Introduction to Reinforcement Learning

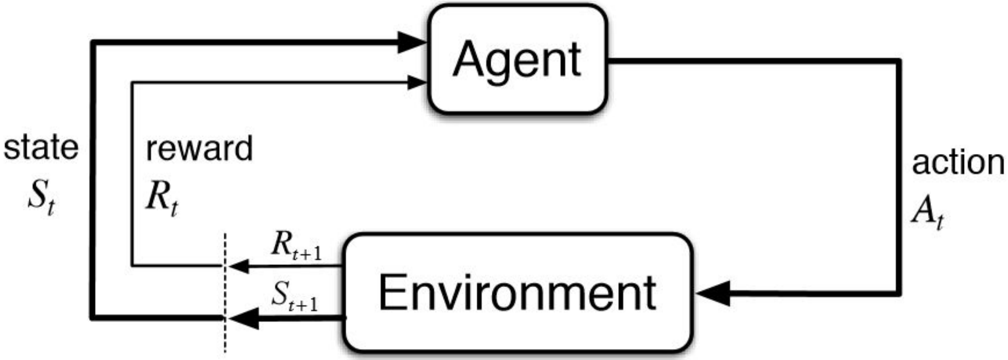
About me

- ➔ Higher National School of Computer Science (ESI) Alumni.
- ➔ Masters student at Sorbonne University.
- ➔ Intern at MLIA-Sorbonne working on Unsupervised Reinforcement Learning.

Introduction

What is Reinforcement Learning?

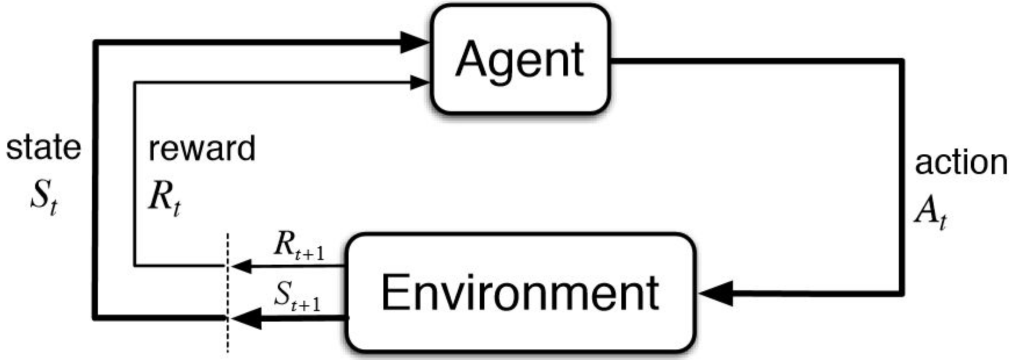
Reinforcement Learning (RL) is a machine learning approach where an **agent** learns by interacting with its **environment** through **actions**, receiving **rewards** for its actions, and aims to maximize **cumulative rewards** over time.

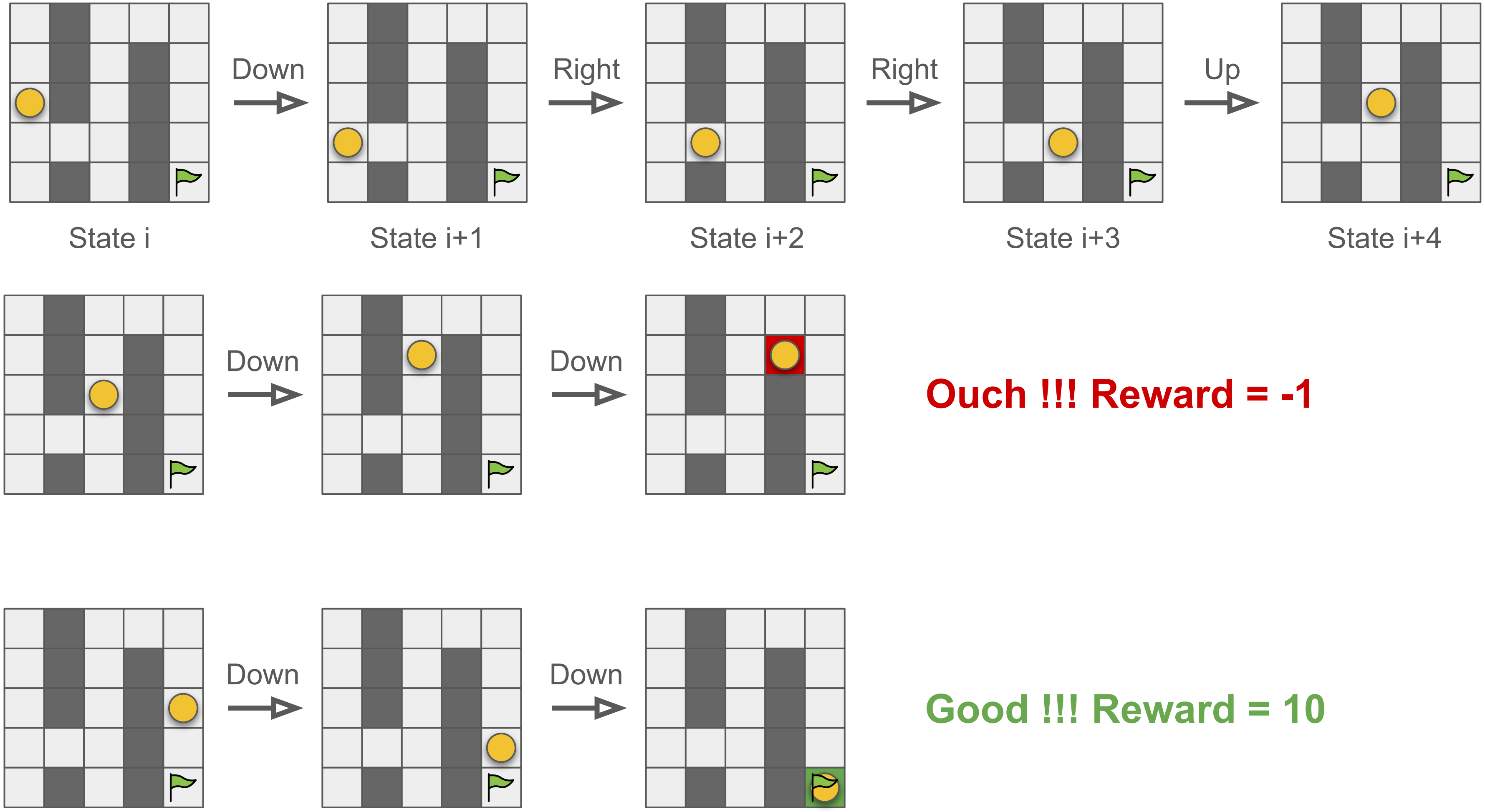


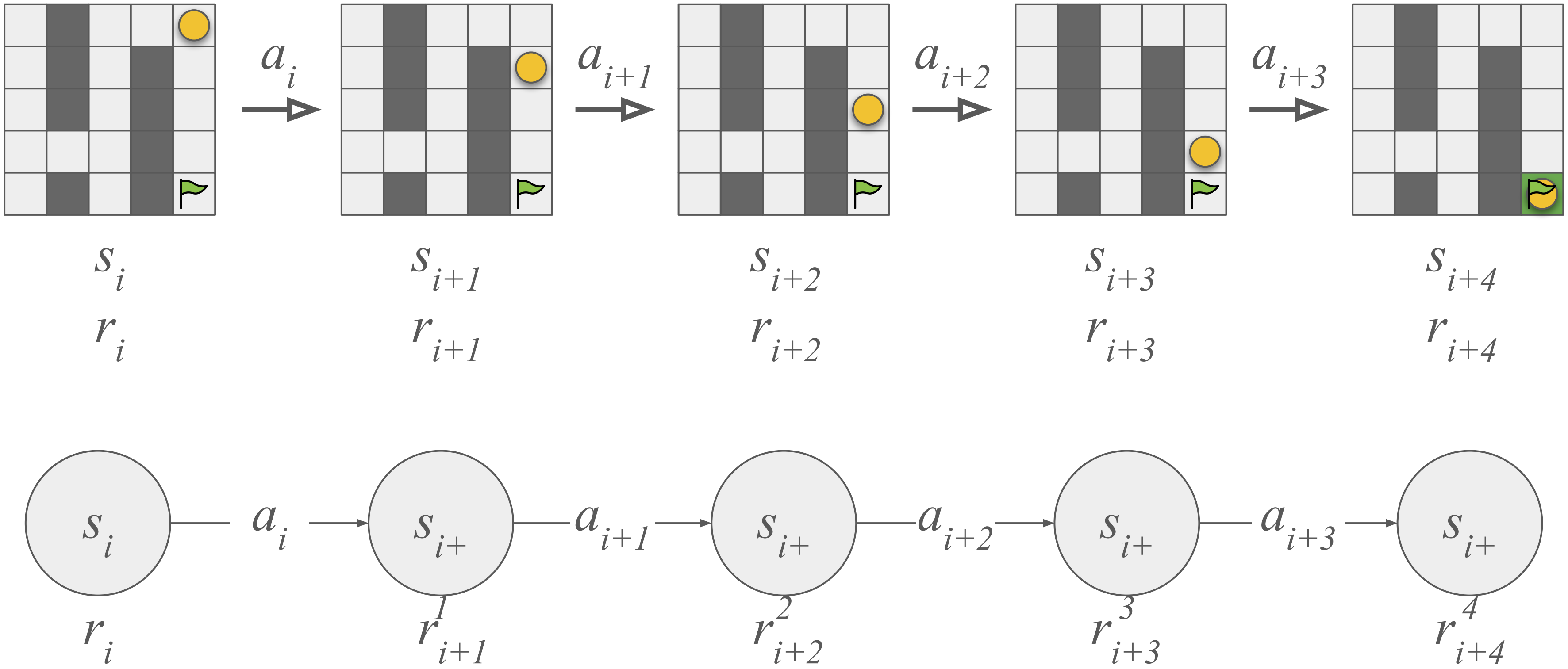
What is Reinforcement Learning?

What is Reinforcement Learning?

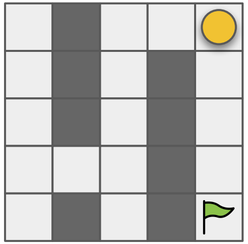
Agent	Environement	Action
Reward	State	Policy




Down
Right
Right
Up
State i
State i+1
State i+2
State i+3
State i+4
Down
Down
Ouch !!! Reward = -1
Down
Down
Good !!! Reward = 10



a_i
a_{i+1}
a_{i+2}
a_{i+3}
s_i
r_i
s_{i+1}
r_{i+1}
s_{i+2}
r_{i+2}
s_{i+3}
r_{i+3}
s_{i+4}
r_{i+4}
s_i
a_i
s_{i+}
a_{i+1}
s_{i+}
a_{i+2}
s_{i+}
a_{i+3}
s_{i+}
r_i
1
r_{i+1}
2
r_{i+2}
3
r_{i+3}
4
r_{i+4}

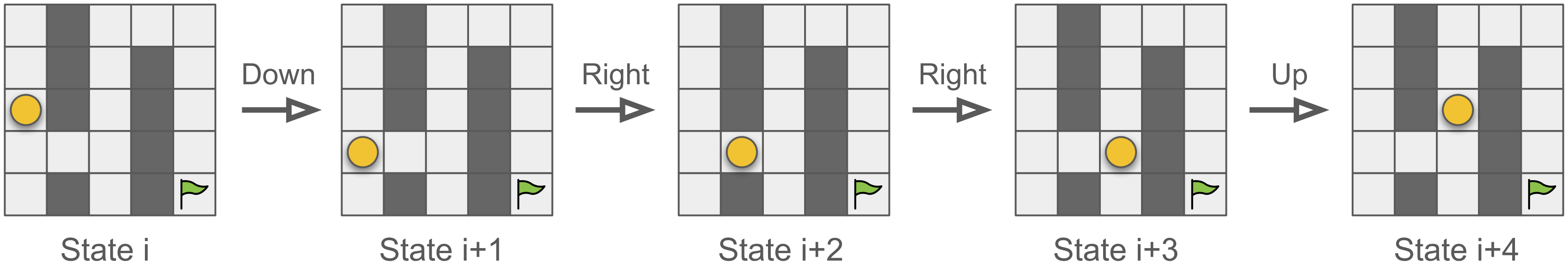


State	Action
(0, 0)	Down
(0, 2)	Right
(0, 3)	Right
(0, 4)	Down
(1, 0)	Down
(1, 2)	Up
…	…
(3, 4)	Down
(4, 0)	Up
(4, 2)	Up
(4, 4)	Down

State / Action

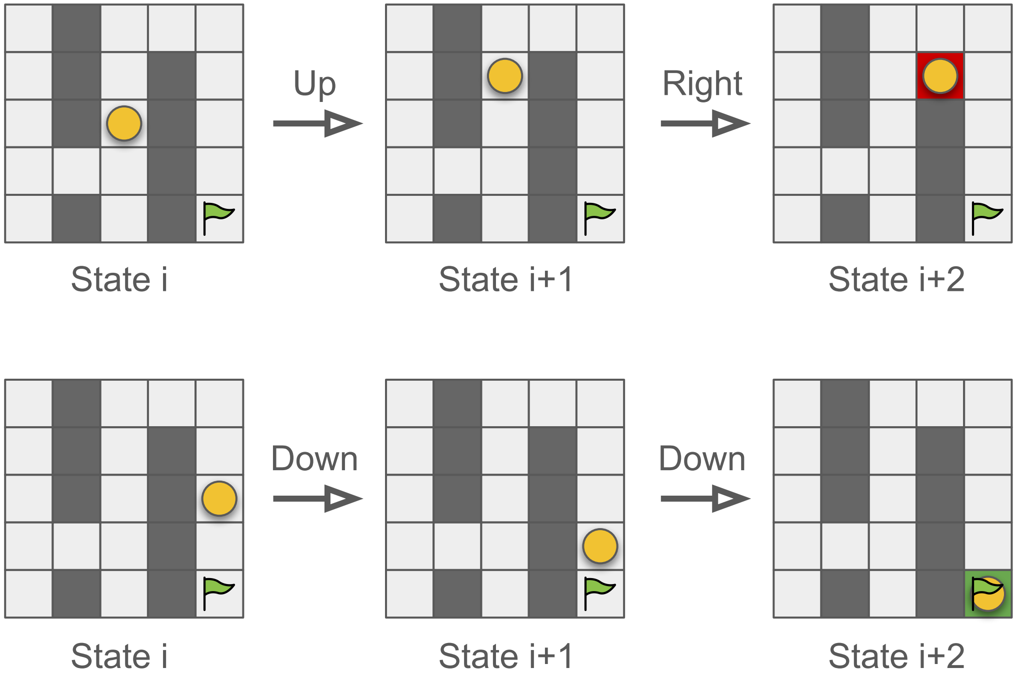
- **State:** A state represents the current situation or configuration of the environment that the agent is in at a given time.

- **Action:** An action is a decision made by the agent that affects the state of the environment, leading to transitions between states.



Reward

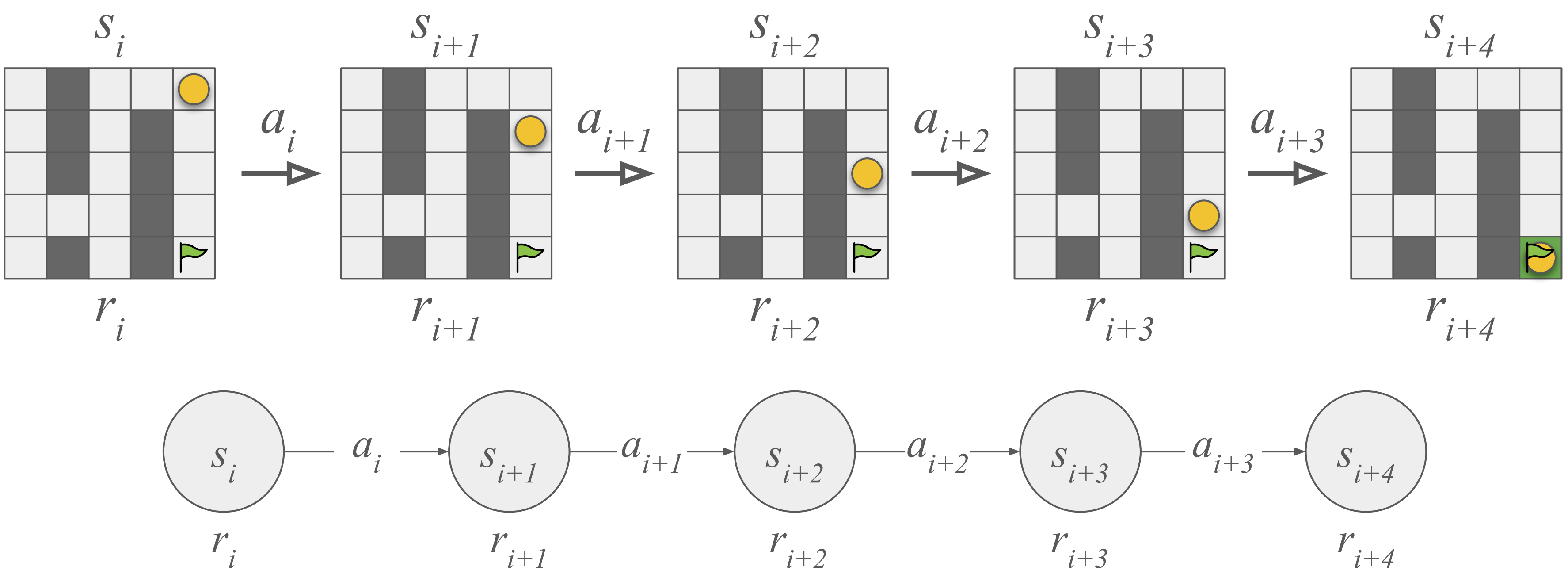
Reward: A feedback signal indicating how good or bad an action is, guiding the agent to maximize long-term gains.



Ouch !!! Reward = -1

Good !!! Reward = 10

Reward



s_i
s_{i+1}
s_{i+2}
s_{i+3}
s_{i+4}
a_i
a_{i+1}
a_{i+2}
a_{i+3}
r_i
r_{i+1}
r_{i+2}
r_{i+3}
r_{i+4}
s_i
a_i
s_{i+1}
a_{i+1}
s_{i+2}
a_{i+2}
s_{i+3}
a_{i+3}
s_{i+4}
r_i
r_{i+1}
r_{i+2}
r_{i+3}
r_{i+4}

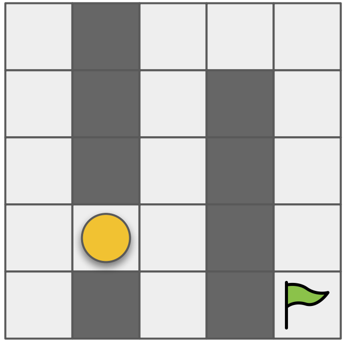
Policy

Policy: it represents the strategy that the agent follow. It maps states to actions:

$$\pi : S \rightarrow A$$

$$\pi(s_i) = a$$

State	Action
(0, 0)	Down
(0, 2)	Right
(0, 3)	Right
(0, 4)	Down
(1, 0)	Down
(1, 2)	Up
...	...
(3, 4)	Down
(4, 0)	Up
(4, 2)	Up
(4, 4)	Down



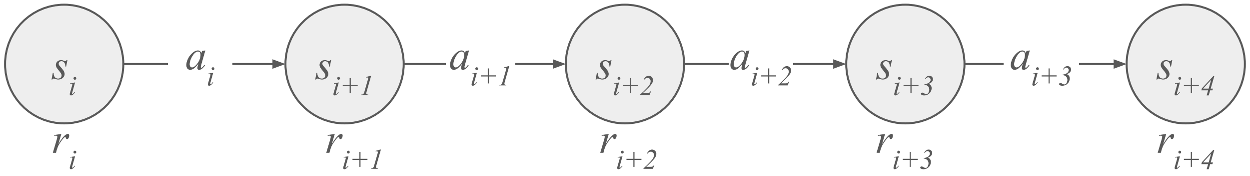
How to learn a policy

$$V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right]$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

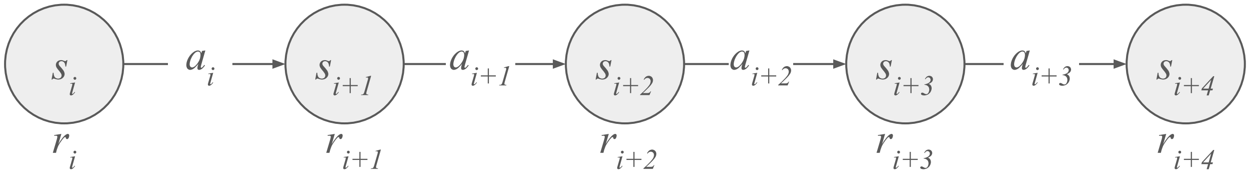
$$Q(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a \right]$$

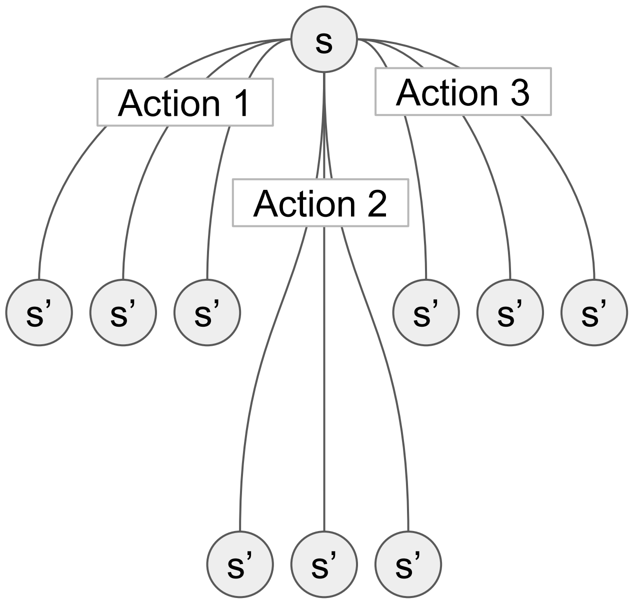
$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma V_\pi(s') \right]$$

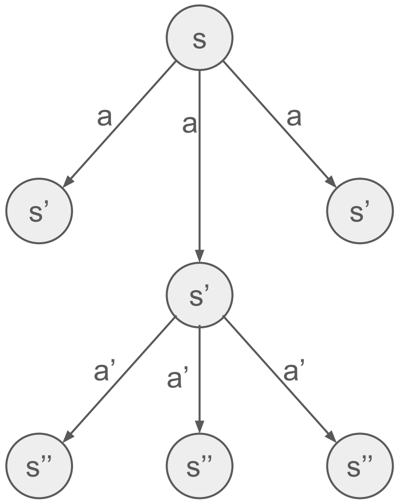


$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V_\pi(s')\right]$$

$$\pi'(s) = \arg\max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V(s')\right]$$



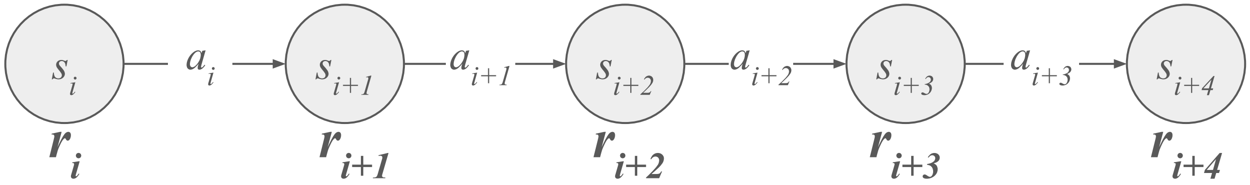

s
Action 1
Action 2
Action 3
s'
s'
s'
s'
s'
s'
s'
s'
s'


s
a
a
a
s'
s'
s'
a'
a'
a'
s''
s''
s''

Cumulative Reward

The goal in RL is not simply maximizing the reward, but rather maximizing the cumulative reward denoted G_t:

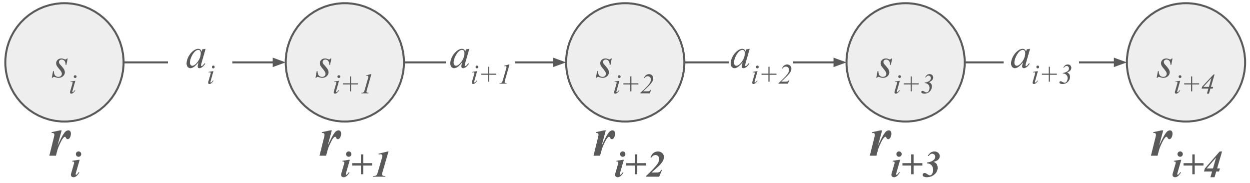
$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$



State Value

Represents the expected cumulative reward starting from a given state s and following a policy **π**. The formula for the state value function is:

$$V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right]$$



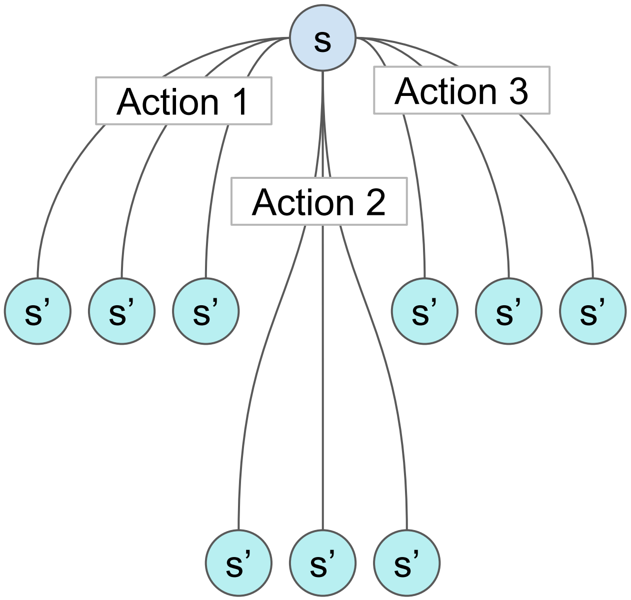
Dynamic Programing

Dynamic Programing

Collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a MDP.

Transition probablity:

$$P(s' \mid s, a)$$



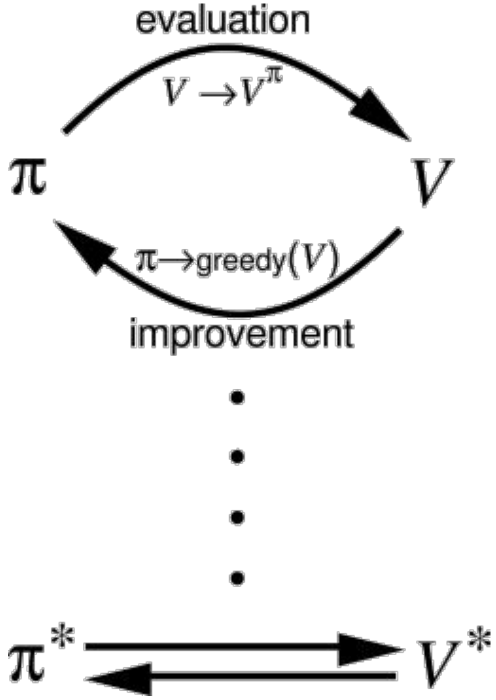
Policy Iteration

1. **Initialize:** Start with an arbitrary policy **π** and value function **V(s)**.

2. **Policy Evaluation:** Update **V(s)** for all states until convergence based on **π**.

3. **Policy Improvement:** Update the policy to **π'** by choosing actions that maximize expected returns based on **V(s)**.

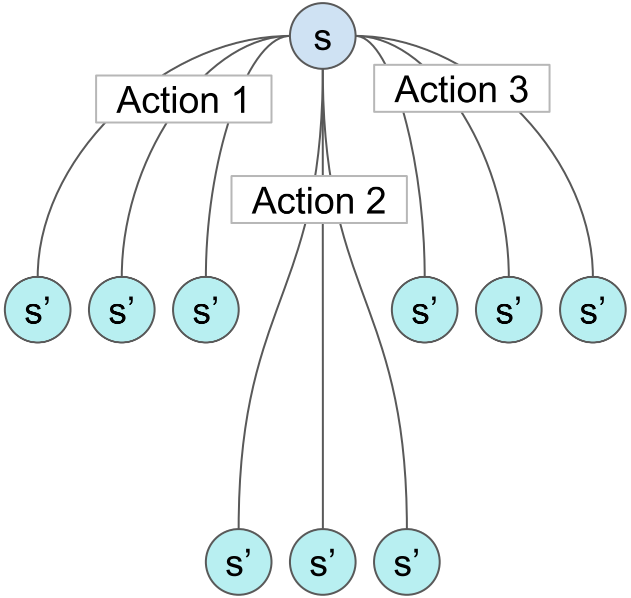
4. **Repeat:** Continue until **π' = π**.



Policy Evaluation

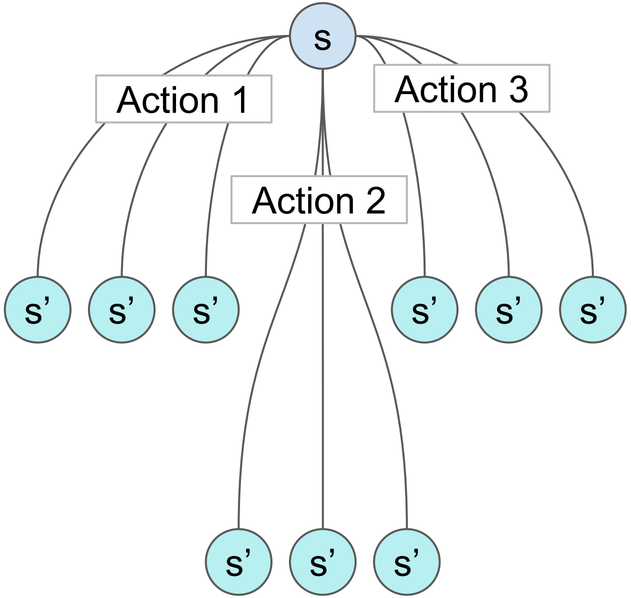
The policy evaluation formula calculates the value of a state under **a given policy π**. It is based on the Bellman expectation equation:

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma V_\pi(s') \right]$$



Policy Improvement

The process of updating a current policy **π** to a new policy **π'** that maximizes expected rewards based on the value of states **V** under the current policy.



$$\pi'(s) = \arg\max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V(s')\right]$$

Temporal Differences

Temporal Difference error

> **Ideally, we have:**

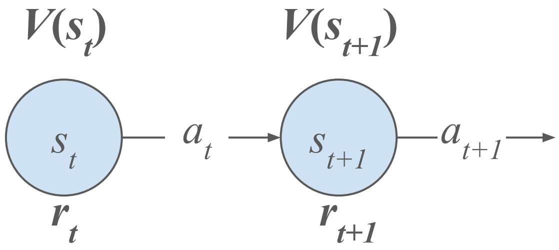
$$V(s_t) = r_{t+1} + \gamma V(s_{t+1})$$

> **The approximation error is:**

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

> **should decrease the error:**

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t$$

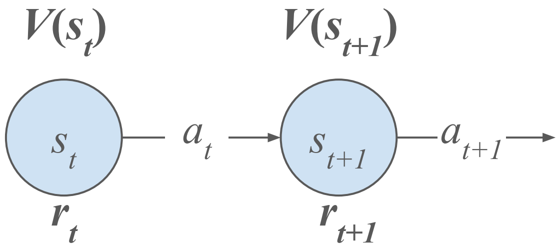


$$V(s_t) \leftarrow V(s_t) + \alpha[\, r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \,]$$

TD Prediction

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)) \quad [6.2]$$

Inputs: π - the policy to be evaluated
Params: step size $\alpha \in]0,1]$
Initialize: $V(s) \in \mathbb{R}$ for all $s \in \mathcal{S}^+$ except for
V(terminal)=0
foreach *episode* **do**
  Initialize S
  foreach *step of episode - until S is terminal* **do**
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha(R + \gamma V(S') - V(S))$
    $S \leftarrow S'$
  end
end

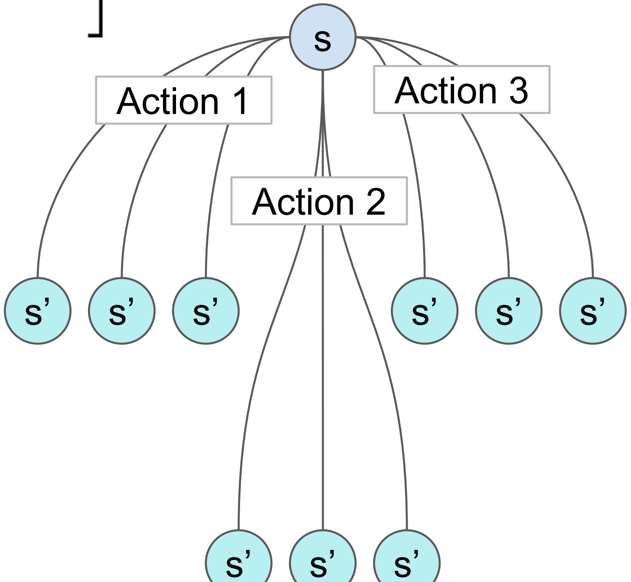


State Action Value

Rrepresents the expected cumulative reward an agent can obtain by taking action **a** in state **s** and subsequently following a specified policy.

$$Q(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a \right]$$

$$V(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right]$$



SARSA:

Params: step size $\alpha \in]0,1]$, small $\epsilon > 0$
Initialize $Q(s,a)$ for all $s \in \mathcal{S}^+$ and $a \in \mathcal{A}(s)$,
arbitrarily except that $Q(terminal - state, \cdot) = 0$

foreach *episode* **do**
  Initialize S
  Choose A from S using policy derived from Q (e.g. ϵ-greedy)
  foreach *step of episode - until S is terminal* **do**
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g. ϵ-greedy)
    $Q(S,A) \leftarrow$
    $Q(S,A) + \alpha\,[R + \gamma \boxed{Q(S',A')} - Q(S,A)]$
    $S \leftarrow S'$
    $A \leftarrow A'$
  end
end

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

Q learning

Q learning:

SARSA:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma Q(s',a') - Q(s,a)\right]$$

Q learning:

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

Q learning:

Params: step size $\alpha \in]0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$ for all $s \in \mathcal{S}^+$ and $a \in \mathcal{A}(s)$,
arbitrarily except that $Q(terminal - state, \cdot) = 0$

foreach *episode* **do**
  Initialize S
  foreach *step of episode - until S is terminal* **do**
    Choose A from S using policy derived from Q (e.g. ϵ-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \boxed{\max_a Q(S', a)} - Q(S, A)\right]$
    $S \leftarrow S'$
  end
end

Conclusion

- ➔ What is Reinforcement Learning?
- ➔ What are the components of Reinforcement Learning ?
- ➔ State Value and State-Action Value.

- ➔ Policy Evaluation
- ➔ Policy Improvement.
- ➔ Policy Iteration.

- ➔ Temporal Differences.
- ➔ SARSA.
- ➔ Q-learning.

Thank you!

jn_bendib@esi.dz