Introduction to Machine Learning

Lecture 18: Elementary Reinforcement Learning – Stochastic Environment

Dec 19, 2023

Jie Wang

Machine Intelligence Research and Applications Lab

Department of Electronic Engineering and Information Science (EEIS)

http://staff.ustc.edu.cn/~jwangx/

jiewangx@ustc.edu.cn

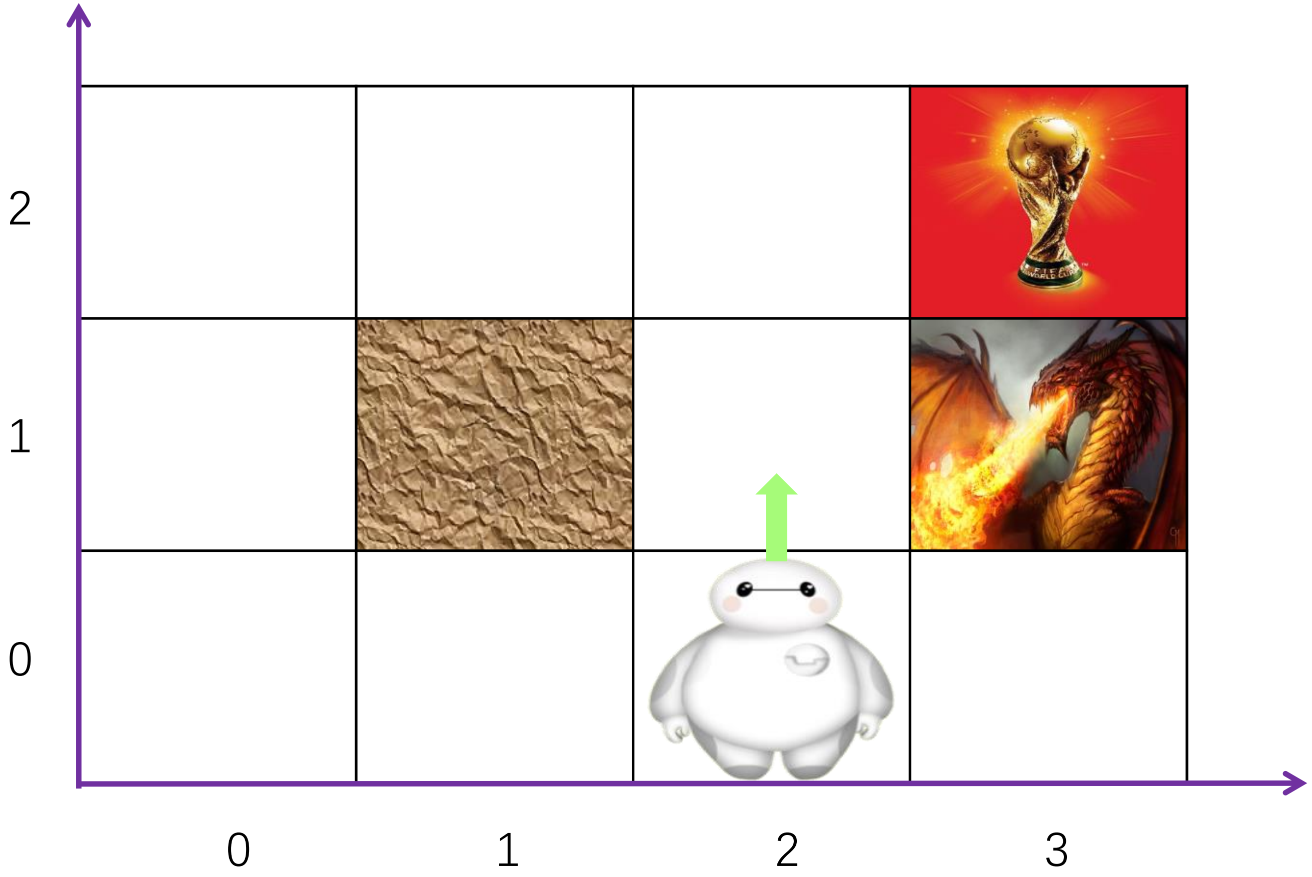
Machine Intelligence Research and Applications Lab

Contents

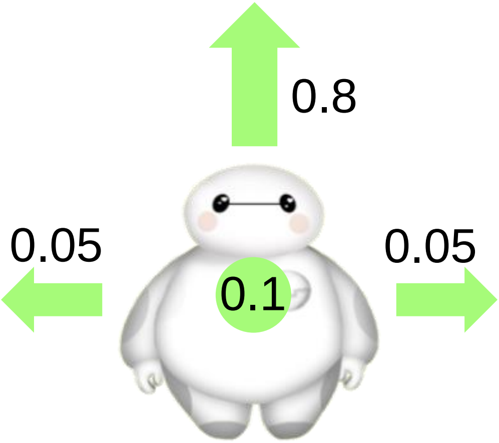
- **Stochastic Environment**
- **Planning Algorithms**
- **Learning Algorithms**

Stochastic Environment

Grid World



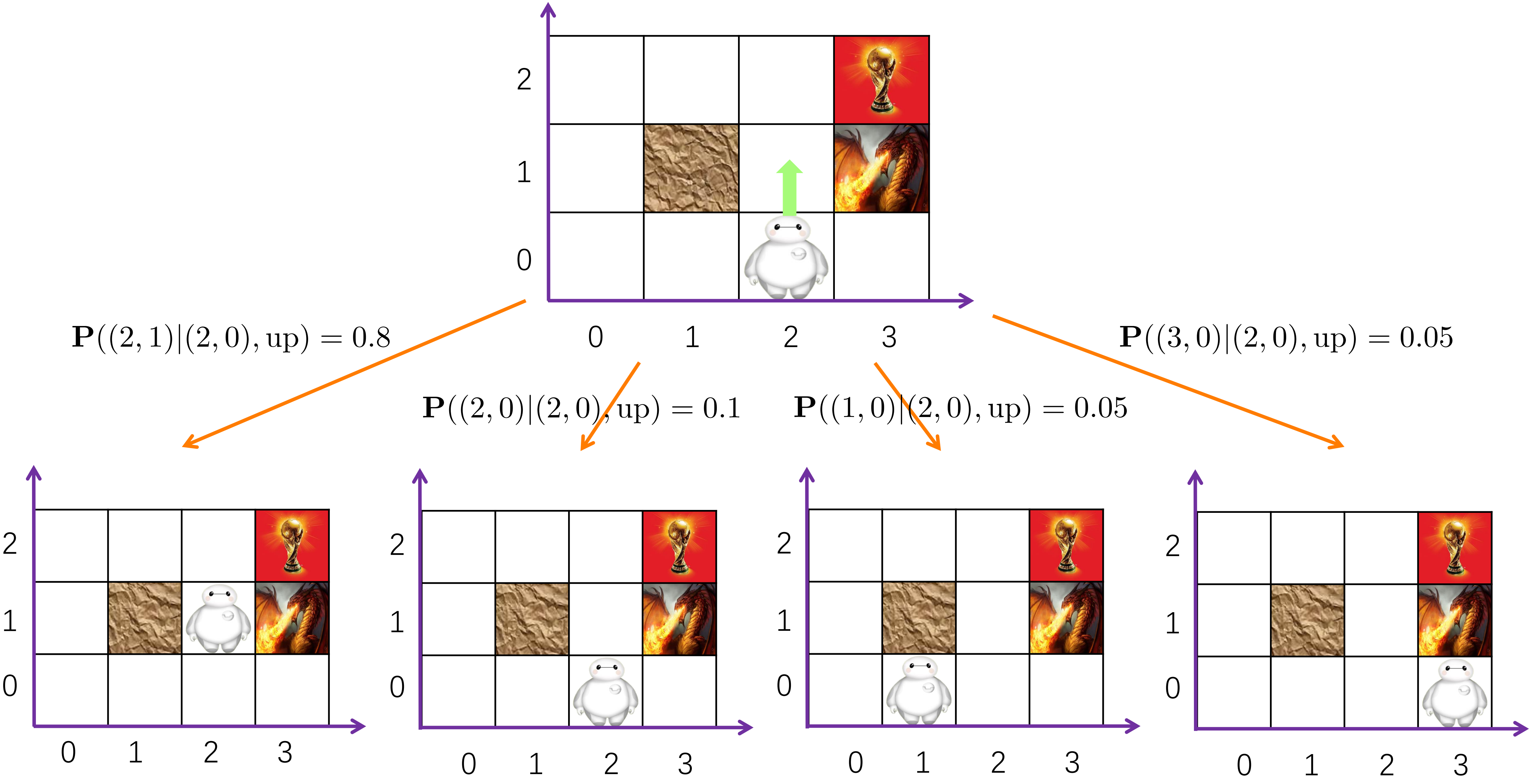
State Transition



State transition probabilities:

After the agent picks and performs a certain action, there are four possibilities for the next state: the destination state, the current state, the states to the right and left of the current state. If the states are reachable, the corresponding probabilities are 0.8, 0.1, 0.05, and 0.05, respectively; otherwise, the agent stays where it is.

State Transition



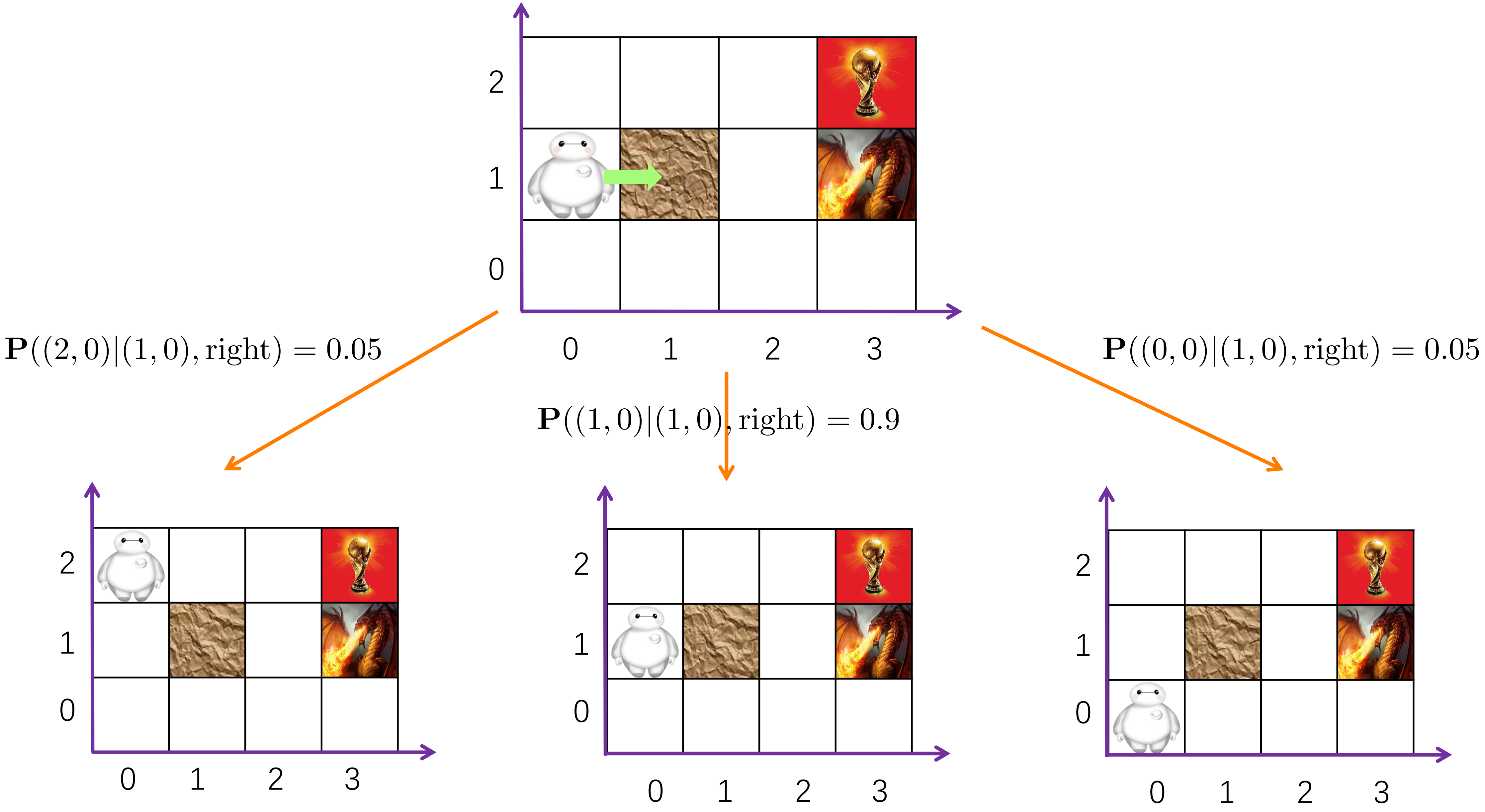
$$\mathbf{P}((2,1)|(2,0),\mathrm{up}) = 0.8$$

$$\mathbf{P}((2,0)|(2,0),\mathrm{up}) = 0.1$$

$$\mathbf{P}((1,0)|(2,0),\mathrm{up}) = 0.05$$

$$\mathbf{P}((3,0)|(2,0),\mathrm{up}) = 0.05$$

State Transition

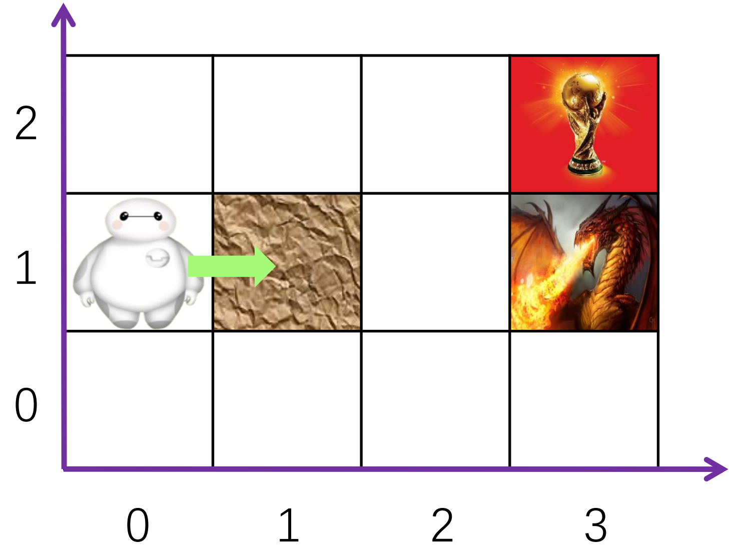


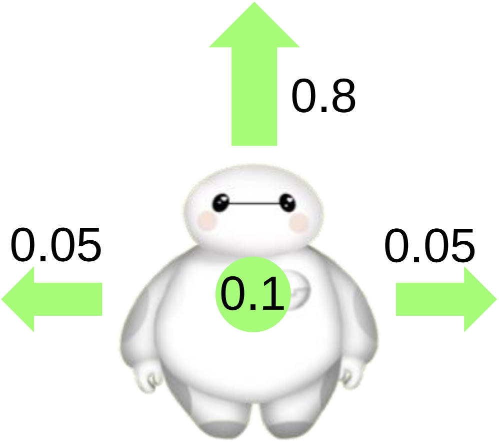
$\mathbf{P}((2,0)|(1,0),\text{right}) = 0.05$

$\mathbf{P}((1,0)|(1,0),\text{right}) = 0.9$

$\mathbf{P}((0,0)|(1,0),\text{right}) = 0.05$

Reward

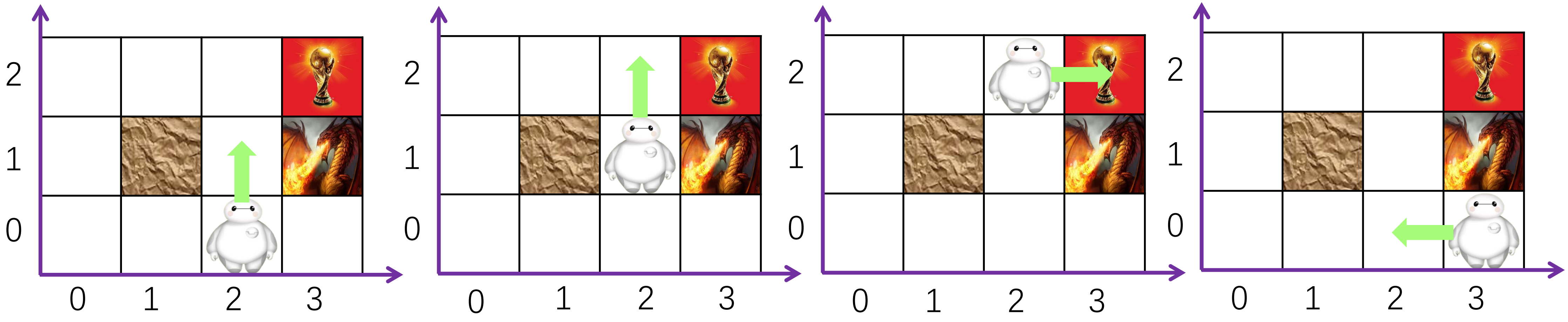




Reward:

After the agent picks and performs a certain action at its current state, it receives rewards of 100, -100, and 0, if it arrives at states (3,2), (2,2), and all the other states, respectively.

Reward



$$\mathbf{E}[r((2,0),\text{up})] = 0.8 \times 0 + 0.1 \times 0 + 0.05 \times 0 + 0.05 \times 0 = 0$$

$$\mathbf{E}[r((2,1),\text{up})] = 0.8 \times 0 + 0.1 \times 0 + 0.05 \times -100 + 0.05 \times 0 = -5$$

$$\mathbf{E}[r((2,2),\text{right})] = 0.8 \times 100 + 0.1 \times 0 + 0.05 \times 0 + 0.05 \times 0 = 80$$

$$\mathbf{E}[r((3,0),\text{left})] = 0.8 \times 0 + 0.1 \times 0 + 0.05 \times -100 + 0.05 \times 0 = -5$$

Markov Decision Process (MDP)

- Indeed, we have already introduced the so-called MDP, which is defined (rigorously) by
  - a set of states $\mathcal{S}$, possibly infinite
  - a set of actions $\mathcal{A}$, possibly infinite
  - an initial state $s_0 \in \mathcal{S}$
  - a transition probability $\mathbf{P}[s'|s, a]$: distribution over destination states $s' = \delta(s, a)$
  - a reward probability $\mathbf{P}[r|s, a]$: distribution over rewards $r' = r(s, a)$

MRT Chapter 14

- This model is Markovian because the transition and reward probabilities only depend on the current state and the action picked and performed at the current state, instead of the previous sequence of states and actions performed.

- In this lecture, we assume that
  - the states and the actions are finite

Value Function

- Suppose that a policy π is given.

- Starting from an arbitrary state s_t, the expected cumulative reward by following π is

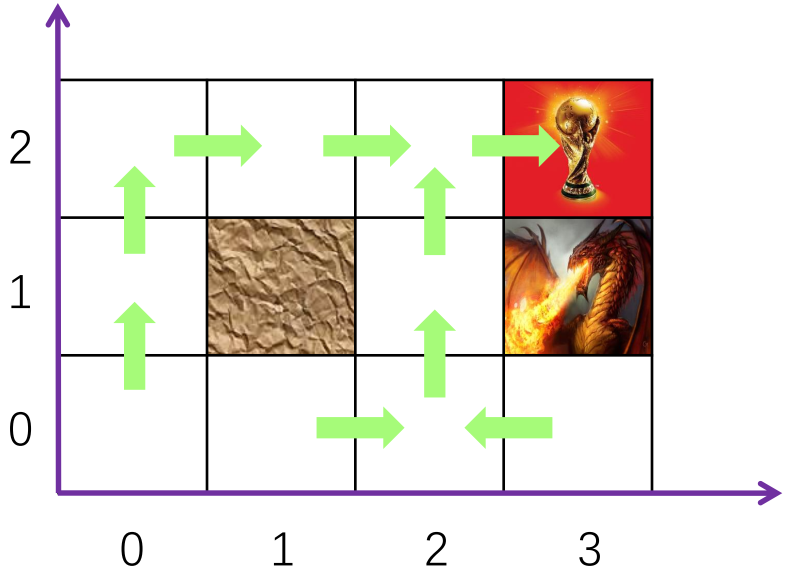
$$V^{\pi}(s_t) := \mathbf{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots | S_t = s_t] = \mathbf{E}\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | S_t = s_t\right]$$

$$\begin{tikzcd} s_t \arrow[r, "a_t", "r_t"'] & s_{t+1} \arrow[r, "a_{t+1}", "r_{t+1}"'] & s_{t+2} \arrow[r, "a_{t+2}", "r_{t+2}"'] & \cdots \end{tikzcd}$$

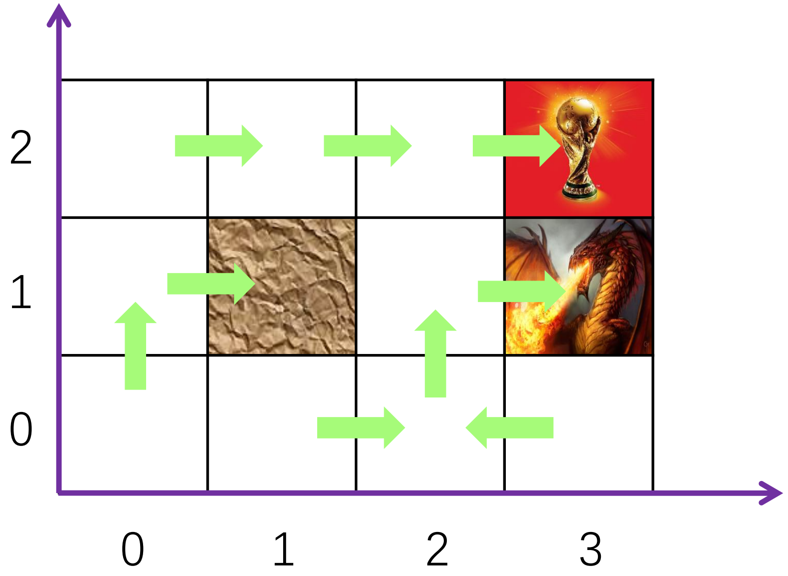
$$a_t = \pi(s_t) \quad r_t = r(s_t, a_t) \quad s_{t+1} = \delta(s_t, a_t)$$

random variable

Value Function



A “good” policy π_1



A bad policy π_2

$\gamma = 0.9$

How to find V^{π_1} and V^{π_2}?

Value Function

- Tower property

$$\mathbf{E}[X|Y] = \mathbf{E}[\mathbf{E}[X|Y,Z]|Y]$$

- A simpler version

$$\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X|Z]]$$

- Example: how to find the average height of the men in China?

$$\mathbf{E}[\text{height}] = \mathbf{E}\big[\mathbf{E}[\text{height}|\text{province}]\big] = \sum_{\text{province}} \mathbf{P}(\text{province})\mathbf{E}[\text{height}|\text{province}]$$

Value Function – Bellman Equation

- Starting from an arbitrary state s_t, the expected cumulative reward by following π is

$$V^{\pi}(s_t) := \mathbf{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots | S_t = s_t] = \mathbf{E}\left[\sum_{i=0}^{\infty} \gamma^i R_{t+i} | S_t = s_t\right]$$

$$s_t \xrightarrow[r_t]{a_t} s_{t+1} \xrightarrow[r_{t+1}]{a_{t+1}} s_{t+2} \xrightarrow[r_{t+2}]{a_{t+2}} \cdots$$

$$a_t = \pi(s_t) \quad r_t = r(s_t, a_t) \quad s_{t+1} = \delta(s_t, a_t)$$

random variable

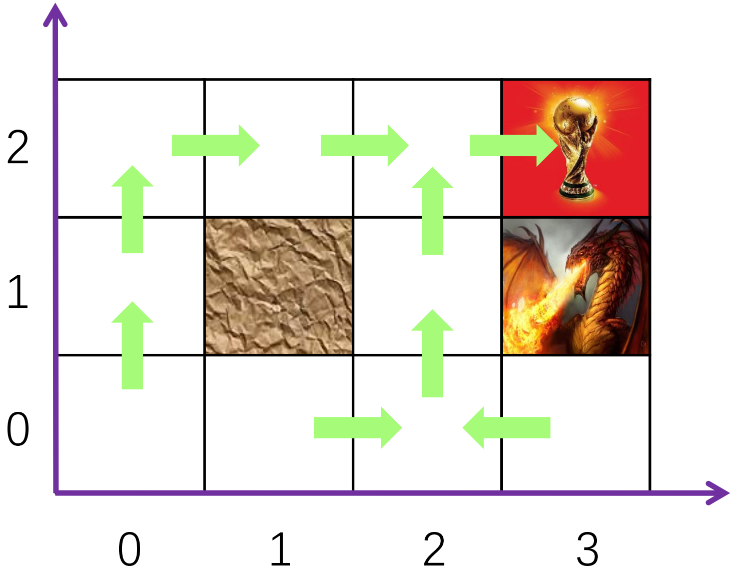
- **Bellman Equation**

$$V^{\pi}(s) = \mathbf{E}[r(s, \pi(s))] + \gamma \sum_{s'} \mathbf{P}(s' | s, \pi(s)) V^{\pi}(s')$$

Value Function – Bellman Equation

- **Bellman Equation**

$$V^{\pi}(s) = \mathbf{E}[r(s,\pi(s))] + \gamma \sum_{s'} \mathbf{P}(s'|s,\pi(s))V^{\pi}(s')$$



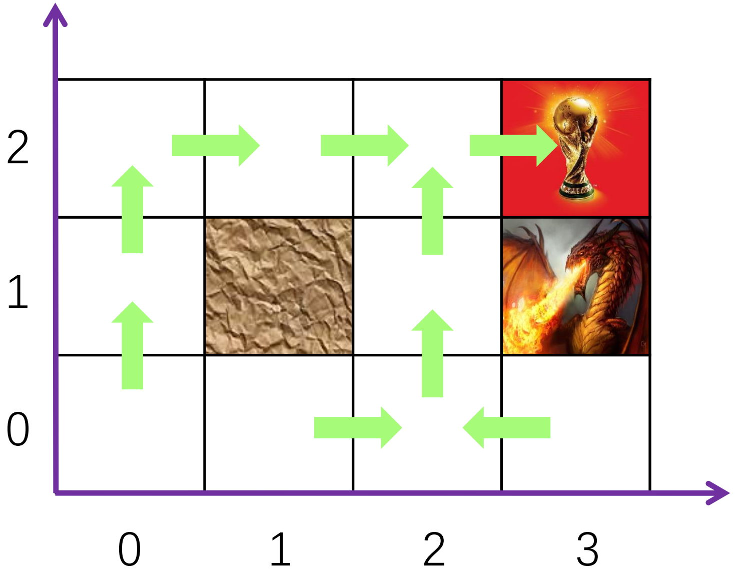
A “good” policy π_1?

$$\begin{pmatrix} V^{\pi_1}((0,0)) \\ V^{\pi_1}((1,0)) \\ V^{\pi_1}((2,0)) \\ V^{\pi_1}((3,0)) \\ V^{\pi_1}((0,1)) \\ V^{\pi_1}((1,1)) \\ V^{\pi_1}((2,1)) \\ V^{\pi_1}((3,1)) \\ V^{\pi_1}((0,2)) \\ V^{\pi_1}((1,2)) \\ V^{\pi_1}((2,2)) \\ V^{\pi_1}((3,2)) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -5 \\ 0 \\ 0 \\ -5 \\ 0 \\ 0 \\ 0 \\ 80 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} 0.15 & 0.05 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.1 & 0.05 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0.15 & 0 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.15 & 0.05 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.15 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0.15 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V^{\pi_1}((0,0)) \\ V^{\pi_1}((1,0)) \\ V^{\pi_1}((2,0)) \\ V^{\pi_1}((3,0)) \\ V^{\pi_1}((0,1)) \\ V^{\pi_1}((1,1)) \\ V^{\pi_1}((2,1)) \\ V^{\pi_1}((3,1)) \\ V^{\pi_1}((0,2)) \\ V^{\pi_1}((1,2)) \\ V^{\pi_1}((2,2)) \\ V^{\pi_1}((3,2)) \end{pmatrix}$$

Value Function – Bellman Equation

- **Bellman Equation**

$$V^{\pi}(s) = \mathbf{E}[r(s,\pi(s))] + \gamma \sum_{s'} \mathbf{P}(s'|s,\pi(s))V^{\pi}(s')$$



A “good” policy π_1

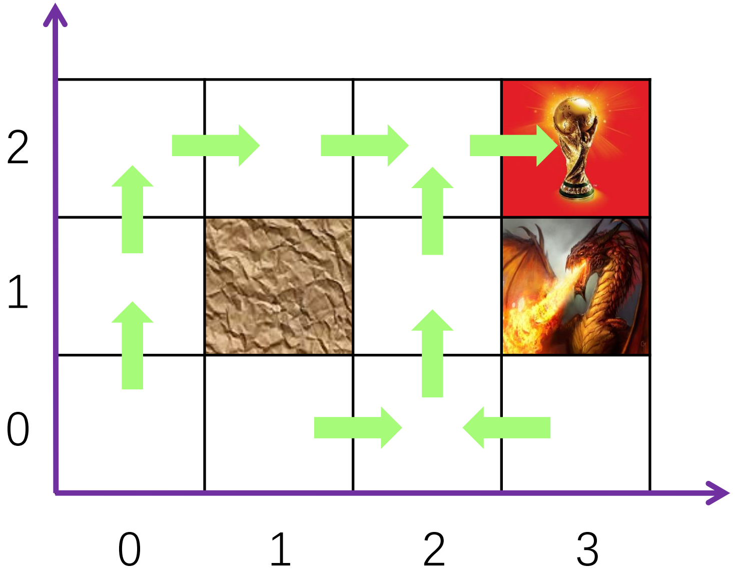
$$\begin{pmatrix} V^{\pi_1}((0,0)) \\ V^{\pi_1}((1,0)) \\ V^{\pi_1}((2,0)) \\ V^{\pi_1}((3,0)) \\ V^{\pi_1}((0,1)) \\ V^{\pi_1}((1,1)) \\ V^{\pi_1}((2,1)) \\ V^{\pi_1}((3,1)) \\ V^{\pi_1}((0,2)) \\ V^{\pi_1}((1,2)) \\ V^{\pi_1}((2,2)) \\ V^{\pi_1}((3,2)) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -5 \\ 0 \\ 0 \\ -5 \\ 0 \\ 0 \\ 0 \\ 80 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} 0.15 & 0.05 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.1 & 0.05 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0.15 & 0 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.15 & 0.05 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.15 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0.15 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V^{\pi_1}((0,0)) \\ V^{\pi_1}((1,0)) \\ V^{\pi_1}((2,0)) \\ V^{\pi_1}((3,0)) \\ V^{\pi_1}((0,1)) \\ V^{\pi_1}((1,1)) \\ V^{\pi_1}((2,1)) \\ V^{\pi_1}((3,1)) \\ V^{\pi_1}((0,2)) \\ V^{\pi_1}((1,2)) \\ V^{\pi_1}((2,2)) \\ V^{\pi_1}((3,2)) \end{pmatrix}$$

$$V = R + \gamma TV$$

Value Function – Bellman Equation

- **Bellman Equation**

$$V^{\pi}(s) = \mathbf{E}[r(s,\pi(s))] + \gamma \sum_{s'} \mathbf{P}(s'|s,\pi(s))V^{\pi}(s')$$



A "good" policy π_1

$$V = R + \gamma TV$$

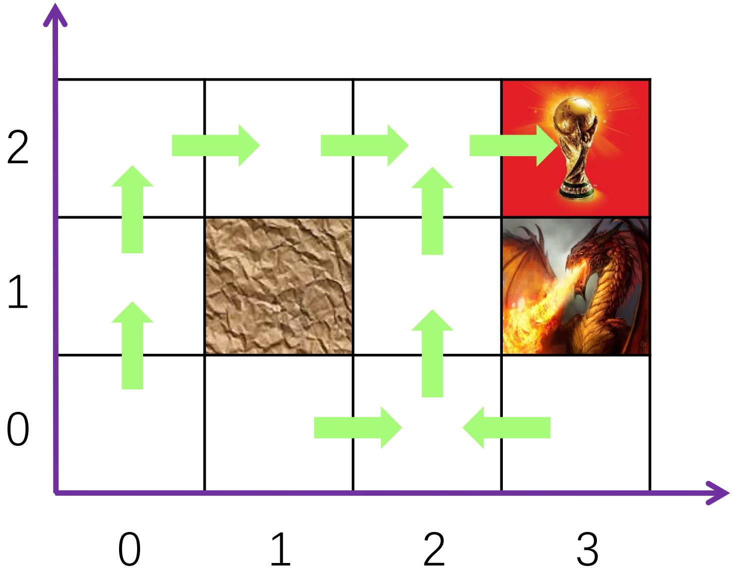
$$\Downarrow$$

$$V = (I - \gamma T)^{-1} R$$

Value Function – Bellman Equation

- **Bellman Equation**

$$V^{\pi}(s) = \mathbf{E}[r(s,\pi(s))] + \gamma \sum_{s'} \mathbf{P}(s'|s,\pi(s))V^{\pi}(s')$$



A “good” policy π_1

$$V = R + \gamma T V$$

$$\Downarrow$$

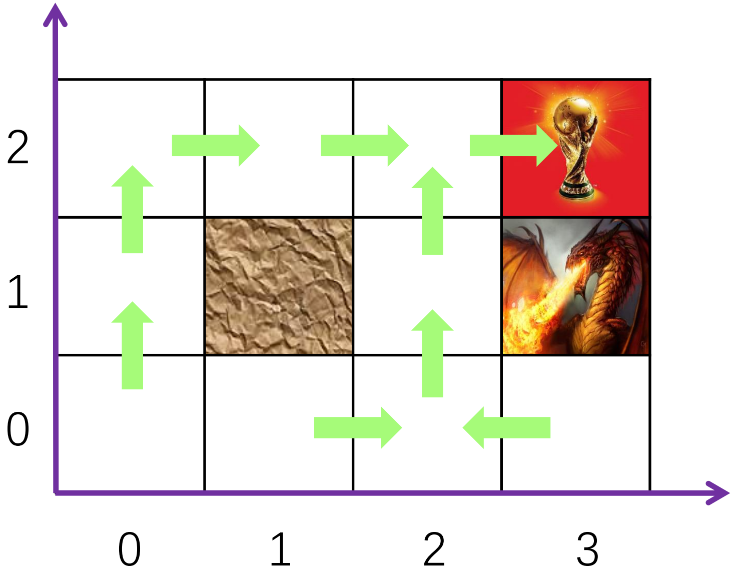
$$V = (\boxed{I - \gamma T})^{-1} R$$

invertible?

Value Function – Bellman Equation

- **Bellman Equation**

$$V^{\pi}(s) = \mathbf{E}[r(s,\pi(s))] + \gamma \sum_{s'} \mathbf{P}(s'|s,\pi(s))V^{\pi}(s')$$



A “good” policy π_1

Theorem: *For a finite MDP, Bellman’s equation admits a unique solution that is given by*

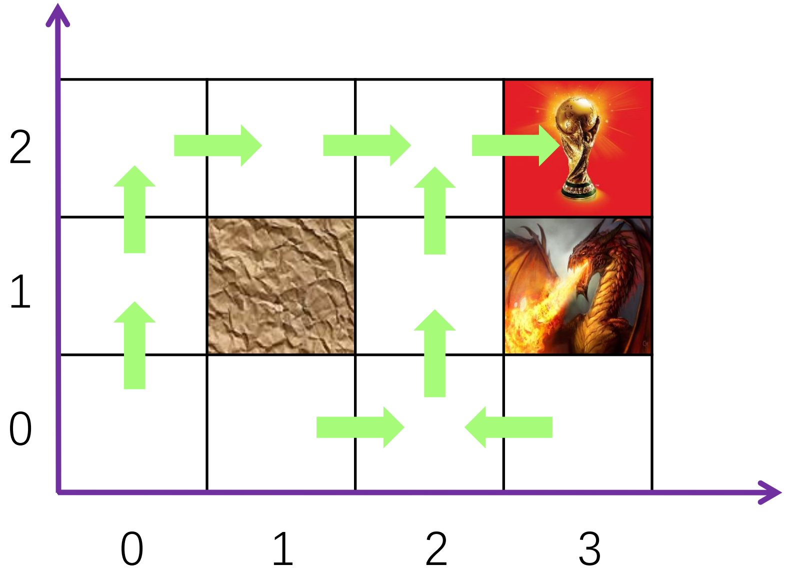
$$V = (I - \gamma T)^{-1} R$$

- The vector R and matrix T depend on the policy

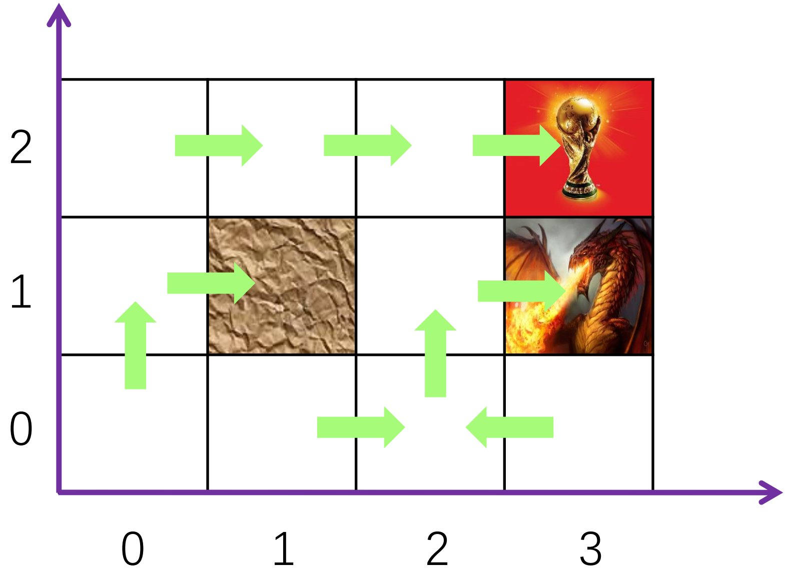
The Learning Task Revisited

- The learning task for RL scenarios is to learn an optimal policy in the sense that

$$\pi^* := \mathrm{argmax}_\pi V^\pi(s), \ \forall s.$$



A good policy π_1



A bad policy π_2

- For π_1 and π_2, we have

$$V^{\pi_1}(s) \geq V^{\pi_2}(s), \ \forall s.$$

- Indeed, π_1 is the optimal policy.

The Q Function

- Learning the optimal policy is challenging

- An alternative approach to find the optimal policy indirectly is by computing the state-action value function (Q function)

$$Q(s,a) = \mathbf{E}[r(s,a)] + \gamma \sum_{s'} \mathbf{P}(s'|s,a) V^*(s')$$

$Q(s,a)$ is the expected accumulated reward by performing the action a first and then following the optimal policy

- The definition of the optimal policy implies that

$$\pi^*(s) = \operatorname{argmax}_a Q(s,a)$$

- Notice that

$$V^*(s) = \max_a Q(s,a)$$

- All together, we have

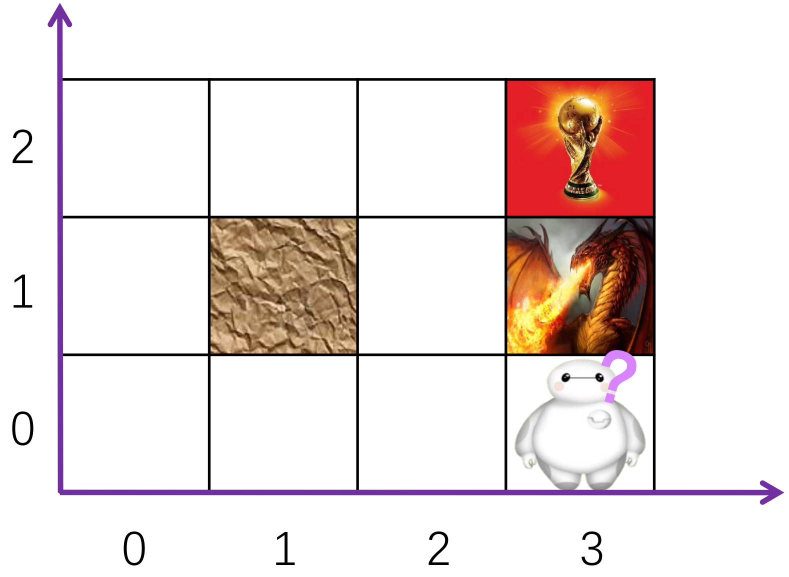
$$Q(s,a) = \mathbf{E}[r(s,a)] + \gamma \sum_{s'} \left[\mathbf{P}(s'|s,a) \max_{a'} Q(s',a') \right]$$

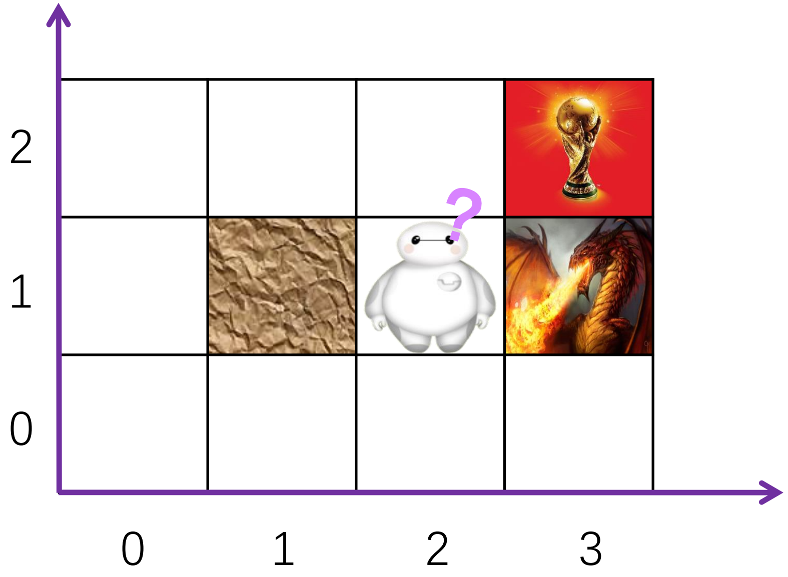
Bellman Equations

Quiz

- The learning task for RL scenarios is to learn an optimal policy in the sense that

$$\pi^* := \operatorname{argmax}_\pi V^\pi(s),\ \forall\, s.$$



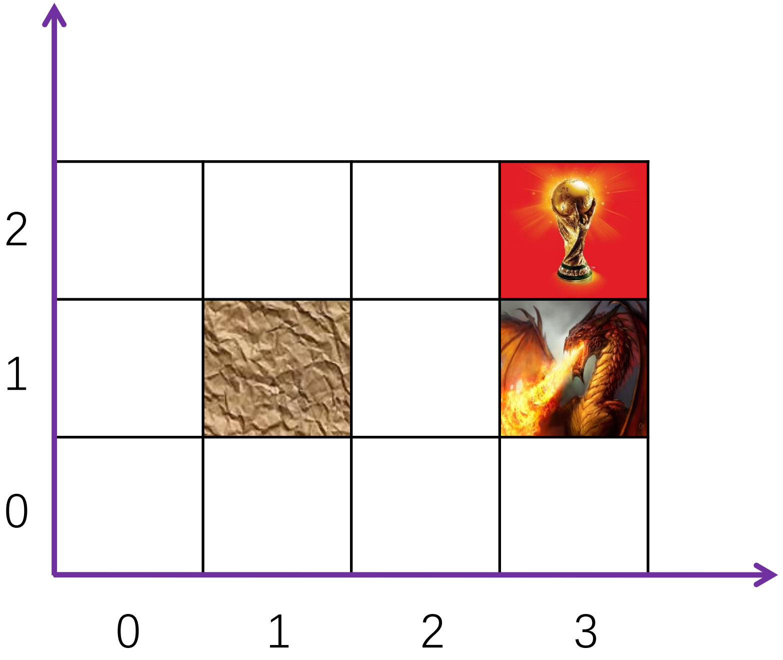


What are the best actions at states (3,0) and (2,1), i.e., $\pi^*\big((3,0)\big)$ and $\pi^*\big((2,1)\big)$?

Planning Algorithms

Planning

- Planning: we assume that the agent has perfect knowledge of the environment; thus, to find the optimal policy, there is no need for the agent to actually perform actions and interact with the environment



Known

$\mathbf{P}(s'|s,a)$: state transition

$\mathbf{P}(r|s,a)$: reward

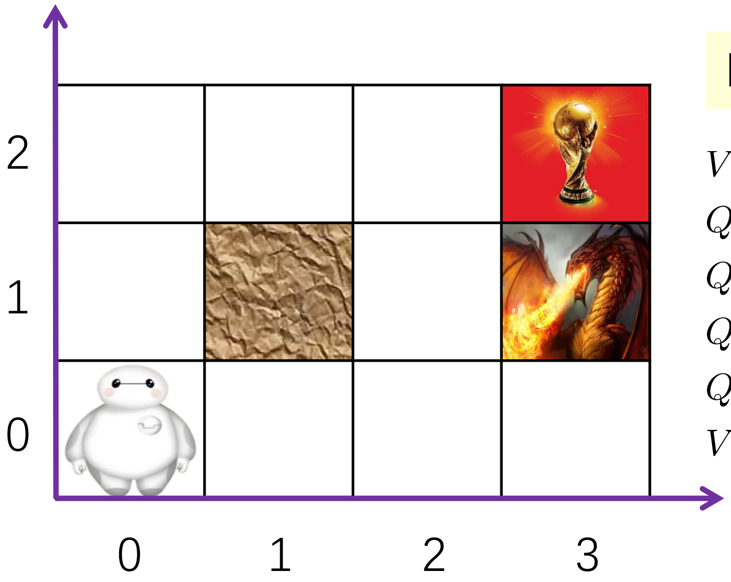
Value Iteration

- Value iteration aims to find the optimal value function and thus the optimal policy

```
**Initialize** $V(s)$ to arbitrary values
**while** termination conditions does not hold
    **For** $s \in \mathcal{S}$
        **For** $a \in \mathcal{A}$
            $Q(s,a) \leftarrow \mathbf{E}[r(s,a)] + \gamma \sum_{s'} \mathbf{P}(s'|s,a)V(s')$
        $V(s) \leftarrow \max_a Q(s,a)$
```

Value Iteration

- Value iteration aims to find the optimal value function and thus the optimal policy



Example

$$V \leftarrow 0$$

$$Q((0,0),\mathrm{up}) \leftarrow 0 + 0.9 \times (0.8 \times V((0,1)) + 0.1 \times V((0,0)) + 0.05 \times V((0,0)) + 0.05 \times V((1,0))) = 0$$

$$Q((0,0),\mathrm{down}) \leftarrow 0 + 0.9 \times (0.95 \times V((0,0)) + 0.05 \times V((1,0))) = 0$$

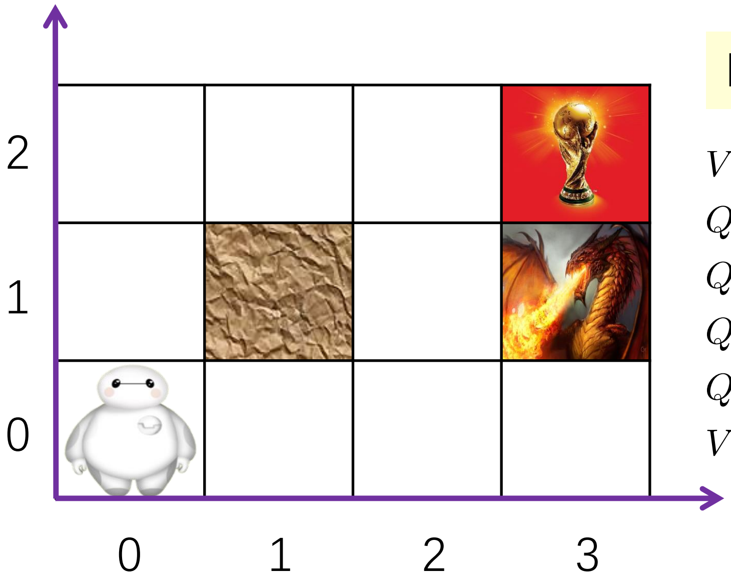
$$Q((0,0),\mathrm{left}) \leftarrow 0 + 0.9 \times (0.95 \times V((0,0)) + 0.05 \times V((0,1))) = 0$$

$$Q((0,0),\mathrm{right}) \leftarrow 0 + 0.9 \times (0.8 \times V((1,0)) + 0.15 \times V((0,0)) + 0.05 \times V((0,1))) = 0$$

$$V((0,0)) \leftarrow \max\{Q((0,0),\mathrm{up}), Q((0,0),\mathrm{down}), Q((0,0),\mathrm{left}), Q((0,0),\mathrm{right})\} = 0$$

Value Iteration

- Value iteration aims to find the optimal value function and thus the optimal policy



Example

$V \leftarrow 0$

$Q((0,0),\text{up}) \leftarrow 0 + 0.9 \times (0.8 \times V((0,1)) + 0.1 \times V((0,0)) + 0.05 \times V((0,0)) + 0.05 \times V((1,0))) = 0$

$Q((0,0),\text{down}) \leftarrow 0 + 0.9 \times (0.95 \times V((0,0)) + 0.05 \times V((1,0))) = 0$

$Q((0,0),\text{left}) \leftarrow 0 + 0.9 \times (0.95 \times V((0,0)) + 0.05 \times V((0,1))) = 0$

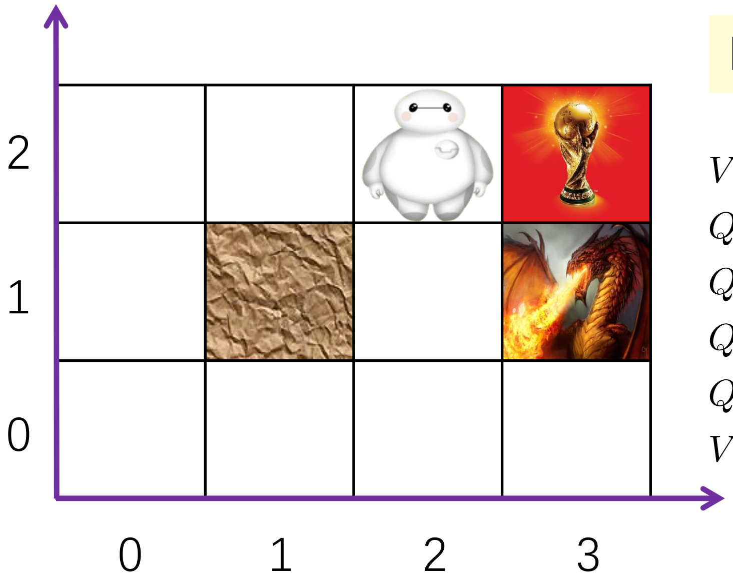
$Q((0,0),\text{right}) \leftarrow 0 + 0.9 \times (0.8 \times V((1,0)) + 0.15 \times V((0,0)) + 0.05 \times V((0,1))) = 0$

$V((0,0)) \leftarrow \max\{Q((0,0),\text{up}), Q((0,0),\text{down}), Q((0,0),\text{left}), Q((0,0),\text{right})\} = 0$

Nothing happens

Value Iteration

- Value iteration aims to find the optimal value function and thus the optimal policy



Example

$$V \leftarrow 0$$

$$Q((2,2),\mathrm{up}) \leftarrow 5 + 0.9 \times (0.9 \times V((2,2)) + 0.05 \times V((1,2)) + 0.05 \times V((3,2))) = 5$$

$$Q((2,2),\mathrm{down}) \leftarrow 5 + 0.9 \times (0.8 \times V((2,1)) + 0.1 \times V((2,2)) + 0.05 \times V((1,2)) + 0.05 \times V((3,2))) = 5$$

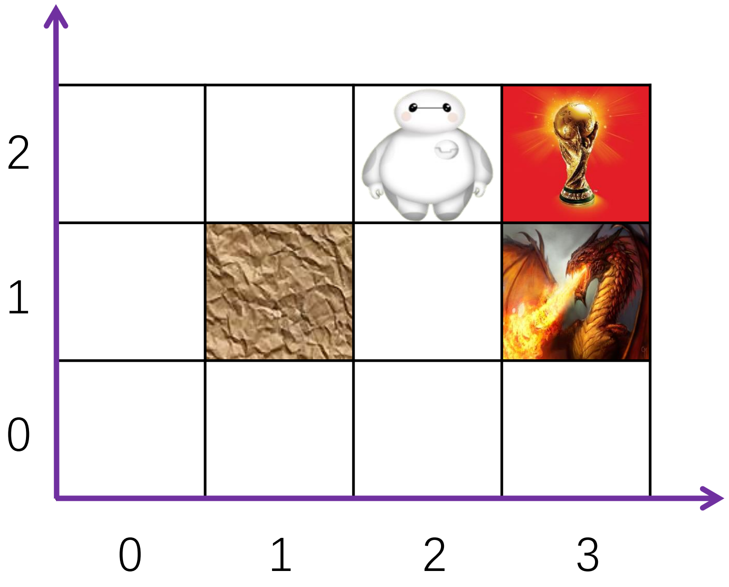
$$Q((2,2),\mathrm{left}) \leftarrow 0 + 0.9 \times (0.8 \times V((1,2)) + 0.15 \times V((2,2)) + 0.05 \times V((2,1))) = 0$$

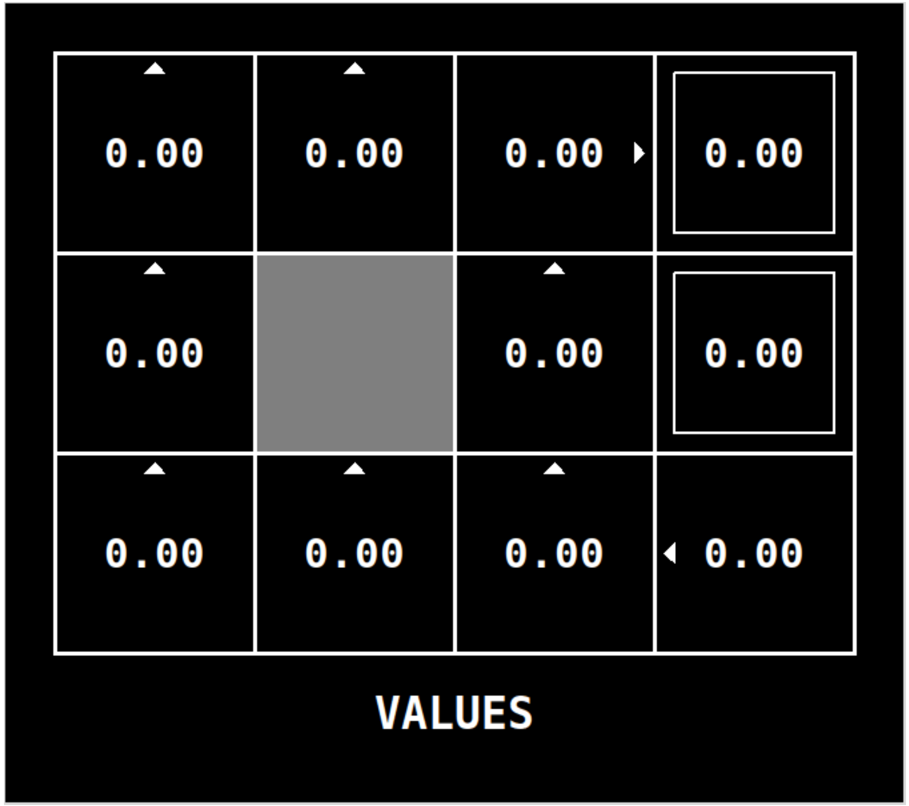
$$Q((2,2),\mathrm{right}) \leftarrow 80 + 0.9 \times (0.8 \times V((3,2)) + 0.15 \times V((2,2)) + 0.05 \times V((2,1))) = 80$$

$$V((2,2)) \leftarrow \max\{Q((0,0),\mathrm{up}), Q((0,0),\mathrm{down}), Q((0,0),\mathrm{left}), Q((0,0),\mathrm{right})\} = 80$$

Value Iteration

- Value iteration aims to find the optimal value function and thus the optimal policy





▲ 0.00	▲ 0.00	0.00 ▶	0.00
▲ 0.00		▲ 0.00	0.00
▲ 0.00	▲ 0.00	▲ 0.00	◀ 0.00

VALUES

Value Iteration

- Value iteration aims to find the optimal value function and thus the optimal policy

Theorem: *For any initial value V, the sequence generated by the value iteration algorithm converges to V^*.*

- The key to the proof is the contraction mapping theorem

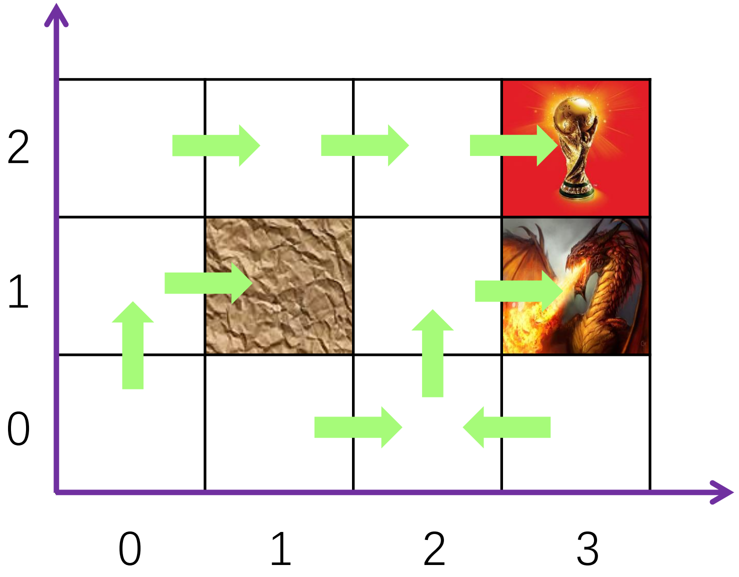
Policy Iteration

- Policy iteration improves the policy directly

```
**Initialize** $\pi \leftarrow \pi_2, \pi' \neq \pi_2$
**while**$(\pi \neq \pi')$
    $V \leftarrow (I - \gamma T^{\pi})^{-1} R^{\pi}$
    $\pi' \leftarrow \pi$
        **For** $s \in \mathcal{S}$
        $\pi(s) \leftarrow \mathrm{argmax}_a \, \mathbf{E}[r(s,a)] + \gamma \sum_{s'} \mathbf{P}(s'|s,a)V(s')$
```

Policy Iteration

- Policy iteration improves the policy directly

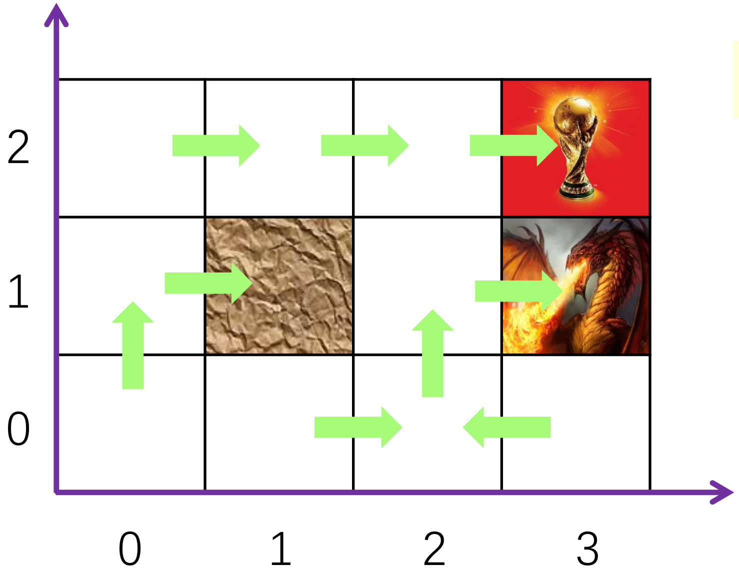


A bad policy π_2

```
    Initialize $\pi \leftarrow \pi_2, \pi' \neq \pi_2$
    while$(\pi \neq \pi')$
$1^{st}$    $V \leftarrow (I - \gamma T^{\pi})^{-1} R^{\pi}$
            $\pi' \leftarrow \pi$
                For $s \in \mathcal{S}$
                $\pi(s) \leftarrow \mathrm{argmax}_a \, \mathbf{E}[r(s,a)] + \gamma \sum_{s'} \mathbf{P}(s'|s,a) V(s')$
```

Policy Iteration

- Policy iteration improves the policy directly



A bad policy π_2

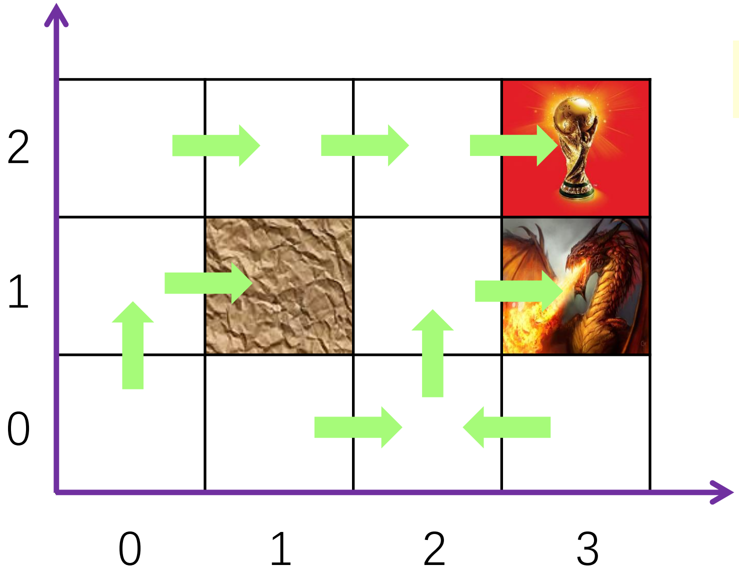
1^{st} iteration

$$V^{\pi} = \begin{pmatrix} V^{\pi}(0,0) \\ V^{\pi}(1,0) \\ V^{\pi}(2,0) \\ V^{\pi}(3,0) \\ V^{\pi}(0,1) \\ V^{\pi}(2,1) \\ V^{\pi}(3,1) \\ V^{\pi}(0,2) \\ V^{\pi}(1,2) \\ V^{\pi}(2,2) \\ V^{\pi}(3,2) \end{pmatrix}$$

11 states in total

Policy Iteration

- Policy iteration improves the policy directly



A bad policy π_2

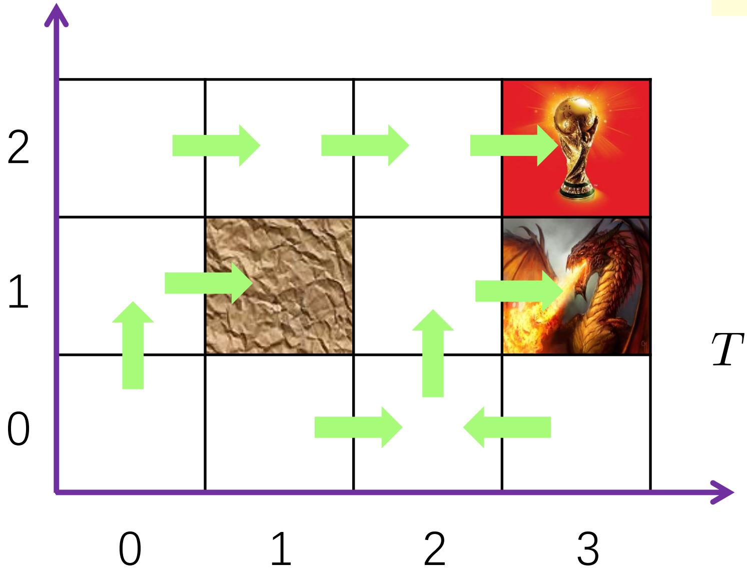
1^{st} iteration

$$R^{\pi} = \begin{pmatrix} r((0,0),\mathrm{up}) \\ r((1,0),\mathrm{right}) \\ r((2,0),\mathrm{up}) \\ r((3,0),\mathrm{left}) \\ r((0,1),\mathrm{right}) \\ r((2,1),\mathrm{right}) \\ r((3,1),\mathrm{END}) \\ r((0,2),\mathrm{right}) \\ r((1,2),\mathrm{right}) \\ r((2,2),\mathrm{right}) \\ r((3,2),\mathrm{END}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -5 \\ 0 \\ -80 \\ 0 \\ 0 \\ 0 \\ 80 \\ 0 \end{pmatrix}$$

Policy Iteration

- Policy iteration improves the policy directly

1^{st} iteration

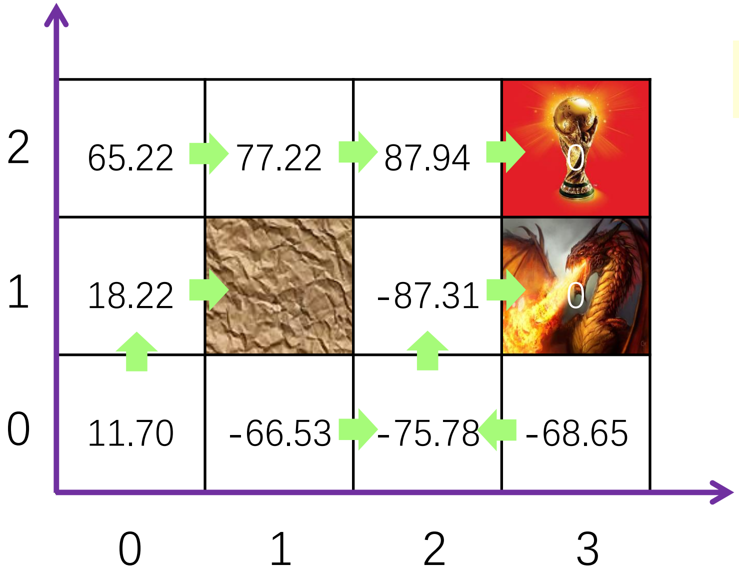


A bad policy π_2

$$T^{\pi} = \begin{pmatrix} 0.15 & 0.05 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0.1 & 0.05 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0.15 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0 \\ 0.05 & 0 & 0 & 0 & 0.9 & 0 & 0 & 0.05 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 & 0 & 0.1 & 0.8 & 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.05 & 0 & 0 & 0.15 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.05 & 0 & 0 & 0 & 0.15 & 0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Policy Iteration

- Policy iteration improves the policy directly



A bad policy π_2

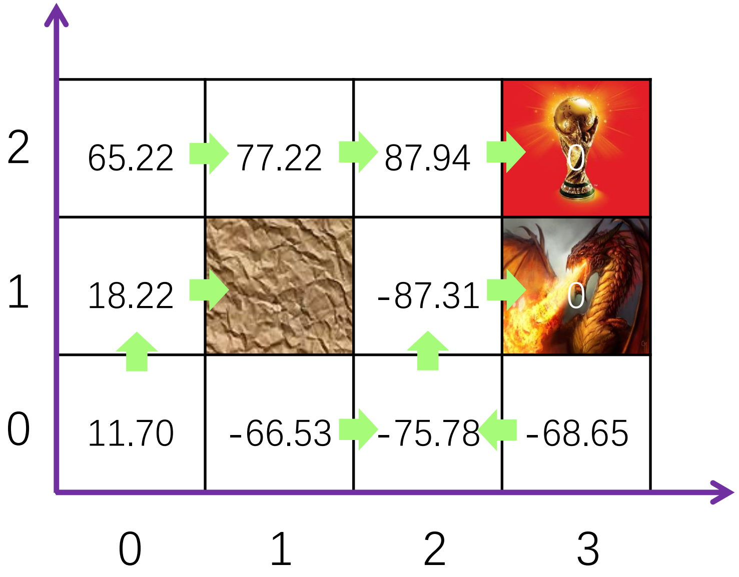
1^{st} iteration

$$V \leftarrow (I - \gamma T^{\pi})^{-1} R^{\pi} = \begin{pmatrix} 11.70 \\ -66.53 \\ -75.78 \\ -68.85 \\ 18.22 \\ -87.31 \\ 0 \\ 65.22 \\ 77.22 \\ 87.94 \\ 0 \end{pmatrix}$$

Policy Iteration

- Policy iteration improves the policy directly

1^{st} iteration: update the policy



A bad policy π_2

$$\begin{aligned} Q((0,0),\mathrm{up}) =& \mathbf{E}[r((0,0),\mathrm{up})] + 0.9 \times (0.8 \times V((0,1)) + 0.15 \times V((0,0)) + 0.05 \times V((1,0))) \\ =& 0 + 0.9 \times (0.8 \times 18.22 + 0.15 \times 11.70 + 0.05 \times -66.53) \\ =& 11.70 \end{aligned}$$

$$\begin{aligned} Q((0,0),\mathrm{down}) =& \mathbf{E}[r((0,0),\mathrm{down})] + 0.9 \times (0.95 \times V((0,0)) + 0.05 \times V((1,0))) \\ =& 0 + 0.9 \times (0.95 \times 11.70 + 0.05 \times (-66.53)) \\ =& 7.01 \end{aligned}$$

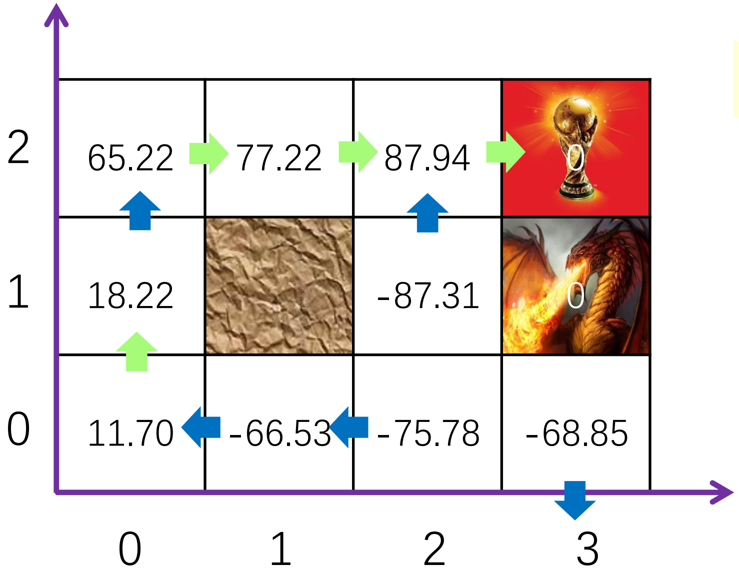
$$\begin{aligned} Q((0,0),\mathrm{left}) =& \mathbf{E}[r((0,0),\mathrm{left})] + 0.9 \times (0.95 \times V((0,0)) + 0.05 \times V((0,1))) \\ =& 0 + 0.9 \times (0.95 \times 11.70 + 0.05 \times 18.22) \\ =& 10.82 \end{aligned}$$

$$\begin{aligned} Q((0,0),\mathrm{right}) =& \mathbf{E}[r((0,0),\mathrm{right})] + 0.9 \times (0.8 \times V((1,0)) + 0.15 \times V((0,0)) + 0.05 \times V((0,1))) \\ =& 0 + 0.9 \times (0.8 \times (-66.53) + 0.15 \times 11.70 + 0.05 \times 18.22) \\ =& -45.50 \end{aligned}$$

$$\begin{aligned} \pi((0,0)) =& \mathrm{argmax}_{\{\mathrm{up,\ down,\ left,\ right}\}} \{Q((0,0),\mathrm{up}), Q((0,0),\mathrm{down}), Q((0,0),\mathrm{left}), Q((0,0),\mathrm{right})\} \\ =& \mathrm{up} \end{aligned}$$

Policy Iteration

- Policy iteration improves the policy directly

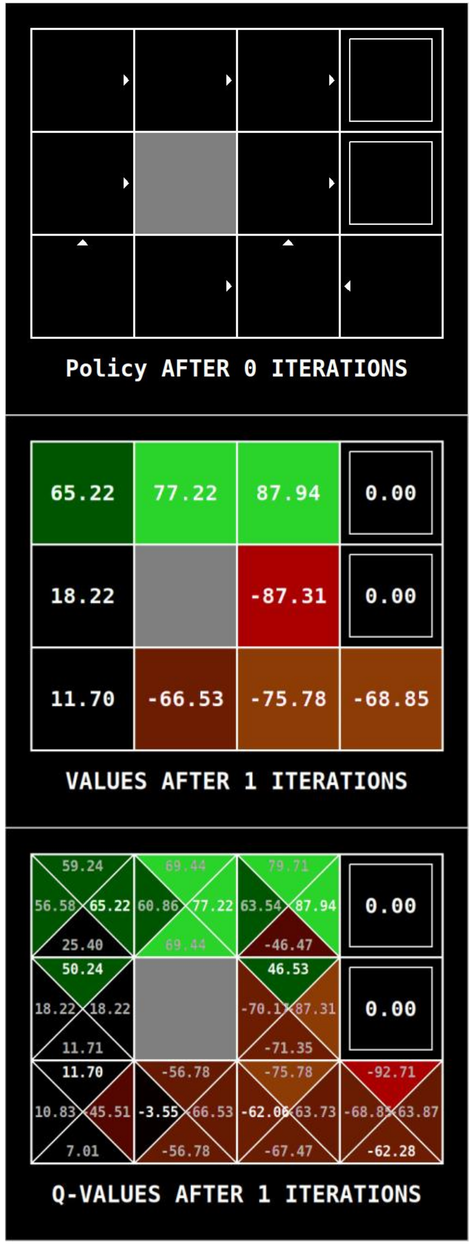


A bad policy π_2

1^{st} iteration: update the policy

$$\pi((0,0)) = \text{up}$$
$$\pi((1,0)) = \text{left}$$
$$\pi((2,0)) = \text{left}$$
$$\pi((3,0)) = \text{down}$$
$$\pi((0,1)) = \text{up}$$
$$\pi((2,1)) = \text{up}$$
$$\pi((3,1)) = \text{END}$$
$$\pi((0,2)) = \text{right}$$
$$\pi((1,2)) = \text{right}$$
$$\pi((2,2)) = \text{right}$$
$$\pi((3,2)) = \text{END}$$

Policy Iteration



→	→	→	□
→		→	□
↑	→	↑	←

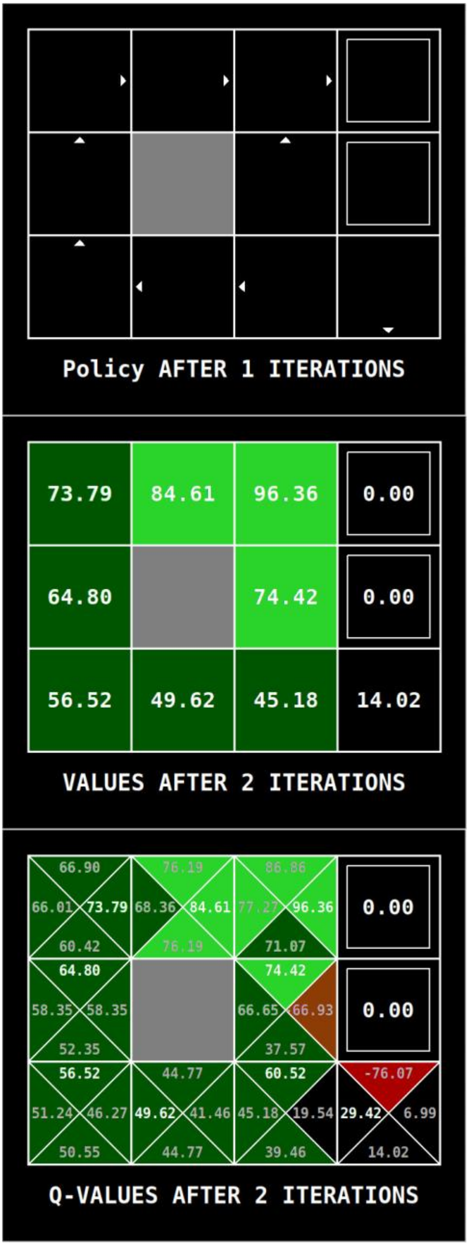
Policy AFTER 0 ITERATIONS

65.22	77.22	87.94	0.00
18.22		-87.31	0.00
11.70	-66.53	-75.78	-68.85

VALUES AFTER 1 ITERATIONS

59.24 56.58 65.22 25.40	69.44 60.86 77.22 69.44	79.71 63.54 87.94 -46.47	0.00
50.24 18.22 18.22 11.71		46.53 -70.17 -87.31 -71.35	0.00
11.70 10.83 -45.51 7.01	-56.78 -3.55 -66.53 -56.78	-75.78 -62.06 -63.73 -67.47	-92.71 -68.85 -63.87 -62.28

Q-VALUES AFTER 1 ITERATIONS



→	→	→	□
↑		↑	□
↑	←	←	↓

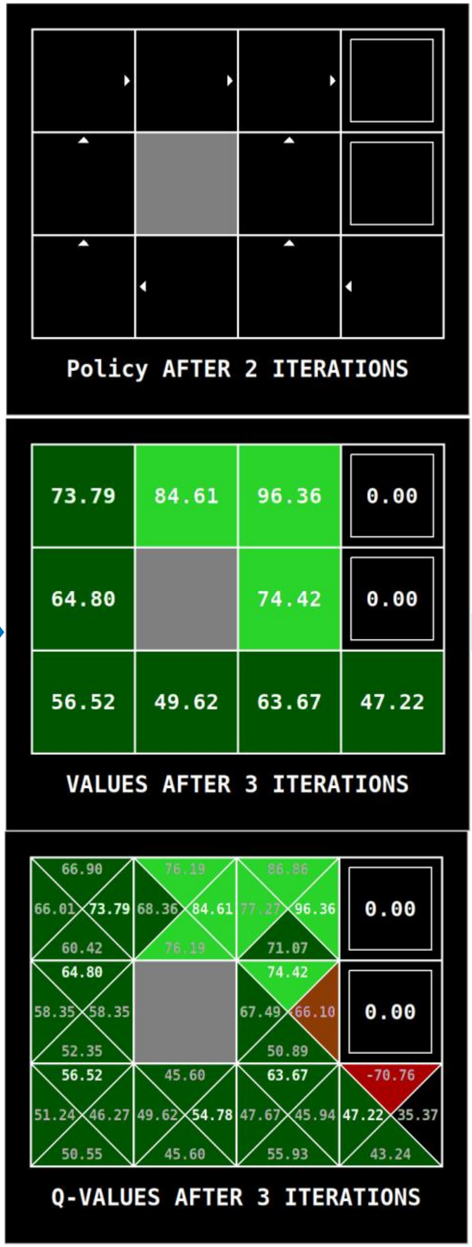
Policy AFTER 1 ITERATIONS

73.79	84.61	96.36	0.00
64.80		74.42	0.00
56.52	49.62	45.18	14.02

VALUES AFTER 2 ITERATIONS

66.90 66.01 73.79 60.42	76.19 68.36 84.61 76.19	86.86 77.27 96.36 71.07	0.00
64.80 58.35 58.35 52.35		74.42 66.65 66.93 37.57	0.00
56.52 51.24 46.27 50.55	44.77 49.62 41.46 44.77	60.52 45.18 19.54 39.46	-76.07 29.42 6.99 14.02

Q-VALUES AFTER 2 ITERATIONS



→	→	→	□
↑		↑	□
↑	←	↑	←

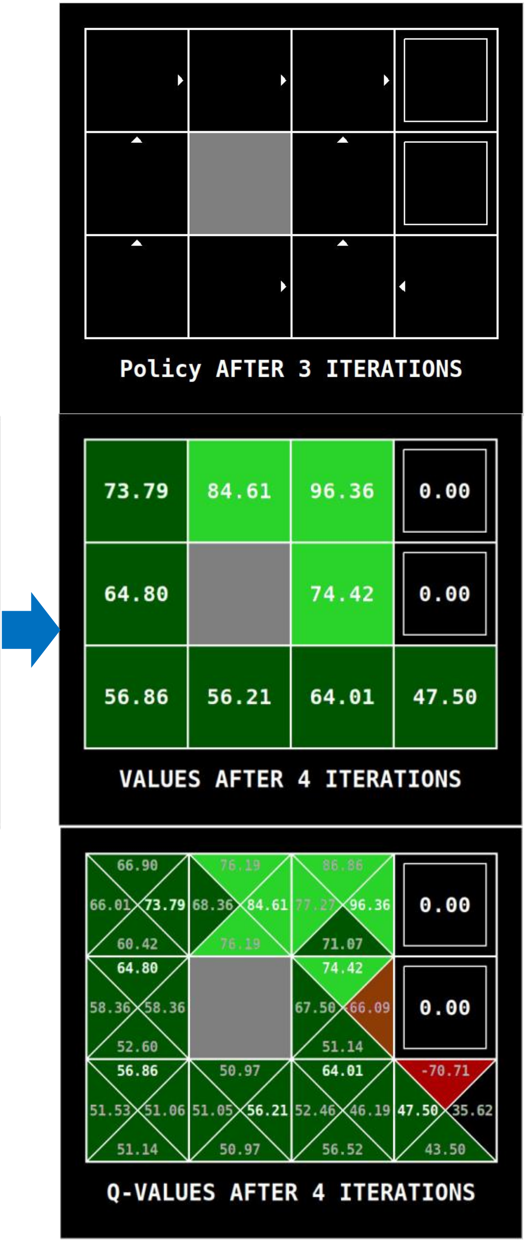
Policy AFTER 2 ITERATIONS

73.79	84.61	96.36	0.00
64.80		74.42	0.00
56.52	49.62	63.67	47.22

VALUES AFTER 3 ITERATIONS

66.90 66.01 73.79 60.42	76.19 68.36 84.61 76.19	86.86 77.27 96.36 71.07	0.00
64.80 58.35 58.35 52.35		74.42 67.49 66.10 50.89	0.00
56.52 51.24 46.27 50.55	45.60 49.62 54.78 45.60	63.67 47.67 45.94 55.93	-70.76 47.22 35.37 43.24

Q-VALUES AFTER 3 ITERATIONS



→	→	→	□
↑		↑	□
↑	→	↑	←

Policy AFTER 3 ITERATIONS

73.79	84.61	96.36	0.00
64.80		74.42	0.00
56.86	56.21	64.01	47.50

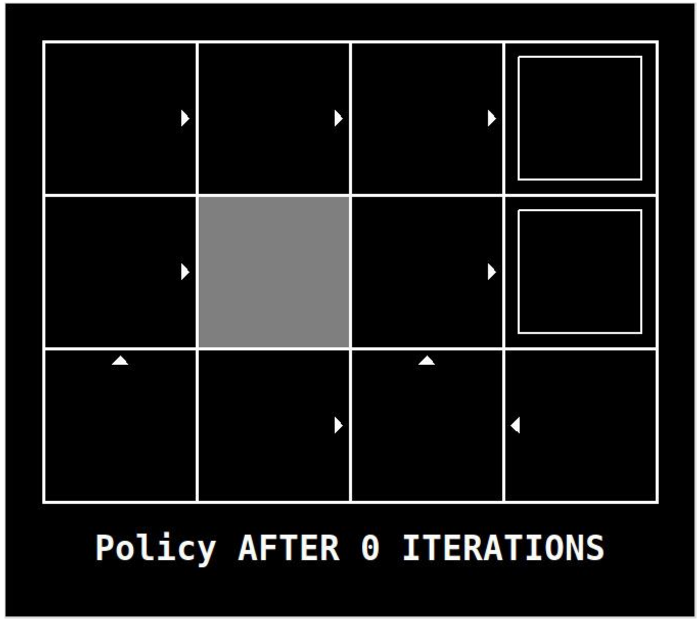
VALUES AFTER 4 ITERATIONS

66.90 66.01 73.79 60.42	76.19 68.36 84.61 76.19	86.86 77.27 96.36 71.07	0.00
64.80 58.36 58.36 52.60		74.42 67.50 66.09 51.14	0.00
56.86 51.53 51.06 51.14	50.97 51.05 56.21 50.97	64.01 52.46 46.19 56.52	-70.71 47.50 35.62 43.50

Q-VALUES AFTER 4 ITERATIONS

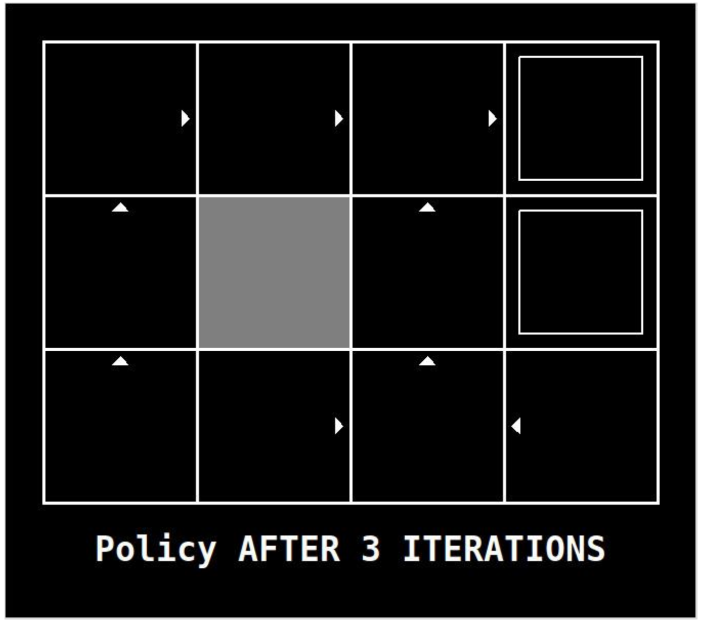
three iterations to converge

Policy Iteration



Policy AFTER 0 ITERATIONS

The optimal policy

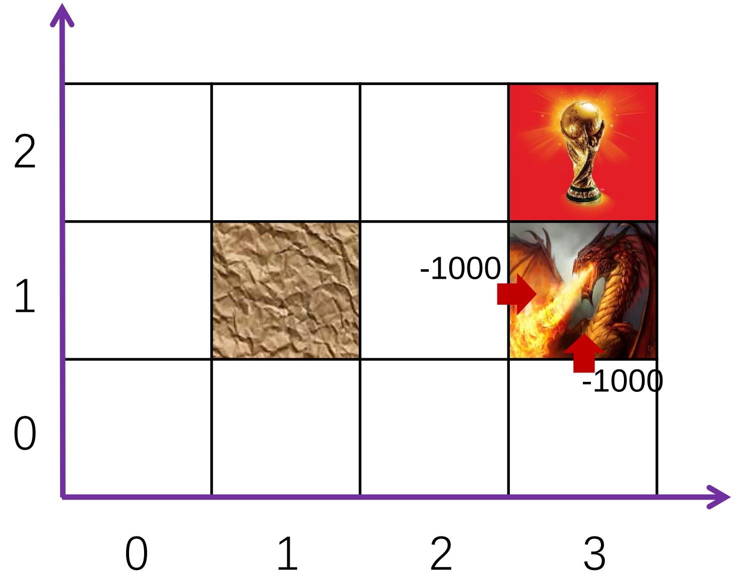


Policy AFTER 3 ITERATIONS

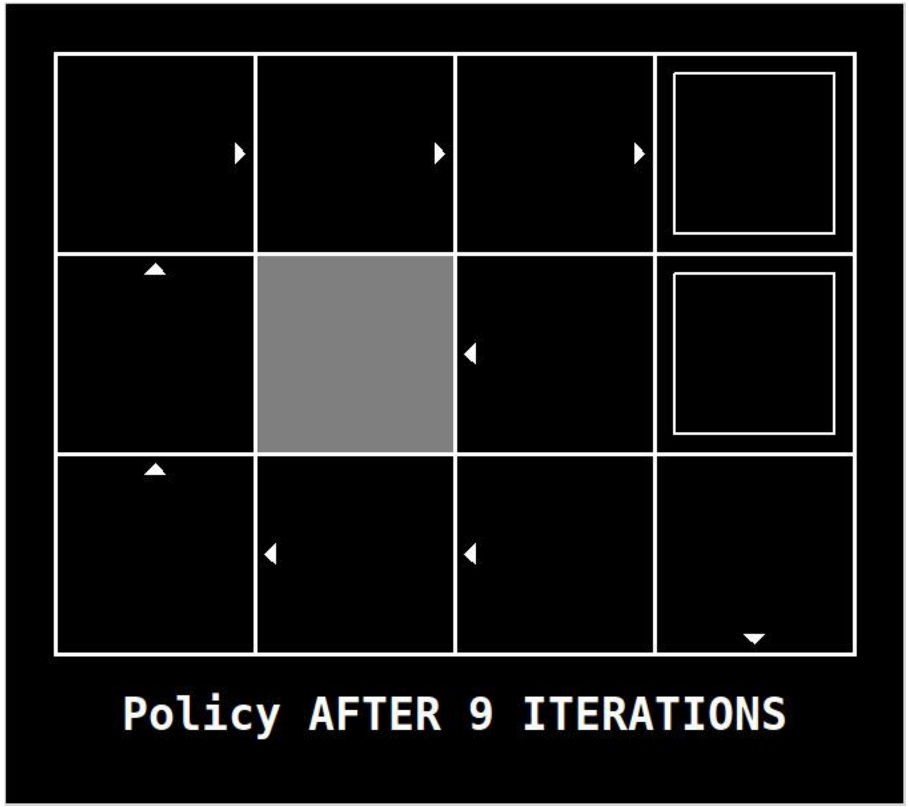
Is this an always winning policy?

Policy Iteration

- What if the reward for getting into (3,1) is -1000?



The optimal policy

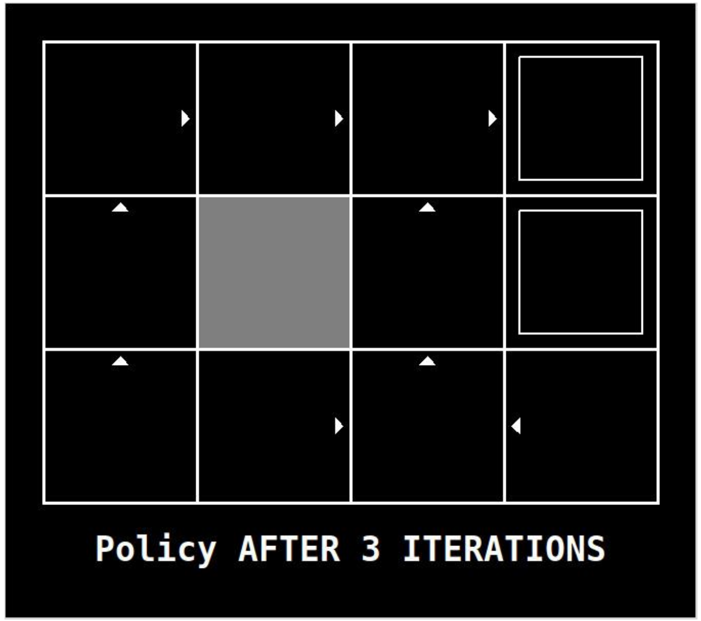


Policy AFTER 9 ITERATIONS

This is an always winning policy (why?).

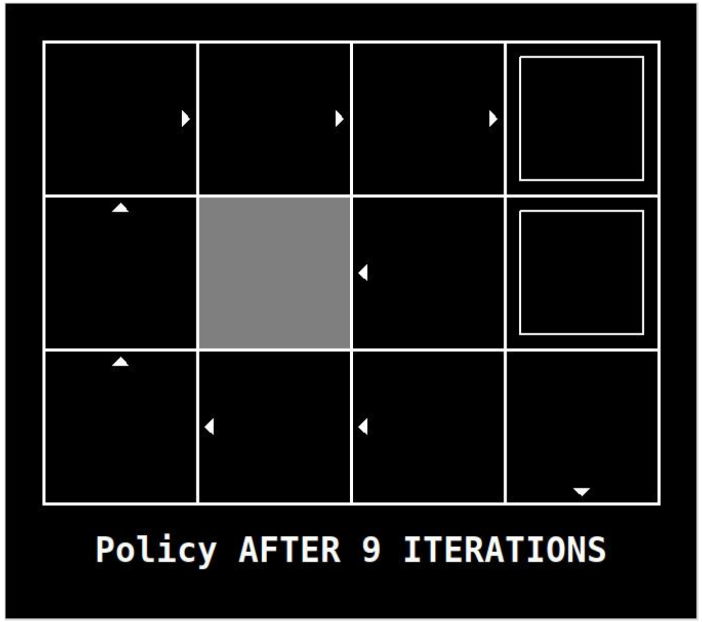
Policy Iteration

A mostly winning policy



Policy AFTER 3 ITERATIONS

A never lose policy



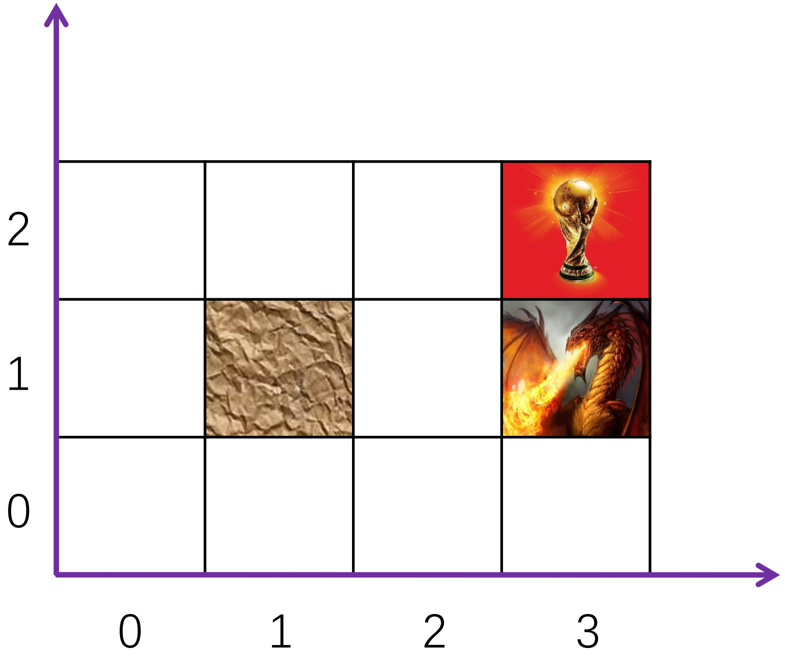
Policy AFTER 9 ITERATIONS

- For the same task, different reward strategies lead to different optimal policy
- According to your preference, you need to carefully design your reward strategy

Learning Algorithms

Learning

- Learning: as the environment model, i.e., the transition and reward, is unknown, the agent may need to learn them based on the training information.

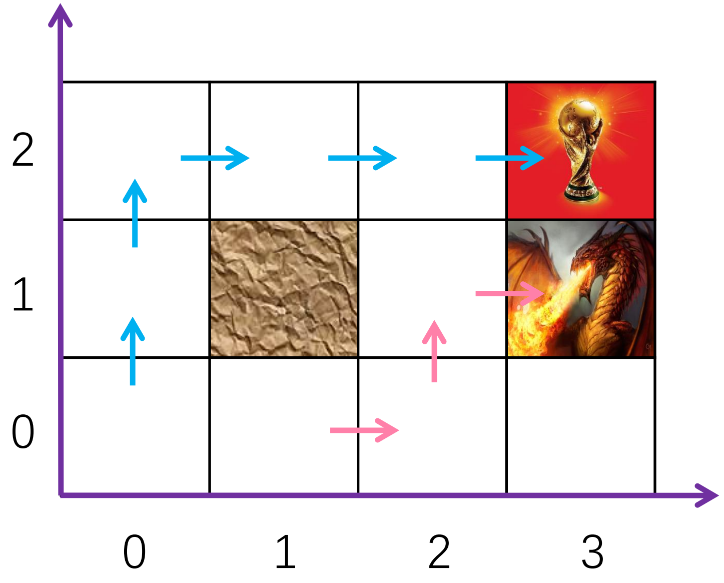


Unknown

$\mathbf{P}(s'|s,a)$: state transition
$\mathbf{P}(r|s,a)$: reward

Learning

- Learning: as the environment model, i.e., the transition and reward probabilities, is unknown, the agent may need to learn them based on the training information.
    - Model-free approach: the agent learns the optimal policy directly, e.g., Q-learning
    - Model-based approach: the agent first learns the environment model and then the optimal policy

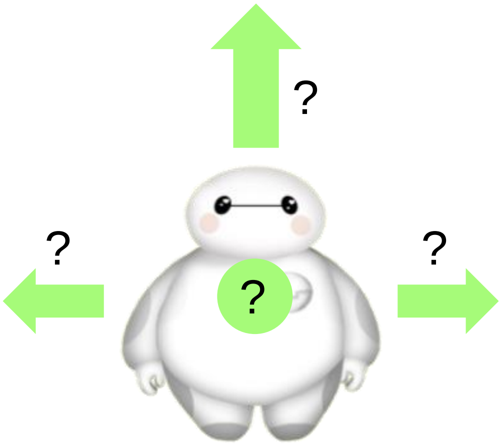


Examples of training data

$$(0,0) \xrightarrow[0]{\text{up}} (0,1) \xrightarrow[0]{\text{up}} (0,2) \xrightarrow[0]{\text{right}} (1,2) \xrightarrow[0]{\text{right}} (2,3) \xrightarrow[100]{\text{right}} (3,2)$$

$$(1,0) \xrightarrow[0]{\text{right}} (2,0) \xrightarrow[0]{\text{up}} (2,1) \xrightarrow[-100]{\text{right}} (3,1)$$

Nondeterministic Rewards and Actions



Unknown

$\mathbf{P}(s'|s,a)$: state transition
$\mathbf{P}(r|s,a)$: reward

How to find the optimal policy without the state transition and reward probabilities?

The Q-learning Algorithm

Recall the Q-learning algorithm for the deterministic environment

- Initialize the matrix $\hat{Q}$ to zero
- Observe the current state s
- Do forever:
  - Pick and perform an action a
  - Receive immediate reward r
  - Observe the new state s'
  - Update

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

  - $s \leftarrow s'$

What if r and s' become random variables?

A sufficient condition for $\hat{Q}(s,a)$ to converge is to visit each state-action pair infinitely often

The Q-learning Algorithm

- For the stochastic environment, we replace the random variables with their expectations in the definition of Q values.

$$Q(s,a) = \boxed{\mathbf{E}[r(s,a)]} + \gamma \sum_{s'} \boxed{\left[\mathbf{P}(s'|s,a)\max_{a'} Q(s',a')\right]}$$

expectations

Q: How to find the expectation of a random variable X?

A: Keep sampling and recording its running average

$$\boxed{\frac{x_1+x_2+\ldots+x_{n+1}}{n+1}} = \boxed{\frac{x_1+x_2+\ldots+x_n}{n}} + \frac{1}{n+1}\left(x_{n+1} - \boxed{\frac{x_1+x_2+\ldots+x_n}{n}}\right) \Rightarrow \boxed{\hat{X} \leftarrow \hat{X} + \frac{1}{n+1}(x_{n+1} - \hat{X})}$$

the running average on n+1 samples

the running average on n samples

the estimation of the expectation of X

The Q-learning Algorithm

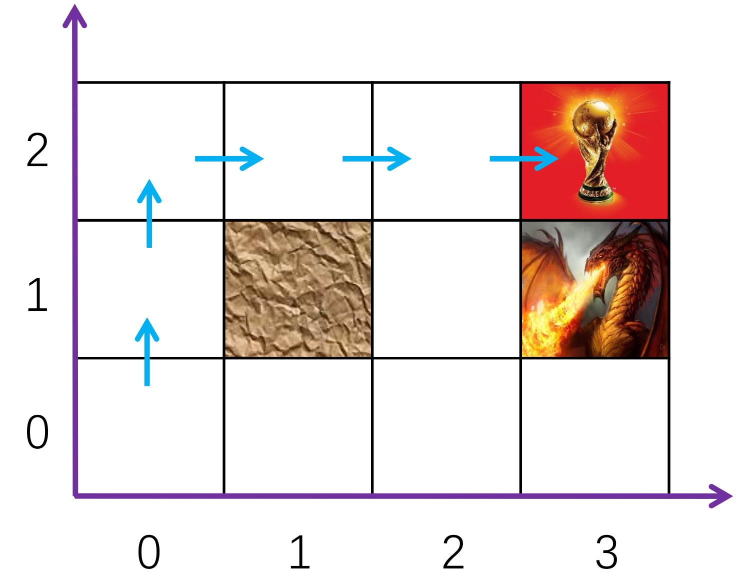
Alpaydin 2014, Chapter 18

```
**Initialize** $\hat{Q}$ arbitrarily
**For** all episodes
    **Initialize** $s$
    **Repeat**
    **Choose** $a$ using policy derived from $Q$, e.g., $\epsilon$-greedy
        Take action $a$, observe $r$ and $s'$
        Update $\hat{Q}(s,a)$:
            $\alpha_n = \frac{1}{1 + n((s,a))}$
            $\hat{Q}(s,a) \leftarrow \hat{Q}(s,a) + \alpha_n(r + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a))$
        $s \leftarrow s'$
    Until $s$ is goal state
```

the number of visits of (s, a)

There are other ways to select α_n to guarantee that $\hat{Q}$ converges to its optimal value. Mitchell 1997, Chapter 13

The Q-learning Algorithm

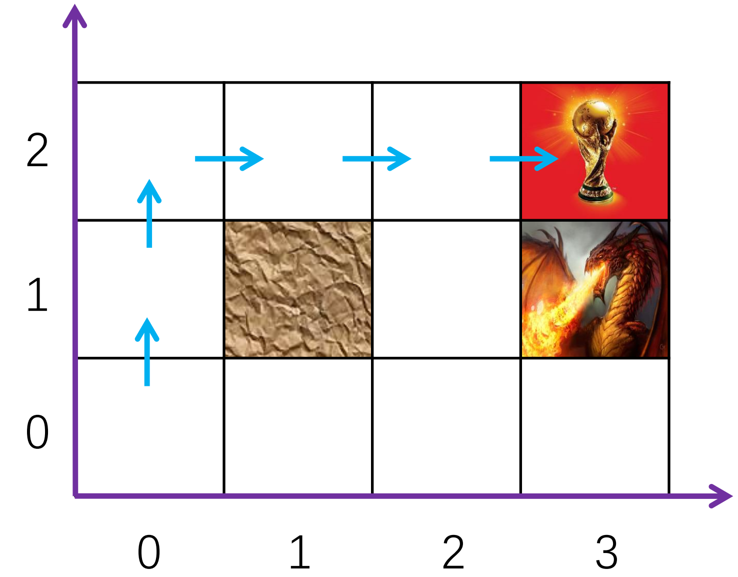


$$(0,0) \xrightarrow[0]{\text{up}} (0,1) \xrightarrow[0]{\text{up}} (0,2) \xrightarrow[0]{\text{right}} (1,2) \xrightarrow[0]{\text{right}} (2,3) \xrightarrow[100]{\text{right}} (3,2)$$

- an example episode
- the initial state in each episode could NOT be fixed (why?)

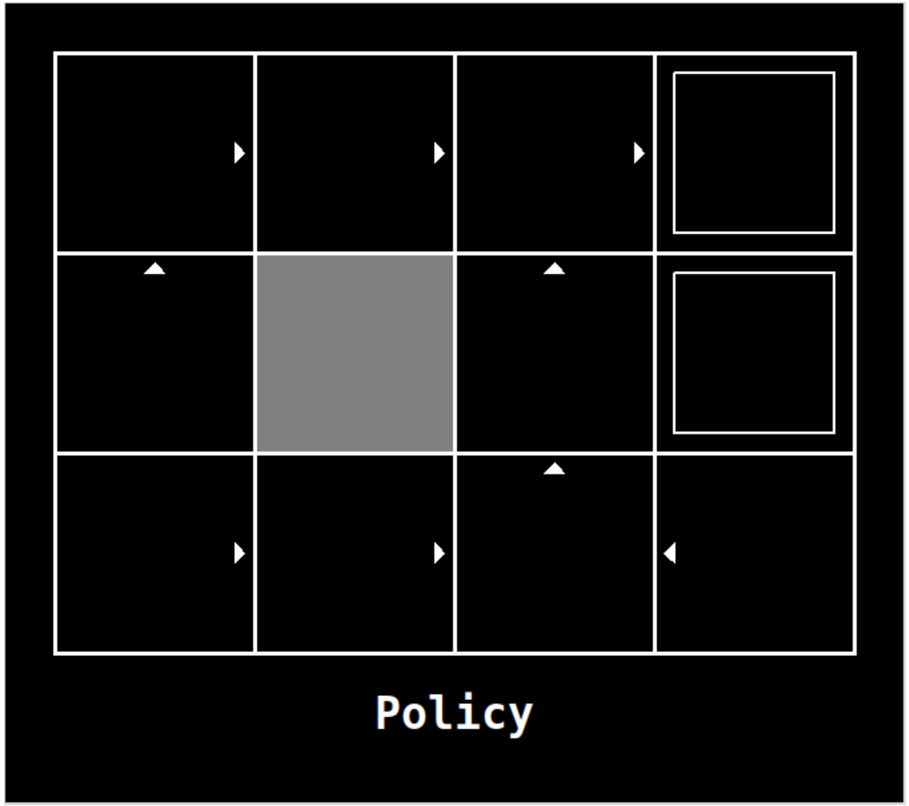
$\epsilon = 0.3$

The Q-learning Algorithm



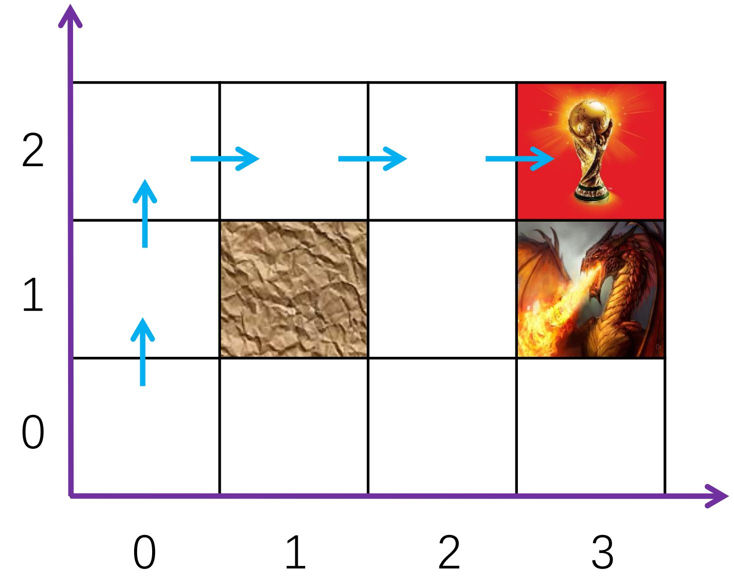
$$(0,0)\xrightarrow[0]{\text{up}}(0,1)\xrightarrow[0]{\text{up}}(0,2)\xrightarrow[0]{\text{right}}(1,2)\xrightarrow[0]{\text{right}}(2,3)\xrightarrow[100]{\text{right}}(3,2)$$

- an example episode
- the initial state in each episode could NOT be fixed (why?)



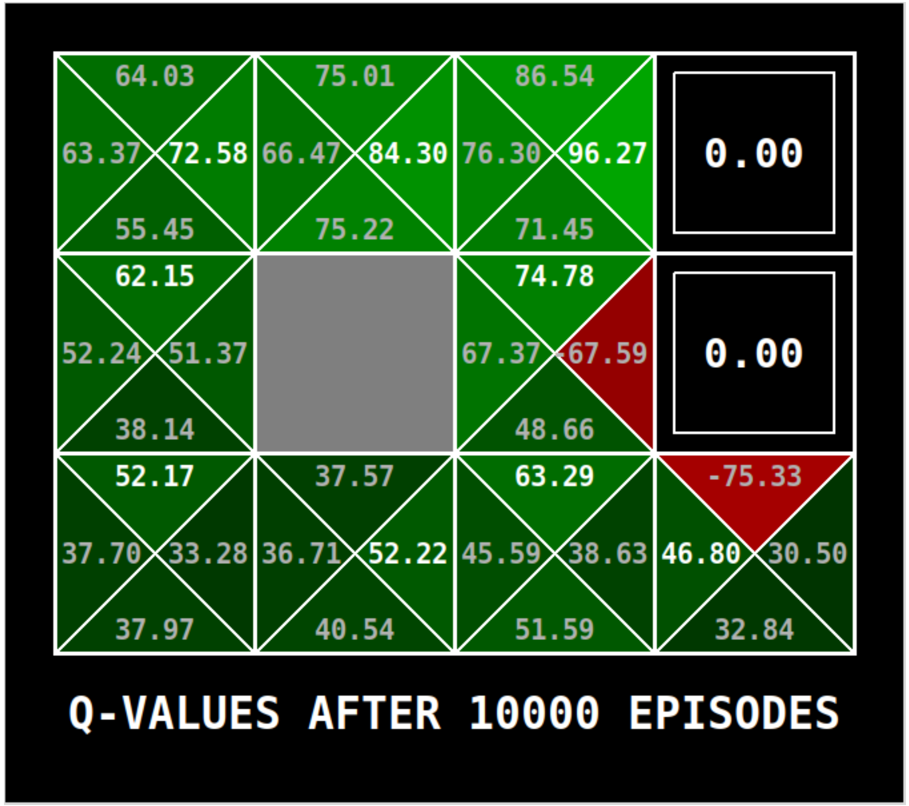
$\epsilon = 0.3$

The Q-learning Algorithm



$$(0,0) \xrightarrow[0]{\text{up}} (0,1) \xrightarrow[0]{\text{up}} (0,2) \xrightarrow[0]{\text{right}} (1,2) \xrightarrow[0]{\text{right}} (2,3) \xrightarrow[100]{\text{right}} (3,2)$$

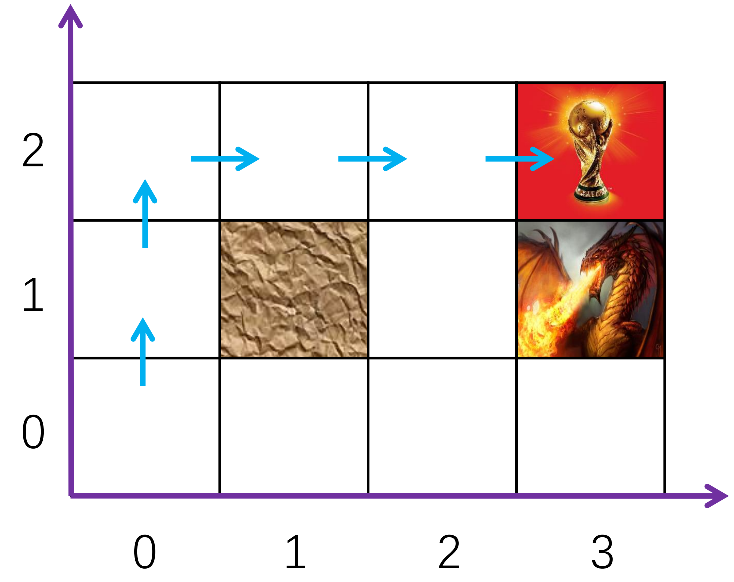
- an example episode
- the initial state in each episode could NOT be fixed (why?)



Q-VALUES AFTER 10000 EPISODES

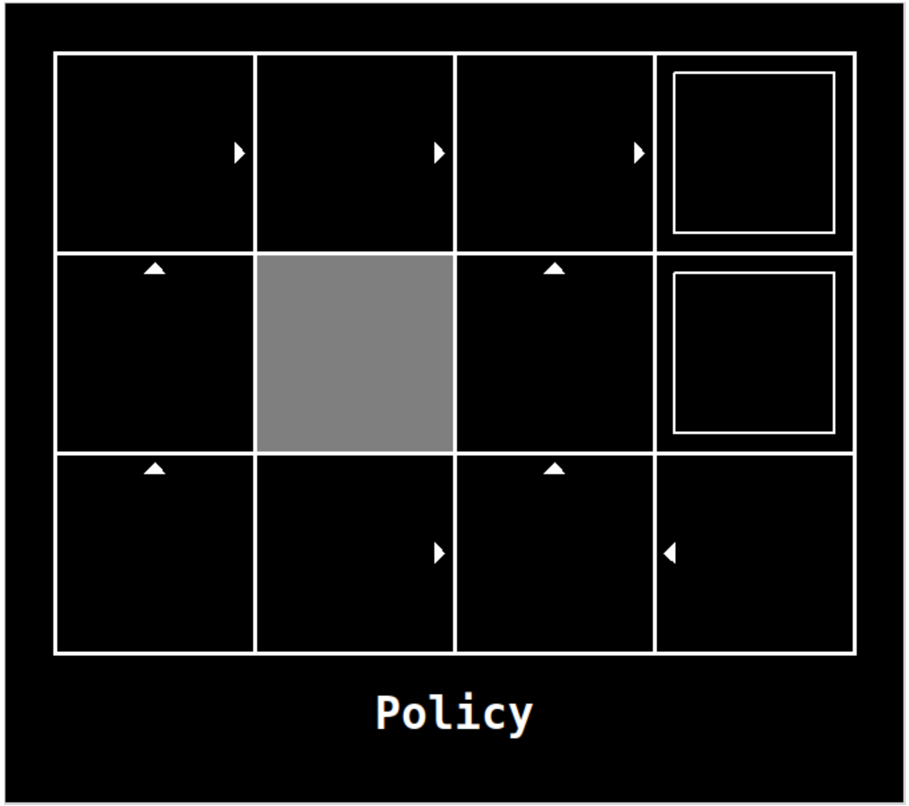
$\epsilon = 0.3$

The Q-learning Algorithm



$$(0,0)\xrightarrow[0]{\text{up}}(0,1)\xrightarrow[0]{\text{up}}(0,2)\xrightarrow[0]{\text{right}}(1,2)\xrightarrow[0]{\text{right}}(2,3)\xrightarrow[100]{\text{right}}(3,2)$$

- an example episode
- the initial state in each episode could NOT be fixed (why?)



$\epsilon = 0.3$

Questions

Getty Images