




# AI in Logistics

AUTUMN 2021

Christina Imdahl

IE & IS, OPAC

# Schedule of Second Part

Monday	Tuesday	Wednesday	Thursday	Friday
Oct 4	Oct 5	Oct 6	Oct 7	Oct 8
	Introduction to RL			Case Study Descriptives and Orientation
Oct 11	Oct 12	Oct 13	Oct 14	Oct 15
	Reinforcement Learning – Key Concept 			(Homework: Implement RL) <i>Momentum</i>
Oct 18	Oct 19	Oct 20	Oct 21	Oct 22
	Inventory Management - Heuristics	No Availability		
				Case Study Benchmark
Oct 25	Oct 26	Oct 27	Oct 28	Oct 29
	Wrap-up / Case Study			Case Study
Nov 1	Nov 2	Nov 3	Nov 4	Nov 5
	Submission Case			Case Presentation

# Agenda

I.	Recap	3
II.	Bellman equation	10
III.	Exploration and Exploitation	20
IV.	SARSA	25
V.	Q-Learning	35
VI.	On- and Off-Policy Learning	43

# Terminology MDP

## State Value Function of a policy

$$v_{\pi}(s) = \mathbb{E}_{\pi} \left( \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}(s_{t+k}, \pi(a|s_{t+k}), s'_{t+k}) \right)$$

Disc. reward of following policy  $\pi$

## Action-Value Function of a policy

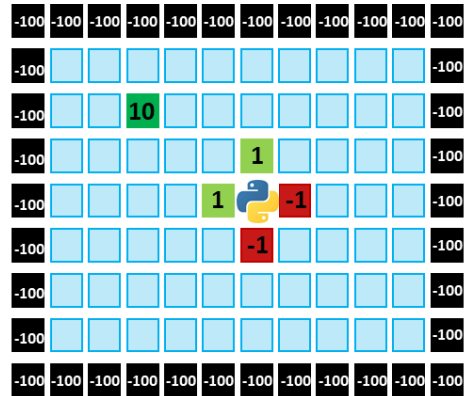
$$q_{\pi}(a, s) = \mathbb{E}_{\pi} (R_{t+1}(s_{t+k}, a, s'_{t+k})) + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1}(s_{t+k}, \pi(a|s_{t+k}), s'_{t+k}))$$

Immediate reward taking a

Disc. reward of following policy  $\pi$

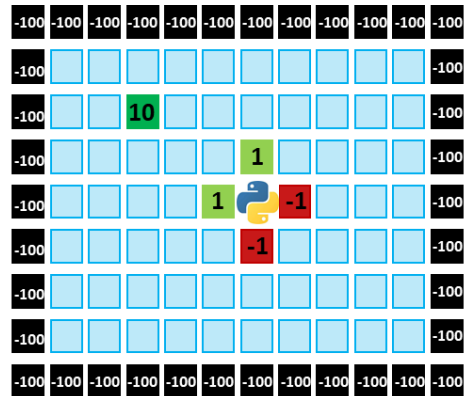
# Modeling influences learning

- Formulation of states  
influences how fast we learn
- Formulation of rewards  
influences what we learn  
(first)
- Snake game similar to the  
KIVA robot

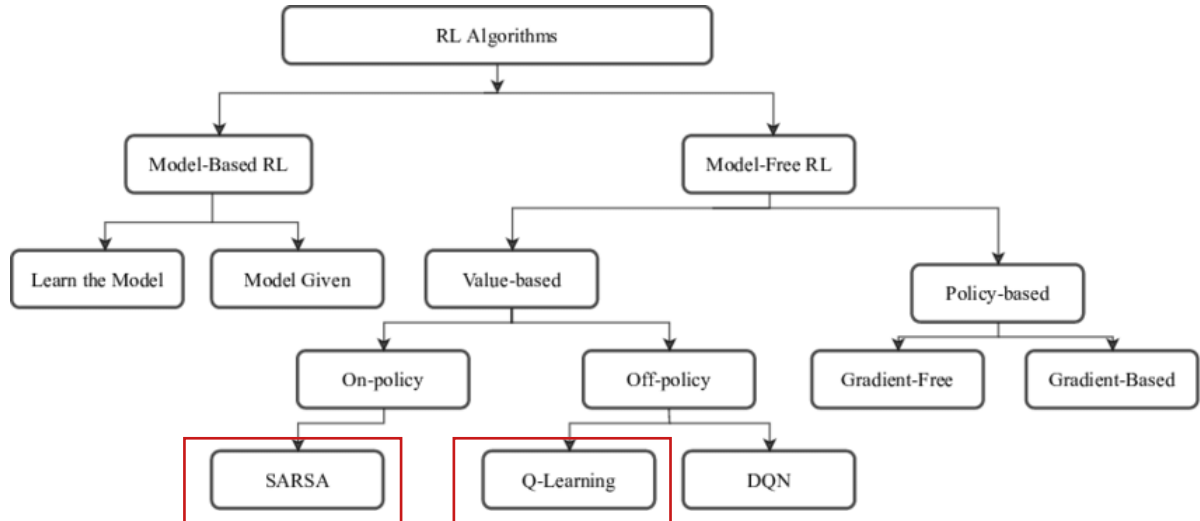


# What does the agent know?

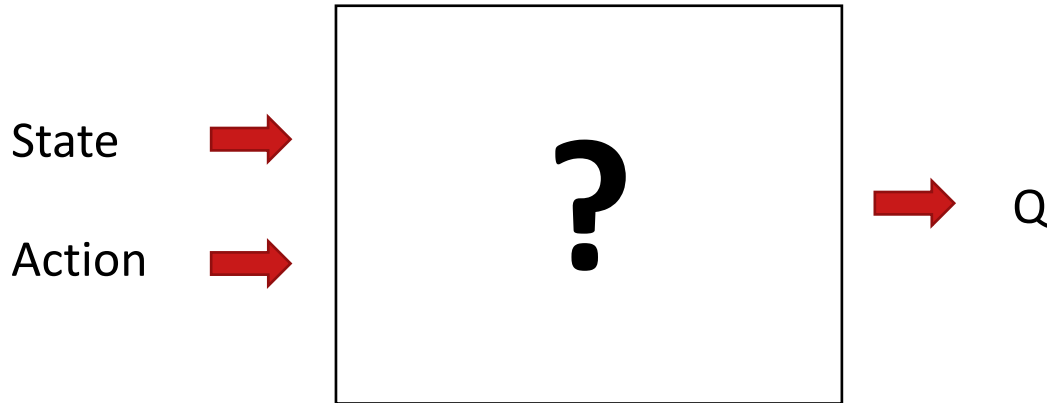
- What is your goal of learning?  
(Poacher vs. Kiva Example)
- What else should the agent be able to learn?



# Overview of ML Algorithms (incomplete)



# Today: How to estimate Q-function?



Why?  $\pi_*(s) = \operatorname{argmax}_a q_*(s, a)$



# Objectives of Today

- Understand the Bellman Equation (!)
- Be aware of the main techniques for exploitation and exploration
- Learn two temporal difference techniques for RL (SARSA, Q-Learning)
- Understand differences between on- and off-policy learning

# Agenda

I.	Recap	3
II.	Bellman equation	10
III.	Exploration and Exploitation	20
IV.	SARSA	25
V.	Q-Learning	35
VI.	On- and Off-Policy Learning	43

# Bellman Equation (1)

State Value Function of a policy

$$v_{\pi}(s) = \mathbb{E}_{\pi} \left( \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}(s_{t+k}, \pi(a|s_{t+k}), s'_{t+k}) \right)$$

---

Disc. reward of following policy  $\pi$

$$= \mathbb{E}_{\pi}(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s)$$

$$= \mathbb{E}_{\pi}(G_t | S_t = s)$$

# Bellman Equation (2)

$$\begin{aligned}v_{\pi}(s) &= \mathbb{E}_{\pi}(G_t | S_t = s) \\&= \mathbb{E}_{\pi}(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s) \\&= \mathbb{E}_{\pi}(R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s) \\&= \mathbb{E}_{\pi}(R_{t+1} + \gamma G_{t+1} | S_t = s) \\&= \mathbb{E}_{\pi}(\underbrace{R_{t+1}}_{\text{Immediate reward}} + \underbrace{\gamma v_{\pi}(S_{t+1})}_{\text{Disc. value of successor state}} | S_t = s)\end{aligned}$$

Immediate reward

Disc. value of successor state

# Bellman Equation (3)

## State Value Function

$$v_{\pi}(s) = \mathbb{E}(R_{t+1} + \gamma v(S_{t+1}) | S_t = s)$$

## Action Value Function

$$q_{\pi}(s, a) = \mathbb{E}(R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a)$$

# Notation

$\mathcal{P}$  is a state transition probability matrix, we note

$$\mathcal{P}_{ss'}^a = \mathbb{P}(S_{t+1} = s' | S_t = s, A_t = a)$$

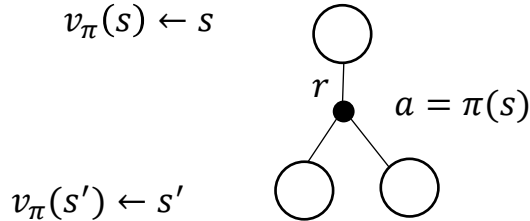
$\mathcal{R}$  is a reward function, we note

$$\mathcal{R}_s^a = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$$

# Bellman Equation (4)

## Deterministic Policy

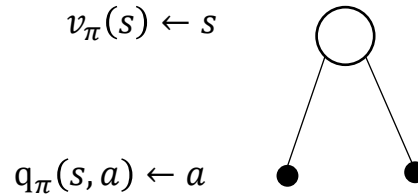
$$v_{\pi}(s) = \mathbb{E}(R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s)$$



$$v_{\pi}(s) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_{\pi}(s')$$

# Bellman Equation (5)

## Stochastic Policy

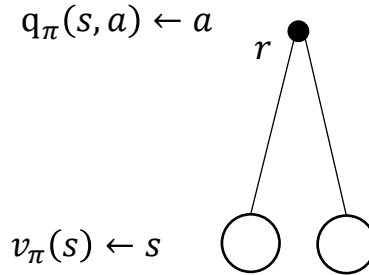


$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s) q_{\pi}(s, a)$$



# Bellman Equation (6)

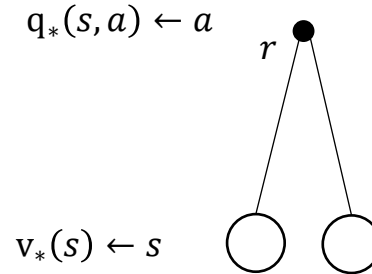
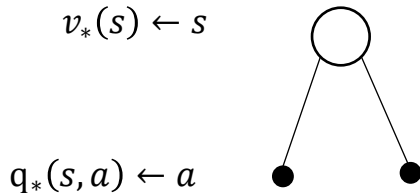
## Q-Function



$$q_{\pi}(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a v_{\pi}(s')$$

# Bellman Optimality Equation

## Value Function



$$v_*(s) = \max_a q_*(s, a)$$

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a v_*(s')$$

$$v_*(s, a) = \max_a (\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a v_*(s'))$$

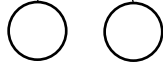
# Bellman Optimality Equation

## Q-Function

$$q_*(s, a) \leftarrow a$$

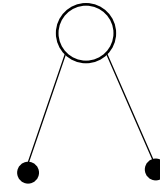
$r$

$$v_*(s') \leftarrow s'$$



$$v_*(s') \leftarrow s'$$

$$q_*(s', a') \leftarrow a'$$



$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a v_*(s')$$













$$v_*(s') = \max_{a'} q_*(s', a')$$

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} p_{ss'}^a \max_{a'} q_*(s', a')$$

# Agenda

I.	Recap	3
II.	Bellman equation	10
III.	Exploration and Exploitation	20
IV.	SARSA	25
V.	Q-Learning	35
VI.	On- and Off-Policy Learning	43

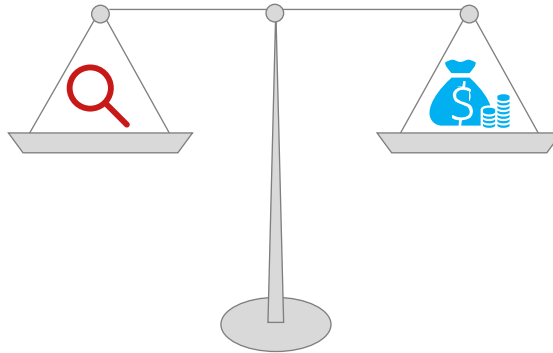
# Exploration and Exploitation

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	+5 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	+10 

# The Trade-Off

## 1 Exploration

- The agent's view of the world is incomplete.
- **Exploration** is the process of getting knowledge about things which we didn't know.
- The agent can gain by completing his knowledge of the world.
- The agent risks choosing a bad action.



## 2 Exploitation

- The agent has some idea of the rewards.
- **Exploitation** is the process of taking benefits from things which we know about.
- The agent takes the trajectory that is optimal given his knowledge.
- The agent may be stucked in suboptimal policies.

# $\epsilon$ -greedy policy

Under the  $\epsilon$ -greedy policy, the action that optimizes the Q-Function is chosen with probability  $1 - \epsilon$ . A random action is taken with probability  $\epsilon$ .

$$\pi(s) = \begin{cases} \operatorname{argmax}_a Q(s, a), & \text{with probability } 1 - \epsilon \\ \text{any } a \in A, & \text{with probability } \epsilon \end{cases}$$

In a decaying  $\epsilon$ -greedy policy,  $\epsilon$  decays over time.

# $\epsilon$ -greedy policy implementation

```
def epsilon_greedy_action(state: int, q_table: np.array, epsilon: float) -> int:
    """
    Select action based on the  $\epsilon$ -greedy policy
    Random action with prob.  $\epsilon$ , greedy action with prob.  $1-\epsilon$ 
    """

    # Random uniform sample from [0,1]
    sample = np.random.random()

    # Set to 'explore' if sample <=  $\epsilon$ 
    explore = True if sample <= epsilon else False

    if explore: # Explore
        # Select random action
        action = np.random.choice(4)
    else: # Exploit:
        # Select action with largest Q-value
        action = np.argmax(q_table[:, state])

    return action
```



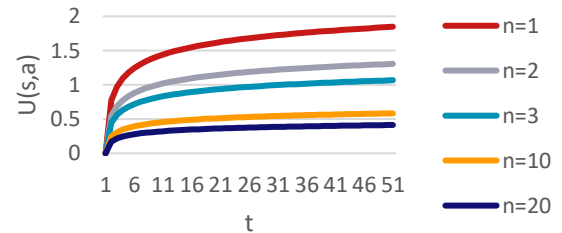
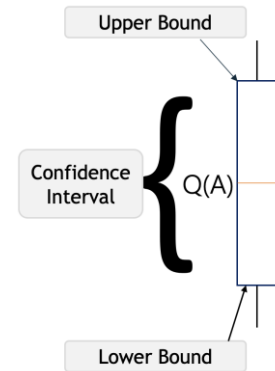
# Upper Confidence Bounds (UCB)

The Upper Confidence Bounds measures the potential of an action „to be rewarding“ by an upper confidence bound of the reward value  $\hat{U}(a)$ , s.t. the true value is below with bound  $Q(s, a) \leq \hat{Q}(s, a) + \hat{U}(s, a)$  with high probability.

$$\pi(s) = \operatorname{argmax}_a (\hat{Q}(s, a) + \hat{U}(s, a))$$

e.g.:













$$\pi(s) = \operatorname{argmax}_a \left( \hat{Q}(s, a) + c \sqrt{\frac{2 \log t}{N(s, a)}} \right)$$



# Agenda

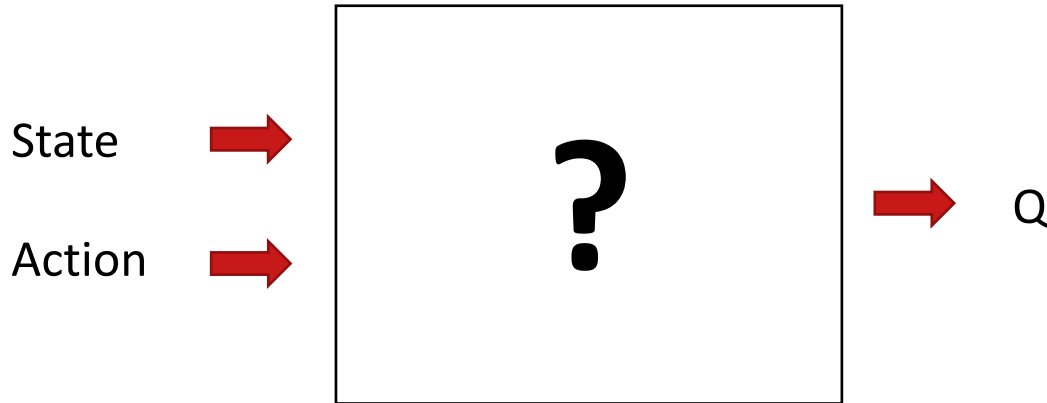
I.	Recap	3
II.	Bellman equation	10
III.	Exploration and Exploitation	20
<b>IV.</b>	<b>SARSA</b>	<b>25</b>
V.	Q-Learning	35
VI.	On- and Off-Policy Learning	43

# Walking the Cliff

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	-100 	+10 

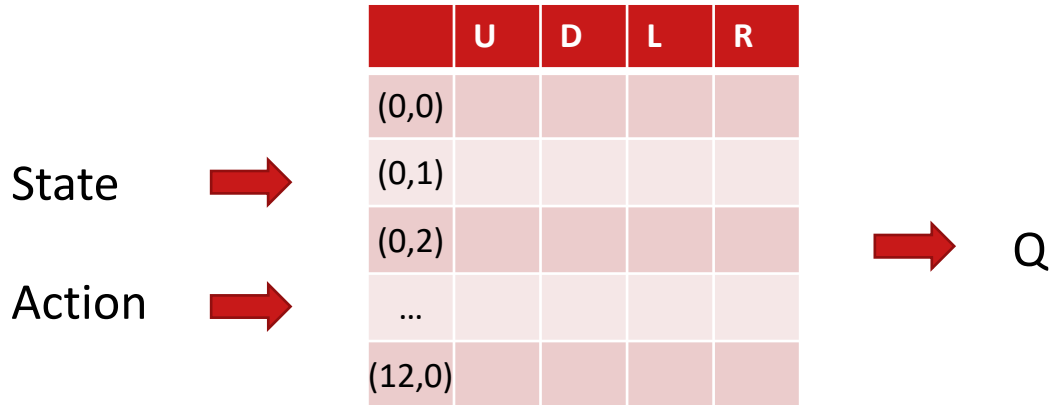
Cliff walking environment with rewards, origin and destination. Stepping into the cliff or on the goal tile ends the learning episode.

# Goal: Learn the Q-Function

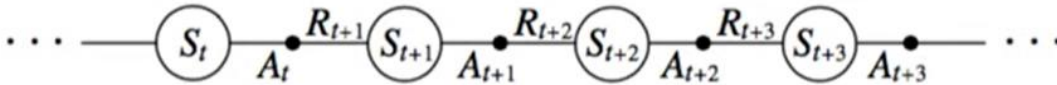


Why?  $\pi_*(s) = \operatorname{argmax}_a q_*(s, a)$

# Goal: Learn the Q-Function



# SARSA: State-Action-Reward-State-Action



*SARSA- Update Rule (on policy)*

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \underbrace{(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}))}_{\text{Experience: „Learned“ Reward}} - \underbrace{Q(S_t, A_t)}_{\text{„Expected“ Reward}}$$

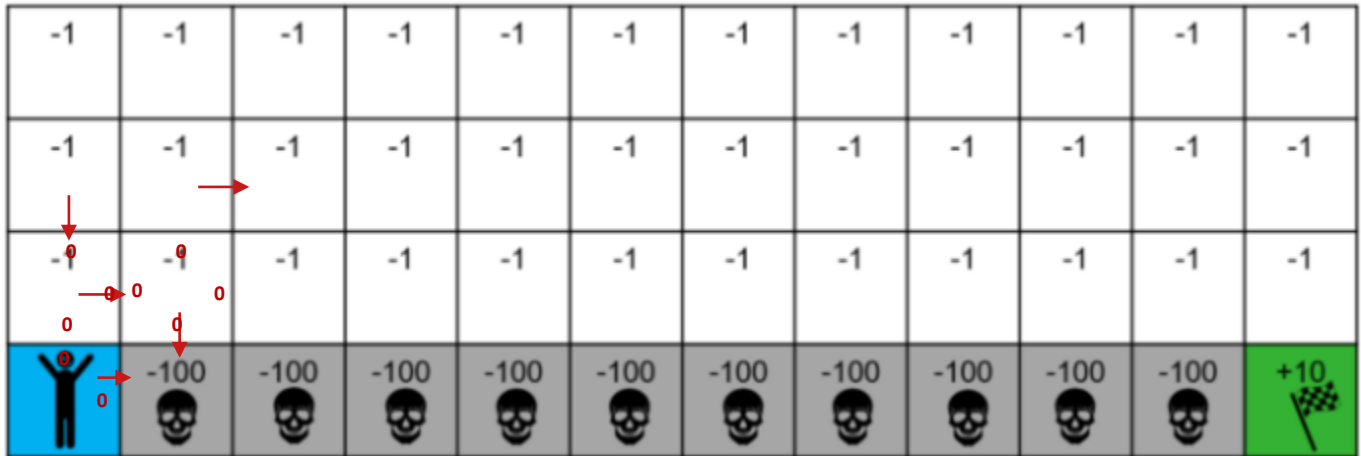
$A_t = \pi(S_t)$                        $A_{t+1} = \pi(S_{t+1})$

# SARSA Iterations

```
Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):  
  Initialize  $S$   
  Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
  Repeat (for each step of episode):  
    Take action  $A$ , observe  $R, S'$   
    Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
     $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$   
     $S \leftarrow S'; A \leftarrow A';$   
  until  $S$  is terminal
```

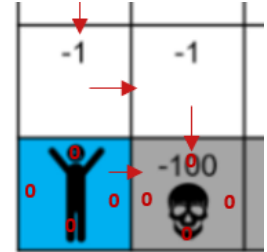
# Walking the Cliff

Initialize  $Q(s,a)$ ,  $\alpha = 0.1$ ,  $\gamma = 0.9$ , get policy  $\pi$  from  $Q$





# Example (1)



## 1. Iteration SARSA

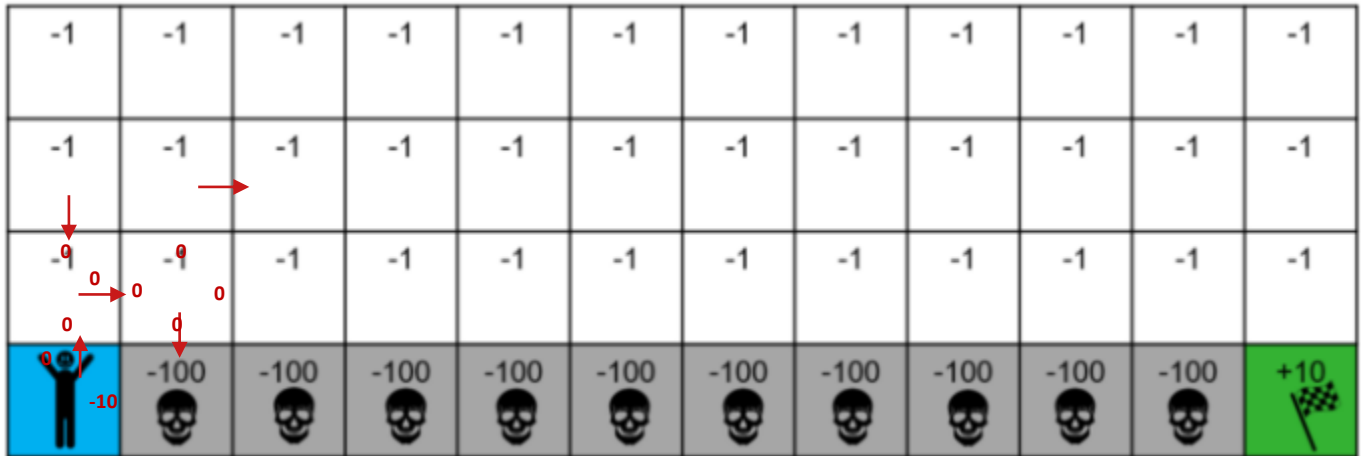
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

$$Q(S_t, A_t) \leftarrow 0 + \alpha(-100 + \gamma \cdot 0 - 0)$$

$$Q(S_t, A_t) \leftarrow -10$$

# Walking the Cliff

Initialize  $Q(s,a)$ ,  $\alpha = 0.1$ ,  $\gamma = 0.9$ , get policy  $\pi$  from  $Q$



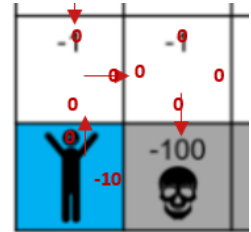
# Example (2)

## 2. Iteration SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

$$Q(S_t, A_t) \leftarrow 0 + \alpha(-1 + \gamma \cdot 0 - 0)$$

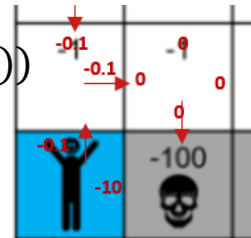
$$Q(S_t, A_t) \leftarrow -0.1$$



## xx. Iteration SARSA

$$Q(S_t, A_t) \leftarrow -0.1 + \alpha(-1 + \gamma \cdot (-0.1) - (-0.1))$$

$$Q(S_t, A_t) \leftarrow -0.199$$



# Agenda

I.	Recap	3
II.	Bellman equation	10
III.	Exploration and Exploitation	20
IV.	SARSA	25
V.	Q-Learning	35
VI.	On- and Off-Policy Learning	43

# Q-Learning

*Bellman Optimality Equation*

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} p_{ss'}^a \max_{a'} q_*(s', a')$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R_{t+1} + \gamma \max_{a' \in A(s_t)} Q(s_{t+1}, a') - Q(s_t, a_t))$$

Experience: „Learned“ Reward„Expected“ Reward

$A_{t+1} = a' \neq \pi(S_{t+1})$   
(can be equal, but not necessarily)

# Q-Learning Iterations

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$   
Repeat (for each episode):  
    Initialize  $S$   
    Repeat (for each step of episode):  
        Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)  
        Take action  $A$ , observe  $R, S'$   
         $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$   
         $S \leftarrow S'$ ;  
    until  $S$  is terminal

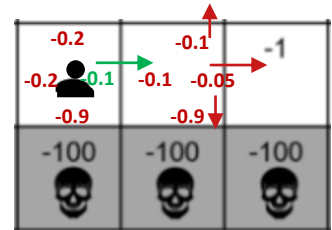
# Walking the Cliff



# SARSA and Q-Learning

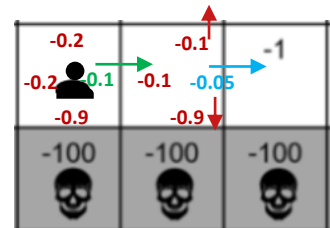
## SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$$

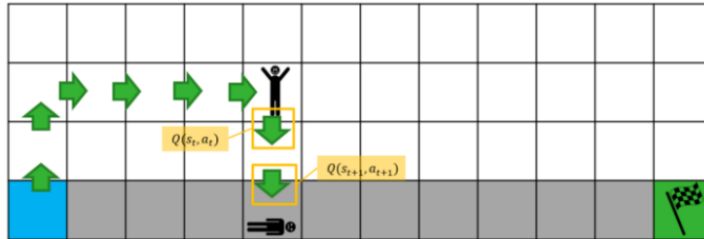


## Q-Learning

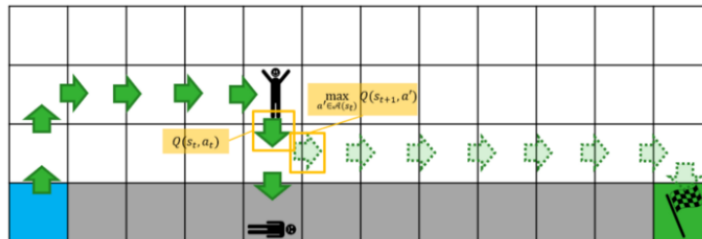
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a' \in A(S_t)} Q(S_{t+1}, a') - Q(S_t, A_t))$$







Temporal difference update with SARSA (on-policy). The large negative reward obtained when falling down the cliff is used to update the  $Q(s,a)$  values for all preceding states and actions, using the actual trajectory.

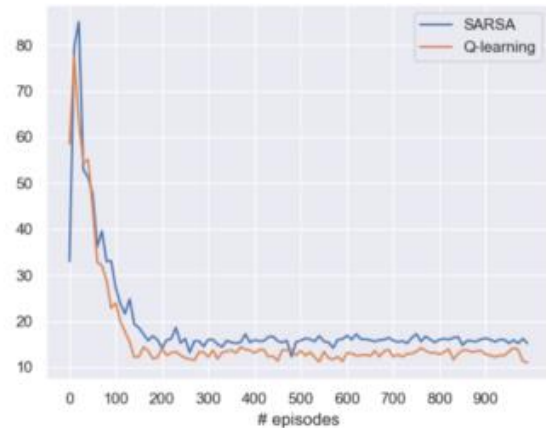
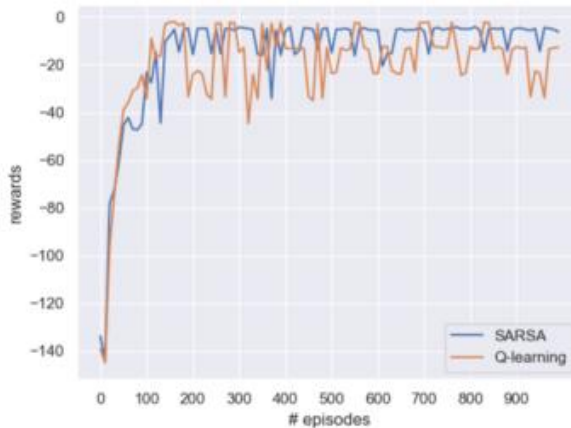


Temporal difference update with Q-learning (off-policy). The large negative reward obtained when falling down the cliff does not influence earlier  $Q$ -values, as the optimal trajectory is used for the updates.

# Comparison of the target policy



# Comparison on the behavioral policy



# Agenda

I.	Recap	3
II.	Bellman equation	10
III.	Exploration and Exploitation	20
IV.	SARSA	25
V.	Q-Learning	35
<b>VI.</b>	<b>On- and Off-Policy Learning</b>	<b>43</b>

# On-Policy Learning and Off-Policy Learning

**On-Policy** learning algorithms are the algorithms that evaluate and improve the same policy which is being used to select actions. That means we will try to evaluate and improve the same policy that the agent is already using for action selection.

**Off-Policy** learning algorithms evaluate and improve a policy that is different from Policy that is used for action selection.

# „Soft“ Comparison

USE....

## On-Policy (e.g. Sarsa), if

- Learning is computational cheap
- Rewards while training matters
- You can invest time for fine-tuning exploration

## Off-Policy (e.g. Q-Learning), if

- Learning is computational expensive
- Only performance of target policy is important
- You cannot invest time for fine-tuning exploration

In all cases, the relative performance matters on the task and the hyperparameters (e.g. small or decaying  $\epsilon \rightarrow$  less differences in target policy)

# Acknowledgements

Bellman Eq.: <https://towardsdatascience.com/reinforcement-learning-markov-decision-process-part-2-96837c936ec3>

Sutton, Barto. Reinforcement Learning: An Introduction  
<https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>

TOSSINGBOT: <https://www.youtube.com/watch?v=-O-E1nFm6-A>

Cliff-Walking

[https://github.com/woutervanheeswijk/cliff\\_walking\\_public/](https://github.com/woutervanheeswijk/cliff_walking_public)  
<https://towardsdatascience.com/walking-off-the-cliff-with-off-policy-reinforcement-learning-7fdbcdfe31ff>