Basic tabular methods: Temporal Difference SARSA and Q-learning

Reinforcement Learning

Edgar J. Andrade-Lotero, Ph.D.

Last revision: January 2025

Outline

1. Introduction
2. Learning Rule
3. Explore vs. Exploit
4. Long-term Reward
5. Temporal Difference Methods

Outline

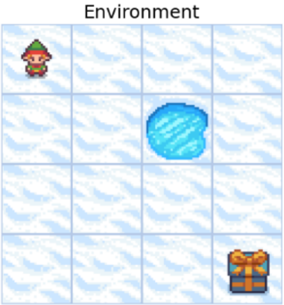
1. Introduction
2. Learning Rule
3. Explore vs. Exploit
4. Long-term Reward
5. Temporal Difference Methods

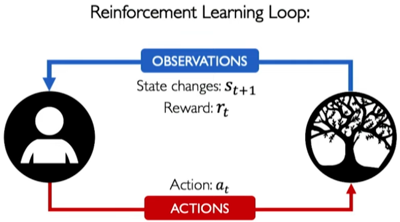
Solving an Environment

Problem: How can we solve the environment if we do not know the underlying model?

Solving an Environment

Problem: How can we solve the environment if we do not know the underlying model?





☞ The agent must learn to act based on its experience over an extended period of time.

Components of Reinforcement Learning

- Learning rule
- Explore vs. Exploit
- Long-term reward

Components of Reinforcement Learning

- Learning rule
- Explore vs. Exploit
- Long-term reward

Learning as a correction of the utility estimator's error based on experience.

Components of Reinforcement Learning

- Learning rule
- Explore vs. Exploit
- Long-term reward

Balancing between exploiting current information and exploring to gather better information.

Components of Reinforcement Learning

- Learning rule
- Explore vs. Exploit
- Long-term reward

The agent must learn to maximize the total reward of the episode, not just the reward for the current action.

Outline

1 Introduction

2 Learning Rule

3 Explore vs. Exploit

4 Long-term Reward

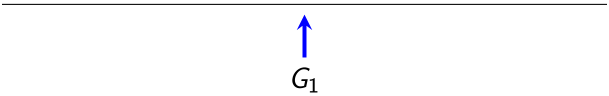
5 Temporal Difference Methods

Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.

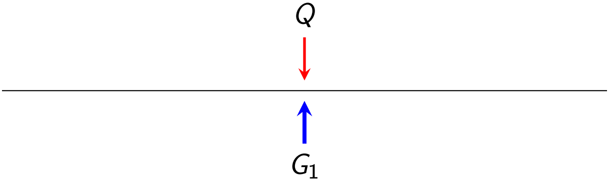
Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



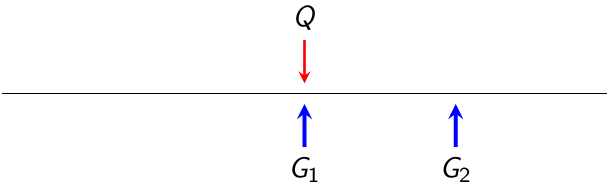
Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



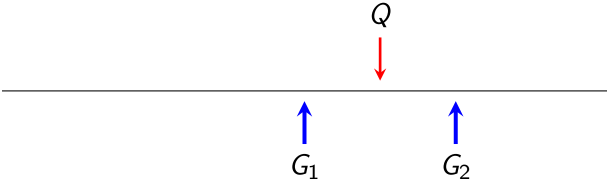
Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



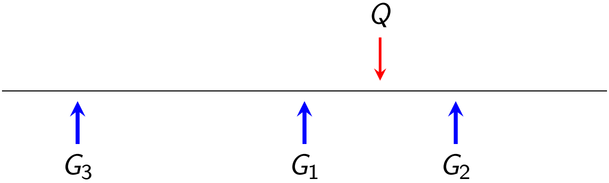
Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



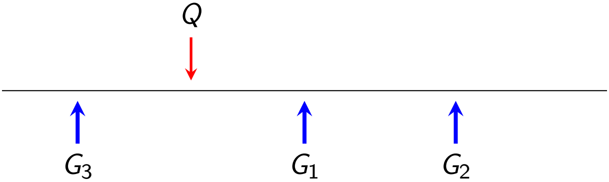
Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



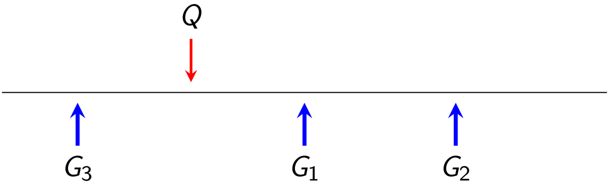
Estimation

Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



Estimation

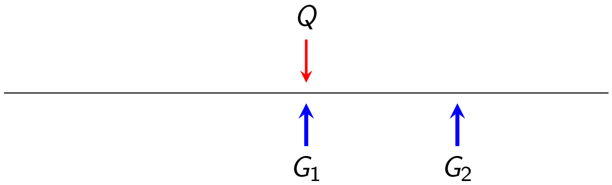
Each round, we want to estimate a natural number q based on imperfect observations $G_1, \ldots, G_T$, where G_i is the observation obtained in the i-th round. Suppose Q is our current estimate.



What formula can we use to measure the direction and magnitude of the change in the estimate?

Error Correction in Estimation

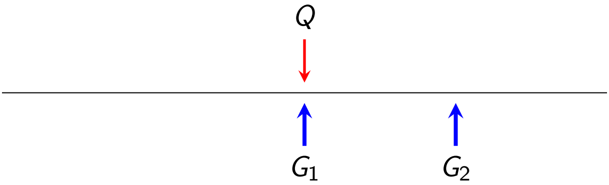
Each round, we observe a G_i and note the variation with respect to our current estimate q.



The estimation error is $\delta = G_2 - Q$.

Error Correction in Estimation

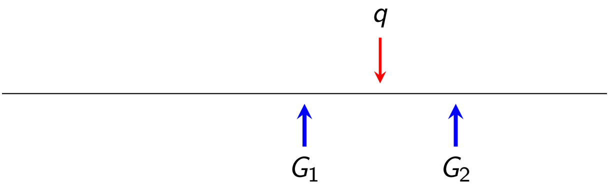
Each round, we observe a G_i and note the variation with respect to our current estimate q.



G_2 is not the actual data q, so we must weight the error by a learning rate α.

Error Correction in Estimation

Each round, we observe a G_i and note the variation with respect to our current estimate q.

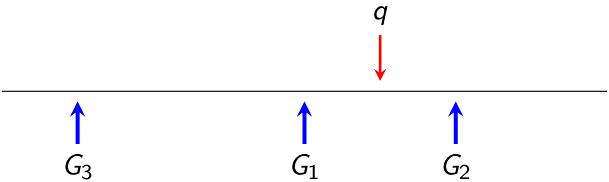


We update Q using the rule

$$Q \leftarrow Q + \alpha\delta$$

Error Correction in Estimation

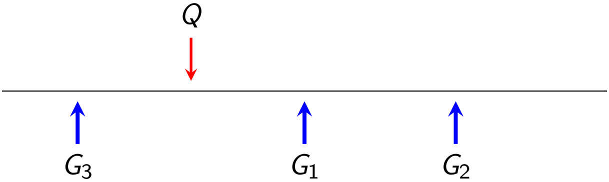
Each round, we observe a G_i and note the variation with respect to our current estimate q.



The estimation error is $\delta = G_3 - Q$.

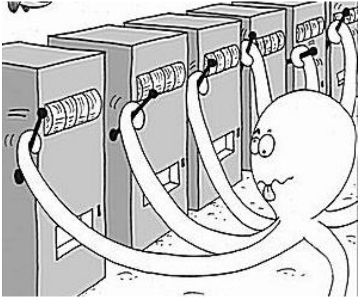
Error Correction in Estimation

Each round, we observe a G_i and note the variation with respect to our current estimate q.



$$Q \leftarrow Q + \alpha\big(G_3 - Q\big)$$

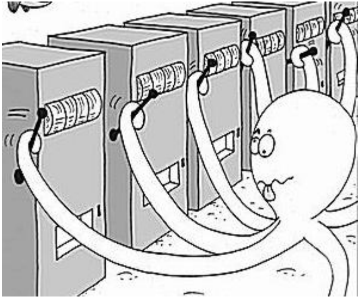
Multi-armed Bandits



Stochastic scheduling problem

- The agent pulls the lever of one of the machines and gets a reward of 0 or 1.

Multi-armed Bandits



Stochastic scheduling problem

- The success probability of each machine is different and initially unknown to the agent.

Multi-armed Bandit Problem

$r(1)$

$r(2)$

Problem: Which machine a offers success with the highest probability $Q(a)$?

Solution Plan

☞ Estimate the success probability $q(a)$ of each machine a.

Solution Plan

☞ Estimate the success probability $q(a)$ of each machine a.

☞ Each round i, select a machine a and observe the reward r_i.

Solution Plan

☞ Estimate the success probability $q(a)$ of each machine a.

☞ Each round i, select a machine a and observe the reward r_i.

☞ Maintain the estimators $Q_i(a)$ and update using the rule:

$$Q_i(a) = \begin{cases} Q_{i-1}(a) + \alpha\big(r_i - Q_{i-1}(a)\big), & \text{if } a \text{ is selected} \\ Q_{i-1}(a), & \text{otherwise} \end{cases}$$

Example

Suppose $\alpha = \frac{1}{2}$.

Machine 1

- *Initial state:* $Q_0(1) = 0$

Machine 2

- *Initial state:* $Q_0(2) = 0$

Both estimators start at 0.

Example

Suppose $\alpha = \frac{1}{2}$.

Machine 1

- *Initial state:* $Q_0(1) = 0$
- *Round 1:*
  $Q_1(1) = 0 + \alpha(1 - 0) = \frac{1}{2}$

Machine 2

- *Initial state:* $Q_0(2) = 0$
- *Round 1:* $Q_1(2) = 0$

We select 1 and achieve success ($r_1 = 1$).

Example

Suppose $\alpha = \frac{1}{2}$.

Machine 1

- *Initial state:* $Q_0(1) = 0$
- *Round 1:* $Q_1(1) = 0 + \alpha(1 - 0) = \frac{1}{2}$
- *Round 2:* $Q_2(1) = \frac{1}{2} + \alpha(0 - \frac{1}{2}) = \frac{1}{4}$

Machine 2

- *Initial state:* $Q_0(2) = 0$
- *Round 1:* $Q_1(2) = 0$
- *Round 2:* $Q_2(2) = 0$

We select 1 and do not achieve success ($r_2 = 0$).

Example

Suppose $\alpha = \frac{1}{2}$.

Machine 1

- *Initial state:* $Q_0(1) = 0$
- *Round 1:* $Q_1(1) = 0 + \alpha(1 - 0) = \frac{1}{2}$
- *Round 2:* $Q_2(1) = \frac{1}{2} + \alpha(0 - \frac{1}{2}) = \frac{1}{4}$
- *Round 3:* $Q_3(1) = \frac{1}{4}$

Machine 2

- *Initial state:* $Q_0(2) = 0$
- *Round 1:* $Q_1(2) = 0$
- *Round 2:* $Q_2(2) = 0$
- *Round 3:* $Q_3(2) = 0 + \alpha(1 - 0) = \frac{1}{2}$

We select 2 and achieve success ($r_3 = 1$).

Outline

1. Introduction
2. Learning Rule
3. Explore vs. Exploit
4. Long-term Reward
5. Temporal Difference Methods

Explore vs. Exploit (1/3)

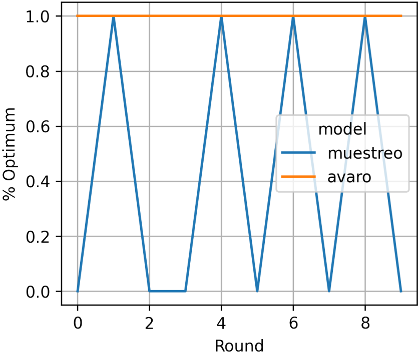
Two extreme approaches:

- **Explore**: Sample both arms.
- **Exploit**: Select the arm that has provided the best rewards so far (greedy strategy).

Explore vs. Exploit (1/3)

Two extreme approaches:

- **Explore**: Sample both arms.
- **Exploit**: Select the arm that has provided the best rewards so far (greedy strategy).

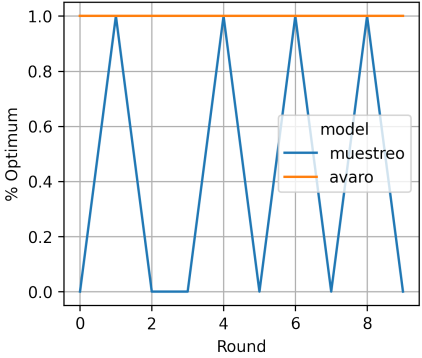


- The sampling strategy exhibits random behavior.

Explore vs. Exploit (1/3)

Two extreme approaches:

- **Explore**: Sample both arms.
- **Exploit**: Select the arm that has provided the best rewards so far (greedy strategy).

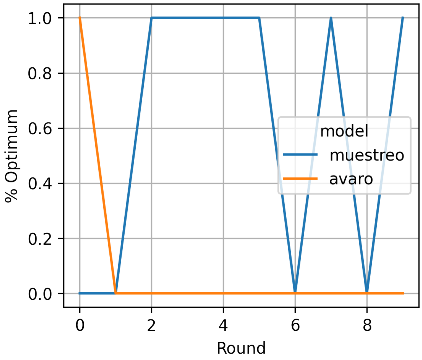


- The sampling strategy exhibits random behavior.
- The greedy strategy tried the optimal arm and succeeded.

Explore vs. Exploit (2/3)

Two extreme approaches:

- **Explore**: Sample both arms.
- **Exploit**: Select the arm that has provided the best rewards so far (greedy strategy).

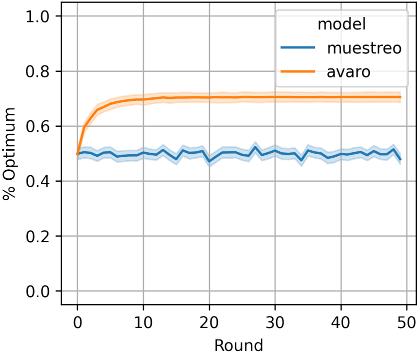


- The greedy strategy tried the optimal arm without success.
- Then, the greedy strategy succeeded by trying the arm that is NOT optimal.

Explore vs. Exploit (3/3)

Two extreme approaches:

- **Explore**: Sample both arms.
- **Exploit**: Select the arm that has provided the best rewards so far (greedy strategy).



Average over 50 experiments of 50 trials each.

Possible Solutions

There are various ways to address the dilemma between explore and exploit:

- Optimistic initial greedy
- ϵ-greedy
- ϵ-greedy with annealing
- Upper Confidence Bound
- Softmax
- Etc.

☞ Here, we will only discuss the ϵ-greedy strategy.

ϵ-greedy (1/2)

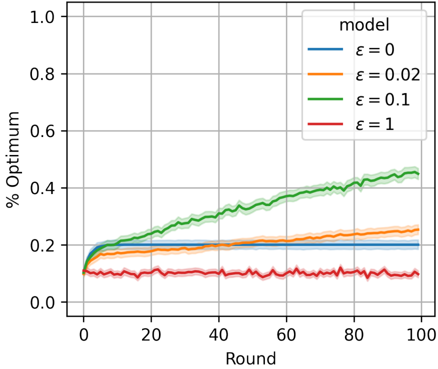
Balance between exploit (with probability $1 - \epsilon$) and explore (with probability ϵ).

Algorithm 1: ϵ-greedy bandit algorithm

Data: una probabilidad de exploración ϵ (donde $0 \leq \epsilon \leq 1$)
Result: índice del brazo seleccionado
$Q(a) \leftarrow 0$ para cada brazo a;
while *True* **do**
    if *probabilidad* $1 - \epsilon$ **then**
        $a \leftarrow \arg\max Q(a)$;
    else
        $a \leftarrow a$ aleatoria;
    end
    Presentar la acción a al entorno y obtener la recompensa r;
    $Q(a) \leftarrow Q(a) + \alpha\left[r - Q(a)\right]$
end

ϵ-greedy (2/2)

Results:



Protocol:

10 arms. 50 experiments of 50 trials.

Outline

1. Introduction
2. Learning Rule
3. Explore vs. Exploit
4. Long-term Reward
5. Temporal Difference Methods

Definitions

- **Utility**:

$$G = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots = \sum_{t=0}^{\infty} \gamma^k r_{t+1}$$

Definitions

- **Utility**:
$$G = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots = \sum_{t=0}^{\infty} \gamma^k r_{t+1}$$
- **Policy**: A function π that for each state s returns a probability distribution over possible actions, such that $\pi(a|s)$ is the probability of taking action a in state s.

Definitions

- **Utility**:

$$G = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots = \sum_{t=0}^{\infty} \gamma^k r_{t+1}$$

- **Policy**: A function π that for each state s returns a probability distribution over possible actions, such that $\pi(a|s)$ is the probability of taking action a in state s.
- **State Value**: The expected utility $v_\pi(s)$ of following policy π from state s: $v_\pi(s) = \mathbb{E}[G|s]$.

Definitions

- **Utility**:

$$G = r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots = \sum_{t=0}^{\infty} \gamma^k r_{t+1}$$

- **Policy**: A function π that for each state s returns a probability distribution over possible actions, such that $\pi(a|s)$ is the probability of taking action a in state s.
- **State Value**: The expected utility $v_\pi(s)$ of following policy π from state s: $v_\pi(s) = \mathbb{E}[G|s]$.
- **Action Value**: The expected utility of performing an action a in state s and then following π: $q_\pi(s, a) = \mathbb{E}[G|s, a]$

Transition Stochasticity

After the agent executes action a in state s, it transitions to state s_i with probability $p(s_i|s,a)$.

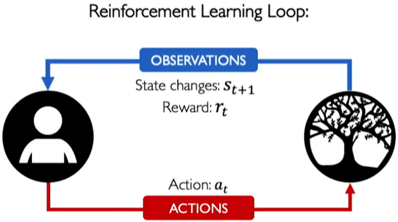
$$\{p(s_1|s,a); p(s_2|s,a); \ldots; p(s_n|s,a)\}$$

Markov Property

☞ Path Independence

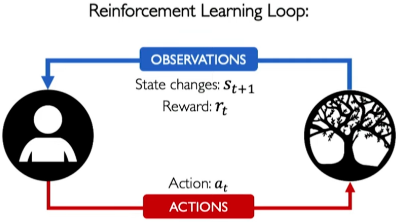
$$p(s_{t+1}|s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = p(s_{t+1}|s_t, a_t)$$

Components of MDPs



- Set of states
- Subset of terminal states
- Set of actions
- Transitions $p(s'|s, a)$
- Rewards $r(s, a, s')$

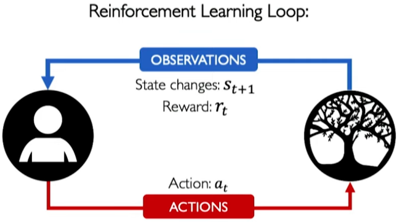
Components of MDPs



- Set of states
- Subset of terminal states
- Set of actions
- Transitions $p(s'|s,a)$
- Rewards $r(s,a,s')$

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \left(p(s'|s,a) \left[r(s,a,s') + \gamma v_\pi(s') \right] \right)$$

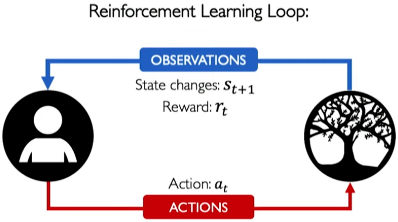
Components of MDPs



- Set of states
- Subset of terminal states
- Set of actions
- ~~Transitions $p(s'|s, a)$~~
- ~~Rewards $r(s, a, s')$~~

☞ Let us assume that we do not know the MDP model.

Components of MDPs



- Set of states
- Subset of terminal states
- Set of actions
- ~~Transitions $p(s'|s, a)$~~
- ~~Rewards $r(s, a, s')$~~

☞ We aim to estimate v_* and q_* directly.

Outline

1 Introduction

2 Learning Rule

3 Explore vs. Exploit

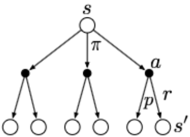
4 Long-term Reward

5 Temporal Difference Methods

Using the Bellman Equation

Dynamic Programming:

$$V_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \Big(p(s'|s,a)\Big[r + \gamma V_k(s')\Big]\Big)$$



Backup diagram for v_π

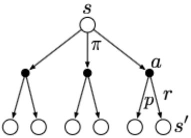
Temporal Difference:

The backup diagram for TD(0)

Using the Bellman Equation

Dynamic Programming:

$$V_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \Big(p(s'|s,a) \Big[r + \gamma V_k(s') \Big] \Big)$$



Backup diagram for v_π

Temporal Difference:

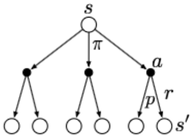
$$V_{k+1}(s) \leftarrow V_k(s) + \alpha \left(G - V_k(s) \right)$$


The backup diagram for TD(0)

Using the Bellman Equation

Dynamic Programming:

$$V_{k+1}(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} \left(p(s'|s,a) \left[r + \gamma V_k(s') \right] \right)$$



Backup diagram for v_π

Temporal Difference:

$$V_{k+1}(s) \leftarrow V_k(s) + \alpha \left(r + \gamma V_k(s') - V_k(s) \right)$$

The backup diagram for TD(0)

Learning Rule

$$V(s) \leftarrow \overbrace{V(s)}^{\text{previous estimate}} + \underbrace{\alpha}_{\text{step size}} \Big(\overbrace{r_1 + \gamma \underbrace{V(s_1)}_{\textit{bootstrap}}}^{\text{new data}} - \underbrace{V(s)}_{\text{previous estimate}} \Big)$$

Learning a Policy (SARSA)

Suppose a policy π.

Learning a Policy (SARSA)

Suppose a policy π.

- Rule to update state-action values:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$$

  Where s' is the state reached after performing a in s,
  and $a' \leftarrow$ action by sampling $\pi(s')$.

Learning a Policy (SARSA)

Suppose a policy π.

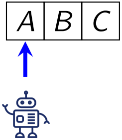
- Rule to update state-action values:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$$

  Where s' is the state reached after performing a in s,
  and $a' \leftarrow$ action by sampling $\pi(s')$.
- Improve $\pi(s)$ with ϵ-greedy using Q for all s.

The Q Table

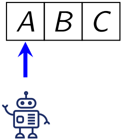
Consider the ABC environment presented two classes ago.



	Left	Right
A	0	0
B	0	0
C	0	0

The Q Table

Consider the ABC environment presented two classes ago.

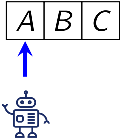


	Left	Right
A	0	0
B	0	0
C	0	0

Randomly, the agent selects the action Left.

The Q Table

Consider the ABC environment presented two classes ago.



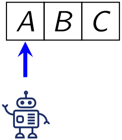
	Left	Right
A	0	0
B	0	0
C	0	0

The agent remains in state A.

$$s = A$$
$$a = \text{Left}$$
$$s' = A$$
$$a' \leftarrow \text{random action}$$

The Q Table

Consider the ABC environment presented two classes ago.



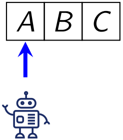
	Left	Right
A	0	0
B	0	0
C	0	0

The agent remains in state A.

$$s = A$$
$$a = \text{Left}$$
$$s' = A$$
$$a' \leftarrow \text{Right}$$

The Q Table

Consider the ABC environment presented two classes ago.

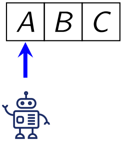


	Left	Right
A	0	0
B	0	0
C	0	0

$$q(A, \text{Left}){+} \;=\; \alpha\Big(-1 \;+\; \gamma q(A, \text{Right}) \;-\; q(A, \text{Left})\Big)$$

The Q Table

Consider the ABC environment presented two classes ago.



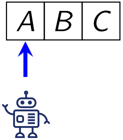
	Left	Right
A	0	0
B	0	0
C	0	0

$$q(A, \text{Left}) += 0.1\Big(-1 + 0.8 \times 0 - 0\Big)$$

Suppose $\alpha = 0.1$

The Q Table

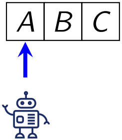
Consider the ABC environment presented two classes ago.



	Left	Right
A	-0.1	0
B	0	0
C	0	0

The Q Table

Consider the ABC environment presented two classes ago.

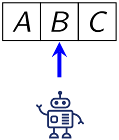


	Left	Right
A	-0.1	0
B	0	0
C	0	0

The agent selects the action with the highest q value, namely, Right (this occurs with probability $1 - \epsilon$).

The Q Table

Consider the ABC environment presented two classes ago.



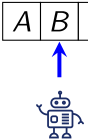
	Left	Right
A	-0.1	0
B	0	0
C	0	0

$$s = A$$
$$a = \text{Right}$$
$$s' = B$$
$$a' \leftarrow \text{Right}$$

Suppose the agent reaches B
(this occurs with probability 0.9).

The Q Table

Consider the ABC environment presented two classes ago.

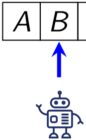


	Left	Right
A	-0.1	0
B	0	0
C	0	0

$$q(A, \text{Right}) += \alpha\Big(-1 + \gamma q(B, \text{Right}) - q(A, \text{Right})\Big)$$

The Q Table

Consider the ABC environment presented two classes ago.

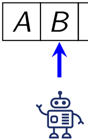


	Left	Right
A	-0.1	0
B	0	0
C	0	0

$$q(A, \text{Right}) += 0.1\Big(-1 + 0.8 \times 0 - 0\Big)$$

The Q Table

Consider the ABC environment presented two classes ago.



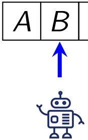
	Left	Right
A	-0.1	-0.1
B	0	0
C	0	0

Randomly, the agent selects the action Right.

$$s = B$$
$$a = \text{Right}$$
$$s' = C$$
$$a' \leftarrow \text{Right}$$

The Q Table

Consider the ABC environment presented two classes ago.

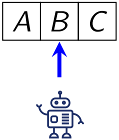


	Left	Right
A	-0.1	-0.1
B	0	0
C	0	0

The agent reaches C.

The Q Table

Consider the ABC environment presented two classes ago.

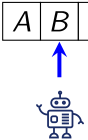


	Left	Right
A	-0.1	-0.1
B	0	0
C	0	0

$$q(B, \text{Right}) += \alpha\Big(10 + \gamma q(C, \text{Right}) - q(B, \text{Right})\Big)$$

The Q Table

Consider the ABC environment presented two classes ago.

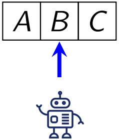


	Left	Right
A	-0.1	-0.1
B	0	0
C	0	0

$$q(B, \text{Right}) += 0.1\Big(10 + 0.8 \times 0 - 0\Big)$$

The Q Table

Consider the ABC environment presented two classes ago.



	Left	Right
A	-0.1	-0.1
B	0	1
C	0	0

SARSA Pseudocode

Algorithm 3: SARSA agent (update rule)

Data: Una acción a, un estado s' y una recompensa r

$Q(s, a) \leftarrow \text{self}.Q(s, a)$ (action-value para cada (s, a));

$\pi \leftarrow \text{self}.\pi$ (política ϵ-greedy sobre Q);

$s \leftarrow$ self.s (estado anterior);

$a' \leftarrow$ acción dada por π en s';

$\text{self}.Q(s, a) \leftarrow Q(s, a) + \alpha\Big(r + \gamma Q(s', a') - Q(s, a)\Big)$;

$\text{self}.\pi \leftarrow$ mejorar $\pi(s)$ con ϵ-greedy sobre Q;

$\text{self}.s \leftarrow s'$;

SARSA vs. Q-learning (1/2)

SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$$

Improve $\pi(s)$ with ϵ-greedy and Q

Q-learning:

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

Improve $\pi(s)$ with ϵ-greedy and Q

SARSA vs. Q-learning (2/2)

Algorithm 3: SARSA agent (update rule)

Data: Una acción a, un estado s' y una recompensa r
$Q(s,a) \leftarrow$ self.$Q(s,a)$ (action-value para cada (s,a));
$\pi \leftarrow$ self.π (política ϵ-greedy sobre Q);
$s \leftarrow$ self.s (estado anterior);
$a' \leftarrow$ acción dada por π en s';
self.$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$;
self.$\pi \leftarrow$ mejorar $\pi(s)$ con ϵ-greedy sobre Q;
self.$s \leftarrow s'$;

Algorithm 5: Q-learning agent

Data: Una acción a, un estado s' y una recompensa r
$Q(s,a) \leftarrow$ self.$Q(s,a)$ (action-value para cada (s,a));
$\pi \leftarrow$ self.π (política ϵ-greedy sobre Q);
$s \leftarrow$ self.s (estado anterior);
self.$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \operatorname{máx}_{a'} Q(s',a') - Q(s,a)\Big)$;
self.$\pi \leftarrow$ actualizar $\pi(s)$ con ϵ-greedy sobre Q;
self.$s \leftarrow s'$;

The Cliff Problem

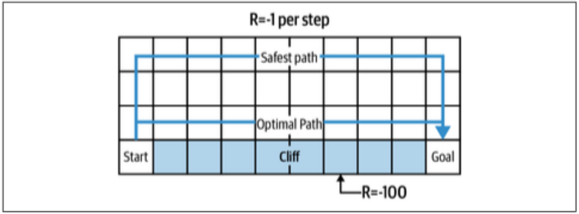


Figure 3-1. A depiction of the grid environment with a cliff along one side.[1]

Optimal Policy — SARSA vs. Q-learning

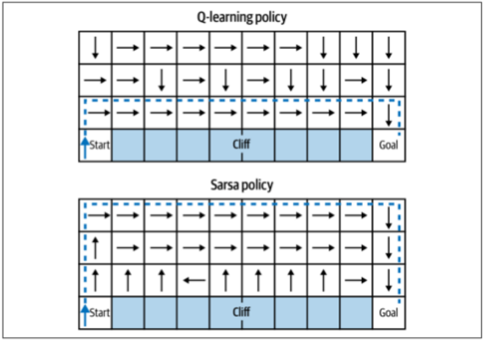


Figure 3-3. The policies derived by Q-learning and SARSA agents. Q-learning tends to prefer the optimal route. SARSA prefers the safe route.

Q Table

	Left	Right	Up	Down
20	-11	-10	-11	-100
21	-10	-9	-10	-100

What is the updated value of $q(20, \text{Right})$

if $a' = \text{Down}$, using:

- SARSA?
- Q-learning?

Utility — SARSA vs. Q-learning

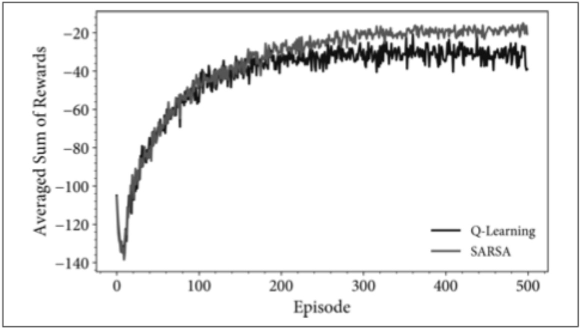


Figure 3-2. A comparison of Q-learning against SARSA for a simple grid problem. The agents were trained upon the environment in Figure 3-1. I used $\gamma \doteq 1.0$, $\epsilon \doteq 0.1$, and $\alpha \doteq 0.5$. The rewards for each episode were captured and averaged over 100 trials.

Takeaway

In this session, you learned:

- To analyze reinforcement learning as the combination of estimating long-term reward, correcting estimation error, and balancing exploit vs. explore.
- The temporal difference methods SARSA and Q-learning.