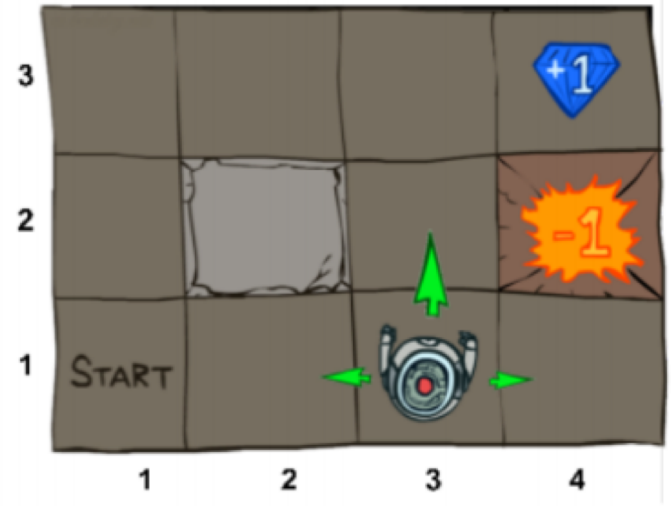
Introduction of RL

- Goal: to learn the best ***policy*** $\pi^*(s^{(t)}) = a^{(t)}$ that maximizes the ***total*** reward $\sum_t R^{(t)}$
- Training:
  1. Perform trial-and-error runs
  2. Learn from the experience



Compared to Supervised Learning

- $\pi^*(\boldsymbol{s}^{(t)}) = \boldsymbol{a}^{(t)}$ maximizing total reward $\sum_t R^{(t)}$ vs. $f^*(\boldsymbol{x}^{(i)}) = \boldsymbol{y}^{(i)}$ minimizing total loss
- Examples $\boldsymbol{x}^{(i)}$'s are i.i.d., but not in RL
  - $\boldsymbol{s}^{(t+1)}$ may depend on $\boldsymbol{s}^{(t)}, \boldsymbol{s}^{(t-1)}, \cdots$ and $\boldsymbol{a}^{(t)}, \boldsymbol{a}^{(t-1)}, \cdots$
- No ***what*** to predict ($\boldsymbol{y}^{(i)}$'s), just ***how good*** a prediction is ($R^{(t)}$'s)
  - $R^{(t)}$'s are also called the ***critics***



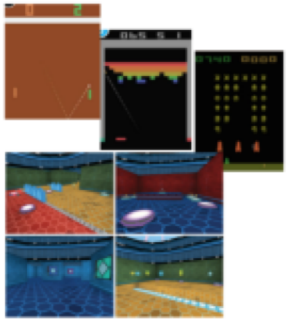
Applications

Kohl and Stone, 2004

Ng et al, 2004

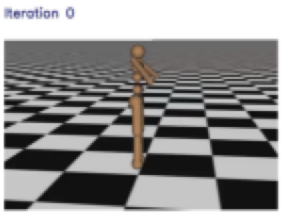
Tedrake et al, 2005

Kober and Peters, 2009

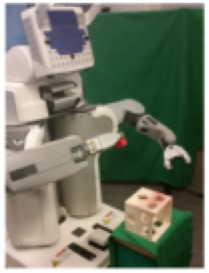


Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

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)

Outline

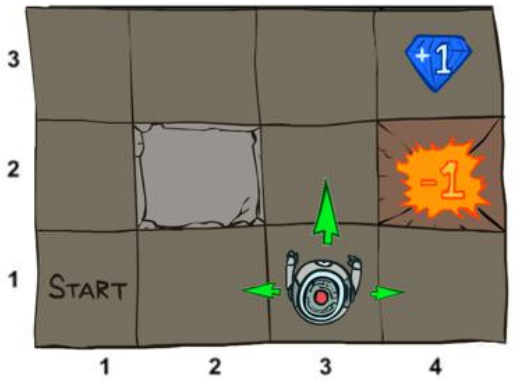
- Introduction of RL
- Markov Decision Process(MDP)
- Value Iteration
- Policy Iteration

MDP

- Definition:
  - A random process is called a ***Markov process*** if it satisfies the ***Markov property***:

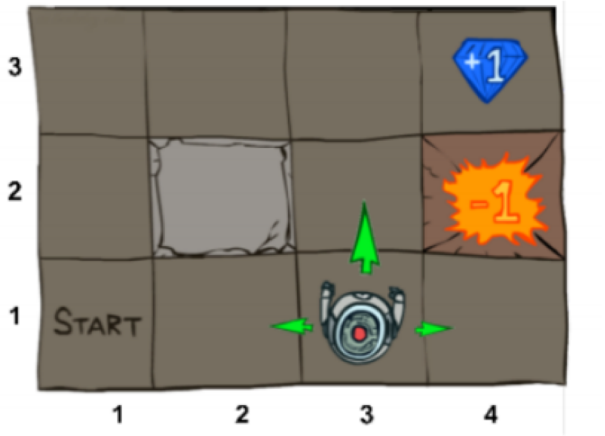
$$\mathrm{P}(\boldsymbol{s}^{(t+1)} | \boldsymbol{s}^{(t)}, \boldsymbol{s}^{(t-1)}, \cdots) = \mathrm{P}(\boldsymbol{s}^{(t+1)} | \boldsymbol{s}^{(t)})$$

    - "Markov" generally means that given the present state, the next state and the past are independent



MDP

- A Markov decision process (MDP) is defined by
  - $\mathbb{S}$ the state space; $\mathbb{A}$ the action space
  - Start state $\boldsymbol{s}^{(0)}$
  - $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ the ***transition distribution*** controlled by actions; fixed over time t
  - $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') \in \mathbb{R}$ (or simply $R(\boldsymbol{s}')$) the deterministic reward function
  - $\gamma \in [0,1]$ is the ***discount factor***
  - $H \in \mathbb{N}$ the ***horizon***; can be infinite
- An absorbing/terminal state transit to itself with probability 1



MDP

- Given a policy $\pi(s) = a$, an MDP proceeds as follows:

$$s^{(0)} \xrightarrow{a^{(0)}} s^{(1)} \xrightarrow{a^{(1)}} \cdots \xrightarrow{a^{(H-1)}} s^{(H)},$$

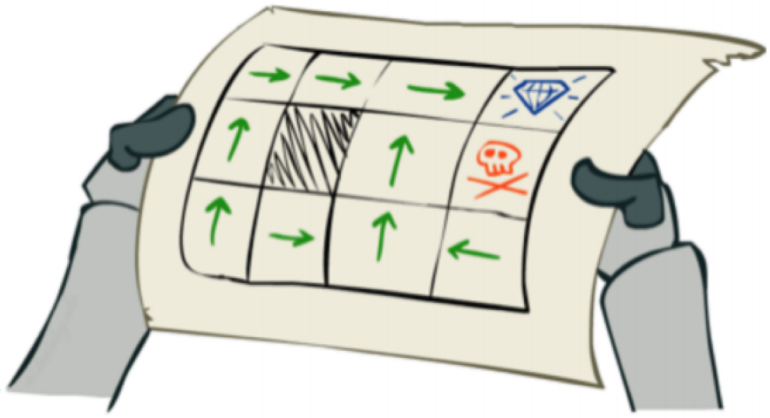
with the accumulative reward

$$R(s^{(0)}, a^{(0)}, s^{(1)}) + \gamma R(s^{(1)}, a^{(1)}, s^{(2)}) + \cdots + \gamma^{H-1} R(s^{(H-1)}, a^{(H-1)}, s^{(H)})$$

- *To accrue rewards as soon as possible (prefer a shot path)*
- *Different accumulative rewards in different trials*

MDP

π



Goal of MDP

- Given a policy π, the expected accumulative reward collected by taking actions following π can be express by:

$$V_\pi = \mathrm{E}_{\mathbf{s}^{(0)},\ldots,\mathbf{s}^{(H)}}\left(\sum_{t=0}^{H}\gamma^t R(\mathbf{s}^{(t)},\pi(\mathbf{s}^{(t)}),\mathbf{s}^{(t+1)});\pi\right)$$

- Goal: to find the optimal policy

$$\pi^* = \arg\max_\pi V_\pi$$

- How?

Outline

- Introduction of RL
- Markov Decision Process(MDP)
- Value Iteration
- Policy Iteration

Value iteration

- Optimal Value Function

$$\pi^* = \arg\max_{\pi} \mathrm{E}_{\mathbf{s}^{(0)},\cdots,\mathbf{s}^{(H)}} \left(\sum_{t=0}^{H} \gamma^t R(\mathbf{s}^{(t)}, \pi(\mathbf{s}^{(t)}), \mathbf{s}^{(t+1)}); \pi \right)$$

  - ***Optimal value function***:

$$V^{*(h)}(\boldsymbol{s}) = \max_{\pi} \mathrm{E}_{\mathbf{s}^{(1)},\cdots,\mathbf{s}^{(h)}} \left(\sum_{t=0}^{h} \gamma^t R(\mathbf{s}^{(t)}, \pi(\mathbf{s}^{(t)}), \mathbf{s}^{(t+1)}) | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi \right)$$

    - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$ and acting optimally for h steps
  - Having $V^{*(H-1)}(\boldsymbol{s})$ for each $\boldsymbol{s}$, we cab solve π^* easily by

$$\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(H-1)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

  - How to obtain $V^{*(H-1)}(\boldsymbol{s})$ for each $\boldsymbol{s}$?

Value iteration

- Dynamic Programming

$$V^{*(h)}(\boldsymbol{s}) = \max_{\pi} \mathrm{E}_{\mathbf{s}^{(1)},\cdots,\mathbf{s}^{(H)}} \left(\sum_{t=0}^{h} \gamma^t R(\mathbf{s}^{(t)}, \pi(\mathbf{s}^{(t)}), \mathbf{s}^{(t+1)}) | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi \right)$$

- $h = H - 1$:

$$V^{*(H-1)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(H-2)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

- $h = H - 2$:

$$V^{*(H-2)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(H-3)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

- $h = 0$:

$$V^{*(0)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(-1)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

- $h = -1$:

$$V^{*(-1)}(\boldsymbol{s}) = 0, \forall \boldsymbol{s}$$

Value iteration

- Dynamic Programming

$$V^{*(h)}(\boldsymbol{s}) = \max_{\pi} \mathrm{E}_{\mathbf{s}^{(1)},\cdots,\mathbf{s}^{(H)}} \left(\sum_{t=0}^{h} \gamma^t R(\mathbf{s}^{(t)}, \pi(\mathbf{s}^{(t)}), \mathbf{s}^{(t+1)}) | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi \right)$$

- $h = H-1$:

$$V^{*(H-1)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(H-2)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

- $h = H-2$:

$$V^{*(H-2)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(H-3)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

- $h = 0$:

$$V^{*(0)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(-1)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

- $h = -1$:

$$V^{*(-1)}(\boldsymbol{s}) = 0, \forall \boldsymbol{s}$$

Value iteration

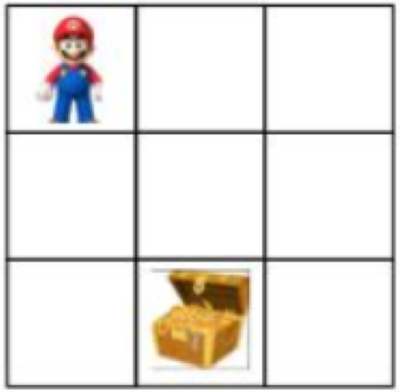
- Algorithm

Input: MDP $(\mathbb{S}, \mathbb{A}, \mathrm{P}, R, \gamma, H \to \infty)$
Output: $\pi^*(\boldsymbol{s})$'s for all $\boldsymbol{s}$'s

For each state $\boldsymbol{s}$, initialize $V^*(\boldsymbol{s}) \leftarrow 0$;
repeat
  foreach $\boldsymbol{s}$ **do**
    $V^*(\boldsymbol{s}) \leftarrow \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$;
  end
until $V^*(\boldsymbol{s})$*'s converge*;
foreach $\boldsymbol{s}$ **do**
  $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$;
end

Examples for value iteration

- Game: Super Mario
  - State: each grid(9 states)
  - Action: up, down, left, right(4 actions) in each state(excluding terminal state)
  - P(s'|s, a) = 1(no noise)
  - Get -1 reward after each move
  - Terminal state(box): Reward = 0

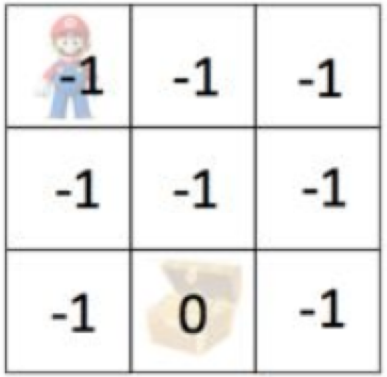


Examples for value iteration

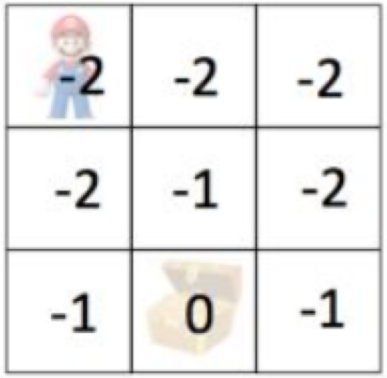
- Game: Super Mario

$$V^{*(h)}(\boldsymbol{s}) = \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^{*(h-1)}(\boldsymbol{s}')], \forall \boldsymbol{s}$$

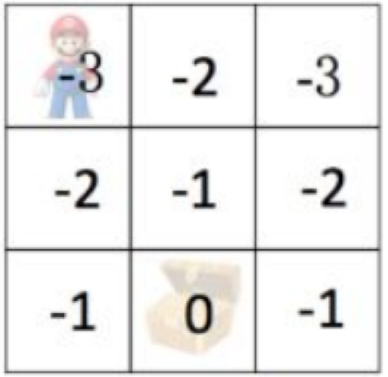
初始化



第一轮迭代



第二轮迭代



第三轮迭代

Examples for value iteration

- Recall the algorithm

Input: MDP $(\mathbb{S}, \mathbb{A}, \mathrm{P}, R, \gamma, H \to \infty)$
Output: $\pi^*(s)$'s for all s's

For each state s, initialize $V^*(s) \leftarrow 0$;

初始化

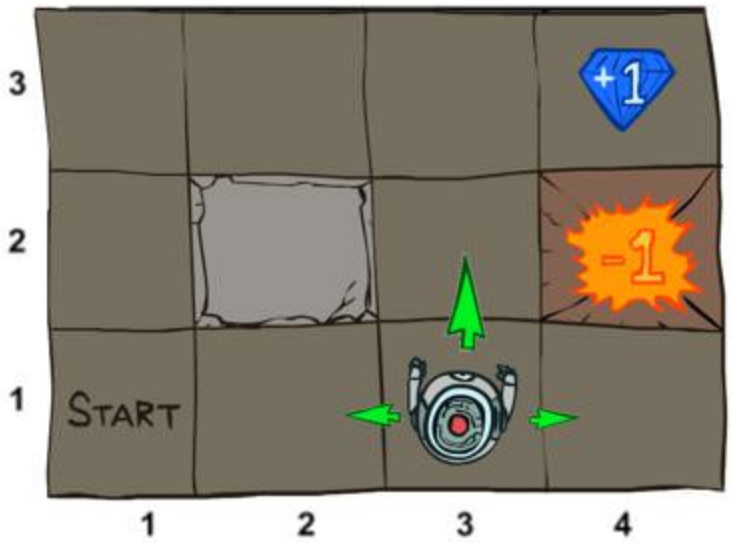
repeat
  foreach s **do**
    $V^*(s) \leftarrow \max_a \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V^*(s')]$;
  end
until $V^*(s)$'s *converge*;

第一轮迭代 第二轮迭代 第三轮迭代

foreach s **do**
  $\pi^*(s) \leftarrow \arg\max_a \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V^*(s')]$;
end

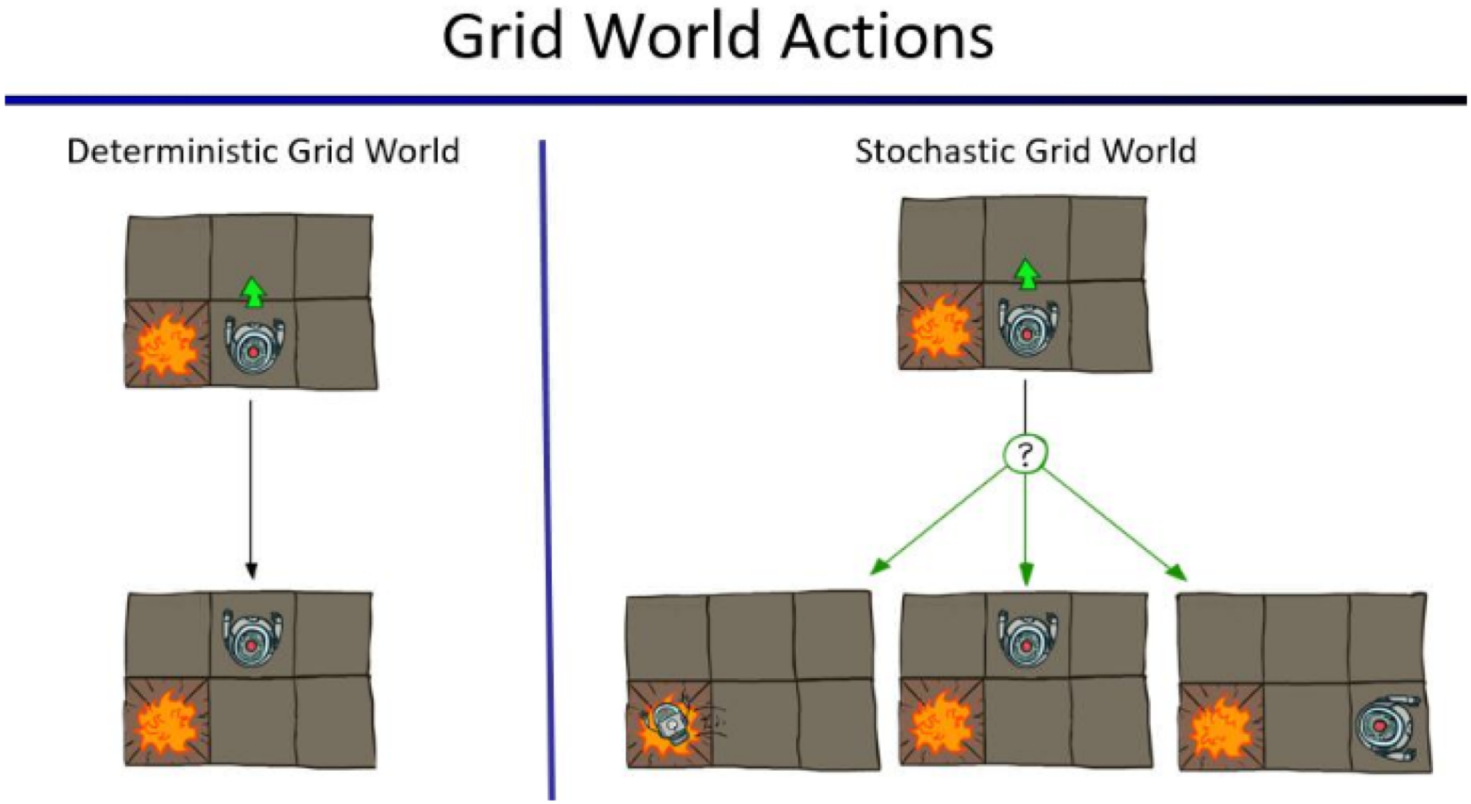
Examples for value iteration

- Game: Gridworld
  - Noise of transition probability $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$: 0.2
  - γ: 0.9



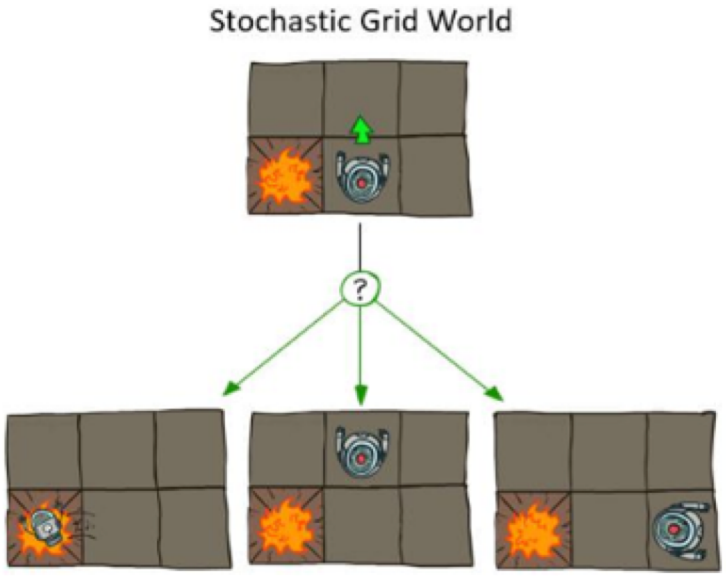
Examples for value iteration

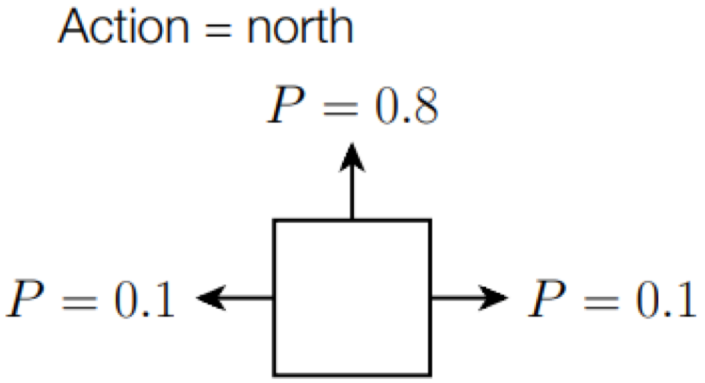
- Game: Gridworld
  - Noise of transition probability $\mathrm{P}(s'|s;a)$: 0.2



Examples for value iteration

- Game: Gridworld
  - Noise: 0.2(0.1 go to the left, 0.1 go to the right)

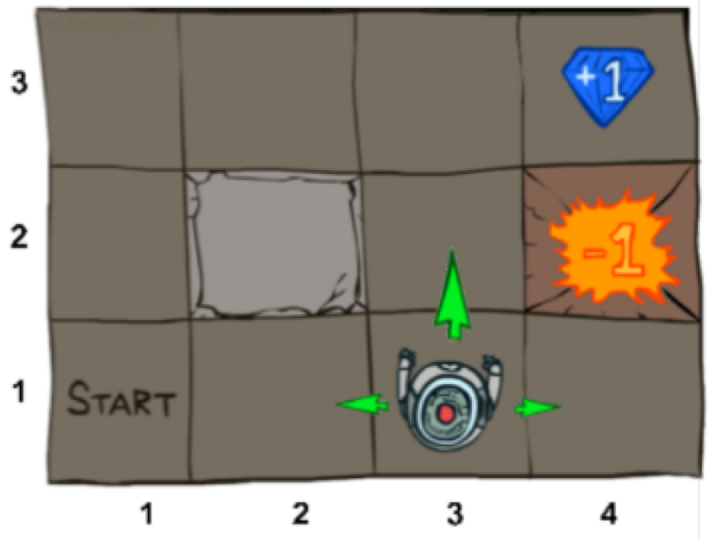


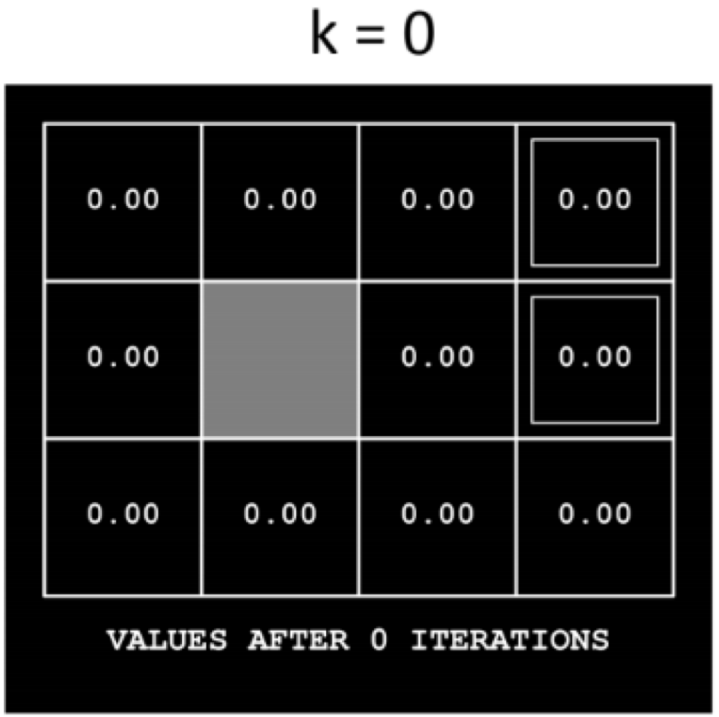


Examples for value iteration

- Game: Gridworld

$$V_0(s) \leftarrow 0$$

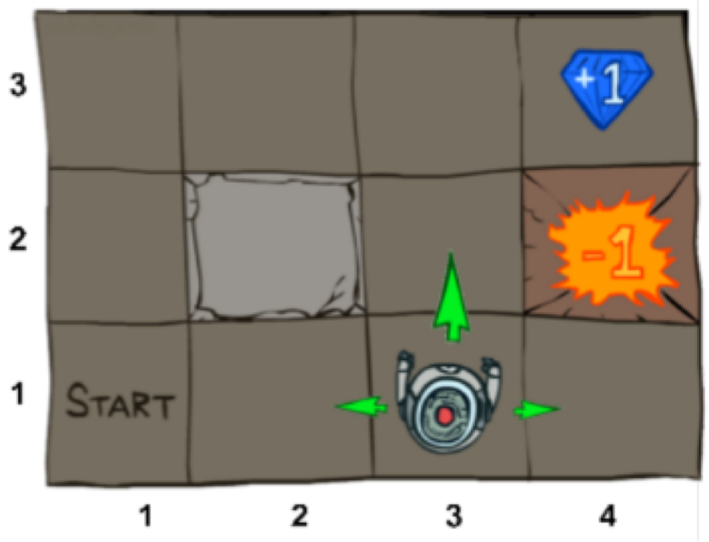




Examples for value iteration

- Game: Gridworld

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



k = 1

0.00	0.00	0.00	1.00
0.00		0.00	-1.00
0.00	0.00	0.00	0.00

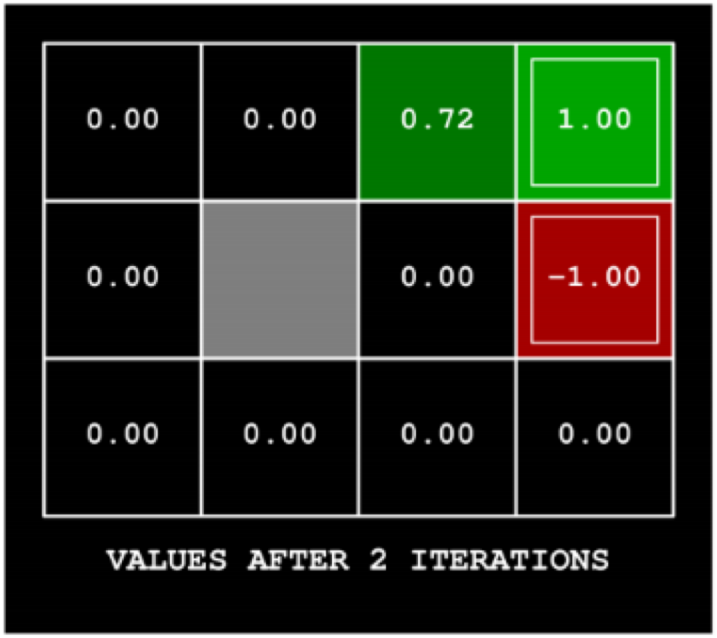
VALUES AFTER 1 ITERATIONS

Examples for value iteration

- Game: Gridworld

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

k = 2

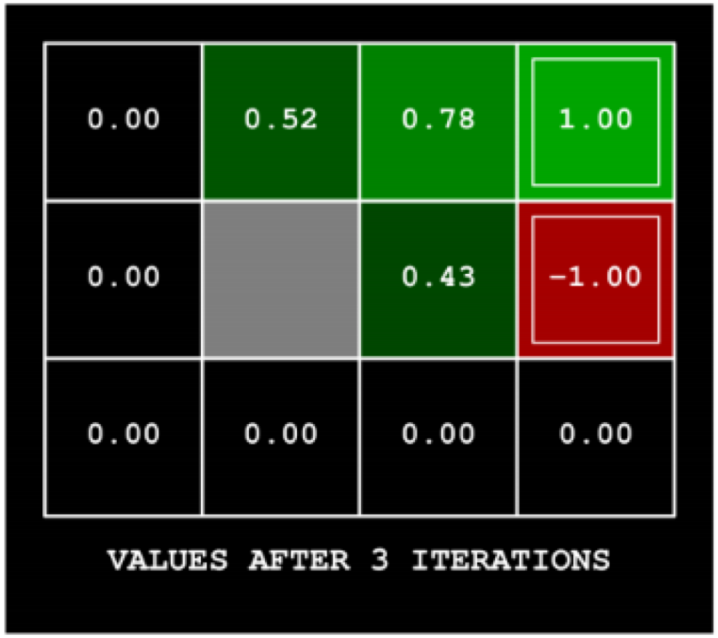


Examples for value iteration

- Game: Gridworld

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

k = 3



Examples for value iteration

- Game: Gridworld

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

k = 4

Examples for value iteration

- Game: Gridworld

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

k = 100

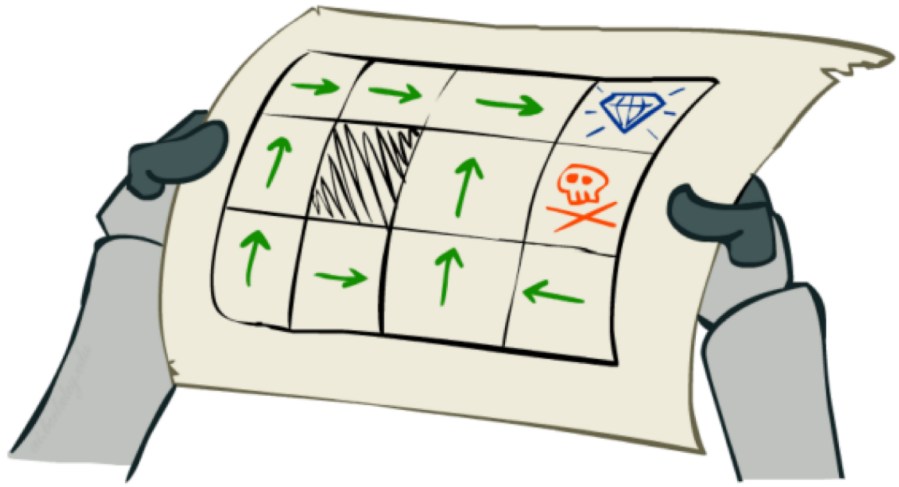
Examples for value iteration

- Game: Gridworld
  - Policy Extraction

foreach s **do**

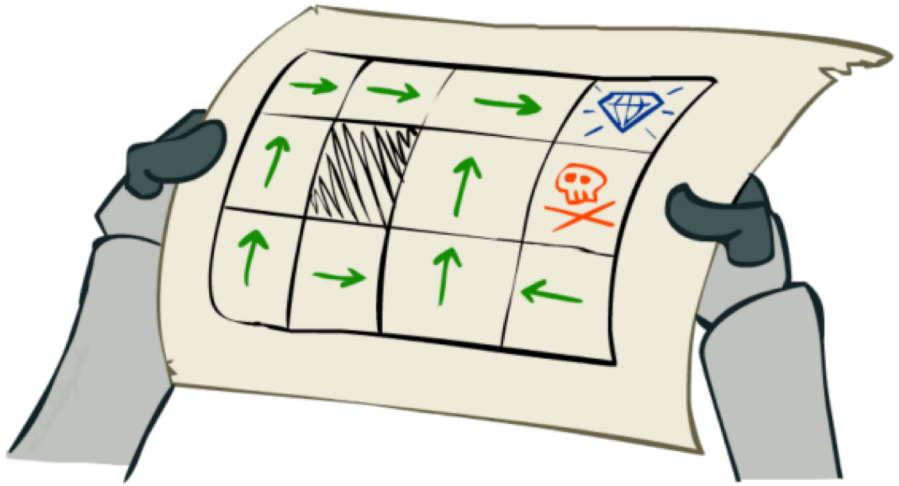
$$\pi^*(s) \leftarrow \arg\max_a \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V^*(s')]$$

end



Examples for value iteration

- Game: Gridworld
  - We get the optimal policy after 100 iterations
  - The policy π gives an action for each state
  - Following the optimal π, we can get the maximum expected total reward



Examples for value iteration

- Theorem
  - Value iteration converges and gives the optimal policy π^* when $H \rightarrow \infty$
  - Proof: Please see the slide of the teacher

Outline

- Introduction of RL
- Markov Decision Process(MDP)
- Value Iteration
- Policy Iteration

Policy iteration

- Goal

- Given an MDP $(\mathbb{S}, \mathbb{A}, \mathrm{P}, R, \gamma, H \to \infty)$
- Expected accumulative reward collected by taking actions following a policy π:

$$V_\pi = \mathrm{E}_{\mathbf{s}^{(0)},\cdots,\mathbf{s}^{(H)}} \left(\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}^{(t)}, \pi(\mathbf{s}^{(t)}), \mathbf{s}^{(t+1)}); \pi \right)$$

- Goal: to find the optimal policy

$$\pi^* = \arg\max_\pi V_\pi$$

Policy iteration

- Goal

- Given an MDP $(\mathbb{S}, \mathbb{A}, \mathrm{P}, R, \gamma, H \to \infty)$
- Expected accumulative reward collected by taking actions following a policy π:

$$V_\pi = \mathrm{E}_{\mathbf{s}^{(0)},\cdots,\mathbf{s}^{(H)}} \left(\sum_{t=0}^{\infty} \gamma^t R(\mathbf{s}^{(t)}, \pi(\mathbf{s}^{(t)}), \mathbf{s}^{(t+1)}); \pi \right)$$

- Goal: to find the optimal policy

$$\pi^* = \arg\max_\pi V_\pi$$

Policy iteration

- Policy Evaluation
- Policy Improvement

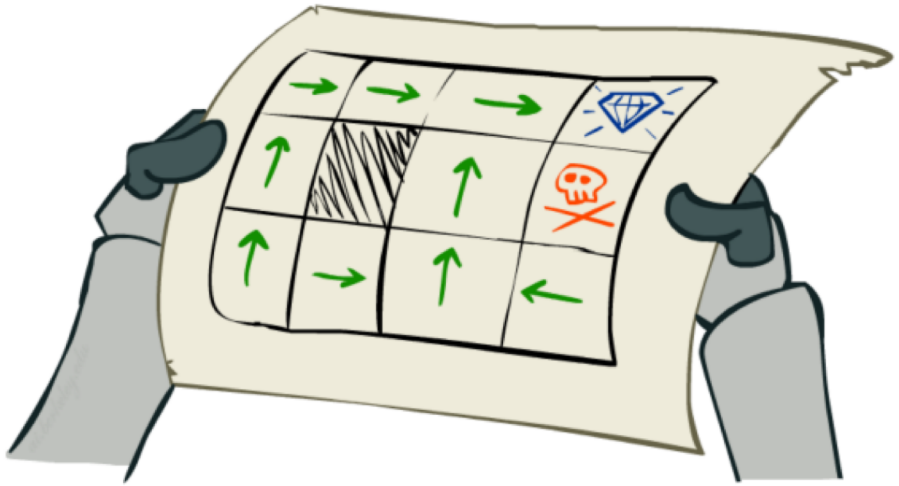
Policy iteration

- Policy Evaluation
  - How to evaluate the value function of a given π?

$$V_\pi(s) = \sum_{s'} \mathrm{P}(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')], \forall s$$

  - Recall optimal value function in value iteration

$$V^*(s) \leftarrow \max_a \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V^*(s')]$$

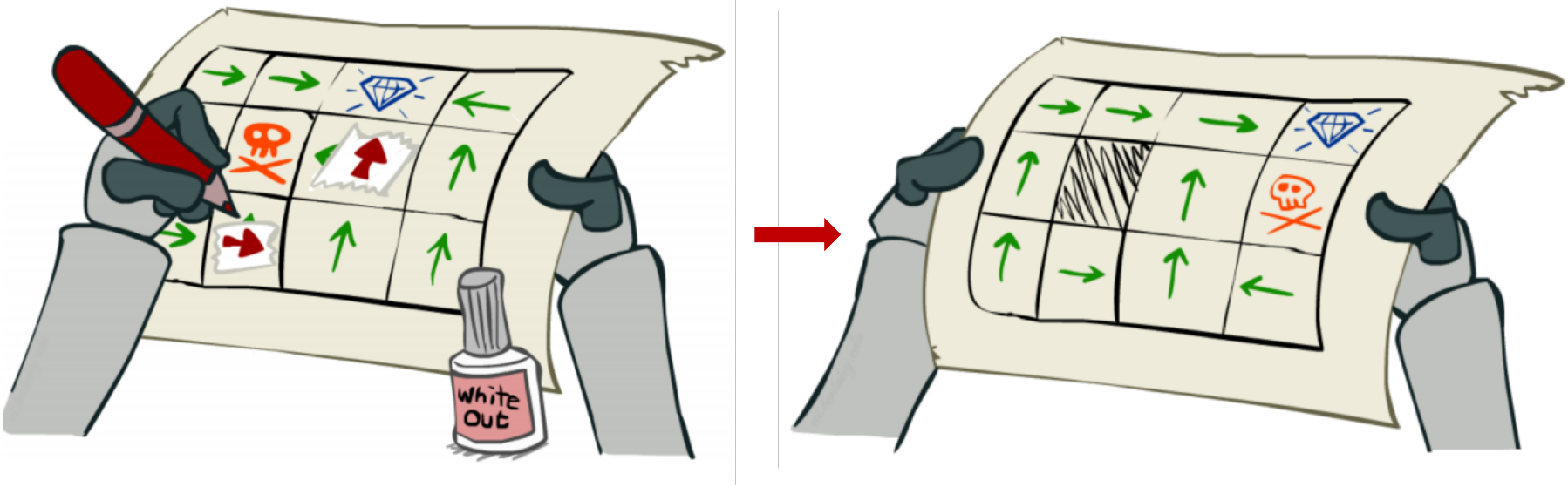


Policy iteration

- Policy Improvement

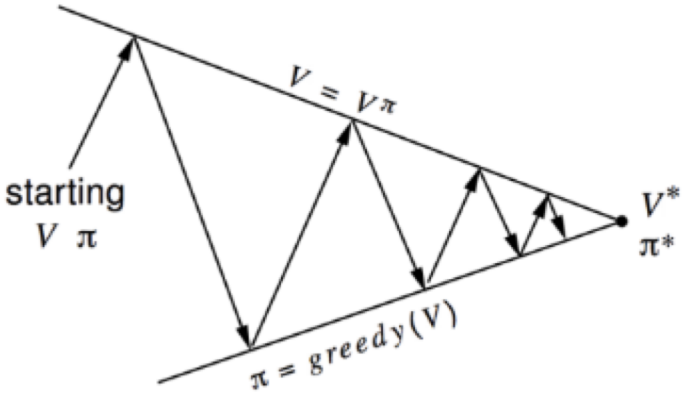
Update rule: for all s do

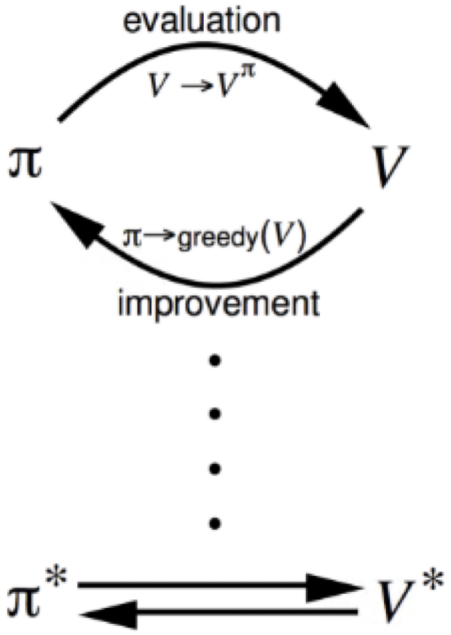
$$\hat{\pi}(s) \leftarrow \arg\max_{a} \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V_{\pi}(s')]$$



Policy iteration

- Introduction





Policy iteration

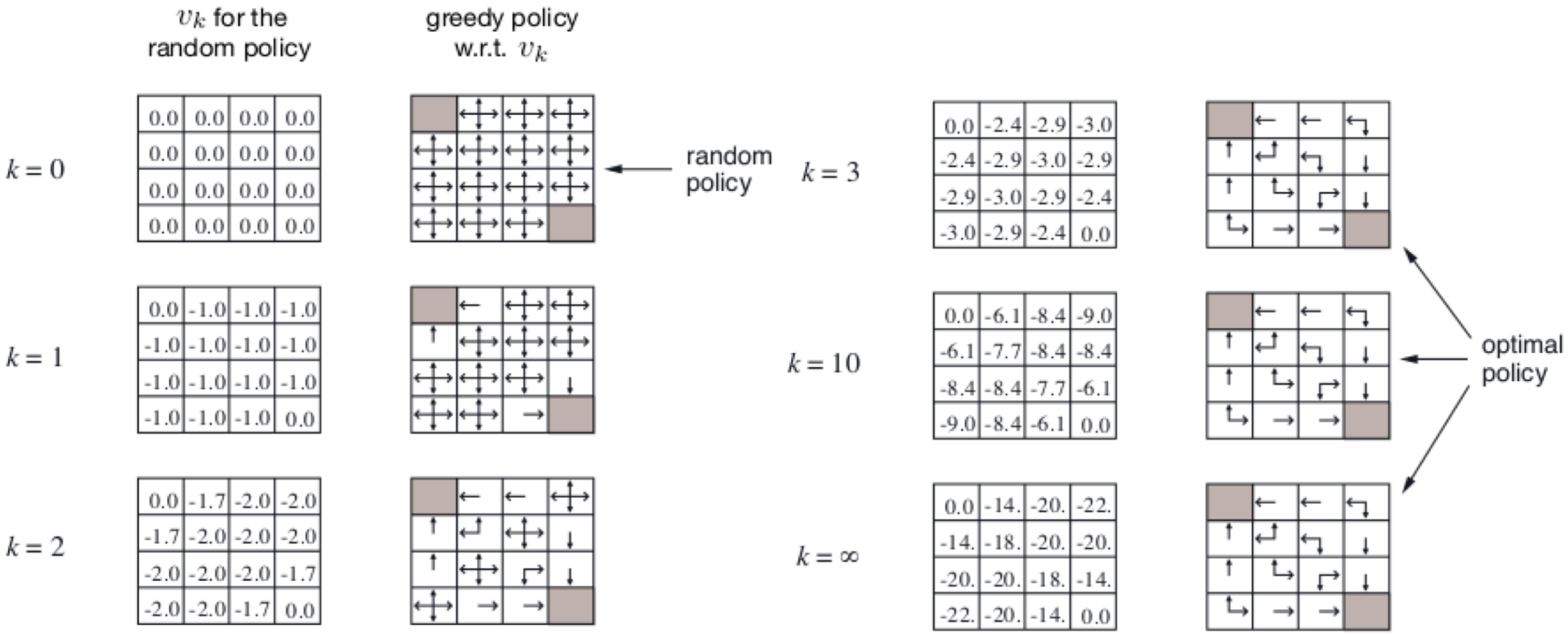
- Algorithm

Input: MDP $(\mathbb{S}, \mathbb{A}, \mathrm{P}, R, \gamma, H \to \infty)$
Output: $\pi(s)$'s for all s's

For each state s, initialize $\pi(s)$ randomly;
repeat
  For each state s, initialize $V_\pi(s) \leftarrow 0$;
  repeat
    foreach s **do**
      $V_\pi(s) \leftarrow \sum_{s'} \mathrm{P}(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')]$;
    end
  until $V_\pi(s)$*'s converge*;
  foreach s **do**
    $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V_\pi(s')]$;
  end
until $\pi(s)$*'s converge*;

Examples for policy iteration

- Gridworld



Examples for policy iteration

- Theorem
  - Policy iteration converges and gives the optimal policy π^* when $H \rightarrow \infty$.
  - Proof: Please see the slide of the teacher

Value iteration VS. Policy iteration

- Difference:

1. Initialization
   $v(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$
      For each $s \in \mathcal{S}$:
         $temp \leftarrow v(s)$
         $v(s) \leftarrow \sum_{s'} p(s'|s, \pi(s)) \left[r(s, \pi(s), s') + \gamma v(s') \right]$
         $\Delta \leftarrow \max(\Delta, |temp - v(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      $temp \leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s'} p(s'|s, a) \left[r(s, a, s') + \gamma v(s') \right]$
      If $temp \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return v and π; else go to 2

Figure 4.3: Policy iteration (using iterative policy evaluation) for v_*. This algorithm has a subtle bug, in that it may never terminate if the policy continually switches between two or more policies that are equally good. The bug can be fixed by adding additional flags, but it makes the pseudocode so ugly that it is not worth it. :-)

finding optimal value function

Initialize array v arbitrarily (e.g., $v(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
      $temp \leftarrow v(s)$
      $v(s) \leftarrow \max_a \sum_{s'} p(s'|s, a)[r(s, a, s') + \gamma v(s')]$
      $\Delta \leftarrow \max(\Delta, |temp - v(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
$\pi(s) = \arg\max_a \sum_{s'} p(s'|s, a) \left[r(s, a, s') + \gamma v(s') \right]$

Figure 4.5: Value iteration.

one policy update (extract policy from the optimal value function

Value iteration VS. Policy iteration

- Similarity:
  - Both are Model-based methods
  - We need to know $P(s'|s, a)$ and $R(s, a, s')$

Next class

- We need to know P(s'|s, a) and R(s, a, s') to use value iteration and policy iteration
- In the real world, the transition probability and reward function may be unknown, what should we do to get the optimal policy?

Thanks!