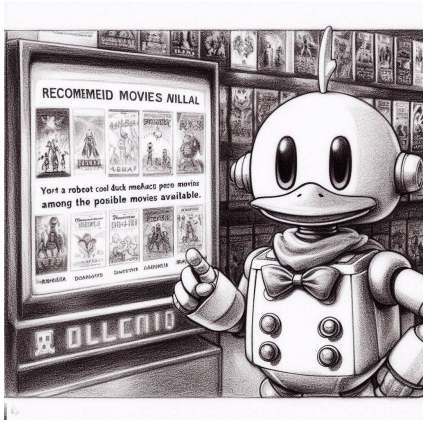
INSA TOULOUSE

WikiStat

Introduction to Reinforcement Learning



AI frameworks

Course: 8 videos

- Introduction
- Markov decision process
- Policy and Value function
- Bellman equations
- Dynamic programming
- Monte-Carlo methods
- Temporal differences
- DQN

Introduction

The three main branches in Machine Learning:

Supervised Learning

Acces to:
- Data: X
- Labels: y

Goal:
- Learn a mapping between X and y

Unsupervised Learning

Acces to:
- Data: X

Goal:
- Discover patterns in the data, learn the distributions of X

Reinforcement Learning

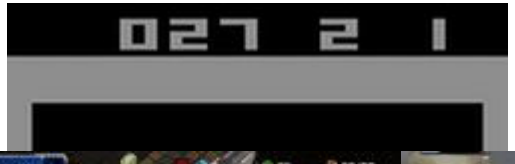
Acces to:
- A simulator

Goal:
- Learn the optimal policy

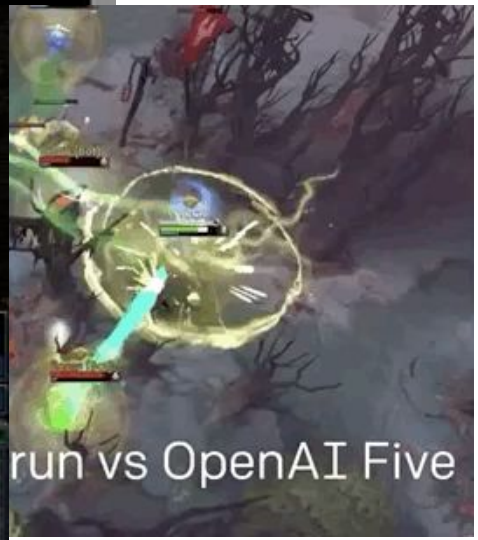
Recent successes in RL

Video games:

- Atari



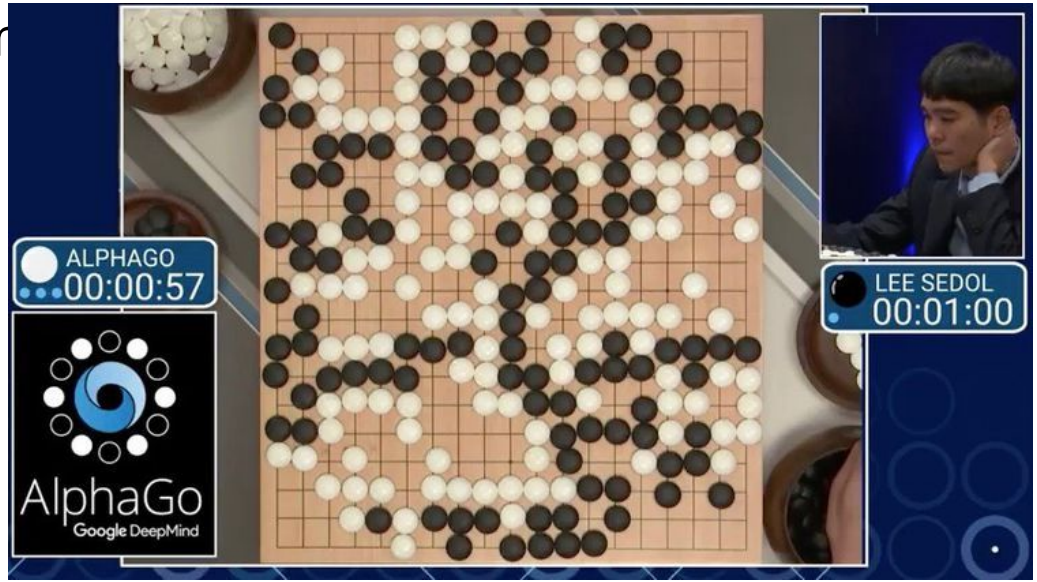
- Dota 2
- Starcraft



Recent successes in RL

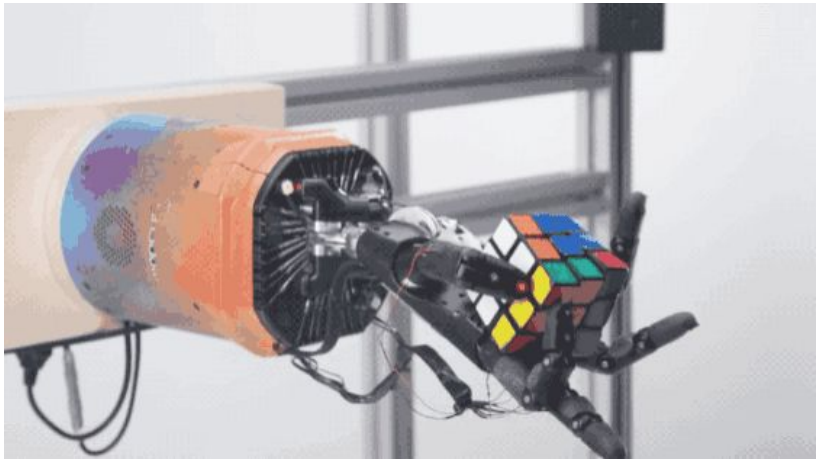
Two players gam

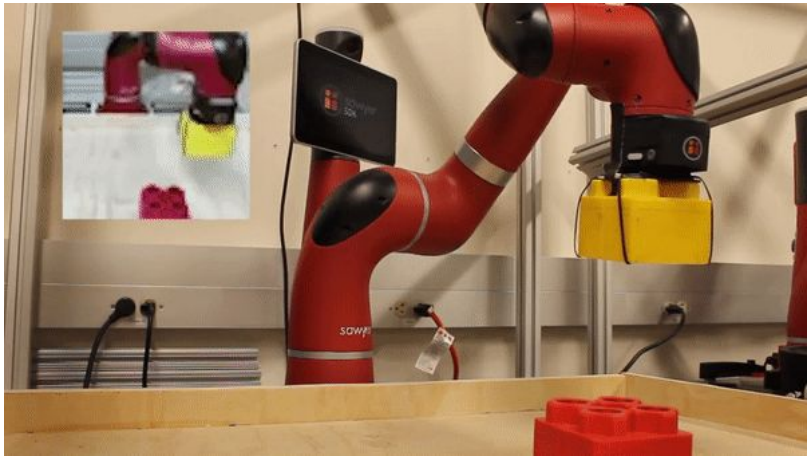
- AlphaGo
- AlphaZero



Recent successes in RL

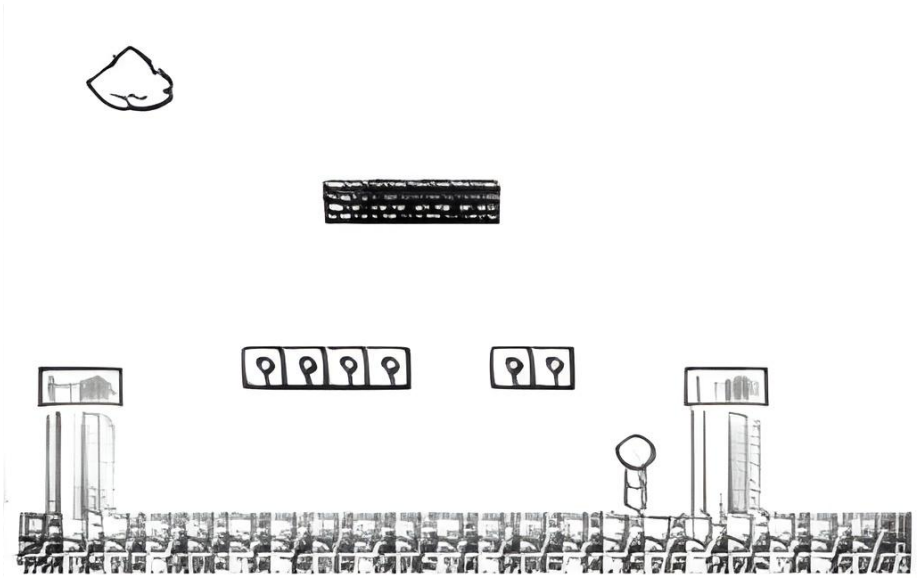
Robotics:





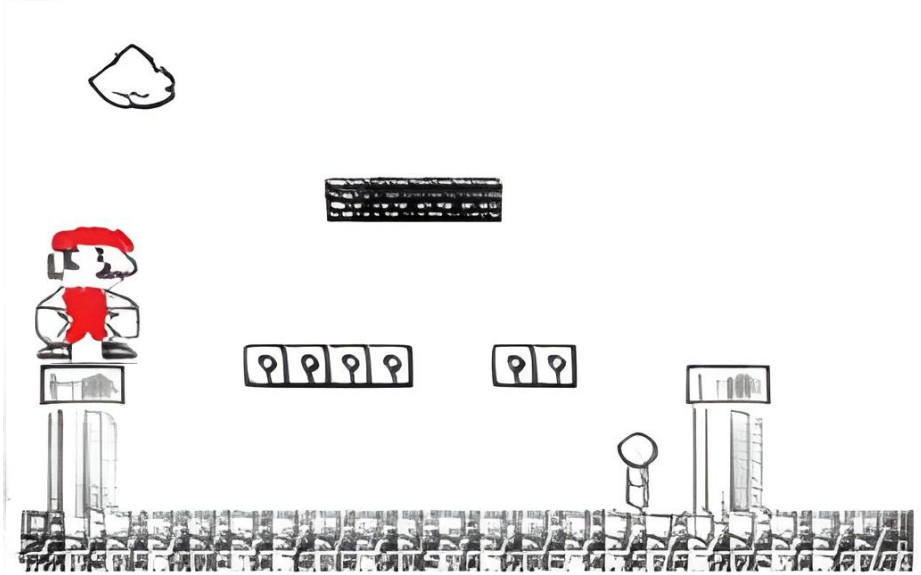
Reinforcement Learning

- **Environment**



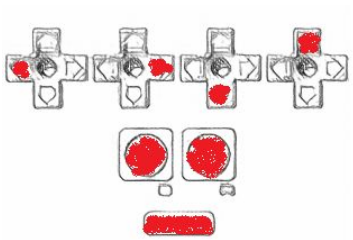
Reinforcement Learning

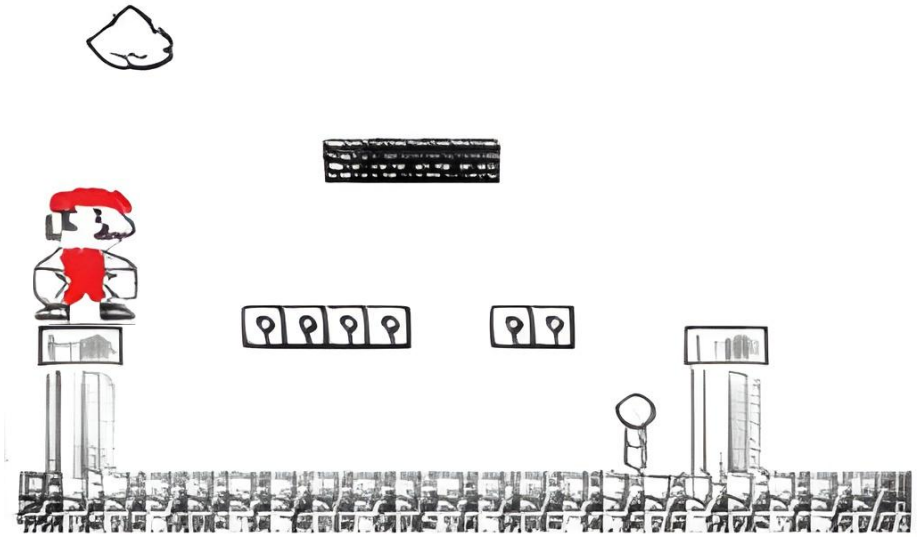
- **Environment**
- **Agent**



Reinforcement Learning

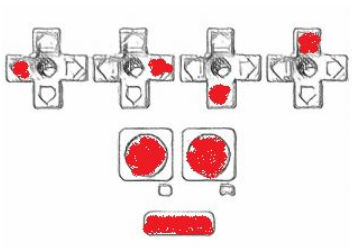
- **Environment**
- **Agent**
- **Action** $a \in \mathcal{A}$

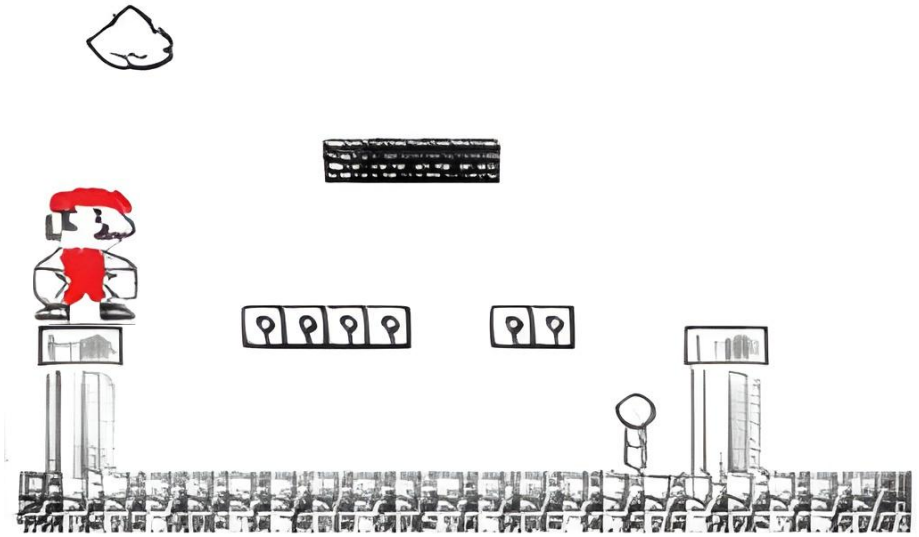




Reinforcement Learning

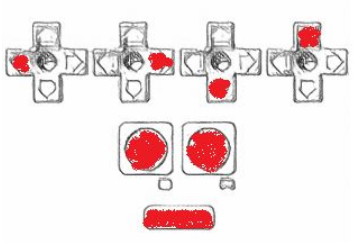
- **Environment**
- **Agent**
- **Action** $a \in \mathcal{A}$
- **State** $s \in \mathcal{S}$

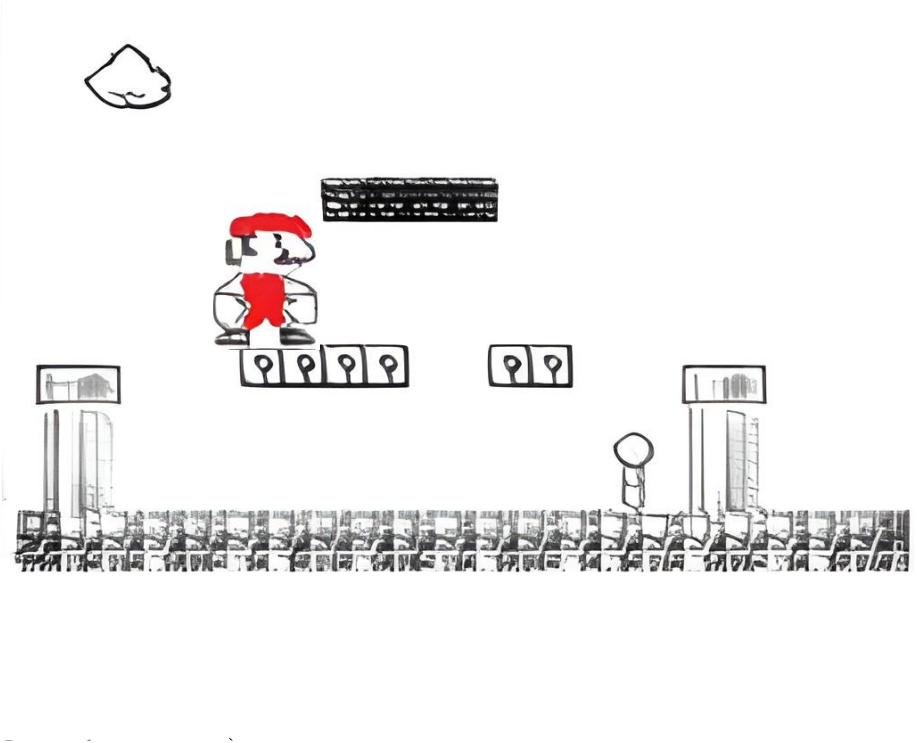




Reinforcement Learning

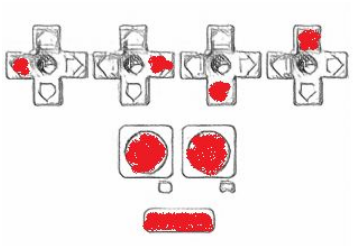
- **Environment**
- **Agent**
- **Action** $a \in \mathcal{A}$
- **State** $s \in \mathcal{S}$
- **Transition model** $p(s'|s, a)$

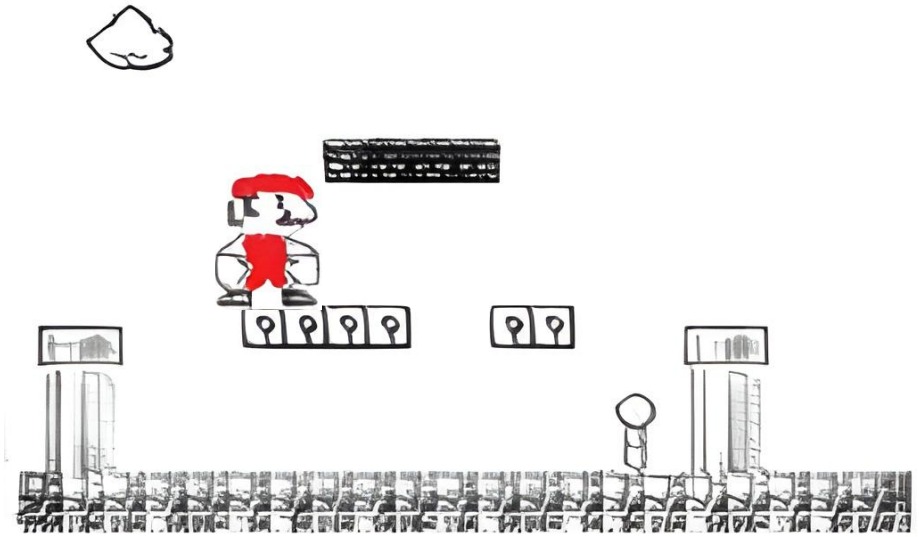




Reinforcement Learning

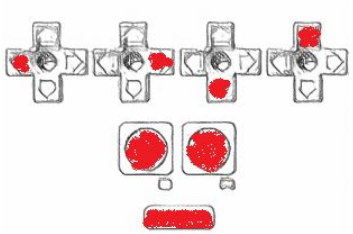
- **Environment**
- **Agent**
- **Action** $a \in \mathcal{A}$
- **State** $s \in \mathcal{S}$
- **Transition model** $p(s'|s, a)$
- **Policy** $\pi : \mathcal{S} \rightarrow \mathcal{A}$

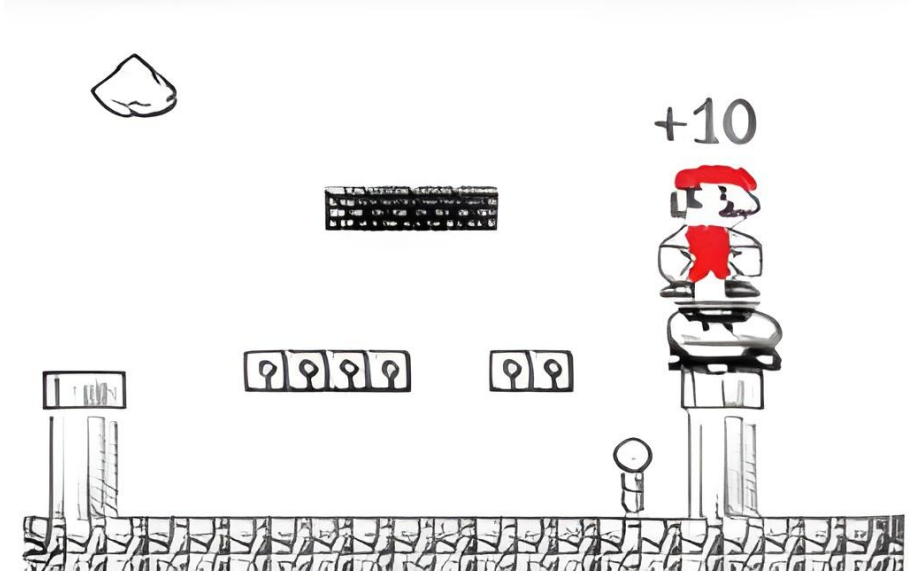




Reinforcement Learning

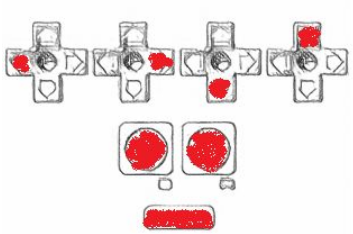
- **Environment**
- **Agent**
- **Action** $a \in \mathcal{A}$
- **State** $s \in \mathcal{S}$
- **Transition model** $p(s'|s,a)$
- **Policy** $\pi : \mathcal{S} \rightarrow \mathcal{A}$
- **Reward** r

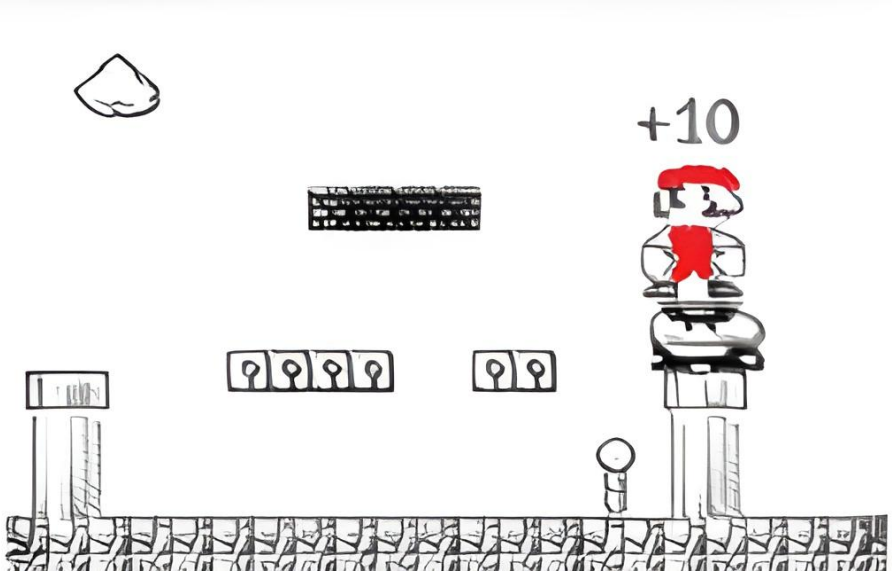




Reinforcement Learning

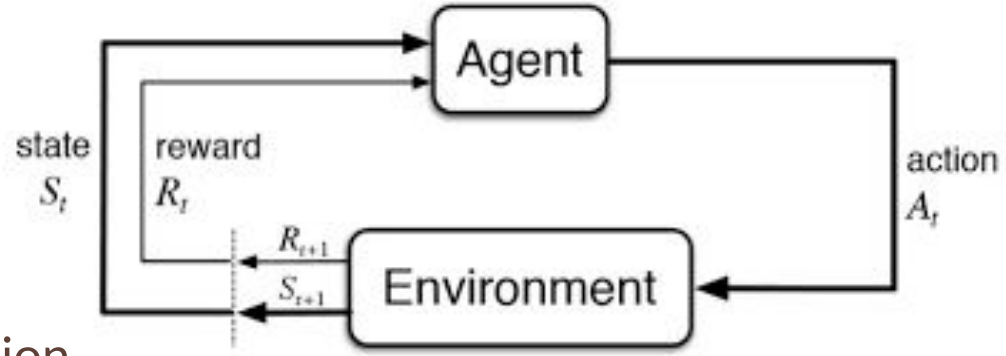
- **Environment**
- **Agent**
- **Action** $a \in \mathcal{A}$
- **State** $s \in \mathcal{S}$
- **Transition model** $p(s'|s,a)$
- **Policy** $\pi : \mathcal{S} \to \mathcal{A}$
- **Reward** r
- **Markov Decision Process** $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r)$





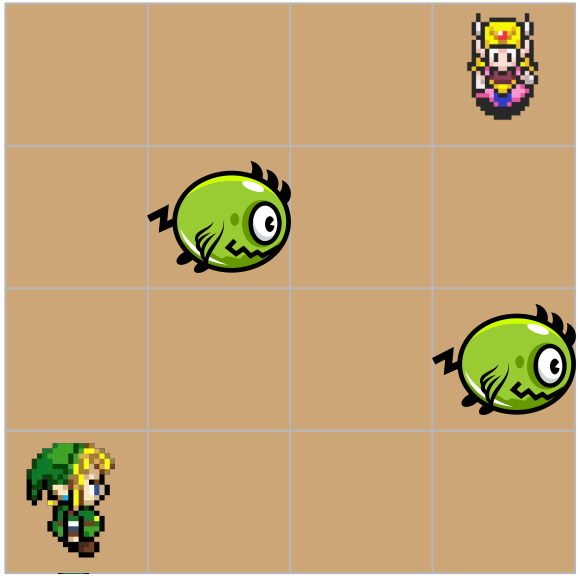
Markov Decision Process:

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action Space
- $P(S_{t+1}|s_t, a_t)$: transition model
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: reward function



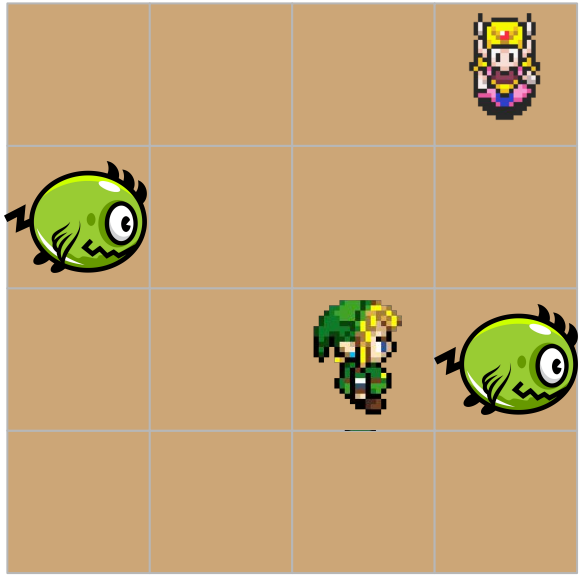
Markov Decision Process:

- $\mathcal{S}$: State space



Markov Decision Process:

- $\mathcal{S}$: State space

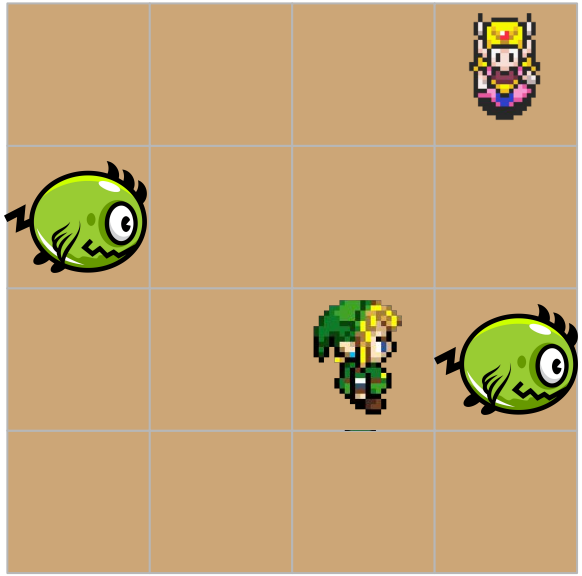


Markov Decision Process:

- $\mathcal{S}$: State space

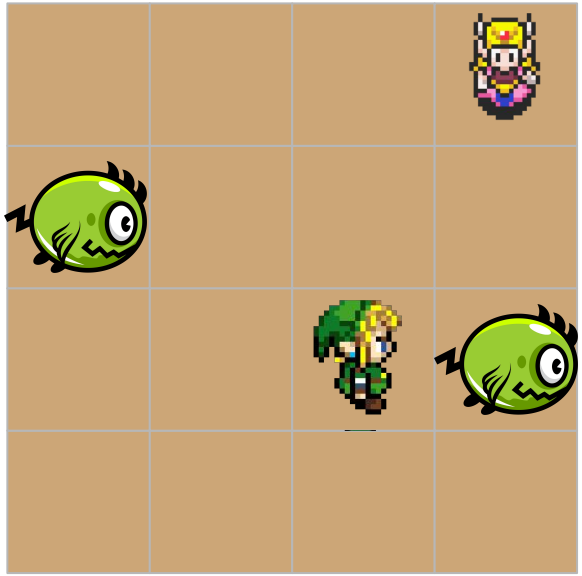
Markov Decision Process:

- $\mathcal{A}$: Action space
  - $\leftarrow, \uparrow, \rightarrow, \downarrow$
  - attack



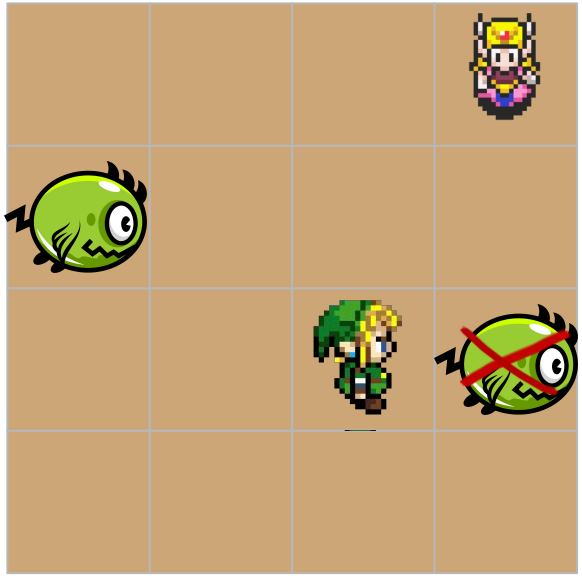
Markov Decision Process:

- $P(S_{t+1}|s_t, a_t)$: transition model



Markov Decision Process:

- $P(S_{t+1}|s_t, a_t)$: transition model
  - Deterministic

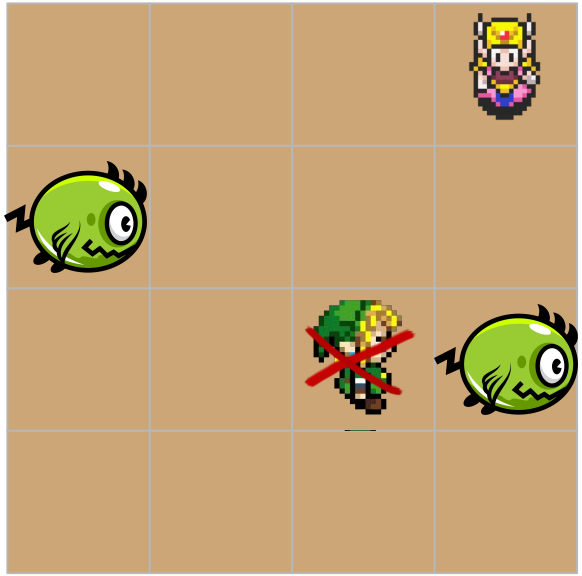


Action: Attack!

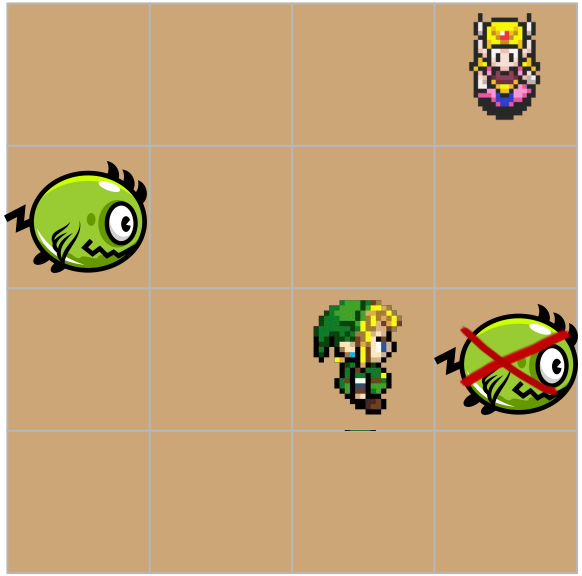
Markov Decision Process:

- $P(S_{t+1}|s_t, a_t)$: transition model
  - Deterministic
  - Stochastic

Action: Attack!



Markov Decision Process:



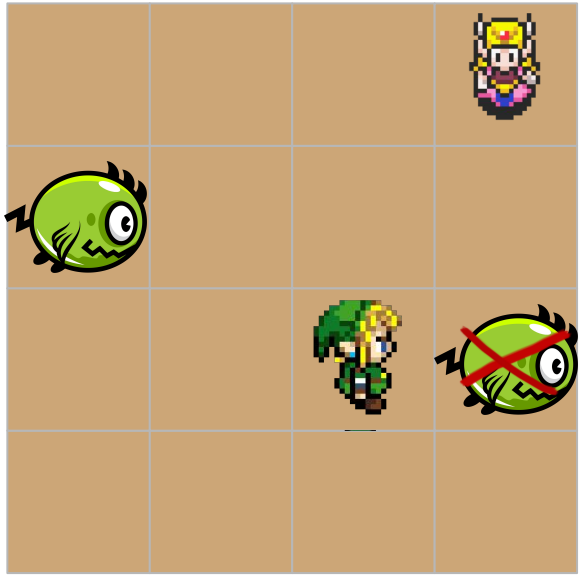
- $P(S_{t+1}|s_t, a_t)$: transition model
  - Deterministic
  - Stochastic

- Markov property:

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, S_3, ..., S_t)$$

Markov Decision Process:

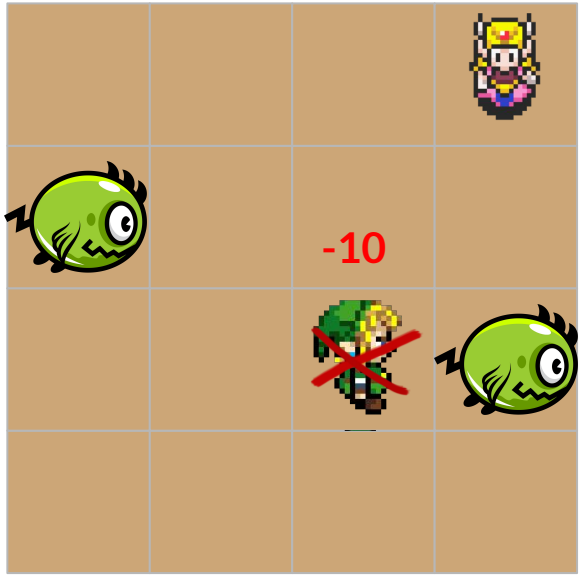
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: reward function



Action: Attack!

Markov Decision Process:

- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: reward function



Action: Attack!

Policy

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action space
  - $\leftarrow, \uparrow, \rightarrow, \downarrow$
  - attack

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

Policy

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action space
  - $\leftarrow, \uparrow, \rightarrow, \downarrow$
  - attack

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

When  -> Attack with p=0.5 | Run with p=0.5

When  -> Save

Markov Decision Process:

- **Return:**

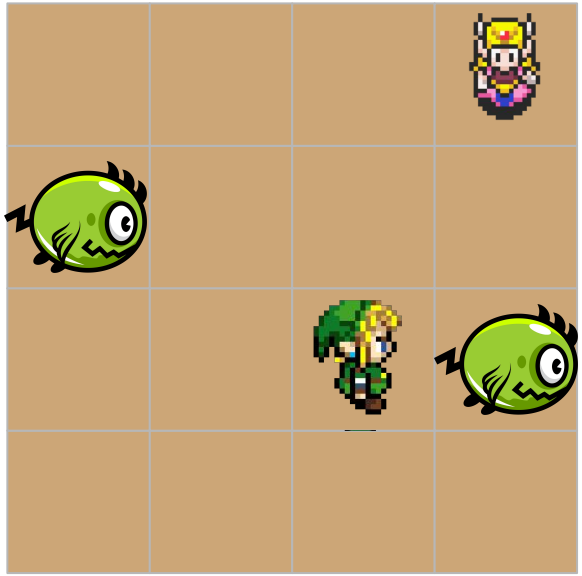
$$G_t = R_{t+1} + R_{t+2} + ... + R_T$$

- **Discount factor:** $\gamma \in [0, 1]$

- **Discounted return:**

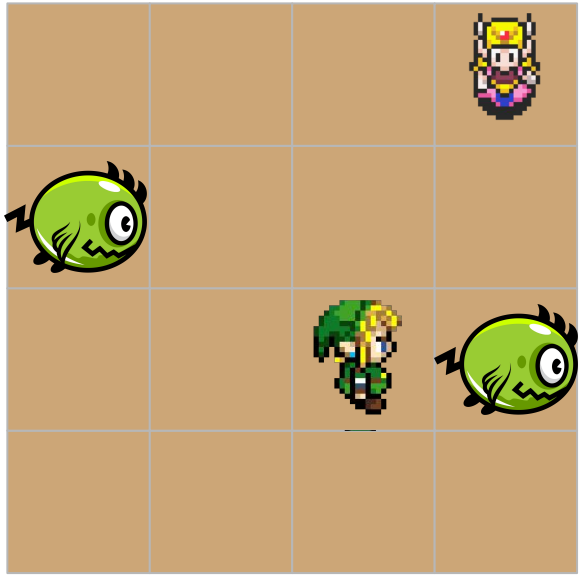
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$$

$$= \sum_{k=0} \gamma^k R_{t+k+1}$$

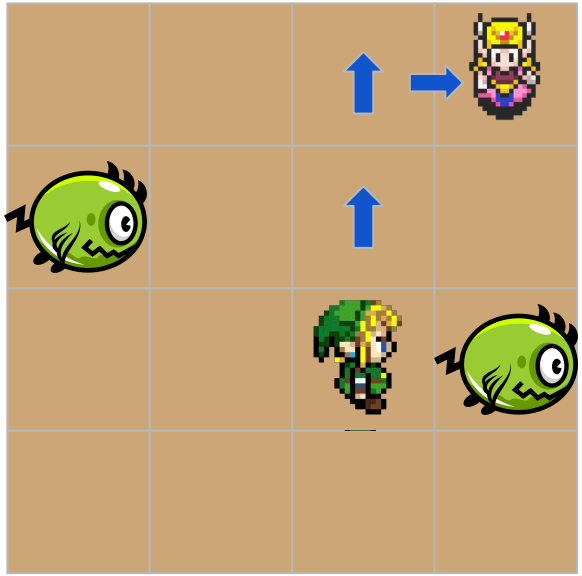


Markov Decision Process:

- $\gamma = 0.9$
-  = +10
- One step = -1



Markov Decision Process:



- $\gamma = 0.9$
-  = +10
- One step = -1

$$G_t = -1 + 0.9 * -1 + 0.9^2 * 10$$

INSA TOULOUSE

Value Function

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

$$= \mathbb{E}_\pi[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s] \text{ , for all } s \in \mathcal{S}$$

INSA TOULOUSE

+10

Deterministic policy

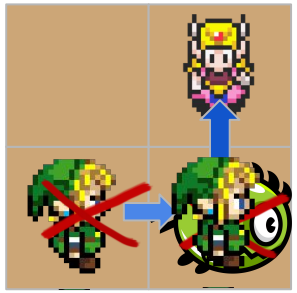
Coward Link:

$$v_{\pi}(s) = 0 + 0.9 * 10 = 9$$

Deterministic policy

Brave Link:

$$v_{\pi}(s) = -10 * 0.2 + 1 * 0.8 + 0.9 * 10 = 10.2$$

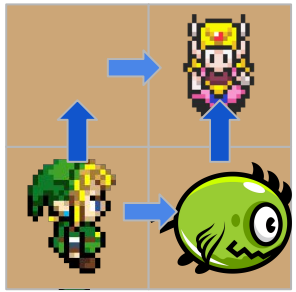


- kill: p = 0.8
- die: p = 0.2

Stochastic policy

Tired Link:

- Goes directly with p=0.5
- Tries to kill monster with p=0.5



- kill: p = 0.8
- die: p = 0.2

$$v_\pi(s) = 0.5 * (0 + 0.9 * 10)$$
$$+0.5 * (-10 * 0.2 + 0.8 * (1 + 0.9 * 10))$$

State-action value function or Q-Function

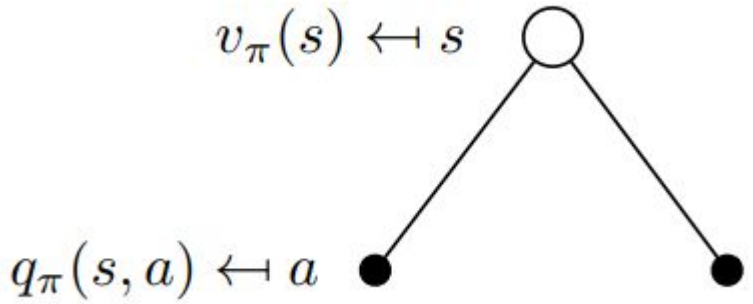
$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

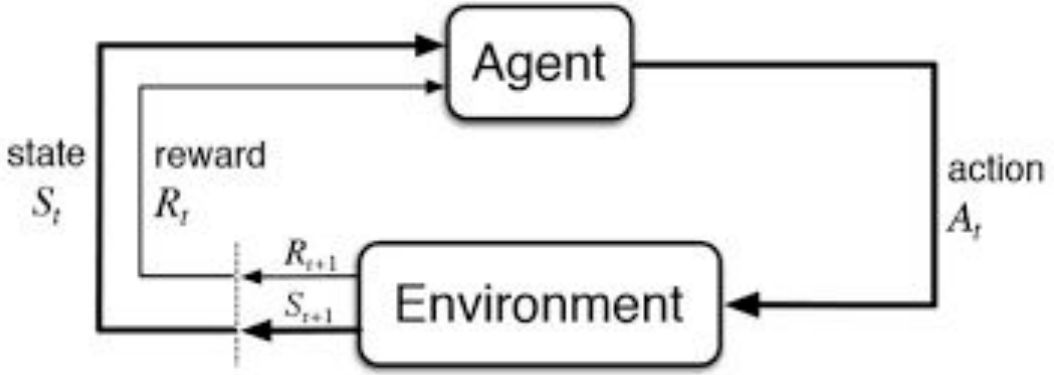
, for all $s \in \mathcal{S}$

Value Function and Q-Function

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



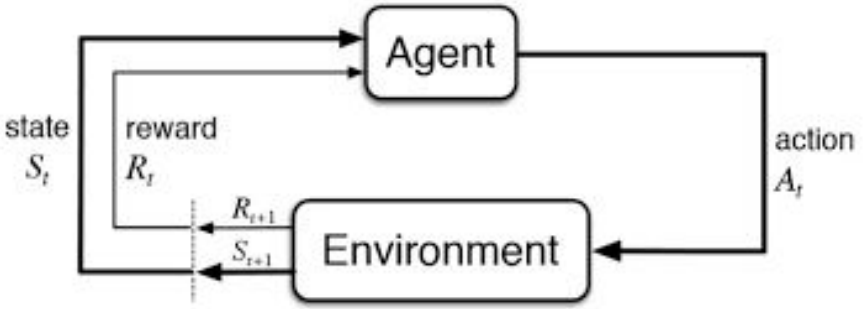
Recap:



Recap:

Markov Decision Process:

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action Space
- $P(S_{t+1}|s_t, a_t)$: Transition model
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: Reward function



Markov property: $P(S_{t+1}|S_t) = P(S_{t+1}|S_1, S_2, S_3, ..., S_t)$

Recap:

Policy:

- $\mathcal{S}$: State space
- $\mathcal{A}$: Action space
  - $\leftarrow, \uparrow, \rightarrow, \downarrow$
  - attack

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

Recap:

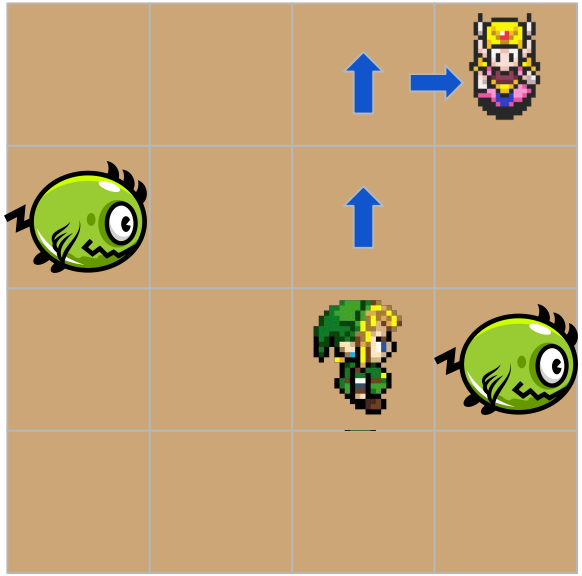
- **Return:**

$$G_t = R_{t+1} + R_{t+2} + ... + R_T$$

- **Discounted return:**

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$$

$$= \sum_{k=0} \gamma^k R_{t+k+1}$$



INSA TOULOUSE

Recap:

- **Value Function:**

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

- **Q-function:**

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

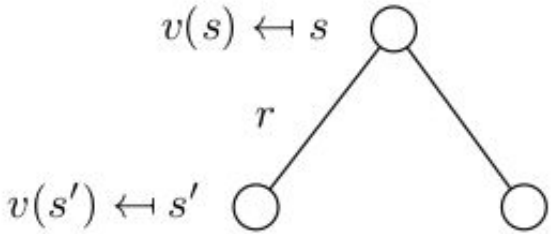
Bellman equation

Bellman Equation

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s] \quad \text{, for all } s \in \mathcal{S}$$

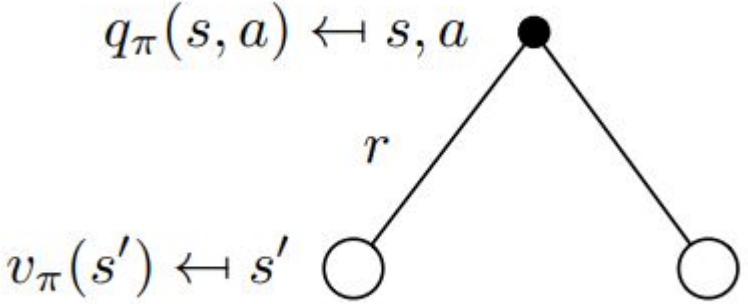
$$= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_t = s]$$

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



Bellman Equation

$$q_\pi(s,a) = \mathbb{E}_\pi[\sum_{k=} \quad \quad a]$$



$$= \mathbb{E}_\pi[R \quad \quad = s, A_t = a]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

INSA TOULOUSE

Bellman Equation

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) Q_\pi(s, a)$$

INSA TOULOUSE

Optimality

Optimal state value function:

$$v_*(s) = \max_\pi v_\pi(s)$$

Optimal state-action value function:

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

INSA TOULOUSE

Optimal policy

$$\pi_*(s) = \underset{a}{\operatorname{argmax}}(q_*(s, a))$$

How do we find these q*(s,a) values ?

INSA TOULOUSE

Bellman Optimal Equations

$$V_*(s) = \max_{a \in \mathcal{A}} Q_*(s,a)$$

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_* \left(s'\right)$$

$$V_*(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_* \left(s'\right) \right)$$

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} \max_{a' \in \mathcal{A}} Q_* \left(s', a'\right)$$

Dynamic Programming

- **Principle**
  - solve complex problems by breaking them down into sub-problems
  - solutions to the sub-problems are combined to solve overall problem

- **Required properties:**
  - Optimal substructure

  - Overlapping sub-problems

    ->Bellman Equation gives recursive decomposition

    -> The Value function stores and reuses solutions

- **Problem:** Evaluate a given policy
- **Solution:** Iterative application of Bellman Expectation

$$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_\pi$$

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{s'}^a v_k\left(s'\right) \right)$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

0	0	0	0

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

0	0	0	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$v(s) = -1 + \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 10$$

$$= 4$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

0	0	4	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$v(s) = -1 + \frac{1}{2} \cdot 4 + \frac{1}{2} \cdot 0$$

$$= 1$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

0	1	4	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$v(s) = -1 + \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 0$$

$$= -0.5$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

-0.5	1	4	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

-0.5	1	4	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$v(s) = -1 + \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 10$$

$$= 4.5$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

-0.5	1	4.5	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$v(s) = -1 + \frac{1}{2} \cdot 4.5 + \frac{1}{2} \cdot -0.5$$

$$= 1$$

$$\pi(\leftarrow|a) = \pi(\rightarrow|a) = 0.5$$

-0.5	1	4.5	0

Reward:

- -1 at every step
- +10 for diamond

$$\gamma = 1$$

$$v(s) = -1 + \frac{1}{2} \cdot -0.5 + \frac{1}{2} \cdot 1$$
$$= -0.25$$

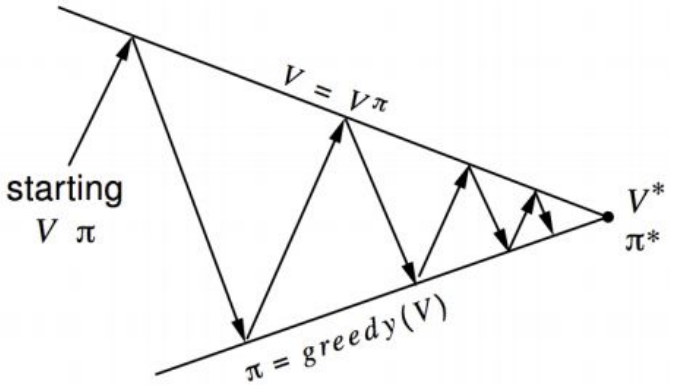
INSA TOULOUSE

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

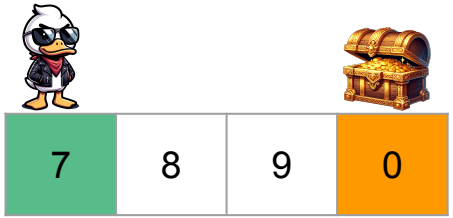
Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

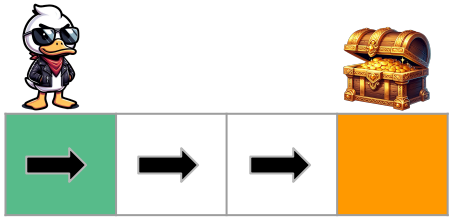
Policy iteration



- **Problem:** Find the optimal policy

- **Solution:**
  - Evaluate values for current policy (**Policy evaluation**)
  - Improve policy **greedily**
  - Repeat until convergence





INSA TOULOUSE

Policy iteration

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   old-action $\leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s') \right]$
   If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

- **Problem:** Find the optimal policy
- **Solution:** Iterative application of Bellman Optimality

$$v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots \rightarrow v_\pi$$

$$v_*(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_* \left(s'\right)$$

INSA TOULOUSE

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $\quad v \leftarrow V(s)$
| $\quad V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$
| $\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a)\big[r + \gamma V(s')\big]$

INSA TOULOUSE

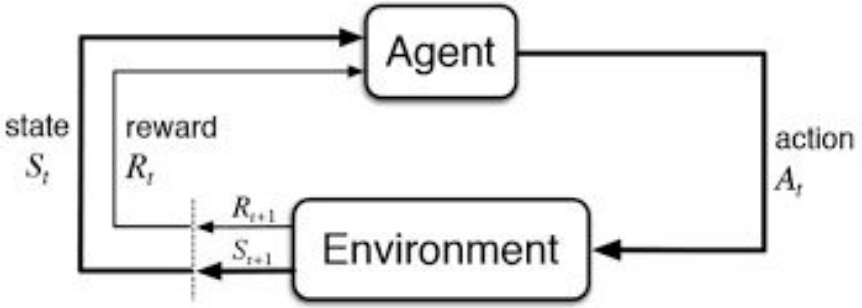
Recap:

Problem	Bellman Equation	Algorithm
Prediction	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

Recap:

Markov Decision Process:

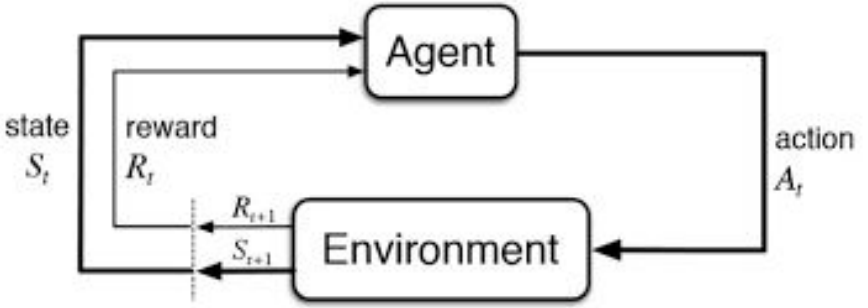
- $\mathcal{S}$: State space
- $\mathcal{A}$: Action Space
- $P(S_{t+1}|s_t, a_t)$: transition model
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$: reward function



Model-Free Reinforcement Learning:

$P(S_{t+1}|s_t, a_t)$: ?

-> Learn from interactions with the environment



INSA TOULOUSE

Monte Carlo Methods

- **Monte Carlo Policy Evaluation:** π is given, evaluate v_π
- **Monte Carlo Control:** approximate π_*

Monte Carlo Policy Evaluation:

Learning $v_\pi(s)$ by sampling:

- Random choice of a starting state $s \in \mathcal{S}$
- Follow the policy π and observe the cumulative gain G_t
- Repeat N times and average: $v_\pi(s) = \mathbb{E}_\pi[G_t]$

INSA TOULOUSE

Monte Carlo Methods

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

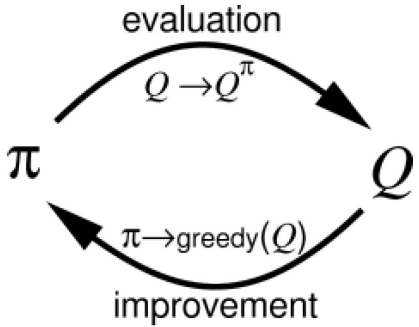
$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte Carlo Control:



Learning $Q_\pi(s, a)$ by sampling:

- Random choice of a starting state $s \in \mathcal{S}$
- Random choice of action a
- Follow the policy π and observe the cumulative gain G_t
- Repeat N times and average: $Q_\pi(s, a) = \mathbb{E}_\pi[G_t]$
- Improve Policy: $\pi(s) = \underset{a}{\text{argmax}}\ Q_\pi(s, a)$

INSA TOULOUSE

Monte Carlo Methods

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

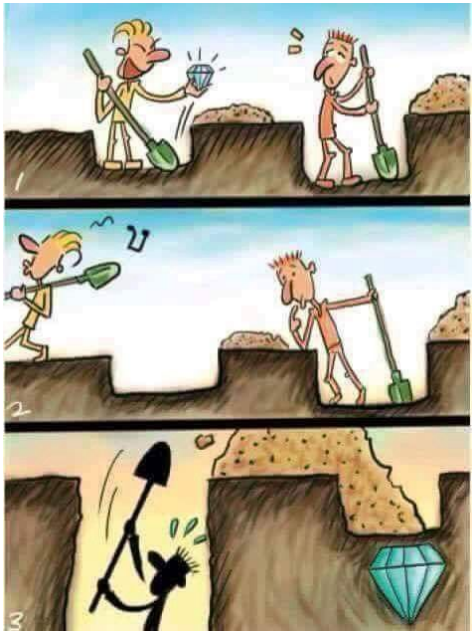
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Exploration vs Exploitation:

Learning $Q_\pi(s,a)$ by sampling:

- Random choice of a starting state $s \in \mathcal{S}$
- Random choice of
- Follow the policy π and observe the cumulative gain G_t
- Repeat N times and average: $Q_\pi(s,a) = \mathbb{E}_\pi[G_t]$
- Improve Policy: $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q_\pi(s,a)$

Exploration vs Exploitation:



Exploration vs Exploitation:

Learning $Q_\pi(s,a)$ by sampling:

- Greedy policy: $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q_\pi(s,a)$
- Epsilon-greedy policy: $\pi(s) = \begin{cases} \underset{a}{\operatorname{argmax}}\, Q_\pi(s,a) \text{ with probability } 1-\epsilon \\ \text{random with probability } \epsilon \end{cases}$

INSA TOULOUSE

Monte Carlo Methods

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

INSA TOULOUSE

Recap:

- **Dynamic programming:**
  - Need to know the transition model $P(S_{t+1}|s_t, a_t)$
  - Benefits of Bellman equation $V_*(s) = \max_{a \in \mathcal{A}}(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s'))$
- **Monte Carlo Methods:**
  - No need to know $P(S_{t+1}|s_t, a_t)$
  - Need to go through the entire episode

Incremental mean:

The mean $\mu_1, \mu_2, \ldots$
of a sequence $x_1, x_2, \ldots$
can be computed incrementally:

$$
\begin{aligned}
\mu_k &= \frac{1}{k}\sum_{j=1}^{k} x_j \\
&= \frac{1}{k}\left(x_k + \sum_{j=1}^{k-1} x_j\right) \\
&= \frac{1}{k}\left(x_k + (k-1)\mu_{k-1}\right) \\
&= \mu_{k-1} + \frac{1}{k}\left(x_k - \mu_{k-1}\right)
\end{aligned}
$$

Incremental Monte Carlo Update:

- **Update $V(s)$ incrementally:**

$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}(G_t - V(S_t))$$

- **For non-stationary problems track the running mean**

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

INSA TOULOUSE

Recap:

- **Dynamic programming:**
  - Need to know the transition model $P(S_{t+1}|s_t, a_t)$
  - Benefits of Bellman equation $V_*(s) = \max_{a \in \mathcal{A}}(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s'))$
- **Monte Carlo Methods:**
  - No need to know $P(S_{t+1}|s_t, a_t)$
  - Need to go through the entire episode

Temporal differences

Monte Carlo Update: $\mathrm{V}\left(\mathrm{S_t}\right) \leftarrow \mathrm{V}\left(\mathrm{S_t}\right) + \alpha\left(\mathrm{G_t} - \mathrm{V}\left(\mathrm{S_t}\right)\right)$

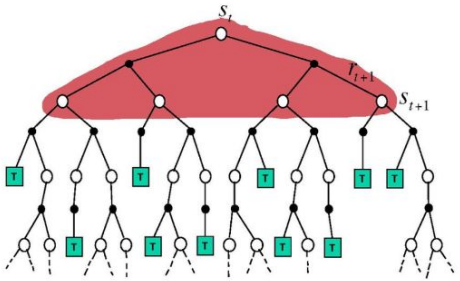
Dynamic Programming Update: $\mathrm{V}\left(\mathrm{S_t}\right) \leftarrow \mathrm{E}_\pi\left(\mathrm{R_{t+1}} + \gamma \mathrm{V}\left(\mathrm{S_{t+1}}\right)\right)$

Temporal Differences Update: $\mathrm{V}\left(\mathrm{S_t}\right) \leftarrow \mathrm{V}\left(\mathrm{S_t}\right) + \alpha\left(\mathrm{R_{t+1}} + \gamma \mathrm{V}\left(\mathrm{S_{t+1}}\right) - \mathrm{V}\left(\mathrm{S_t}\right)\right)$

Temporal differences

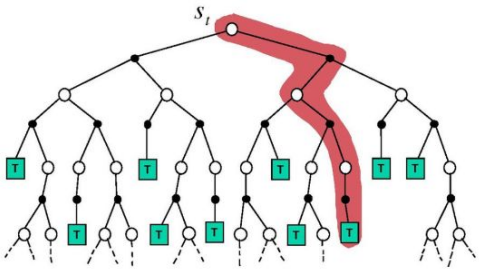
Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right]$$



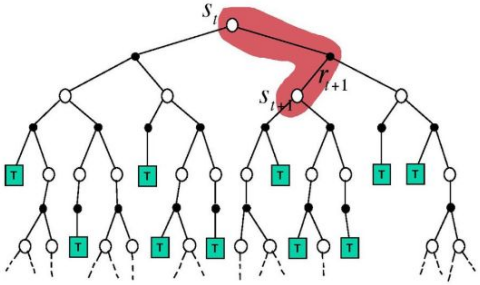
Monte-Carlo Backup

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



Temporal-Difference Backup

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$



INSA TOULOUSE

Temporal differences

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

INSA TOULOUSE

Temporal differences

Sarsa: An on-policy TD control algorithm

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

INSA TOULOUSE

Temporal differences

Q-learning: An off-policy TD control algorithm

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

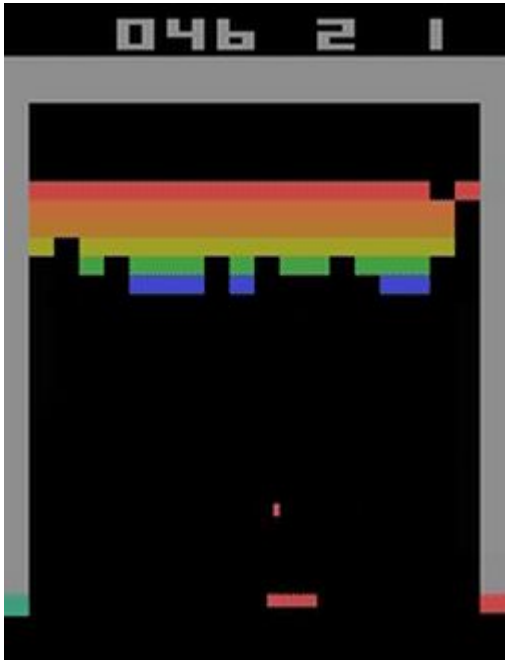
Q-Table:

- $\gamma = 0.9$
- = +10
- = -10

State	←	→
1	-10	5,9
2	5,3	6,56
3	5,9	7,29
4	6,56	8,1
5	7,29	9
6	8,1	+10

Temporal Differences

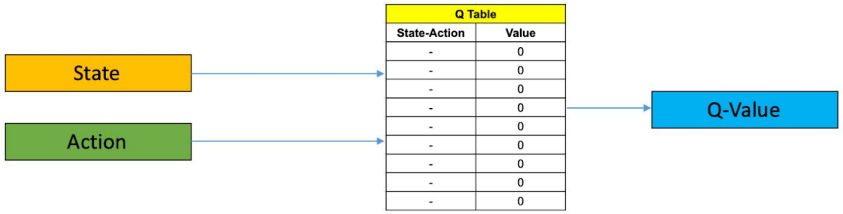
- No need to know $P(S_{t+1}|s_t, a_t)$
- Online learning
- Not adapted to large State spaces:
  - Chess: 10^50 states
  - Go: 10^170 states
  - Atari: 240x160x3 pixels



INSA TOULOUSE

DQN

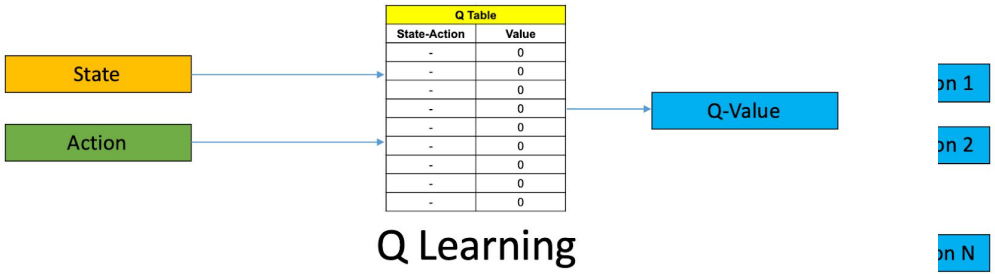
$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q\left(S', a'\right) - Q(S,A) \right)$$



Q Learning

DQN

$$Q_\theta(S,A) \leftarrow Q_\theta(S,A) + \alpha \left(R + \gamma \max_{a'} Q_\theta\left(S', a'\right) - Q_\theta(S,A) \right)$$



Q Learning

Deep Q Learning

INSA TOULOUSE

DQN

$$Q_\theta(S,A) \leftarrow Q_\theta(S,A) + \alpha \left(R + \gamma \max_{a'} Q_\theta\left(S', a'\right) - Q_\theta(S,A) \right)$$

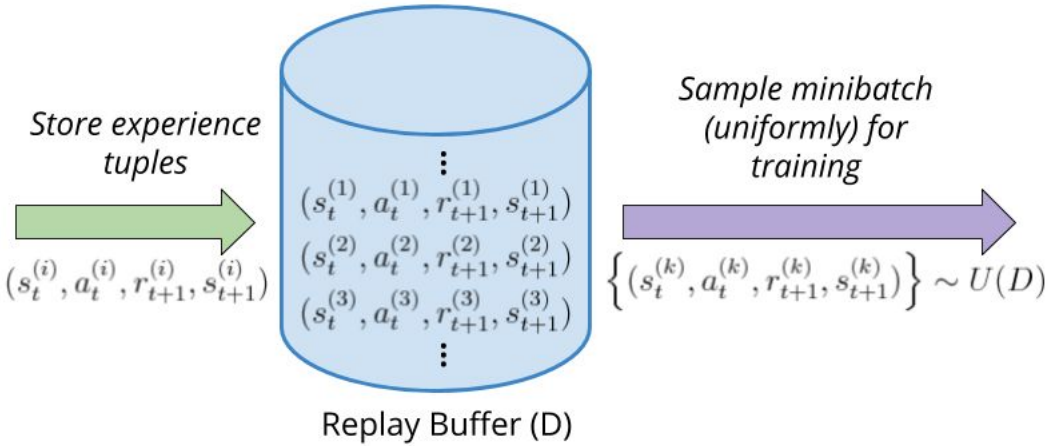
- **Network training:**

$$L_i\left(\theta_i\right) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q\left(s,a;\theta_i\right)\right)^2\right]$$

$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q\left(s', a'; \theta_{i-1}\right) \mid s, a\right]$$

Experience replay

- **Problem: correlated data**



INSA TOULOUSE

DQN

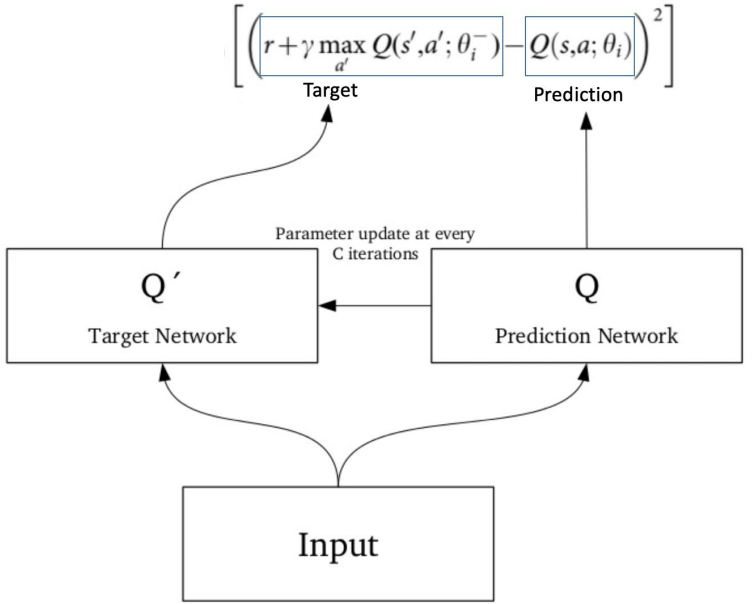
Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$
  end for
end for

Target Network

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$$

$$y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s',a';\theta_{i-1}) \mid s,a\right]$$



DQN with target network

Algorithm 1: deep Q-learning with experience replay.

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode = 1, M **do**

  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

  For t = 1,T **do**

    With probability ε select a random action a_t

    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

    Execute action a_t in emulator and observe reward r_t and image x_{t+1}

    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

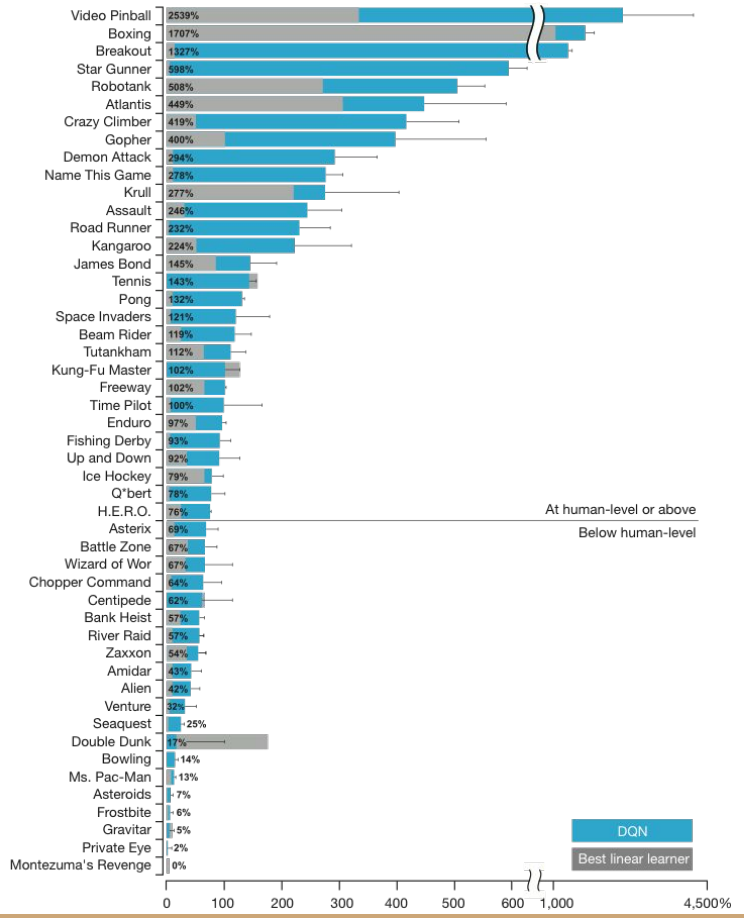
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

    Every C steps reset $\hat{Q} = Q$

  End For

End For

DQN from vizual inputs



References

Reinforcement Learning an introduction by Richard S. Sutton & Andrew G. Barto:
https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

David Sylver's class:
https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver

Olivier Pietquin 's class at RLSS:
https://rlss.inria.fr/files/2019/07/intro_rl_pietquin.pdf

Olivier Sigaud's videos:
https://www.youtube.com/c/OlivierSigaud/playlists

Playing Atari with Deep Reinforcement Learning: V Mnih & al.
https://arxiv.org/abs/1312.5602

Human level control through deep reinforcement learning: V Mnih & al.
https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf