Reinforcement Learning

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

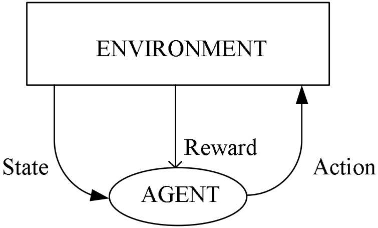
Machine Learning

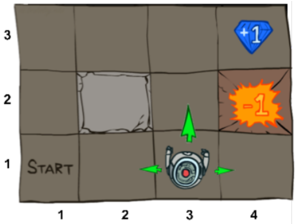
Outline

Outline

Reinforcement Learning I

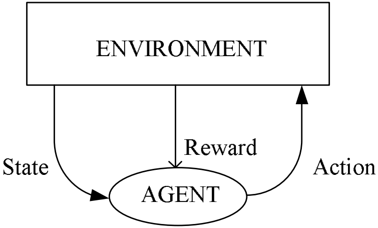
- An agent sees ***states*** $\boldsymbol{s}^{(t)}$'s of an environment, takes ***actions*** $\boldsymbol{a}^{(t)}$'s, and receives ***rewards*** $R^{(t)}$'s (or penalties)

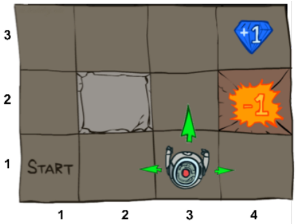




Reinforcement Learning I

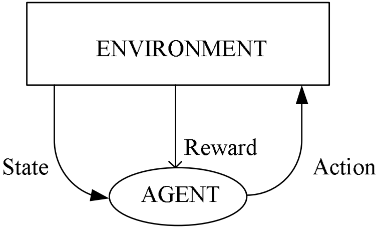
- An agent sees ***states*** $\boldsymbol{s}^{(t)}$'s of an environment, takes ***actions*** $\boldsymbol{a}^{(t)}$'s, and receives ***rewards*** $R^{(t)}$'s (or penalties)
  - Environment does ***not*** change over time

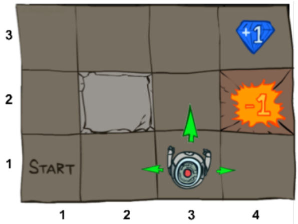




Reinforcement Learning I

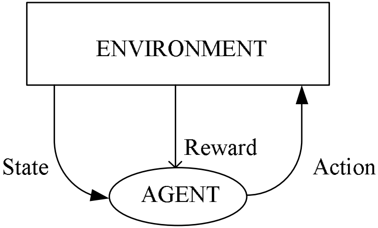
- An agent sees ***states*** $\boldsymbol{s}^{(t)}$'s of an environment, takes ***actions*** $\boldsymbol{a}^{(t)}$'s, and receives ***rewards*** $R^{(t)}$'s (or penalties)
  - Environment does ***not*** change over time
  - The state of the environment may change due to an action

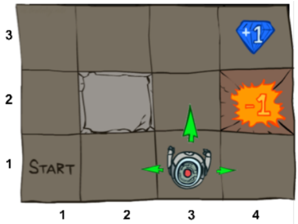




Reinforcement Learning I

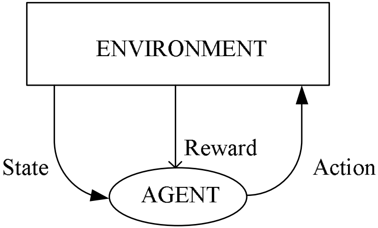
- An agent sees ***states*** $\boldsymbol{s}^{(t)}$'s of an environment, takes ***actions*** $\boldsymbol{a}^{(t)}$'s, and receives ***rewards*** $R^{(t)}$'s (or penalties)
  - Environment does ***not*** change over time
  - The state of the environment may change due to an action
  - Reward $R^{(t)}$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

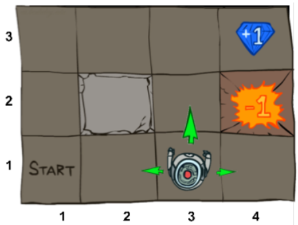




Reinforcement Learning II

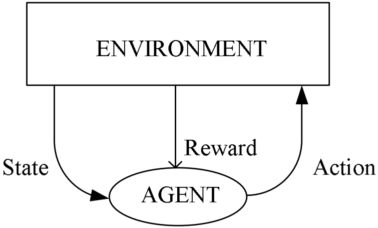
- Goal: to learn the best ***policy*** $\pi^*(\boldsymbol{s}^{(t)}) = \boldsymbol{a}^{(t)}$ that maximizes the ***total*** reward $\sum_t R^{(t)}$

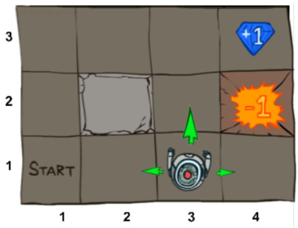




Reinforcement Learning II

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





Compared to Supervised/Unsupervised Learning

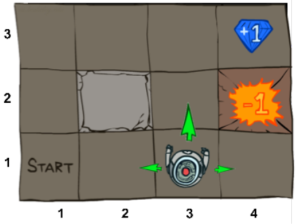
- $\pi^*(\boldsymbol{s}^{(t)}) = \boldsymbol{a}^{(t)}$ maximizing total reward $\sum_t R^{(t)}$ vs. $f^*(\boldsymbol{x}^{(i)}) = \boldsymbol{y}^{(i)}$ minimizing total loss

Compared to Supervised/Unsupervised 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 $\boldsymbol{a}^{(t)}, \boldsymbol{a}^{(t-1)}, \cdots$

Compared to Supervised/Unsupervised 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

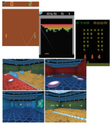
- Sequential decision making and control problems

Kohl and Stone, 2004

Ng et al, 2004

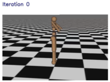
Tedrake et al, 2005

Kober and Peters, 2009

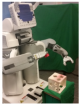


Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

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)

Applications

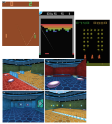
- Sequential decision making and control problems

Kohl and Stone, 2004

Ng et al, 2004

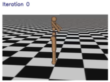
Tedrake et al, 2005

Kober and Peters, 2009

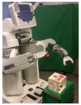


Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

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)

- From machine learning to AI that changes the world

Outline

Markov Processes

- A ***random process*** $\{\mathbf{s}^{(t)}\}_t$ is a collection of time-indexed random variables
  - A classic way to model dependency between input samples $\{\boldsymbol{s}^{(t)}\}_t$
- 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 Processes

- A ***random process*** $\{\mathbf{s}^{(t)}\}_t$ is a collection of time-indexed random variables
  - A classic way to model dependency between input samples $\{\boldsymbol{s}^{(t)}\}_t$
- A random process is called a ***Markov process*** if it satisfies the ***Markov property***:

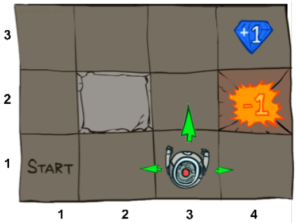
$$\mathrm{P}(\boldsymbol{s}^{(t+1)}|\boldsymbol{s}^{(t)},\boldsymbol{s}^{(t-1)},\cdots)=\mathrm{P}(\boldsymbol{s}^{(t+1)}|\boldsymbol{s}^{(t)})$$

- For those who knows Markov process:**

	States are fully observable	**States are partially observable**
Transition is autonomous	Markov chains	Hidden Markov models
Transition is controlled	***Markov decision processes (MDP)***	Partially observable MDP

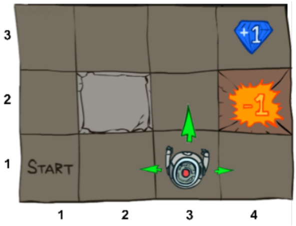
Markov Decision Process I

- 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



Markov Decision Process I

- 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



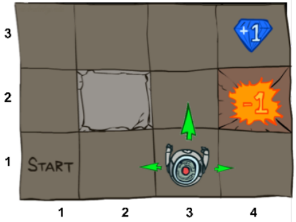
Markov Decision Process II

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

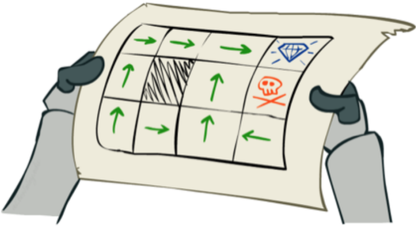
$$\boldsymbol{s}^{(0)} \xrightarrow{\boldsymbol{a}^{(0)}} \boldsymbol{s}^{(1)} \xrightarrow{\boldsymbol{a}^{(1)}} \cdots \xrightarrow{\boldsymbol{a}^{(H-1)}} \boldsymbol{s}^{(H)},$$

with the accumulative reward

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



MDP


π

Markov Decision Process II

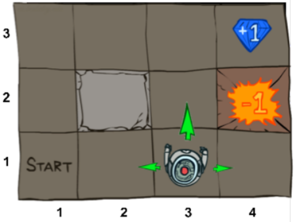
- Given a policy $\pi(s) = a$, an MDP proceeds as follows:

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

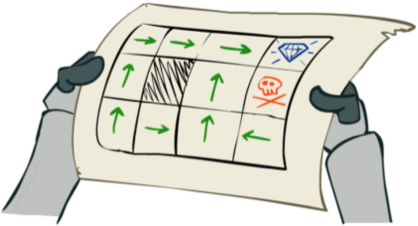
with the accumulative reward

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

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



MDP



π

Goal

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

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

- Goal: to find the optimal policy

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

Goal

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

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

- Goal: to find the optimal policy

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

- How?

Outline

Optimal Value Function

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

Optimal Value Function

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

- ***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

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

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

in $O(|\mathbb{S}||\mathbb{A}|)$ time

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

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

  in $O(|\mathbb{S}||\mathbb{A}|)$ time
- How to obtain $V^{*(H-1)}(\boldsymbol{s})$ for each $\boldsymbol{s}$?

Dynamic Programming

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

Dynamic Programming

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

- $h = H - 1$:

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

Dynamic Programming

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

- $h = H - 1$:

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

- $h = H - 2$:

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

Dynamic Programming

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

- $h = H-1$:

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

- $h = H-2$:

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

- $h = 0$:

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

- $h = -1$:

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

Algorithm: Value Iteration (Finite Horizon)

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

For each state $\boldsymbol{s}$, initialize $V^*(\boldsymbol{s}) \leftarrow 0$;
for $h \leftarrow 0$ **to** $H-1$ **do**
  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
end
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

Algorithm 1: Value iteration with finite horizon.

Example

- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$: 0.2,

    e.g., up:

	.8	
.1		.1
	.0	

    , right:

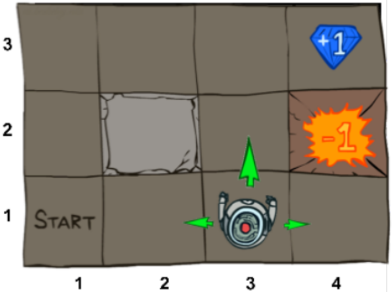
	.1	
0		.8
	.1	

    , right:

	.1	
0	.1	.8
	■	

    ,

    etc.
  - γ: 0.9



Example

- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $\mathrm{P}(s'|s;a)$: 0.2,

    e.g., up:

	.8	
.1		.1
	.0	

    , right:

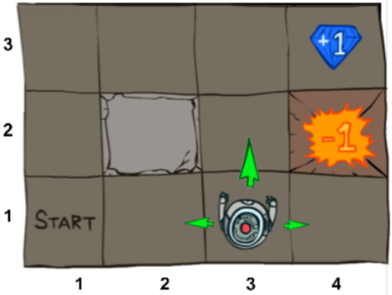
	.1	
0		.8
	.1	

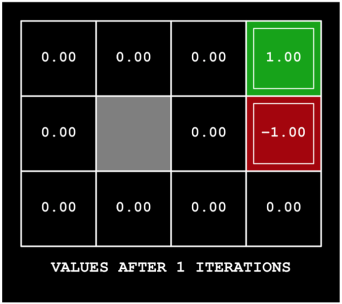
    , right:

	.1	
0	.1	.8
	■	

    ,

    etc.
  - γ: 0.9





Example

- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $\mathrm{P}(s'|s;a)$: 0.2,

    e.g., up:

	.8	
.1		.1
	.0	

    , right:

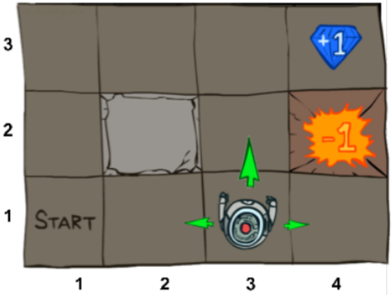
	.1	
0		.8
	.1	

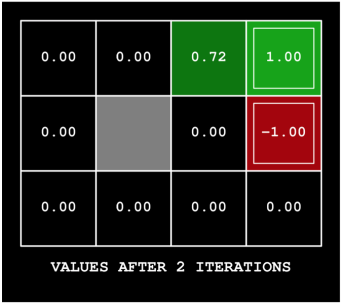
    , right:

	.1	
0	.1	.8
	■	

    ,

    etc.
  - γ: 0.9



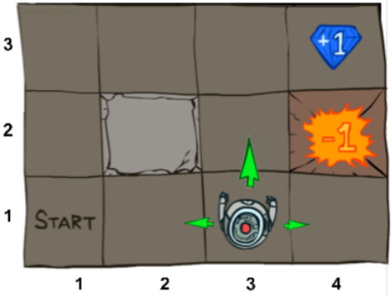


Example

- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $\mathrm{P}(s'|s;a)$: 0.2,

    e.g., up: $\begin{array}{ccc} & \boxed{.8} & \\ \boxed{.1} & \boxed{} & \boxed{.1} \\ & \boxed{.0} & \end{array}$, right: $\begin{array}{ccc} & \boxed{.1} & \\ \boxed{0} & \boxed{} & \boxed{.8} \\ & \boxed{.1} & \end{array}$, right: $\begin{array}{ccc} & \boxed{.1} & \\ \boxed{0} & \boxed{.1} & \boxed{.8} \\ & \boxed{\blacksquare} & \end{array}$,

    etc.
  - γ: 0.9



Example

- MDP settings:
  - Actions: “up,” “down,” “left,” “right”
  - Noise of transition probability $\mathrm{P}(s' | s; a)$: 0.2,

    e.g., up:

	.8	
.1		.1
	.0	

    , right:

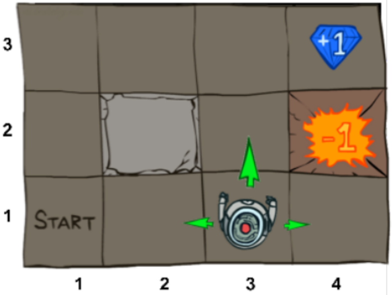
	.1	
0		.8
	.1	

    , right:

	.1	
0	.1	.8
	■	

    ,

    etc.
  - γ: 0.9

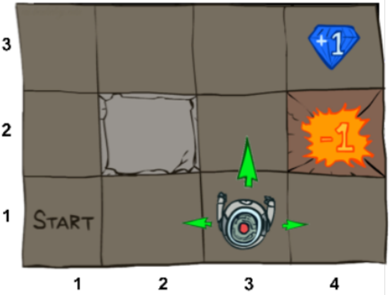


Example

- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $\mathrm{P}(s'|s;a)$: 0.2,

    e.g., up: $\begin{array}{|c|c|c|} \hline & .8 & \\ \hline .1 & & .1 \\ \hline & .0 & \\ \hline \end{array}$, right: $\begin{array}{|c|c|c|} \hline & .1 & \\ \hline 0 & & .8 \\ \hline & .1 & \\ \hline \end{array}$, right: $\begin{array}{|c|c|c|} \hline & .1 & \\ \hline 0 & .1 & .8 \\ \hline & \blacksquare & \\ \hline \end{array}$,

    etc.
  - γ: 0.9

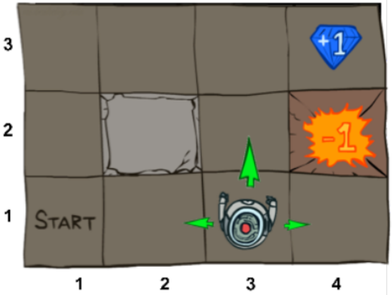


Example

- MDP settings:
  - Actions: "up," "down," "left," "right"
  - Noise of transition probability $\mathrm{P}(s'|s;a)$: 0.2,

    e.g., up: $\begin{array}{|c|c|c|}\hline & .8 & \\ \hline .1 & & .1 \\ \hline & .0 & \\ \hline\end{array}$, right: $\begin{array}{|c|c|c|}\hline & .1 & \\ \hline 0 & & .8 \\ \hline & .1 & \\ \hline\end{array}$, right: $\begin{array}{|c|c|c|}\hline & .1 & \\ \hline 0 & .1 & .8 \\ \hline & \blacksquare & \\ \hline\end{array}$,

    etc.
  - γ: 0.9



Infinite Horizon & Bellman Optimality Equation

- Recurrence of optimal values:

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

Infinite Horizon & Bellman Optimality Equation

- Recurrence of optimal values:

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

- When $h \to \infty$, we have the ***Bellman optimality equation***:

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

Infinite Horizon & Bellman Optimality Equation

- Recurrence of optimal values:

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

- When $h \to \infty$, we have the ***Bellman optimality equation***:

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

- Optimal policy:

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

Infinite Horizon & Bellman Optimality Equation

- Recurrence of optimal values:

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

- When $h \to \infty$, we have the ***Bellman optimality equation***:

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

- Optimal policy:

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

- When $h \to \infty$, π^* is
  - ***Stationary***: the optimal action at a state $\boldsymbol{s}$ is the same at all times (efficient to store)
  - ***Memoryless***: independent with $\boldsymbol{s}^{(0)}$

Algorithm: Value Iteration (Infinite Horizon)

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

Algorithm 2: Value iteration with infinite horizon.

Convergence

Theorem

Value iteration converges and gives the optimal policy π^* *when* $H \to \infty$.

Convergence

Theorem

Value iteration converges and gives the optimal policy π^* when $H \to \infty$.

- Intuition:

$$\begin{aligned} V^*(\boldsymbol{s}) - V^{*(H)}(\boldsymbol{s}) &= \gamma^{H+1} R(\boldsymbol{s}^{(H+1)}, \boldsymbol{a}^{(H+1)}, \boldsymbol{s}^{(H+2)}) \\ &\qquad + \gamma^{H+2} R(\boldsymbol{s}^{(H+2)}, \boldsymbol{a}^{(H+2)}, \boldsymbol{s}^{(H+3)}) + \cdots \\ &\leq \gamma^{H+1} R_{\max} + \gamma^{H+2} R_{\max} + \cdots \\ &= \frac{\gamma^{H+1}}{1-\gamma} R_{\max} \end{aligned}$$

  - Goes to 0 as $H \to \infty$
  - Hence, $V^{*(H)}(\boldsymbol{s}) \xrightarrow{H\to\infty} V^*(\boldsymbol{s})$

Convergence

Theorem

Value iteration converges and gives the optimal policy π^* when $H \to \infty$.

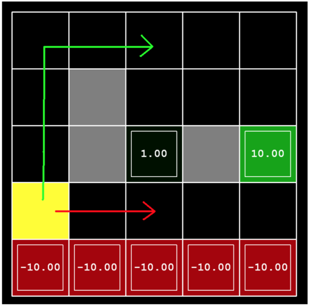
- Intuition:

$$\begin{aligned} V^*(\boldsymbol{s}) - V^{*(H)}(\boldsymbol{s}) &= \gamma^{H+1} R(\boldsymbol{s}^{(H+1)}, \boldsymbol{a}^{(H+1)}, \boldsymbol{s}^{(H+2)}) \\ &\qquad + \gamma^{H+2} R(\boldsymbol{s}^{(H+2)}, \boldsymbol{a}^{(H+2)}, \boldsymbol{s}^{(H+3)}) + \cdots \\ &\leq \gamma^{H+1} R_{\max} + \gamma^{H+2} R_{\max} + \cdots \\ &= \frac{\gamma^{H+1}}{1-\gamma} R_{\max} \end{aligned}$$

  - Goes to 0 as $H \to \infty$
  - Hence, $V^{*(H)}(\boldsymbol{s}) \xrightarrow{H \to \infty} V^*(\boldsymbol{s})$
- Assumed that $R(\cdot) \geq 0$; still holds if rewards can be negative
  - by using $\max|R(\cdot)|$ and bounding from both sides

Effect of MDP Parameters

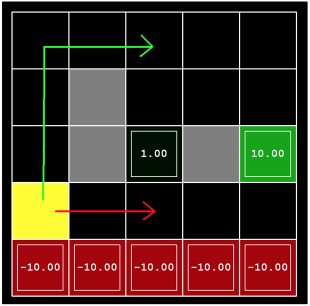
- How does the noise of $\mathrm{P}(s'|s;a)$ and γ affect π^*?



1. $\gamma = 0.99$, noise $= 0.5$
2. $\gamma = 0.99$, noise $= 0$
3. $\gamma = 0.1$, noise $= 0.5$
4. $\gamma = 0.1$, noise $= 0$

Effect of MDP Parameters

- How does the noise of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and γ affect π^*?

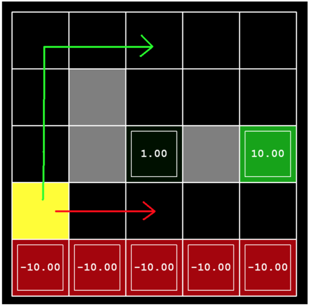


1. $\gamma = 0.99$, noise $= 0.5$
2. $\gamma = 0.99$, noise $= 0$
3. $\gamma = 0.1$, noise $= 0.5$
4. $\gamma = 0.1$, noise $= 0$

- π^* prefers the close exit $(+1)$; risking the cliff (-10)?
- π^* prefers the close exit $(+1)$; avoiding the cliff (-10)?
- π^* prefers the distant exit $(+10)$; risking the cliff (-10)?
- π^* prefers the distant exit $(+10)$; avoiding the cliff (-10)?

Effect of MDP Parameters

- How does the noise of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and γ affect π^*?

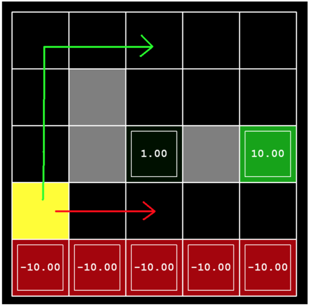


1. $\gamma = 0.99$, noise $= 0.5$
2. $\gamma = 0.99$, noise $= 0$
3. $\gamma = 0.1$, noise $= 0.5$
4. $\gamma = 0.1$, noise $= 0$

- π^* prefers the close exit $(+1)$; risking the cliff (-10)? (4)
- π^* prefers the close exit $(+1)$; avoiding the cliff (-10)?
- π^* prefers the distant exit $(+10)$; risking the cliff (-10)?
- π^* prefers the distant exit $(+10)$; avoiding the cliff (-10)?

Effect of MDP Parameters

- How does the noise of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and γ affect π^*?

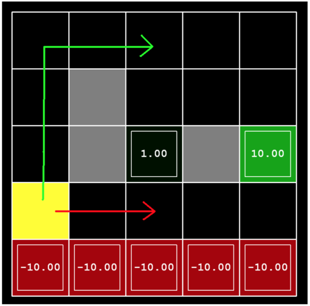


1. $\gamma = 0.99$, noise $= 0.5$
2. $\gamma = 0.99$, noise $= 0$
3. $\gamma = 0.1$, noise $= 0.5$
4. $\gamma = 0.1$, noise $= 0$

- π^* prefers the close exit $(+1)$; risking the cliff (-10)? (4)
- π^* prefers the close exit $(+1)$; avoiding the cliff (-10)? (3)
- π^* prefers the distant exit $(+10)$; risking the cliff (-10)?
- π^* prefers the distant exit $(+10)$; avoiding the cliff (-10)?

Effect of MDP Parameters

- How does the noise of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and γ affect π^*?

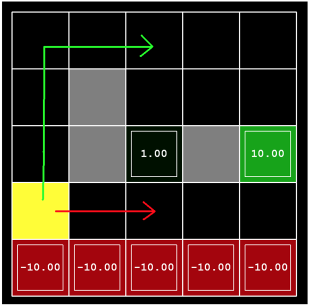


1. $\gamma = 0.99$, noise $= 0.5$
2. $\gamma = 0.99$, noise $= 0$
3. $\gamma = 0.1$, noise $= 0.5$
4. $\gamma = 0.1$, noise $= 0$

- π^* prefers the close exit $(+1)$; risking the cliff (-10)? (4)
- π^* prefers the close exit $(+1)$; avoiding the cliff (-10)? (3)
- π^* prefers the distant exit $(+10)$; risking the cliff (-10)? (2)
- π^* prefers the distant exit $(+10)$; avoiding the cliff (-10)?

Effect of MDP Parameters

- How does the noise of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and γ affect π^*?



1. $\gamma = 0.99$, noise $= 0.5$
2. $\gamma = 0.99$, noise $= 0$
3. $\gamma = 0.1$, noise $= 0.5$
4. $\gamma = 0.1$, noise $= 0$

- π^* prefers the close exit $(+1)$; risking the cliff (-10)? (4)
- π^* prefers the close exit $(+1)$; avoiding the cliff (-10)? (3)
- π^* prefers the distant exit $(+10)$; risking the cliff (-10)? (2)
- π^* prefers the distant exit $(+10)$; avoiding the cliff (-10)? (1)

Outline

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

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

Algorithm: Policy Iteration (Simplified)

- Given a π, define its ***value function***:

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

  - Expected accumulative reward when starting from state s and acting based on π

Algorithm: Policy Iteration (Simplified)

- Given a π, define its ***value function***:

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

  - Expected accumulative reward when starting from state $\boldsymbol{s}$ and acting based on π

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 $\pi(\boldsymbol{s})$ randomly;
repeat
    Evaluate $V_{\pi}(\boldsymbol{s}),\forall\boldsymbol{s}$;
    Improve π such that $V_{\pi}(\boldsymbol{s})$, $\forall\boldsymbol{s}$, becomes higher;
until $\pi(\boldsymbol{s})$*'s converge*;

Algorithm 4: Policy iteration.

Evaluating $V_\pi(s)$

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

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

Evaluating $V_\pi(s)$

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

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

- ***Bellman expectation equation***:

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

M1 Solve the system of linear equations

- $|\mathbb{S}|$ linear equations and $|\mathbb{S}|$ variables
- Time complexity: $O(|\mathbb{S}|^3)$

Evaluating $V_\pi(s)$

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

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

- ***Bellman expectation equation***:

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

M1 Solve the system of linear equations
- $|\mathbb{S}|$ linear equations and $|\mathbb{S}|$ variables
- Time complexity: $O(|\mathbb{S}|^3)$

M2 Dynamic programming (just like value iteration):
- Initializes $V_\pi(s) = 0, \forall s$
- $V_\pi(s) \leftarrow \sum_{s'} \mathrm{P}(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')], \forall s$

Policy Improvement

- How to find $\hat{\pi}$ such that $V_{\hat{\pi}}(\boldsymbol{s}) \geq V_{\pi}(\boldsymbol{s})$ for all $\boldsymbol{s}$?

Policy Improvement

- How to find $\hat{\pi}$ such that $V_{\hat{\pi}}(\boldsymbol{s}) \geq V_{\pi}(\boldsymbol{s})$ for all $\boldsymbol{s}$?
- Update rule: for all $\boldsymbol{s}$ do

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

- Why?

Policy Improvement

- How to find $\hat{\pi}$ such that $V_{\hat{\pi}}(\boldsymbol{s}) \geq V_{\pi}(\boldsymbol{s})$ for all $\boldsymbol{s}$?
- Update rule: for all $\boldsymbol{s}$ do

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

- Why?

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

Policy Improvement

- How to find $\hat{\pi}$ such that $V_{\hat{\pi}}(\boldsymbol{s}) \geq V_{\pi}(\boldsymbol{s})$ for all $\boldsymbol{s}$?
- Update rule: for all $\boldsymbol{s}$ do

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

- Why?

$$\begin{aligned} V_{\pi}(\boldsymbol{s}) &= \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_{\pi}(\boldsymbol{s}')] \\ &\leq \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\hat{\pi}(\boldsymbol{s}))[R(\boldsymbol{s},\hat{\pi}(\boldsymbol{s}),\boldsymbol{s}') + \gamma {\color{red}V_{\pi}(\boldsymbol{s}')}] \end{aligned}$$

Policy Improvement

- How to find $\hat{\pi}$ such that $V_{\hat{\pi}}(\boldsymbol{s}) \geq V_{\pi}(\boldsymbol{s})$ for all $\boldsymbol{s}$?
- Update rule: for all $\boldsymbol{s}$ do

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

- Why?

$$\begin{aligned}
V_{\pi}(\boldsymbol{s}) &= \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_{\pi}(\boldsymbol{s}')] \\
&\leq \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\hat{\pi}(\boldsymbol{s}))[R(\boldsymbol{s},\hat{\pi}(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_{\pi}(\boldsymbol{s}')] \\
&\leq \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\hat{\pi}(\boldsymbol{s}))\,\{R(\boldsymbol{s},\hat{\pi}(\boldsymbol{s}),\boldsymbol{s}') + \\
&\qquad\qquad \gamma \sum_{\boldsymbol{s}''} \mathrm{P}(\boldsymbol{s}''|\boldsymbol{s}';\hat{\pi}(\boldsymbol{s}'))[R(\boldsymbol{s}',\hat{\pi}(\boldsymbol{s}'),\boldsymbol{s}'') + \gamma V_{\pi}(\boldsymbol{s}'')]\}
\end{aligned}$$

Policy Improvement

- How to find $\hat{\pi}$ such that $V_{\hat{\pi}}(\boldsymbol{s}) \geq V_{\pi}(\boldsymbol{s})$ for all $\boldsymbol{s}$?
- Update rule: for all $\boldsymbol{s}$ do

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

- Why?

$$\begin{array}{rl} V_{\pi}(\boldsymbol{s}) & = \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_{\pi}(\boldsymbol{s}')] \\ & \leq \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\hat{\pi}(\boldsymbol{s}))[R(\boldsymbol{s},\hat{\pi}(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_{\pi}(\boldsymbol{s}')] \\ & \leq \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\hat{\pi}(\boldsymbol{s}))\,\{R(\boldsymbol{s},\hat{\pi}(\boldsymbol{s}),\boldsymbol{s}') + \\ & \qquad\qquad \gamma \sum_{\boldsymbol{s}''} \mathrm{P}(\boldsymbol{s}''|\boldsymbol{s}';\hat{\pi}(\boldsymbol{s}'))[R(\boldsymbol{s}',\hat{\pi}(\boldsymbol{s}'),\boldsymbol{s}'') + \gamma V_{\pi}(\boldsymbol{s}'')]\} \\ & \cdots \\ & \leq \mathrm{E}_{\mathbf{s}',\mathbf{s}'',\cdots}\left(R(\mathbf{s},\hat{\pi}(\mathbf{s}),\mathbf{s}') + \gamma R(\mathbf{s}',\hat{\pi}(\mathbf{s}'),\mathbf{s}'') + \cdots | \mathbf{s}^{(0)} = \boldsymbol{s};\, \hat{\pi}\right) \\ & = V_{\hat{\pi}}(\boldsymbol{s}), \forall \boldsymbol{s} \end{array}$$

Algorithm: Policy Iteration

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

Algorithm 5: Policy iteration.

Convergence

Theorem

Policy iteration converges and gives the optimal policy π^* when $H \to \infty$.

Convergence

Theorem

Policy iteration converges and gives the optimal policy π^* when $H \to \infty$.

- Convergence: in every step the policy improves

Convergence

Theorem

Policy iteration converges and gives the optimal policy π^* when $H \to \infty$.

- Convergence: in every step the policy improves
- Optimal policy: at convergence, $V_\pi(\boldsymbol{s})$, $\forall \boldsymbol{s}$, satisfies the Bellman optimality equation

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

Outline

Difference from MDP

- In practice, we may not be able to model the environment as an MDP

Difference from MDP

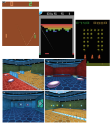
- In practice, we may not be able to model the environment as an MDP
- Unknown transition distribution $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$

Kohl and Stone, 2004

Ng et al, 2004

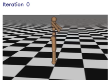
Tedrake et al, 2005

Kober and Peters, 2009

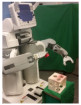


Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

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)

Difference from MDP

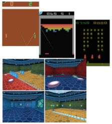
- In practice, we may not be able to model the environment as an MDP
- Unknown transition distribution $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$

Kohl and Stone, 2004

Ng et al, 2004

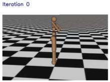
Tedrake et al, 2005

Kober and Peters, 2009

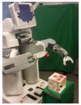


Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

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)

- Unknown reward function $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$

Exploration vs. Exploitation

- What would you do to get some rewards?

Exploration vs. Exploitation

- What would you do to get some rewards?

1. Perform actions randomly to ***explore*** $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ first
    - Collect samples so to estimate $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$

Exploration vs. Exploitation

- What would you do to get some rewards?

1. Perform actions randomly to ***explore*** $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ first
   - Collect samples so to estimate $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$
2. Then, perform actions to ***exploit*** the learned $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$
   - π^* can be computed/planned "in mind" using value/policy iteration

Model-based RL using Monte Carlo Estimation

1. Use some ***exploration policy*** π' to perform one or more ***episodes***/***trails***
   - Each episode records samples of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ from start to terminal state

$$\boldsymbol{s}^{(0)} \xrightarrow{\pi'(\boldsymbol{s}^{(0)})} \boldsymbol{s}^{(1)} \xrightarrow{\pi'(\boldsymbol{s}^{(1)})} \cdots \xrightarrow{\pi'(\boldsymbol{s}^{(H-1)})} \boldsymbol{s}^{(H)}$$

   and

$$R(\boldsymbol{s}^{(0)},\pi'(\boldsymbol{s}^{(0)}),\boldsymbol{s}^{(1)}) \to \cdots \to R(\boldsymbol{s}^{(H-1)},\pi(\boldsymbol{s}^{(H-1)}),\boldsymbol{s}^{(H)})$$

2. Estimate $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ using the samples and update the ***exploitation policy*** π
   - $\hat{\mathrm{P}}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a}) = \dfrac{\#\text{ times the action } \boldsymbol{a} \text{ takes state } \boldsymbol{s} \text{ to state } \boldsymbol{s}'}{\#\text{ times action } \boldsymbol{a} \text{ is taken in state } \boldsymbol{s}}$
   - $\hat{R}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') =$ average of reward values received when $\boldsymbol{a}$ takes $\boldsymbol{s}$ to $\boldsymbol{s}'$
3. Repeat from Step 1, but gradually mix π into π'
   - Mix-in strategy? E.g., ε-greedy (more on this later)

Problems of Model-based RL

- There may be lots of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ to estimate
- If $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ is low, there may be too few samples to have a good estimate
- Low $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ may also lead to a poor estimate of $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$
  - when $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ depends on $\boldsymbol{s}'$

Problems of Model-based RL

- There may be lots of $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ to estimate
- If $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ is low, there may be too few samples to have a good estimate
- Low $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$ may also lead to a poor estimate of $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$
  - when $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$ depends on $\boldsymbol{s}'$
- We estimate $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$'s and $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$'s in order to compute $V^*(\boldsymbol{s})/V_\pi(\boldsymbol{s})$ and solve π^*
- Why not estimate $V^*(\boldsymbol{s})/V_\pi(\boldsymbol{s})$ directly?

Outline

Model-based vs. Model-Free

- How to estimate $\mathrm{E}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \mathrm{P}(\boldsymbol{x}|\boldsymbol{y})f(\boldsymbol{x})$ given samples $(\boldsymbol{x}^{(1)},\boldsymbol{y}^{(1)}),(\boldsymbol{x}^{(2)},\boldsymbol{y}^{(2)}),\cdots$?

Model-based vs. Model-Free

- How to estimate $\mathrm{E}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \mathrm{P}(\boldsymbol{x}|\boldsymbol{y}) f(\boldsymbol{x})$ given samples $(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \cdots$?
- ***Model-based*** estimation:
  1. For each $\boldsymbol{x}$, estimate $\hat{\mathrm{P}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{\text{count}(\boldsymbol{x}^{(i)}=\boldsymbol{x} \text{ and } \boldsymbol{y}^{(i)}=\boldsymbol{y})}{\text{count}(\boldsymbol{y}^{(j)}=\boldsymbol{y})}$
  2. $\hat{\mathrm{E}}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \hat{\mathrm{P}}(\boldsymbol{x}|\boldsymbol{y}) f(\boldsymbol{x})$

Model-based vs. Model-Free

- How to estimate $\mathrm{E}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \mathrm{P}(\boldsymbol{x}|\boldsymbol{y})f(\boldsymbol{x})$ given samples $(\boldsymbol{x}^{(1)},\boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)},\boldsymbol{y}^{(2)}), \cdots$?
- ***Model-based*** estimation:
  1. For each $\boldsymbol{x}$, estimate $\hat{\mathrm{P}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{\text{count}(\boldsymbol{x}^{(i)}=\boldsymbol{x} \text{ and } \boldsymbol{y}^{(i)}=\boldsymbol{y})}{\text{count}(\boldsymbol{y}^{(j)}=\boldsymbol{y})}$
  2. $\hat{\mathrm{E}}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \hat{\mathrm{P}}(\boldsymbol{x}|\boldsymbol{y})f(\boldsymbol{x})$
- ***Model-free*** estimation:

$$\hat{\mathrm{E}}(f(\mathbf{x})|\boldsymbol{y}) = \frac{1}{\text{count}(\boldsymbol{y}^{(j)} = \boldsymbol{y})} \sum_{i:\boldsymbol{x}^{(i)}=\boldsymbol{x} \text{ and } \boldsymbol{y}^{(i)}=\boldsymbol{y}} f(\boldsymbol{x}^{(i)})$$

  - Why does it work?

Model-based vs. Model-Free

- How to estimate $\mathrm{E}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \mathrm{P}(\boldsymbol{x}|\boldsymbol{y})f(\boldsymbol{x})$ given samples $(\boldsymbol{x}^{(1)},\boldsymbol{y}^{(1)}),(\boldsymbol{x}^{(2)},\boldsymbol{y}^{(2)}),\cdots$?
- ***Model-based*** estimation:
  1. For each $\boldsymbol{x}$, estimate $\hat{\mathrm{P}}(\boldsymbol{x}|\boldsymbol{y}) = \frac{\text{count}(\boldsymbol{x}^{(i)}=\boldsymbol{x} \text{ and } \boldsymbol{y}^{(i)}=\boldsymbol{y})}{\text{count}(\boldsymbol{y}^{(j)}=\boldsymbol{y})}$
  2. $\hat{\mathrm{E}}(f(\mathbf{x})|\boldsymbol{y}) = \sum_{\boldsymbol{x}} \hat{\mathrm{P}}(\boldsymbol{x}|\boldsymbol{y})f(\boldsymbol{x})$
- ***Model-free*** estimation:

$$\hat{\mathrm{E}}(f(\mathbf{x})|\boldsymbol{y}) = \frac{1}{\text{count}(\boldsymbol{y}^{(j)}=\boldsymbol{y})} \sum_{i:\boldsymbol{x}^{(i)}=\boldsymbol{x} \text{ and } \boldsymbol{y}^{(i)}=\boldsymbol{y}} f(\boldsymbol{x}^{(i)})$$

  - Why does it work? Because samples appear in right frequencies

Challenges of Model-Free RL

- Value iteration: start from $V^*(\boldsymbol{s}) \leftarrow 0$
  1. Iterate $V^*(\boldsymbol{s}) \leftarrow \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$ until converge
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$
- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $V_\pi(\boldsymbol{s}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_\pi(\boldsymbol{s}')]$ until converge
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V_\pi(\boldsymbol{s}')]$

Challenges of Model-Free RL

- Value iteration: start from $V^*(\boldsymbol{s}) \leftarrow 0$
  1. Iterate $V^*(\boldsymbol{s}) \leftarrow \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$ until converge

     ☹ ***Not easy to estimate*** $V^*(\boldsymbol{s}) = \max_{\pi} \mathrm{E}\left(\sum_t \gamma^t R^{(t)} | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi\right)$
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$

     ☹ ***Need model to solve***
- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $V_\pi(\boldsymbol{s}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma V_\pi(\boldsymbol{s}')]$ until converge

     ☺ ***Can estimate*** $V_\pi(\boldsymbol{s}) = \mathrm{E}\left(\sum_t \gamma^t R^{(t)} | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi\right)$ ***using MC est.***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V_\pi(\boldsymbol{s}')]$

     ☹ ***Need model to solve***

Q Function for π

- Value function for a given π:

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

  with recurrence (Bellman expectation equation):

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

  - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$ and acting based on π onward

Q Function for π

- Value function for a given π:

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

  with recurrence (Bellman expectation equation):

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

  - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$ and acting based on π onward
- Define ***Q function*** for π:

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

  such that $V_\pi(\boldsymbol{s}) = Q_\pi(\boldsymbol{s},\pi(\boldsymbol{s}))$ with recurrence:

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

  - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$, taking action $\boldsymbol{a}$, and then acting based on π

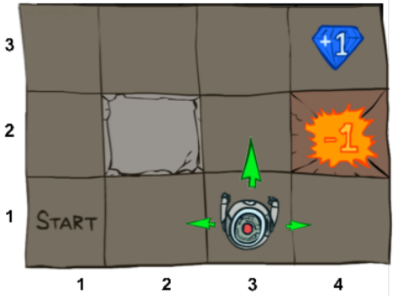
Algorithm: Policy Iteration based on Q_π

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 $V_\pi(s) \leftarrow 0$~~ Initialize $Q_\pi(s,a) = 0, \forall s,a$;
  repeat
    foreach s *and* a **do**
      $\cancel{V_\pi(s) \leftarrow \sum_{s'} \mathrm{P}(s'|s;\pi(s))[R(s,\pi(s),s') + \gamma V_\pi(s')]}$
      $Q_\pi(s,a) \leftarrow \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma Q_\pi(s',\pi(s'))]$;
    end
  until $\cancel{V_\pi(s)\text{'s}}$ $Q_\pi(s,a)$*'s converge*;
  foreach s **do**
    $\cancel{\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathrm{P}(s'|s;a)[R(s,a,s') + \gamma V_\pi(s')]}$
    $\pi(s) \leftarrow \arg\max_a Q_\pi(s,a)$;
  end
until $\pi(s)$*'s converge*;

Algorithm 6: Policy iteration.

Example


3
2
1
+1
-1
START
1
2
3
4


0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS


0.59
0.67
0.77
0.57
0.64
0.60
0.74
0.66
0.85
1.00
0.53
0.67
0.57
0.57
0.57
0.51
0.51
0.53
-0.60
-1.00
0.46
0.30
0.49
0.40
0.48
-0.65
0.45
0.41
0.43
0.42
0.40
0.29
0.28
0.13
0.44
0.40
0.41
0.27
Q-VALUES AFTER 100 ITERATIONS

Policy Iteration and Model-Free RL

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$

Policy Iteration and Model-Free RL

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge
     ☺ ***Can estimate*** $Q_\pi(\boldsymbol{s},\boldsymbol{a}) = \mathrm{E}\left(R^{(0)} + \sum_{t=1} \gamma^t R^{(t)}\right)$ ***using MC est.***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$
     ☻ ***No need for model to solve***

Policy Iteration and Model-Free RL

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge

     ☺ ***Can estimate*** $Q_\pi(\boldsymbol{s},\boldsymbol{a}) = \mathrm{E}\left(R^{(0)} + \sum_{t=1}\gamma^t R^{(t)}\right)$ ***using MC est.***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$

     ☺ ***No need for model to solve***
- RL: start from a random π ***for both exploration & exploitation***, repeat until converge:
  1. Create episodes using π and get MC estimates $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s},\boldsymbol{a}$
  2. Update π by $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s}$

Policy Iteration and Model-Free RL

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge
     ☺ ***Can estimate*** $Q_\pi(\boldsymbol{s},\boldsymbol{a}) = \mathrm{E}\left(R^{(0)} + \sum_{t=1} \gamma^t R^{(t)}\right)$ ***using MC est.***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$
     ☹ ***No need for model to solve***
- RL: start from a random π ***for both exploration & exploitation***, repeat until converge:
  1. Create episodes using π and get MC estimates $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s},\boldsymbol{a}$
  2. Update π by $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s}$
- Problem: π improves little after running lots of trials

Policy Iteration and Model-Free RL

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge

     ☺ ***Can estimate*** $Q_\pi(\boldsymbol{s},\boldsymbol{a}) = \mathrm{E}\left(R^{(0)} + \sum_{t=1} \gamma^t R^{(t)}\right)$ ***using MC est.***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$

     ☺ ***No need for model to solve***
- RL: start from a random π ***for both exploration & exploitation***, repeat until converge:
  1. Create episodes using π and get MC estimates $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s},\boldsymbol{a}$
  2. Update π by $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}), \forall \boldsymbol{s}$
- Problem: π improves little after running lots of trials
- Can we improve π right after each action?

Outline

Policy Iteration Revisited

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$

Policy Iteration Revisited

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge
     ☺ ***Can estimate*** $Q_\pi(\boldsymbol{s},\boldsymbol{a}) = \mathrm{E}\left(R^{(0)} + \sum_{t=1}\gamma^t R^{(t)}\right)$ ***using MC est.***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$
     ☺ ***No need for model to solve***

Policy Iteration Revisited

- Policy iteration: start from a random π, repeat until converge:
  1. Iterate $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$ until converge
     ☺ ***Can estimate*** $Q_\pi(s,a) = \mathrm{E}\left(R^{(0)} + \sum_{t=1} \gamma^t R^{(t)}\right)$ ***using MC est.***
     ☺ ***Can also estimate*** $Q_\pi(s,a)$ ***based on the recurrence***
  2. Solve $\pi(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q_\pi(\boldsymbol{s},\boldsymbol{a})$
     ☺ ***No need for model to solve***

Temporal Difference Estimation I

$$Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$$

- Given the samples

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

and

$$R(\boldsymbol{s}^{(0)},\boldsymbol{a}^{(0)},\boldsymbol{s}^{(1)}) \to \cdots \to R(\boldsymbol{s}^{(H-1)},\boldsymbol{a}^{(H-1)},\boldsymbol{s}^{(H)})$$

- ***Temporal difference (TD) estimation*** of $Q_\pi(\boldsymbol{s},\boldsymbol{a})$:
  1. $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow$ randon value, $\forall \boldsymbol{s},\boldsymbol{a}$
  2. Repeat until converge ***for each action $\boldsymbol{a}^{(t)}$***:
     $$\hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \leftarrow \hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) + \eta\left[(R(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)},\boldsymbol{s}^{(t+1)}) + \gamma\hat{Q}_\pi(\boldsymbol{s}^{(t+1)},\pi(\boldsymbol{s}^{(t+1)}))) - \hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)})\right]$$

Temporal Difference Estimation I

$$Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$$

- Given the samples

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

and

$$R(\boldsymbol{s}^{(0)},\boldsymbol{a}^{(0)},\boldsymbol{s}^{(1)}) \rightarrow \cdots \rightarrow R(\boldsymbol{s}^{(H-1)},\boldsymbol{a}^{(H-1)},\boldsymbol{s}^{(H)})$$

- ***Temporal difference (TD) estimation*** of $Q_\pi(\boldsymbol{s},\boldsymbol{a})$:
  1. $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow$ randon value, $\forall \boldsymbol{s},\boldsymbol{a}$
  2. Repeat until converge ***for each action $\boldsymbol{a}^{(t)}$***:

$$\hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \leftarrow \hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) + \eta\left[\left(R(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)},\boldsymbol{s}^{(t+1)}) + \gamma\hat{Q}_\pi(\boldsymbol{s}^{(t+1)},\pi(\boldsymbol{s}^{(t+1)}))\right) - \hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)})\right]$$

- $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})$ can be updated ***on the fly*** during an episode

Temporal Difference Estimation I

$$Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\pi(\boldsymbol{s}))[R(\boldsymbol{s},\pi(\boldsymbol{s}),\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))]$$

- Given the samples

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

  and

$$R(\boldsymbol{s}^{(0)},\boldsymbol{a}^{(0)},\boldsymbol{s}^{(1)}) \to \cdots \to R(\boldsymbol{s}^{(H-1)},\boldsymbol{a}^{(H-1)},\boldsymbol{s}^{(H)})$$

- ***Temporal difference (TD) estimation*** of $Q_\pi(\boldsymbol{s},\boldsymbol{a})$:
  1. $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow$ randon value, $\forall \boldsymbol{s},\boldsymbol{a}$
  2. Repeat until converge ***for each action $\boldsymbol{a}^{(t)}$***:

$$\hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \leftarrow \hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) + \eta\left[\left(R(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)},\boldsymbol{s}^{(t+1)}) + \gamma \hat{Q}_\pi(\boldsymbol{s}^{(t+1)},\pi(\boldsymbol{s}^{(t+1)}))\right) - \hat{Q}_\pi(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)})\right]$$

- $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})$ can be updated ***on the fly*** during an episode
- η the "learning rate"?

Temporal Difference Estimation II

$$\begin{aligned}\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow\ & \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta \left[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) - \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \right] \\ & = \eta (R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) + (1-\eta)\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\end{aligned},$$

Temporal Difference Estimation II

$$\begin{array}{rl}\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow & \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\left[\left(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma\hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))\right) - \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\right] \\ & = \eta\left(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma\hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))\right) + (1-\eta)\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\end{array},$$

- ***Exponential moving average***:

$$\bar{x}_n = \frac{x^{(n)} + (1-\eta)x^{(n-1)} + (1-\eta)^2 x^{(n-2)} + \cdots}{1 + (1-\eta) + (1-\eta)^2 + \cdots},$$

where $\eta \in [0,1]$ is the “forget rate”

- Recent samples are exponentially more important

Temporal Difference Estimation II

$$\begin{aligned}\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \;& \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\left[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma\hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) - \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\right] \\ & = \eta(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma\hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) + (1-\eta)\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\end{aligned},$$

- ***Exponential moving average***:

$$\bar{x}_n = \frac{x^{(n)} + (1-\eta)x^{(n-1)} + (1-\eta)^2 x^{(n-2)} + \cdots}{1 + (1-\eta) + (1-\eta)^2 + \cdots},$$

  where $\eta \in [0,1]$ is the "forget rate"
  - Recent samples are exponentially more important
- Since $1/\eta = 1 + (1-\eta) + (1-\eta)^2 + \cdots$, we have $\bar{x}_n = \eta x^{(n)} + (1-\eta)\bar{x}_{n-1}$ [Proof]

Temporal Difference Estimation II

$$\begin{aligned}\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \quad & \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\left[\left(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma\hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))\right) - \hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\right] \\ & = \eta\left(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma\hat{Q}_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))\right) + (1-\eta)\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})\end{aligned},$$

- ***Exponential moving average***:

$$\bar{x}_n = \frac{x^{(n)} + (1-\eta)x^{(n-1)} + (1-\eta)^2 x^{(n-2)} + \cdots}{1 + (1-\eta) + (1-\eta)^2 + \cdots},$$

  where $\eta \in [0,1]$ is the "forget rate"
  - Recent samples are exponentially more important
- Since $1/\eta = 1 + (1-\eta) + (1-\eta)^2 + \cdots$, we have $\bar{x}_n = \eta x^{(n)} + (1-\eta)\bar{x}_{n-1}$ [Proof]
- As long as η gradually decreases to 0:
  - $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})$ degenerates to average of accumulative rewards
  - $\hat{Q}_\pi(\boldsymbol{s},\boldsymbol{a})$ converges to $Q_\pi(\boldsymbol{s},\boldsymbol{a})$ (more on this later)

Algorithm: SARSA (Simplified)

Input: $\mathbb{S}$, $\mathbb{A}$, and γ
Output: $\pi^*(\boldsymbol{s})$'s for all $\boldsymbol{s}$'s

For each state $\boldsymbol{s}$ and $\boldsymbol{a}$, initialize $Q_\pi(\boldsymbol{s},\boldsymbol{a})$ arbitrarily;
foreach *episode* **do**
  Set $\boldsymbol{s}$ to initial state;
  repeat
    Take action $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s},\boldsymbol{a}')$;
    Observe $\boldsymbol{s}'$ and reward $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$;
    $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow$
     $Q_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\left[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) - Q_\pi(\boldsymbol{s},\boldsymbol{a})\right]$;
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$;
  until $\boldsymbol{s}$ *is terminal state*;
end

Algorithm 7: State-Action-Reward-State-Action (SARSA).

Algorithm: SARSA (Simplified)

Input: $\mathbb{S}$, $\mathbb{A}$, and γ
Output: $\pi^*(\boldsymbol{s})$'s for all $\boldsymbol{s}$'s

For each state $\boldsymbol{s}$ and $\boldsymbol{a}$, initialize $Q_\pi(\boldsymbol{s},\boldsymbol{a})$ arbitrarily;
foreach *episode* **do**
  Set $\boldsymbol{s}$ to initial state;
  repeat
    Take action $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s},\boldsymbol{a}')$;
    Observe $\boldsymbol{s}'$ and reward $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$;
    $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow$
    $Q_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\,[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) - Q_\pi(\boldsymbol{s},\boldsymbol{a})]$;
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$;
  until $\boldsymbol{s}$ *is terminal state*;
end

Algorithm 8: State-Action-Reward-State-Action (SARSA).

- Policy improves ***each time*** when deciding the next action $\boldsymbol{a}$

Convergence

Theorem

SARSA converges and gives the optimal policy π^* almost surely if

1) π is GLIE (Greedy in the Limit with Infinite Exploration);

2) η small enough eventually, but not decreasing too fast.

Convergence

Theorem

SARSA converges and gives the optimal policy π^* almost surely if
1) π is GLIE (Greedy in the Limit with Infinite Exploration);
2) η small enough eventually, but not decreasing too fast.

- Greedy in the limit: the policy π converges (in the limit) to the exploitation/greedy policy
  - At each step, we choose $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s}, \boldsymbol{a}')$

Convergence

Theorem

SARSA converges and gives the optimal policy π^* almost surely if

1) π is GLIE (Greedy in the Limit with Infinite Exploration);

2) η small enough eventually, but not decreasing too fast.

- Greedy in the limit: the policy π converges (in the limit) to the exploitation/greedy policy
  - At each step, we choose $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s}, \boldsymbol{a}')$
- Infinite exploration: all $(\boldsymbol{s}, \boldsymbol{a})$ pairs are visited infinite times
  - $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s}, \boldsymbol{a}')$ ***cannot*** guarantee this
  - Need a better way (next section)

Convergence

Theorem

SARSA converges and gives the optimal policy π^* almost surely if
1) π is GLIE (Greedy in the Limit with Infinite Exploration);
2) η small enough eventually, but not decreasing too fast.

- Greedy in the limit: the policy π converges (in the limit) to the exploitation/greedy policy
  - At each step, we choose $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s}, \boldsymbol{a}')$
- Infinite exploration: all $(\boldsymbol{s}, \boldsymbol{a})$ pairs are visited infinite times
  - $\boldsymbol{a} \leftarrow \arg\max_{\boldsymbol{a}'} Q_\pi(\boldsymbol{s}, \boldsymbol{a}')$ ***cannot*** guarantee this
  - Need a better way (next section)
- Furthermore, η should satisfy: $\sum_t \eta^{(t)} = \infty$ and $\sum_t \eta^{(t)2} < \infty$
  - E.g., $\eta^{(t)} = O(\frac{1}{t})$
  - $\sum_t \frac{1}{t}$ is a harmonic series known to diverge
  - $\sum_t (\frac{1}{t})^p$, $p > 1$, is a p-series converging to Riemann zeta $\zeta(p)$

Outline

General RL Steps

- Repeat until converge:
  1. Use some exploration policy π' to run episodes/trails
  2. Estimate targets (e.g., $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$, $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$, or $Q_\pi(\boldsymbol{s},\boldsymbol{a})$) and update the exploitation policy π
  3. Gradually mix π into π'

General RL Steps

- Repeat until converge:
  1. Use some exploration policy π' to run episodes/trails
  2. Estimate targets (e.g., $\mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})$, $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$, or $Q_\pi(\boldsymbol{s},\boldsymbol{a})$) and update the exploitation policy π
  3. Gradually mix π into π'
- Goals of mix-in/exploration strategy:
  - Infinite exploration with π'
  - π' is greedy/exploitative in the end

ε-Greedy Strategy

- At every time step, flip a coin
  - With probability ε, act randomly (explore)
  - With probability $(1-\varepsilon)$, compute/update the exploitation policy and act accordingly (exploit)

ε-Greedy Strategy

- At every time step, flip a coin
  - With probability ε, act randomly (explore)
  - With probability $(1-\varepsilon)$, compute/update the exploitation policy and act accordingly (exploit)
- Gradually decrease ε over time

ε-Greedy Strategy

- At every time step, flip a coin
  - With probability ε, act randomly (explore)
  - With probability $(1-\varepsilon)$, compute/update the exploitation policy and act accordingly (exploit)
- Gradually decrease ε over time
- At each time step, the action is at either of two extremes
  - Exploration or exploitation
- "Soft" policy between the two extremes?

Softmax Strategy

- Idea: perform action $\boldsymbol{a}$ more often if $\boldsymbol{a}$ gives more accumulative rewards
- E.g., in SARSA, choose $\boldsymbol{a}$ from $\boldsymbol{s}$ by sampling from the distribution:

$$\mathrm{P}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(Q_\pi(\boldsymbol{s},\boldsymbol{a})/t)}{\sum_{\boldsymbol{a}'}(\exp Q_\pi(\boldsymbol{s},\boldsymbol{a}')/t)}, \ \forall \boldsymbol{a}$$

  - Softmax function converts $Q_\pi(\boldsymbol{s},\boldsymbol{a})$'s to probabilities

Softmax Strategy

- Idea: perform action $\boldsymbol{a}$ more often if $\boldsymbol{a}$ gives more accumulative rewards
- E.g., in SARSA, choose $\boldsymbol{a}$ from $\boldsymbol{s}$ by sampling from the distribution:

$$\mathrm{P}(\boldsymbol{a}|\boldsymbol{s}) = \frac{\exp(Q_\pi(\boldsymbol{s},\boldsymbol{a})/t)}{\sum_{\boldsymbol{a}'}(\exp Q_\pi(\boldsymbol{s},\boldsymbol{a}')/t)}, \forall \boldsymbol{a}$$

  - Softmax function converts $Q_\pi(\boldsymbol{s},\boldsymbol{a})$'s to probabilities
- ***Temperature*** t starts from a high value (exploration), and decreases over time (exploitation)

Exploration Function

- Idea: to explore areas with fewest samples
- E.g., in each step of SARSA, define an exploration function

$$f(q,n) = q + K/n,$$

where

  - q the estimated Q-value
  - n the number of samples for the estimate
  - K some positive constant

Exploration Function

- Idea: to explore areas with fewest samples
- E.g., in each step of SARSA, define an exploration function

$$f(q,n) = q + K/n,$$

  where
  - q the estimated Q-value
  - n the number of samples for the estimate
  - K some positive constant
- Instead of:

$$Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\left[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\boldsymbol{a}')) - Q_\pi(\boldsymbol{s},\boldsymbol{a})\right]$$

- Use f when updating $Q_\pi(\boldsymbol{s},\boldsymbol{a})$:

$$\begin{aligned} Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow\ & Q_\pi(\boldsymbol{s},\boldsymbol{a}) + \\ & \eta\left\{[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma f(Q_\pi(\boldsymbol{s}',\boldsymbol{a}'), \text{count}(\boldsymbol{s}',\boldsymbol{a}'))] - Q_\pi(\boldsymbol{s},\boldsymbol{a})\right\} \end{aligned}$$

  - Infinite exploration
  - Exploit once exploring enough

Outline

Model-Free RL with Value Iteration?

- Value iteration: start from $V^*(\boldsymbol{s}) \leftarrow 0$
  1. Iterate $V^*(\boldsymbol{s}) \leftarrow \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$ until converge
     ☹ ***Not easy to estimate*** $V^*(\boldsymbol{s}) = \max_\pi \mathrm{E}\left(\sum_t \gamma^t R^{(t)} | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi\right)$
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$
     ☹ ***Need model to solve***

Model-Free RL with Value Iteration?

- Value iteration: start from $V^*(\boldsymbol{s}) \leftarrow 0$
  1. Iterate $V^*(\boldsymbol{s}) \leftarrow \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$ until converge
     ☹ ***Not easy to estimate*** $V^*(\boldsymbol{s}) = \max_{\pi} \mathrm{E}\left(\sum_t \gamma^t R^{(t)} | \mathbf{s}^{(0)} = \boldsymbol{s}; \pi\right)$
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]$
     ☹ ***Need model to solve***
- Q-version that helps sample-based estimation?

Optimal Q Function

- Optimal value function:

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

  with recurrence

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

  - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$ and acting optimally onward

Optimal Q Function

- Optimal value function:

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

with recurrence

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

  - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$ and acting optimally onward
- Q^* ***function***:

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

such that $V^*(\boldsymbol{s}) = \max_{\boldsymbol{a}} Q^*(\boldsymbol{s},\boldsymbol{a})$ with recurrence:

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

  - Maximum expected accumulative reward when starting from state $\boldsymbol{s}$, taking actin $\boldsymbol{a}$, and then acting optimally onward

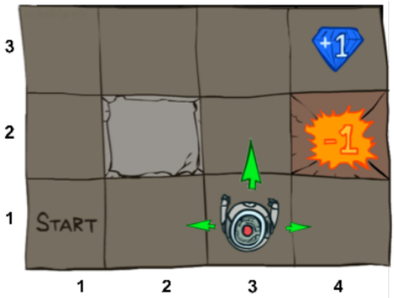
Algorithm: Q-Value Iteration

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 $V^*(\boldsymbol{s}) \leftarrow 0$~~ Initialize $Q^*(\boldsymbol{s},\boldsymbol{a}) = 0, \forall \boldsymbol{s}, \boldsymbol{a}$;
repeat
  foreach $\boldsymbol{s}$ *and* $\boldsymbol{a}$ **do**
    $\cancel{V^*(\boldsymbol{s}) \leftarrow \max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]}$
    $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')]$;
  end
until $\cancel{V^*(\boldsymbol{s})\text{'s}}$ $Q^*(\boldsymbol{s},\boldsymbol{a})$*'s converge*;
foreach $\boldsymbol{s}$ **do**
  $\cancel{\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma V^*(\boldsymbol{s}')]}$
  $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{s},\boldsymbol{a})$;
end

Algorithm 9: Q-Value iteration with infinite horizon.

Example



+1
-1
START
1
2
3
4

0.64
0.74
0.85
1.00
0.57
0.57
-1.00
0.49
0.43
0.48
0.28
VALUES AFTER 100 ITERATIONS

0.59
0.67
0.77
0.57
0.64
0.60
0.74
0.66
0.85
1.00
0.53
0.67
0.57
0.57
0.57
0.51
0.51
0.53
-0.60
-1.00
0.46
0.30
0.49
0.40
0.48
-0.65
0.45
0.41
0.43
0.42
0.40
0.29
0.28
0.13
0.44
0.40
0.41
0.27
Q-VALUES AFTER 100 ITERATIONS

Q-Value Iteration

- Value iteration: start from $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow 0$
  1. Iterate $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')]$ until converge
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{s},\boldsymbol{a})$

Q-Value Iteration

- Value iteration: start from $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow 0$
  1. Iterate $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')]$ until converge
     ☹ ***Still not easy to estimate*** $Q^*(\boldsymbol{s},\boldsymbol{a}) = \max_\pi \mathrm{E}\left(R^{(0)} + \sum_t \gamma^t R^{(t)}\right)$
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{s},\boldsymbol{a})$
     ☺ ***No need for model to solve***

Q-Value Iteration

- Value iteration: start from $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow 0$
  1. Iterate $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \sum_{\boldsymbol{s}'} \mathrm{P}(\boldsymbol{s}'|\boldsymbol{s};\boldsymbol{a})[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')]$ until converge
     ☹ ***Still not easy to estimate*** $Q^*(s,a) = \max_\pi \mathrm{E}\left(R^{(0)} + \sum_t \gamma^t R^{(t)}\right)$
     ☺ ***But we can estimate*** $Q^*(s,a)$ ***based on the recurrence now!***
  2. Solve $\pi^*(\boldsymbol{s}) \leftarrow \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{s},\boldsymbol{a})$
     ☺ ***No need for model to solve***

Temporal Difference Estimation

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

- Given an exploration policy π' and samples

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

  and

$$R(\boldsymbol{s}^{(0)},\boldsymbol{a}^{(0)},\boldsymbol{s}^{(1)}) \to \cdots \to R(\boldsymbol{s}^{(H-1)},\boldsymbol{a}^{(H-1)},\boldsymbol{s}^{(H)})$$

- ***Temporal difference (TD) estimation*** of $Q^*(\boldsymbol{s},\boldsymbol{a})$ for exploitation policy π:
  1. $\hat{Q}^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow$ randon value, $\forall \boldsymbol{s},\boldsymbol{a}$
  2. Repeat until converge ***for each action $\boldsymbol{a}^{(t)}$***:

$$\hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \leftarrow \hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) + \eta\left[(R(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)},\boldsymbol{s}^{(t+1)}) + \gamma \max_{\boldsymbol{a}} \hat{Q}^*(\boldsymbol{s}^{(t+1)},\boldsymbol{a})) - \hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)})\right]$$

Temporal Difference Estimation

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

- Given an exploration policy π' and samples

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

  and

$$R(\boldsymbol{s}^{(0)},\boldsymbol{a}^{(0)},\boldsymbol{s}^{(1)}) \to \cdots \to R(\boldsymbol{s}^{(H-1)},\boldsymbol{a}^{(H-1)},\boldsymbol{s}^{(H)})$$

- ***Temporal difference (TD) estimation*** of $Q^*(\boldsymbol{s},\boldsymbol{a})$ for exploitation policy π:
  1. $\hat{Q}^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow$ randon value, $\forall \boldsymbol{s},\boldsymbol{a}$
  2. Repeat until converge ***for each action $\boldsymbol{a}^{(t)}$***:

$$\hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \leftarrow \hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) + \eta \left[(R(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)},\boldsymbol{s}^{(t+1)}) + \gamma \max_{\boldsymbol{a}} \hat{Q}^*(\boldsymbol{s}^{(t+1)},\boldsymbol{a})) - \hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \right]$$

- $\hat{Q}^*(\boldsymbol{s},\boldsymbol{a})$ can be updated ***on the fly*** during exploration

Temporal Difference Estimation

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

- Given an exploration policy π' and samples

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

  and

$$R(\boldsymbol{s}^{(0)},\boldsymbol{a}^{(0)},\boldsymbol{s}^{(1)}) \to \cdots \to R(\boldsymbol{s}^{(H-1)},\boldsymbol{a}^{(H-1)},\boldsymbol{s}^{(H)})$$

- ***Temporal difference (TD) estimation*** of $Q^*(\boldsymbol{s},\boldsymbol{a})$ for exploitation policy π:
  1. $\hat{Q}^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow$ randon value, $\forall \boldsymbol{s},\boldsymbol{a}$
  2. Repeat until converge ***for each action $\boldsymbol{a}^{(t)}$***:

$$\hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) \leftarrow \hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)}) + \eta\left[(R(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)},\boldsymbol{s}^{(t+1)}) + \gamma \max_{\boldsymbol{a}} \hat{Q}^*(\boldsymbol{s}^{(t+1)},\boldsymbol{a})) - \hat{Q}^*(\boldsymbol{s}^{(t)},\boldsymbol{a}^{(t)})\right]$$

- $\hat{Q}^*(\boldsymbol{s},\boldsymbol{a})$ can be updated ***on the fly*** during exploration
- η the "forget rate" of moving avg. that gradually decreases

Algorithm: Q-Learning

Input: $\mathbb{S}$, $\mathbb{A}$, and γ
Output: $\pi^*(\boldsymbol{s})$'s for all $\boldsymbol{s}$'s

For each state $\boldsymbol{s}$ and $\boldsymbol{a}$, initialize $Q^*(\boldsymbol{s},\boldsymbol{a})$ arbitrarily;
foreach *episode* **do**
  Set $\boldsymbol{s}$ to initial state;
  repeat
    Take action $\boldsymbol{a}$ from $\boldsymbol{s}$ using some exploration policy π' derived from Q^* (e.g., ε-greedy);
    Observe $\boldsymbol{s}'$ and reward $R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')$;
    $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta\,[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')) - Q^*(\boldsymbol{s},\boldsymbol{a})]$;
    $\boldsymbol{s} \leftarrow \boldsymbol{s}'$;
  until $\boldsymbol{s}$ *is terminal state*;
end

Algorithm 10: Q-learning.

Convergence

Theorem

Q-learning converges and gives the optimal policy π^* if

1) π' has explored enough;

2) η small enough eventually, but not decreasing too fast.

Convergence

Theorem

Q-learning converges and gives the optimal policy π^* if
1) π' has explored enough;
2) η small enough eventually, but not decreasing too fast.

- π' has explored enough:
  - All states and actions are visited infinitely often
  - Does ***not*** matter how π' selects actions!

Convergence

Theorem

Q-learning converges and gives the optimal policy π^* if

1) π' has explored enough;

2) η small enough eventually, but not decreasing too fast.

- π' has explored enough:
  - All states and actions are visited infinitely often
  - Does ***not*** matter how π' selects actions!
- η satisfies $\sum_t \eta^{(t)} = \infty$ and $\sum_t \eta^{(t)2} < \infty$
  - E.g., $\eta^{(t)} = O(\frac{1}{t})$

Outline

1. Introduction
2. Markov Decision Process
   - Value Iteration
   - Policy Iteration
3. **Reinforcement Learning**
   - Model-Free RL using Monte Carlo Estimation
   - Temporal-Difference Estimation and SARSA (Model-Free)
   - Exploration Strategies
   - Q-Learning (Model-Free)
   - **SARSA vs. Q-Learning**

Off-Policy vs. On-Policy RL

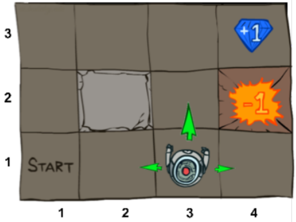
- General RL steps: repeat until converge:
  1. Use some exploration policy π' to run episodes/trails
  2. Use smaples to update the exploitation policy π
  3. Gradually mix π into π'

Off-Policy vs. On-Policy RL

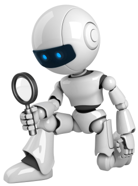
- General RL steps: repeat until converge:
  1. Use some exploration policy π' to run episodes/trails
  2. Use smaples to update the exploitation policy π
  3. Gradually mix π into π'
- ***Off-policy***: π updated toward a greedy policy ***independent with*** π'
  - Q-learning:
    $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta\,[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')) - Q^*(\boldsymbol{s},\boldsymbol{a})]$
- ***On-policy***: π updated to improve (and ***depends on***) π'
  - SARSA: $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\,[(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))) - Q_\pi(\boldsymbol{s},\boldsymbol{a})]$

Off-Policy vs. On-Policy RL

- General RL steps: repeat until converge:
  1. Use some exploration policy π' to run episodes/trails
  2. Use smaples to update the exploitation policy π
  3. Gradually mix π into π'
- ***Off-policy***: π updated toward a greedy policy ***independent with*** π'
  - Q-learning:
    $Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta\left[\left(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')\right) - Q^*(\boldsymbol{s},\boldsymbol{a})\right]$
- ***On-policy***: π updated to improve (and ***depends on***) π'
  - SARSA: $Q_\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow Q_\pi(\boldsymbol{s},\boldsymbol{a}) + \eta\left[\left(R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma Q_\pi(\boldsymbol{s}',\pi(\boldsymbol{s}'))\right) - Q_\pi(\boldsymbol{s},\boldsymbol{a})\right]$



π'



π

Practical Results

- SARSA has the capability to ***avoid the mistakes due to exploration***
  - E.g., the maze-with-cliff problem
  - Advantageous when $\varepsilon > 0$ with ε-greedy exploration strategy
  - Converges to optimal policy π^* (as Q-learning) when $\varepsilon \to 0$

Practical Results

- SARSA has the capability to ***avoid the mistakes due to exploration***
  - E.g., the maze-with-cliff problem
  - Advantageous when $\varepsilon > 0$ with ε-greedy exploration strategy
  - Converges to optimal policy π^* (as Q-learning) when $\varepsilon \to 0$
- But Q-learning has the capability to ***continue learning while changing the exploration policy***

Remarks

- Update rule for Q^* (or Q_π) takes terminal states differently:

$$Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \begin{cases} (1-\eta)Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') & \text{if } \boldsymbol{s}' \text{ is terminal} \\ (1-\eta)Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta \left[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')\right], & \text{otherwise} \end{cases}$$

  - Better convergence in practice

Remarks

- Update rule for Q^* (or Q_π) takes terminal states differently:

$$Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \begin{cases} (1-\eta)Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') & \text{if } \boldsymbol{s}' \text{ is terminal} \\ (1-\eta)Q^*(\boldsymbol{s},\boldsymbol{a}) + \eta\left[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') + \gamma \max_{\boldsymbol{a}'} Q^*(\boldsymbol{s}',\boldsymbol{a}')\right], & \text{otherwise} \end{cases}$$

  - Better convergence in practice
- Watch out your memory usage!
- Space complexity for SARSA/Q-learning: $O(|\mathbb{S}||\mathbb{A}|)$
  - Store $Q^*(\boldsymbol{s},\boldsymbol{a})$'s or $Q_\pi(\boldsymbol{s},\boldsymbol{a})$'s for all $(\boldsymbol{s},\boldsymbol{a})$ combinations

Remarks

- Update rule for Q^* (or Q_π) takes terminal states differently:

$$Q^*(\boldsymbol{s},\boldsymbol{a}) \leftarrow \begin{cases} (1-\eta)Q^*(\boldsymbol{s},\boldsymbol{a})+\eta R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') & \text{if } \boldsymbol{s}' \text{ is terminal} \\ (1-\eta)Q^*(\boldsymbol{s},\boldsymbol{a})+\eta\left[R(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}')+\gamma\max_{\boldsymbol{a}'}Q^*(\boldsymbol{s}',\boldsymbol{a}')\right], & \text{otherwise} \end{cases}$$

  - Better convergence in practice
- Watch out your memory usage!
- Space complexity for SARSA/Q-learning: $O(|\mathbb{S}||\mathbb{A}|)$
  - Store $Q^*(\boldsymbol{s},\boldsymbol{a})$'s or $Q_\pi(\boldsymbol{s},\boldsymbol{a})$'s for all $(\boldsymbol{s},\boldsymbol{a})$ combinations
- Why not train a (deep) regressor for $Q^*(\boldsymbol{s},\boldsymbol{a})$'s or $Q_\pi(\boldsymbol{s},\boldsymbol{a})$'s?
  - Space reduction due to the generalizability of the regressor