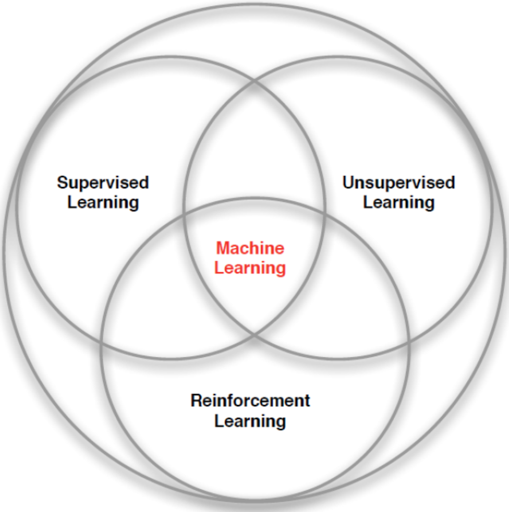
Introduction to Reinforcement Learning

Alexandre Carbonneau, M. Sc.

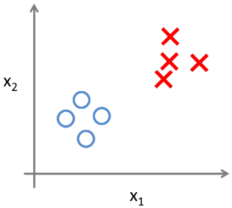
Fin ML

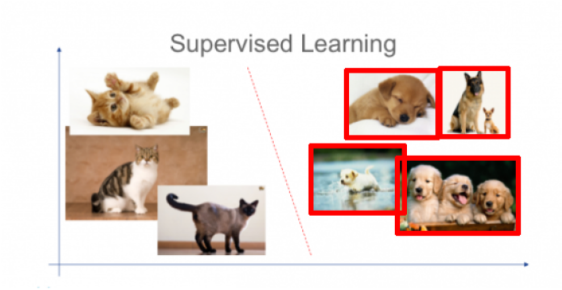
Motivation



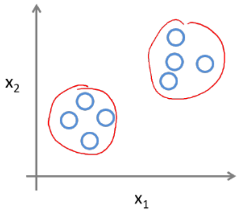
Supervised Learning
Unsupervised Learning
Machine Learning
Reinforcement Learning

Supervised learning example

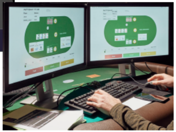


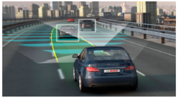


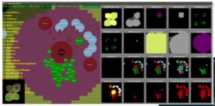
Unsupervised learning example



RL examples







Breakout

https://www.youtube.com/watch?v=TmPfTpjtdgg

Introduction to Reinforcement Learning

Goal of this presentation:

1) Present a general introduction of reinforcement learning (RL).
2) Show one method to solve the RL problem: Q-learning (will implement in Python).

RL vs machine learning

How is RL different than the typical machine learning framework.

- **No supervisor**: only a reward signal that can be sparse (i.e. delayed reward, not instantaneous).

RL vs machine learning

How is RL different than the typical machine learning framework.

- **No supervisor**: only a reward signal that can be sparse (i.e. delayed reward, not instantaneous).
- Sequential **non i.i.d.** data (time is important).

RL vs machine learning

How is RL different than the typical machine learning framework.

- **No supervisor**: only a reward signal that can be sparse (i.e. delayed reward, not instantaneous).
- Sequential **non i.i.d.** data (time is important).
- Agent's actions affect the subsequent data it receives.
  - Chess, self-driving cars, financial trading agent with a limit order book (market impact), etc...

Rewards

Every step, the agent gets a scalar feedback signal $R_t \in \mathbb{R}$ called the **reward** at step t.

- Agent's goal: select a sequence of actions to maximize **the cumulative future reward**.
- Actions may have long-term consequences.
  - Might be better to sacrifice immediate reward to gain more long-term rewards!

Examples

- Shortest path problems

  Make $R_t = -1$ at each step

Examples

- Shortest path problems

  Make $R_t = -1$ at each step
- Financial trading agent

  Make R_t the P&L from time $t-1$ to t.

Examples

- Shortest path problems

  Make $R_t = -1$ at each step
- Financial trading agent

  Make R_t the P&L from time $t-1$ to t.
- Game of chess

  $\pm$ 1 for winning/losing a game, else zero.

Examples

- Shortest path problems

  Make $R_t = -1$ at each step
- Financial trading agent

  Make R_t the P&L from time $t-1$ to t.
- Game of chess

  $\pm$ 1 for winning/losing a game, else zero.
- Atari games

  R_t implicitly defined as the score in each game

Agent and Environment

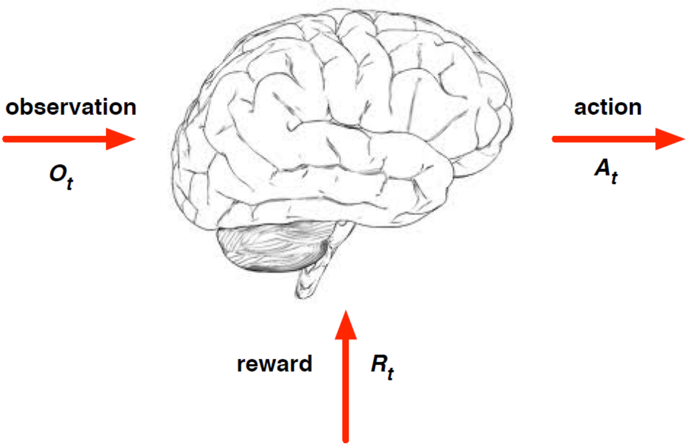


Figure: Slide from David Silver [2015].

History and state

Define the **history** as the sequence of observations, actions, rewards:

$$H_t = \{O_1, A_1, R_1, \ldots, A_{t-1}, R_{t-1}, O_t\}.$$

- Define the **state** S_t at time step t as any function f of the history:

$$S_t = f(H_t).$$

History and state

Define the **history** as the sequence of observations, actions, rewards:

$$H_t = \{O_1, A_1, R_1, \ldots, A_{t-1}, R_{t-1}, O_t\}.$$

- Define the **state** S_t at time step t as any function f of the history:

$$S_t = f(H_t).$$

- The agent has to pick the action A_t based on S_t.

History and state

Define the **history** as the sequence of observations, actions, rewards:

$$H_t = \{O_1, A_1, R_1, \ldots, A_{t-1}, R_{t-1}, O_t\}.$$

- Define the **state** S_t at time step t as any function f of the history:

$$S_t = f(H_t).$$

- The agent has to pick the action A_t based on S_t.
- Examples:

  $S_t = O_t$, simply the last observation.
  $S_t = f(O_{t-k+1}, A_{t-k+1}, R_{t-k+1} \ldots, O_t)$, the last k transitions of observations, actions and rewards.

Markov state

We assume that the state representation $\mathcal{S}$ is **Markov**:

$$\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_1, \ldots, S_t).$$

- Conclusion: the next action to pick A_t **only depends** on S_t.

Rat example

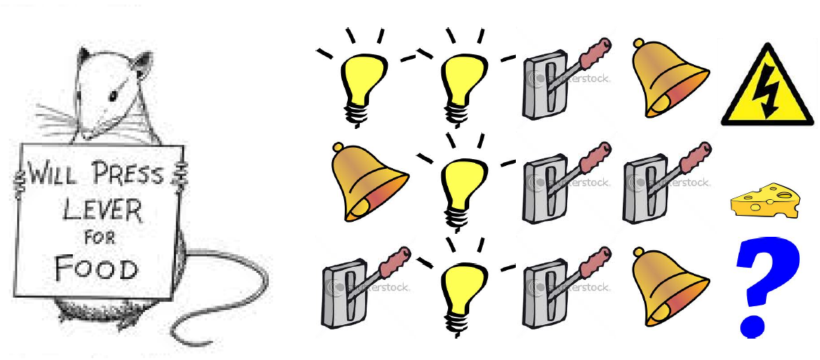
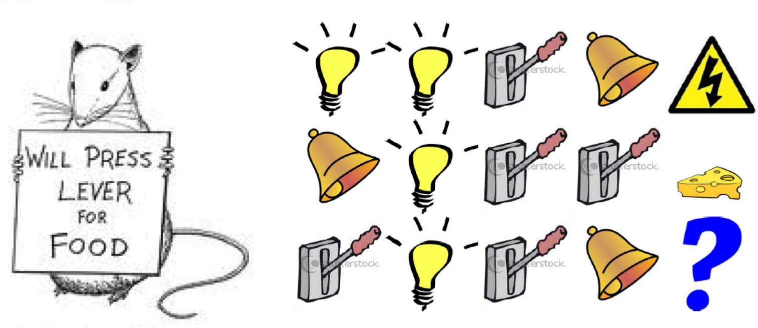


Figure: Slide from David Silver [2015].

Rat example



- What if agent state = last 3 items in sequence?
- What if agent state = counts for lights, bells and levers?
- What if agent state = complete sequence?

Figure: Slide from David Silver [2015].

How to solve the problem

- So far, we only presented the RL problem, but not how to solve it. A **policy** $\pi(a|s)$ is the agent's behavior which we want to learn from **experience** (trial-and-error).

How to solve the problem

- So far, we only presented the RL problem, but not how to solve it. A **policy** $\pi(a|s)$ is the agent's behavior which we want to learn from **experience** (trial-and-error).
- The policy is **stochastic** if $\pi(a|s) := \mathbb{P}(A_t = a|S_t = s)$, i.e. we have a distribution over the possible actions to take.

How to solve the problem

- So far, we only presented the RL problem, but not how to solve it. A **policy** $\pi(a|s)$ is the agent's behavior which we want to learn from **experience** (trial-and-error).
- The policy is **stochastic** if $\pi(a|s) := \mathbb{P}(A_t = a|S_t = s)$, i.e. we have a distribution over the possible actions to take.
- The policy is **deterministic** if $\pi : \mathcal{S} \to \mathcal{A}$, i.e. it's a mapping from state to action.

How to solve the problem

- So far, we only presented the RL problem, but not how to solve it. A **policy** $\pi(a|s)$ is the agent's behavior which we want to learn from **experience** (trial-and-error).
- The policy is **stochastic** if $\pi(a|s) := \mathbb{P}(A_t = a|S_t = s)$, i.e. we have a distribution over the possible actions to take.
- The policy is **deterministic** if $\pi : \mathcal{S} \to \mathcal{A}$, i.e. it's a mapping from state to action.
- Note: $\pi(a|s)$ depend on the **current state** S_t (not the history).

Cumulative discount rewards

The **cumulated future discounted rewards** from time $t \in \{0, 1, \ldots, T\}$ is:

$$G_t := \sum_{k=0}^{T-t} \gamma^k R_{t+k}.$$

where T is the end of the episode.

- $\gamma \in [0, 1]$ is the discount rate.
  - If γ is close to 0: agent has a "myopic" view, i.e. only short-term rewards are important.
  - If γ is close to 1: agent has a "long-term" view.

Cumulative discount rewards

The **cumulated future discounted rewards** from time $t \in \{0, 1, \dots, T\}$ is:

$$G_t := \sum_{k=0}^{T-t} \gamma^k R_{t+k}.$$

where T is the end of the episode.

- $\gamma \in [0, 1]$ is the discount rate.
  - If γ is close to 0: agent has a "myopic" view, i.e. only short-term rewards are important.
  - If γ is close to 1: agent has a "long-term" view.
- Note: T is a random variable. Ex: end of a game of chess, end of an Atari 2600 game, etc...

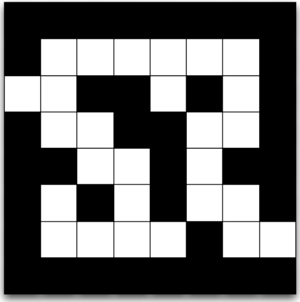
Value function

Define the **value function** of the state s under the policy π, $v_\pi(s) : \mathcal{S} \to \mathbb{R}$, as follows:

$$v_\pi(s) := \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{T-t} \gamma^k R_{t+k} | S_t = s\right].$$

- $v_\pi(s)$ tells us the **goodness/badness** of being in state s.
- Intuitively, we want to pick actions which takes us to the best states, i.e. the states where $v_\pi(s)$ is **large**.

Toy example (1)



- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Figure: Slide from David Silver [2015].

Toy example (2)

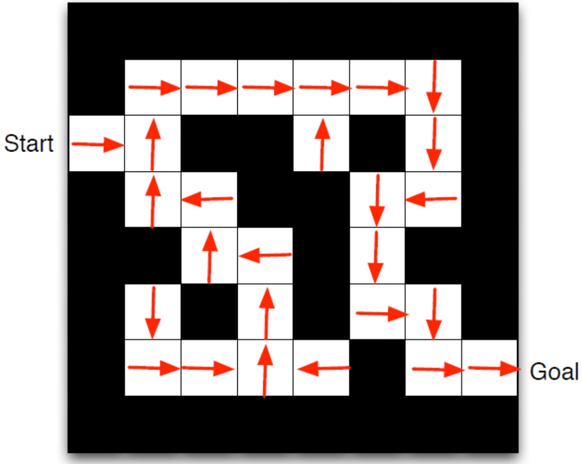
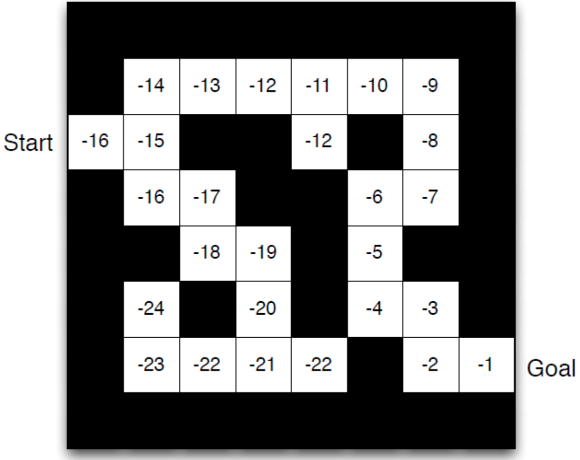


Figure: Arrows represent policy $\pi(a|s)$.

Toy example (3)



- Numbers represent value $v_\pi(s)$ of each state s

Value function - Bellman equation

With simple manipulations and the use of the law of iterated expectations, one can show that:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi\left[\sum_{k=0}^{T-t} \gamma^k R_{t+k} | S_t = s\right] \\
&= \mathbb{E}_\pi[R_t + \gamma v_\pi(S_{t+1}) | S_t = s] \qquad (1)
\end{aligned}
$$

Equation (1) is called the **Bellman equation** for the value function.

Action-value function

The action-value function $Q_\pi(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as follows:

$$\begin{aligned} Q_\pi(s, a) &:= \mathbb{E}_\pi \left[\sum_{k=0}^{T-t} \gamma^k R_{t+k} | S_t = s, A_t = a \right] \\ &= \mathbb{E}_\pi [R_t + \gamma Q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] \end{aligned} \tag{2}$$

where (2) is the **Bellman equation** for the action-value function.

Optimal $Q(s, a)$

- The **optimal action-value** function $Q_\star(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as:

$$Q_\star(s, a) := \max_{\pi \in \Pi} Q_\pi(s, a), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A},$$

  where Π is the set of possible policies.

Optimal $Q(s, a)$

- The **optimal action-value** function $Q_\star(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as:

$$Q_\star(s, a) := \max_{\pi \in \Pi} Q_\pi(s, a), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A},$$

where Π is the set of possible policies.

- Given $Q_\star(s, a)$, the **optimal** policy $\pi_\star(a|s)$ is *defined* as:

$$\pi_\star(a|s) = \begin{cases} 1, & \text{if } a = \operatorname*{argmax}_{a \in \mathcal{A}} Q_\star(s, a), \\ 0, & \text{otherwise} . \end{cases}$$

Goal: method to obtain $Q_\star(s, a)$!

Goal

1) Assumption: agent has **no prior knowledge** of the task and the environment to solve.
2) In other words: for each pair (s, a), $Q(s, a)$ is **unknown** and needs to be estimated.

Goal: method to learn $Q_\star(s, a)$ by simulation in order to learn the optimal behavior $\pi_\star$.

ϵ-greedy policy (1)

Suppose $|\mathcal{A}| = m$. The type of policy we will work with are called ϵ-greedy policy. Let $\epsilon \in [0, 1]$. At each time step:

- Pick a **random** action with probability ϵ, with each action having a probability ϵ/m of being chosen.

ϵ-greedy policy (1)

Suppose $|\mathcal{A}| = m$. The type of policy we will work with are called ϵ-greedy policy. Let $\epsilon \in [0, 1]$. At each time step:

- Pick a **random** action with probability ϵ, with each action having a probability ϵ/m of being chosen.
- Pick the action that maximizes $Q(s, a)$ with probability $1 - \epsilon$, also called the **greedy action**.

ϵ-greedy policy (1)

Suppose $|\mathcal{A}| = m$. The type of policy we will work with are called ϵ-greedy policy. Let $\epsilon \in [0, 1]$. At each time step:

- Pick a **random** action with probability ϵ, with each action having a probability ϵ/m of being chosen.
- Pick the action that maximizes $Q(s, a)$ with probability $1 - \epsilon$, also called the **greedy action**.
- Mathematically:

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon, & \text{if } a = \underset{A \in \mathcal{A}}{\operatorname{argmax}}\, Q(s, A), \\ \epsilon/m, & \text{otherwise.} \end{cases}$$

ϵ-greedy policy (2)

Why choose an ϵ-greedy policy:

1) Simplest idea for ensuring **continual exploration** (all m actions are tried with non-zero probability as long as $\epsilon > 0$).
2) Under some constraints on the ϵ decay, an ϵ-greedy policy converges to $\pi_\star$ (next slides).

Q-learning(1)

The famous algorithm called **Q-learning** from Watkins [1992] can be used with an ϵ-greedy policy to obtain the optimal policy $\pi_\star$.

- You will implement a variation of Q-learning in the context of an optimal trade execution problem!

Q-learning (2)

1) Initialize $Q(s, a)$ randomly $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$. Ex: $Q(s, a) = 0$.
2) Start at the initial state $s \in \mathcal{S}$.
3) Repeat until the **end of the episode**:
   3.1) Select the action $a \sim \pi(a|s)$.
   3.2) Take action a, observe the reward r and move to the state s':

$$\{s, a, r, s'\}$$

   .

   3.3) Update (with $\alpha \in [0, 1)$, the **learning rate**)

$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') \right].$$

4) Go back to step 2).

Fin ML

Q-learning (3)

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') \right].$$

Intuition:

1) $Q(s, a)$ is the current estimate of the action-value function for the state s and action a.
2) The term $(1 - \alpha)Q(s, a)$ is a weighted value of our current estimate.
3) The term $\alpha \left[r + \gamma \max_{a'} Q(s', a') \right]$ *comes* from the Bellman Optimality equation:

$$Q_\star(s, a) = \mathbb{E}_\pi \left[R_t + \gamma \max_{A_{t+1}} Q_\star(S_{t+1}, A_{t+1}) | S_t = s, A_t = a \right].$$

Q-learning (4)

Results:

1. Under **certain constraints** on α and ϵ, Q-learning converges to the optimal action-value function $Q_\star$.

Q-learning (4)

Results:

1. Under **certain constraints** on α and ϵ, Q-learning converges to the optimal action-value function $Q_\star$.
2. Q-learning provides the **optimal policy**:

$$a = \operatorname*{argmax}_{A \in \mathcal{A}} Q_\star(s, A), \quad \forall s \in \mathcal{S}.$$

Q-learning: Toy example - black board

1. State space: $\mathcal{S} = \{0, 1, \ldots, 4, 5\}$ (6 possible states). Starting state and ending state: $S_1 = 3$ and $S_T = 5$.

Q-learning: Toy example - black board

1. State space: $\mathcal{S} = \{0, 1, \dots, 4, 5\}$ (6 possible states). Starting state and ending state: $S_1 = 3$ and $S_T = 5$.
2. Reward matrix (shortest path problem):

$$R = \begin{bmatrix} -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 100 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where rewards of zero are **impossible** transitions (see board).

Q-learning: Toy example - black board

1. State space: $\mathcal{S} = \{0, 1, \ldots, 4, 5\}$ (6 possible states). Starting state and ending state: $S_1 = 3$ and $S_T = 5$.
2. Reward matrix (shortest path problem):

$$R = \begin{bmatrix} -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 100 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

   where rewards of zero are **impossible** transitions (see board).
3. Initialize $Q(s, a) = 0$, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$.

Q-learning: Toy example - black board

1. State space: $\mathcal{S} = \{0, 1, \ldots, 4, 5\}$ (6 possible states). Starting state and ending state: $S_1 = 3$ and $S_T = 5$.
2. Reward matrix (shortest path problem):

$$R = \begin{bmatrix} -1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 100 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

   where rewards of zero are **impossible** transitions (see board).
3. Initialize $Q(s, a) = 0$, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$.
4. Additional hyperparameters: fix $\epsilon = 0.7$, $\gamma = 1.00$ and $\alpha = 0.9$.

Fin ML

Q-learning: Toy example - first transition

1) Start at state $s = 3$. Three possible actions: $a \in \{1, 2, 4\}$.
2) Sample action $a \sim \pi(a|s = 3)$. Two possibilities:
   A) Pick a random action with probability 0.7.
   B) Pick the greedy action with probability 0.3.
3) Suppose the action taken is $a = 4$. The transition is the following:
$$\{s, a, r, s'\} = \{3, 4, -1, 4\}$$
4) Update $Q(3, 4)$[1]:

$$\begin{aligned} Q(3,4) &= (1-0.9)Q(3,4) + 0.9\left[-1 + \max_{a' \in \{0,3,5\}} Q(4,a')\right] \\ &= (1-0.9) \times 0 + 0.9\,[-1+0] = -0.9. \end{aligned}$$

---

[1]
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right].$$

Fin ML

Q-learning: Toy example - second transition

1) Currently in state $s = 4$. Three possible actions: $a \in \{0, 3, 5\}$.
2) Sample action $a \sim \pi(a|s = 4)$. Three possibilities:
   - A) Pick a random action with probability 0.7.
   - B) Pick the greedy action with probability 0.3.
3) Suppose the action taken is $a = 5$. The transition is the following:

$$\{s, a, r, s'\} = \{4, 5, 100, 5\}$$

4) Update $Q(4, 5)$:

$$Q(4,5) = (1 - 0.9)Q(4,5) + 0.9 \left[100 + \max_{a'} Q(5, a')\right]$$
$$= (1 - 0.9) \times 0 + 0.9\,[100 + 0] = 90$$

Q-learning: Toy example - conclusion

1. The transitions of the first episode from the starting state $s = 3$:

$$3 \longrightarrow 4 \longrightarrow 5.$$

Q-learning: Toy example - conclusion

1. The transitions of the first episode from the starting state $s = 3$:
$$3 \longrightarrow 4 \longrightarrow 5.$$
2. Made the following updates to the Q-function:
$$Q(3,4) = -0.9, \quad Q(4,5) = 90,$$
while the other values of $Q(s,a)$ are **unchanged** and still zero.

Q-learning: Toy example - conclusion

1. The transitions of the first episode from the starting state $s = 3$:
$$3 \longrightarrow 4 \longrightarrow 5.$$
2. Made the following updates to the Q-function:
$$Q(3,4) = -0.9, \quad Q(4,5) = 90,$$
while the other values of $Q(s,a)$ are **unchanged** and still zero.
3. Next steps: start a new episode of training (i.e. start back at the initial state $s = 3$ and procede the same way).

Q-learning: Toy example - conclusion

1. The transitions of the first episode from the starting state $s = 3$:
$$3 \longrightarrow 4 \longrightarrow 5.$$
2. Made the following updates to the Q-function:
$$Q(3,4) = -0.9, \quad Q(4,5) = 90,$$
while the other values of $Q(s,a)$ are **unchanged** and still zero.
3. Next steps: start a new episode of training (i.e. start back at the initial state $s = 3$ and procede the same way).
4. Will we converge to the optimal policy? Q-learning converges to $\pi_\star$ **only if** ϵ and α eventually reaches zero under certain constraints!

Fin ML

Large-Scale Reinforcement Learning

Most interesting problems are too large to be solved using a tabular method:

- Backgammon: 10^{20} states;
- Game of Go: 10^{170} states;
- Control a robot: continuous state space;

Q-learning **can't be applied** for large-scale problems.

- In such cases, function approximators for the Q-function (or value function) are often used.
- For example, a neural network!

Additional Ressources

1) *Reinforcement Learning: An introduction* by R. Sutton and A. Barto (2017). Free book on their website.

2) David Silver's online lectures on YouTube (from 2015). Main author on many breakthroughs in reinforcement learning and an excellent teacher!

3) Deep Reinforcement Learning course from UC Berkeley `http://rail.eecs.berkeley.edu/deeprlcourse/` with online lectures (Fall 2018).

4) For a bridge between RL and optimal control: Course of Dimitri P. Bertsekas (2019) with online lectures `http://web.mit.edu/dimitrib/www/RLbook.html`.