Porto Business School

/ University of Porto

BPI Grupo CaixaBank

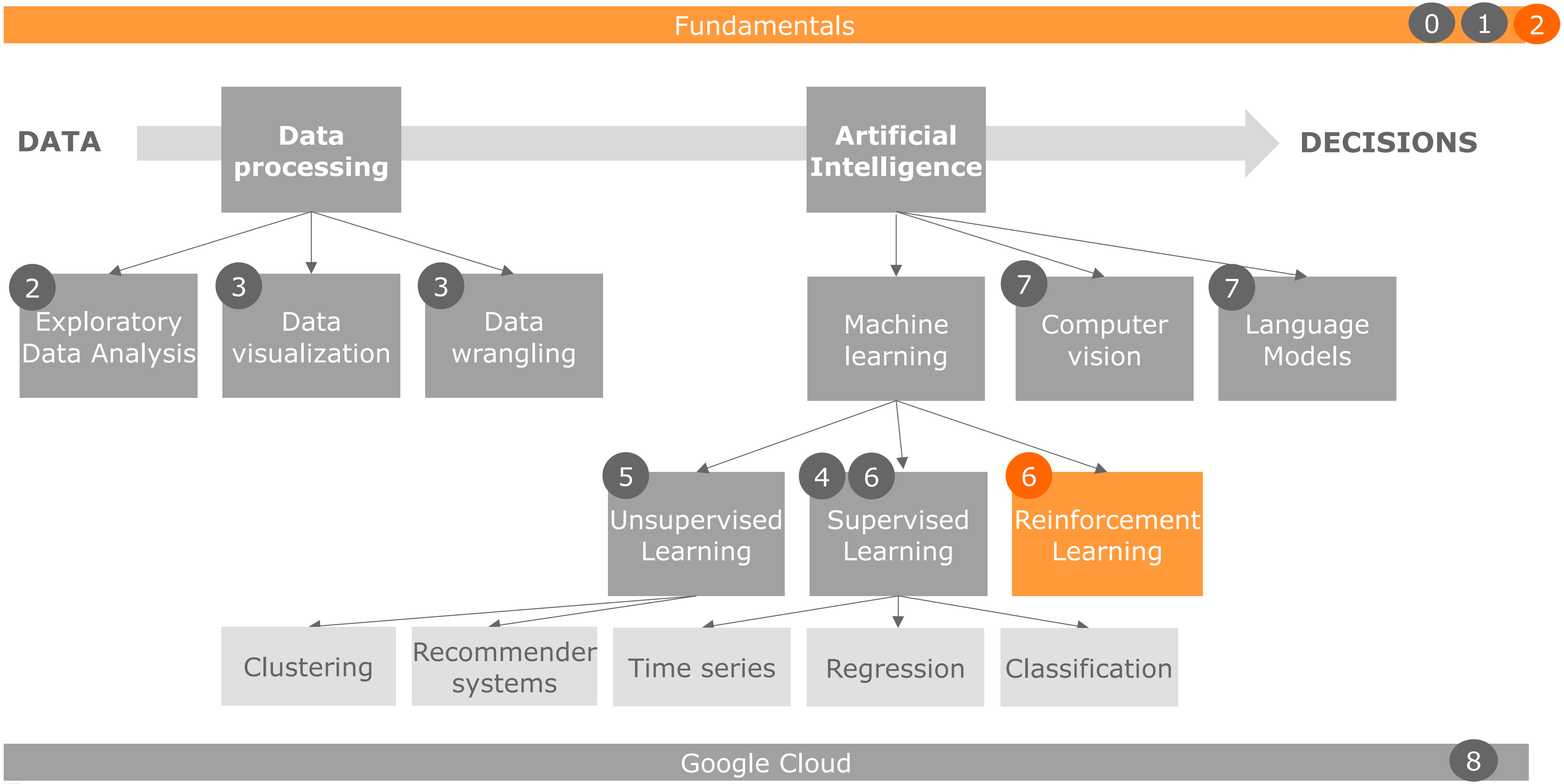
# Advanced Learning models

## Reinforcement Learning

Tiago Cunha | Lisboa, 26/10/2023

ACCREDITATIONS MEMBERSHIPS RANKINGS

# Academia DTI



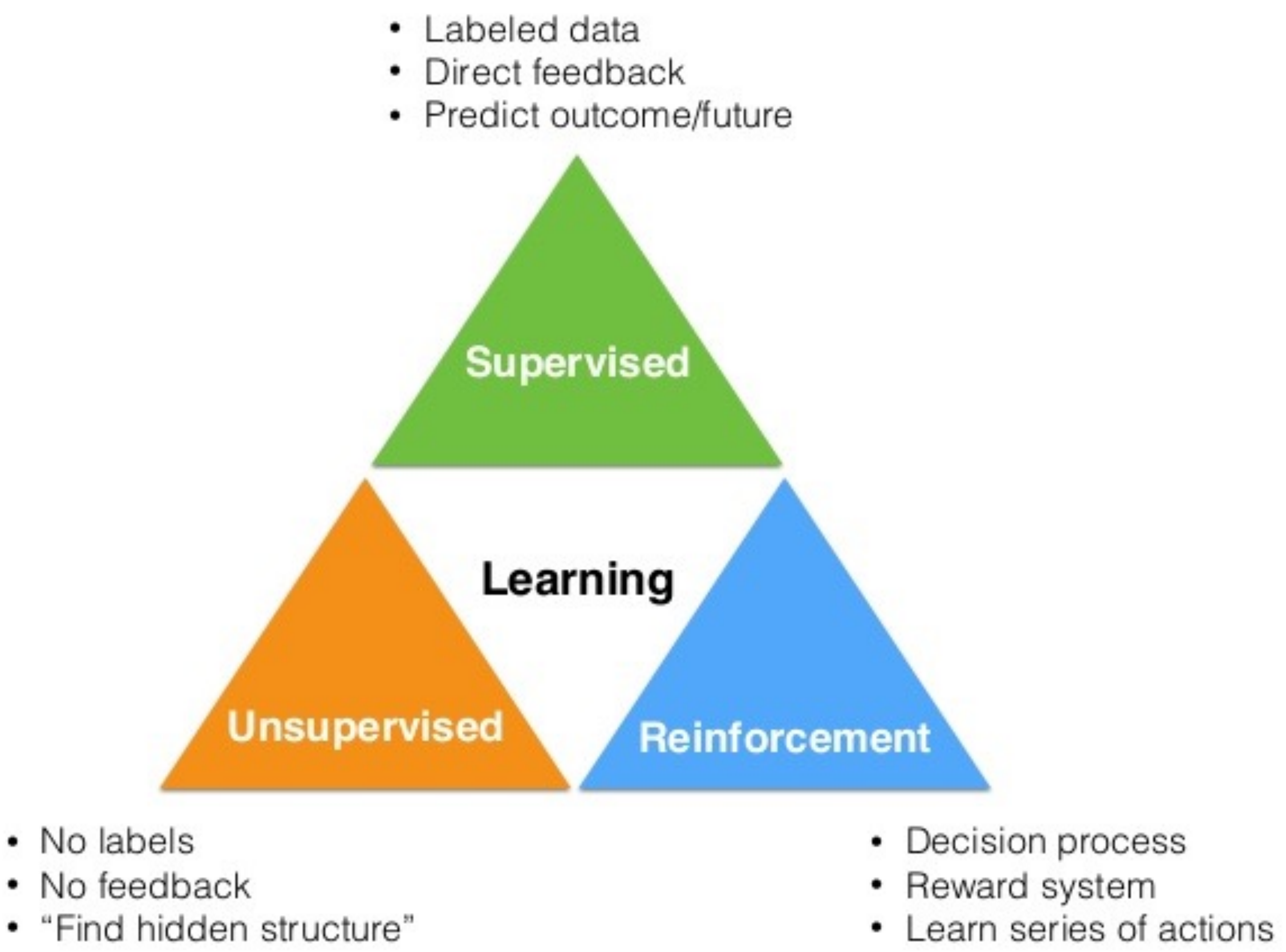
# Advanced Learning models

## Agenda

| | |
|---|---|
| 09h30 – 11h00 | Introduction to Reinforcement Learning |
| | Multi-Armed bandits |
| | Exercises |
| 11h00 – 11h15 | **Break** |
| 11h15 – 13h00 | Q-learning |
| | SARSA |
| | Exercises |
| | Deep RL in the wild |

# Introduction to Reinforcement Learning

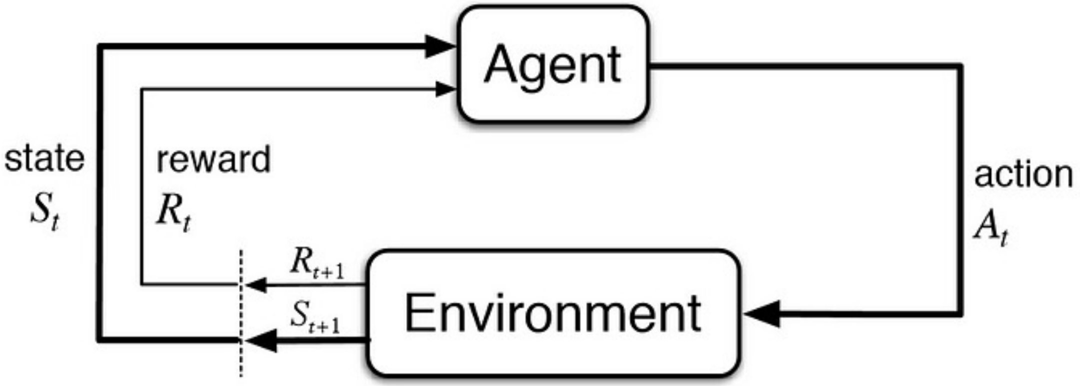
# Reinforcement Learning



# Reinforcement Learning

## Action-Reward feedback loop

- Environment: world
- Agent: learner
- State: information about the agent in the environment
- Reward: feedback from world
- Policy: learning agent's way of behaving at a given time
- Value: future reward

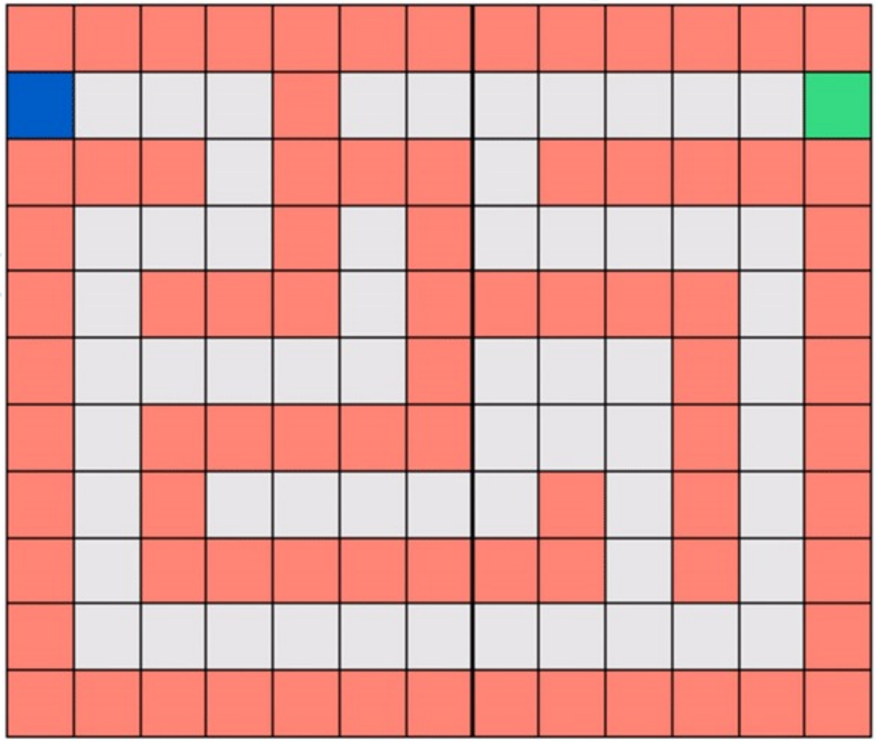



Goal is to maximize total cumulative reward, i.e. Return

$$\text{Return} = \sum_{t=1}^{T} reward_t$$

# Reinforcement Learning

## States, Actions and Policy



| State \ Action | left | up | right | down |
|---|---|---|---|---|
| 2,1 | | | | |
| 2,2 | | | | |
| 2,3 | | | | |
| 2,4 | | | | |
| 3,4 | | | | |
| 4,4 | | | | |
| 4,3 | | | | |
| 4,2 | | | | |
| 5,2 | | | | |
| 6,2 | | | | |
| 6,3 | | | | |
| 7,2 | | | | |
| … | | | | |

# Reinforcement Learning

## Reward function

- Defines good and bad events
- Numerical value indicating desirability of that transaction
- Short-term
- Primary

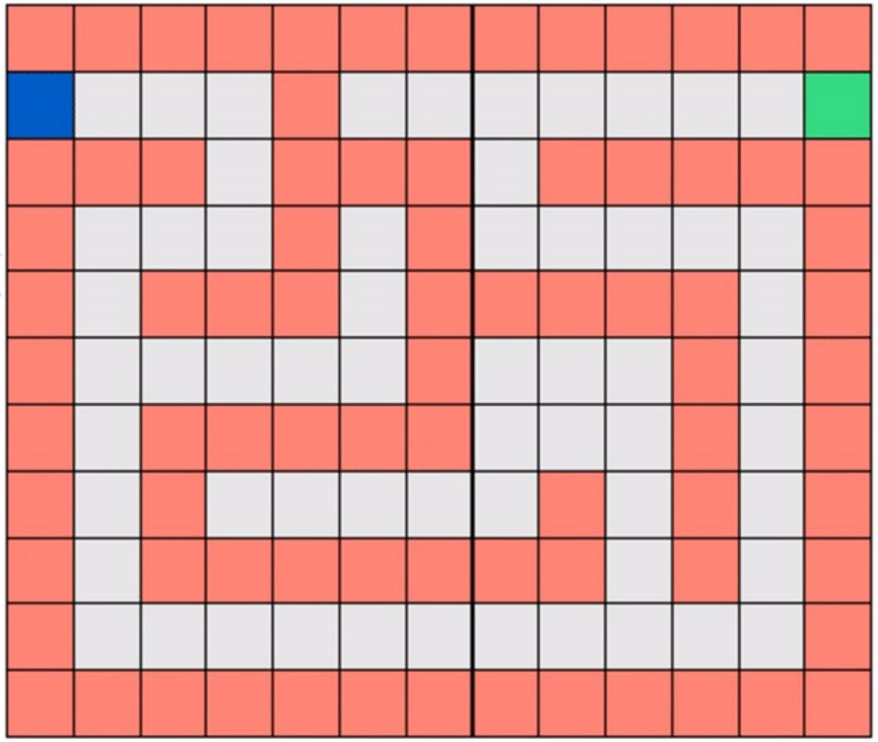
## Value function

- Defines policy's worth
- Total amount of reward accumulated in the future
- Long-term
- Secondary – it needs rewards to exist

RL focuses on actions that maximize Value, rather than Reward. Why?

# Reinforcement Learning

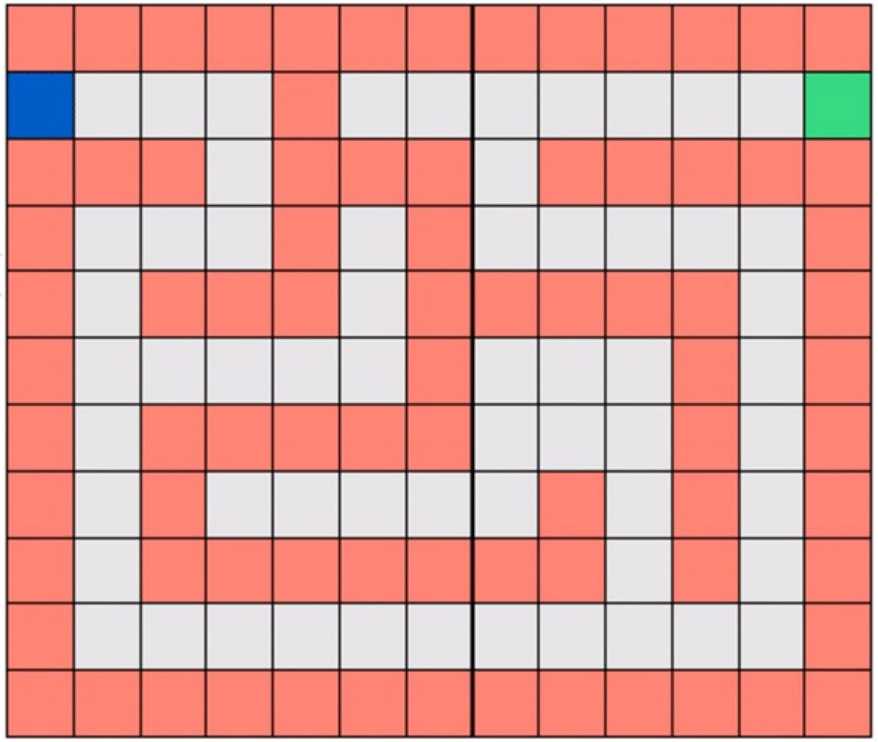
## Rewards



| State \ Action | left | up | right | down |
|---|---|---|---|---|
| 2,1 | 0 | 0 | 1 | 0 |
| 2,2 | -1 | 0 | 1 | 0 |
| 2,3 | -1 | 0 | 1 | 0 |
| 2,4 | -1 | 0 | 0 | 1 |
| 3,4 | 0 | -1 | 0 | 1 |
| 4,4 | 1 | -1 | 0 | 0 |
| 4,3 | 1 | 0 | -1 | 0 |
| 4,2 | 0 | 0 | -1 | 1 |
| 5,2 | 0 | -1 | 0 | 1 |
| 6,2 | 0 | -1 | -1 | 1 |
| 6,3 | 1 | 0 | -1 | 0 |
| 7,2 | 0 | -1 | 0 | 1 |
| … | | | | |

# Reinforcement Learning

## Values



| State \ Action | left | up | right | down |
|---|---|---|---|---|
| 2,1 | 0 | 0 | 200 | 0 |
| 2,2 | -100 | 0 | 180 | 0 |
| 2,3 | -100 | 0 | 150 | 0 |
| 2,4 | -100 | 0 | 0 | 130 |
| 3,4 | 0 | -100 | 0 | 120 |
| 4,4 | 110 | -100 | 0 | 0 |
| 4,3 | 100 | 0 | -100 | 0 |
| 4,2 | 0 | 0 | -100 | 100 |
| 5,2 | 0 | -100 | 0 | 100 |
| 6,2 | 0 | -100 | -300 | 200 |
| 6,3 | 100 | 0 | -400 | 0 |
| 7,2 | 0 | -100 | 0 | 100 |
| ... | | | | |

# Reinforcement Learning

## Episodic

- E.g.: game of chess
- It reaches an end
  - Finite amount of time
- Episodes are independent
  - Reset to initial state
- Return can be calculated more easily

$$\text{Return} = \sum_{t=1}^{T} reward_t$$

## Continuing

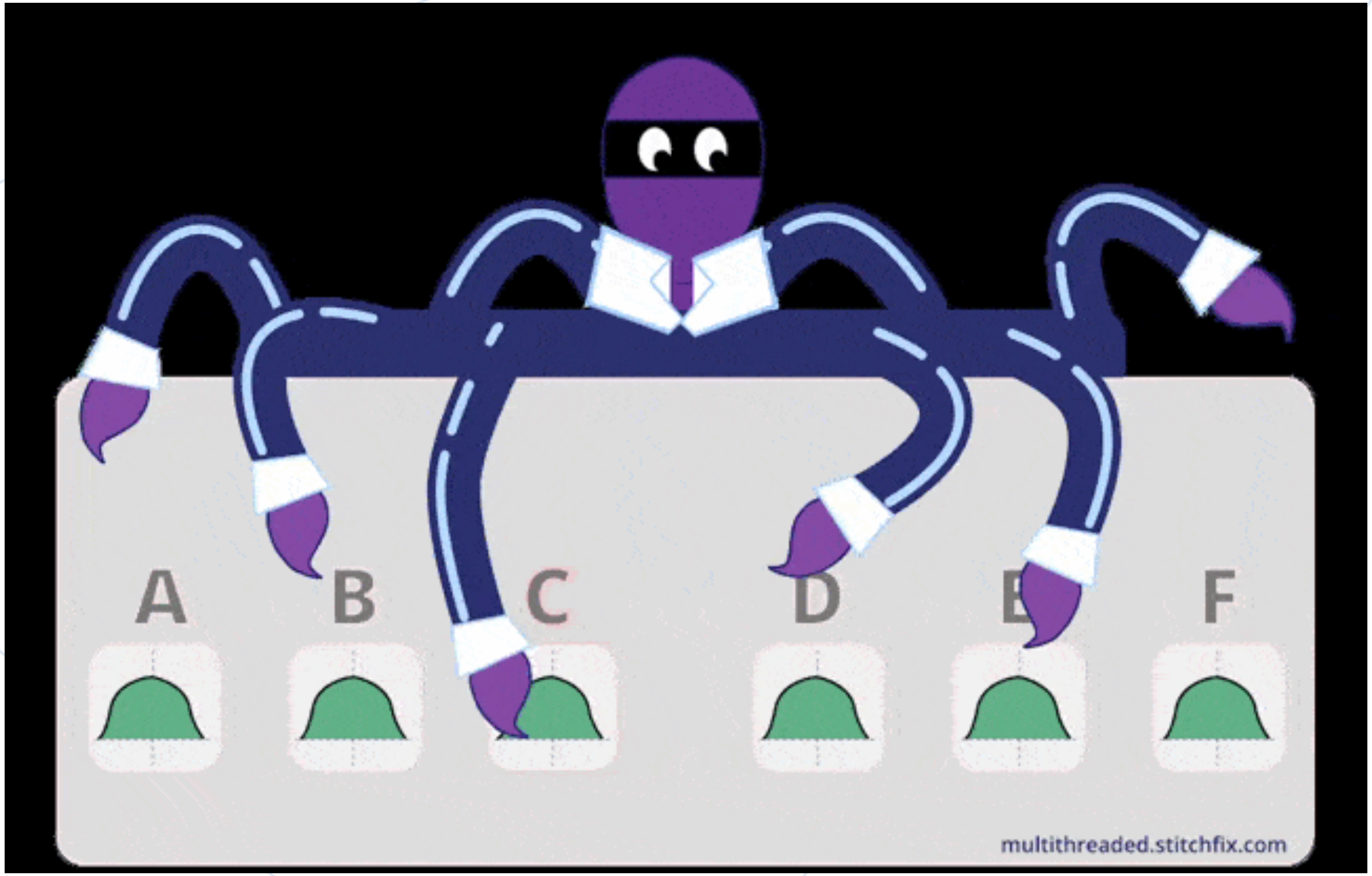
- E.g.: personal assistance robot
- Never reaches an end
  - Infinite amount of time
- There is no terminal state
- Discounting factor $\gamma$
  - Recent actions assigned more reward

$$\text{Return} = \sum_{t=1}^{T} \gamma^t \, reward_t$$

# Multi-Armed bandits

# Reinforcement Learning

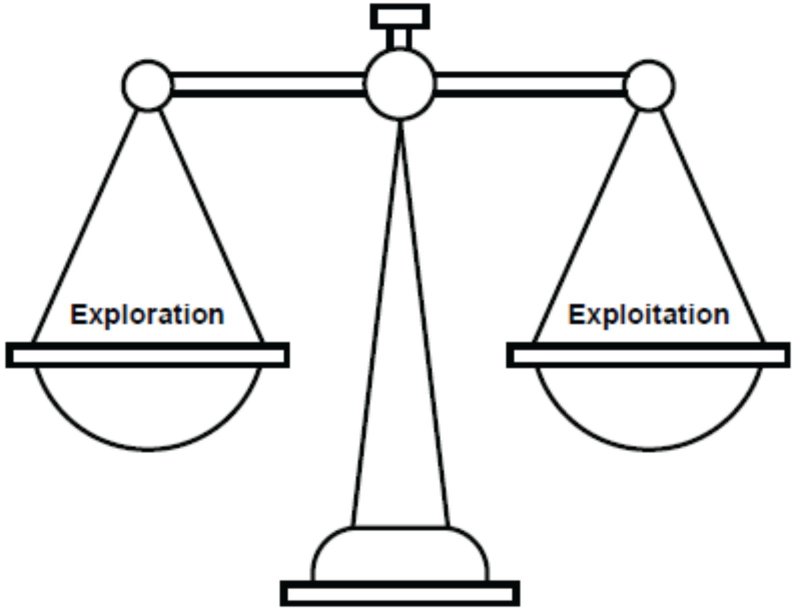
## Multi-Armed bandits



# Reinforcement Learning

## Exploration vs Exploitation

- Try other options
- Allows to find better options
- Optimizes for the long term




- Choose best option
- Best reward given current knowledge
- Optimizes for the short term

**MUST ITERATE BETWEEN BOTH OPTIONS**

# Reinforcement Learning

MAB algorithms

- Greedy vs ε-greedy
- Upper-Confidence Bound (UCB)
- Thompson Sampling

Greedy

- Value of arm is given by average reward

$$Q_k(a) = \frac{1}{k}(r_1 + r_2 + r_3 + \ldots + r_k)$$

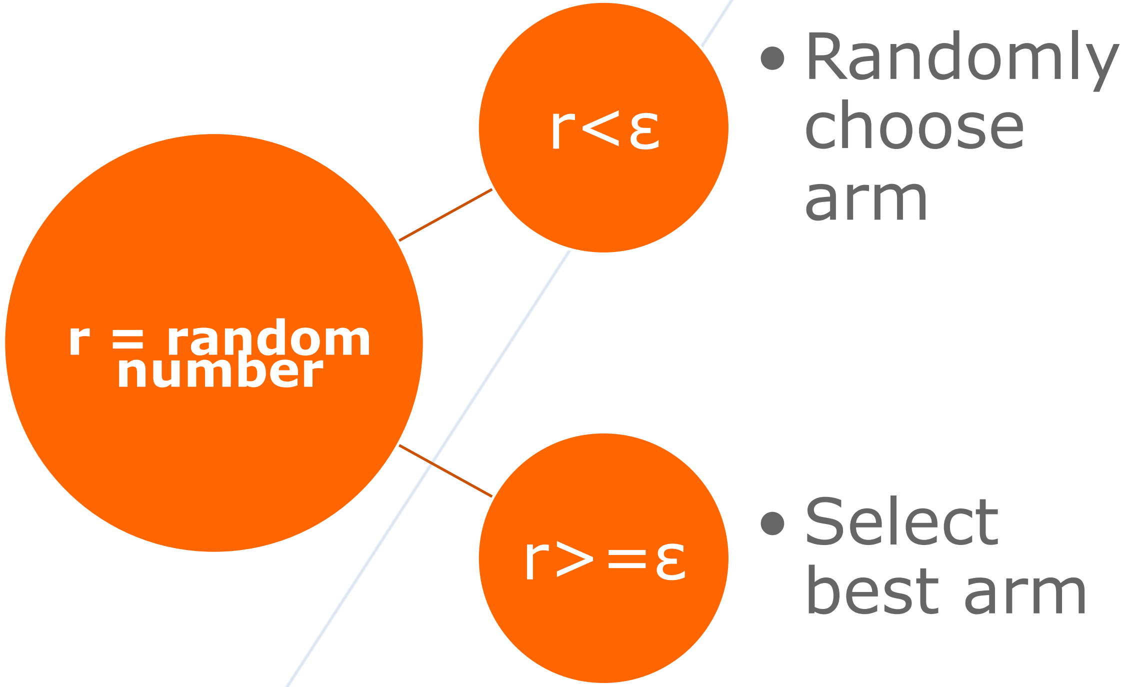
- Best arm is simply choosing argmax

$$a_{greedy} = argmax_a \, Q_k(a)$$

# Reinforcement Learning

## ε-greedy

- Simple method to balance exploration and exploitation
- ε is a hyper-parameter
  - usually a small number
  - Must tune on specific use case
- Greedy method does not trade-off - always chooses highest paying arm
  - Cannot properly explore



# Reinforcement Learning

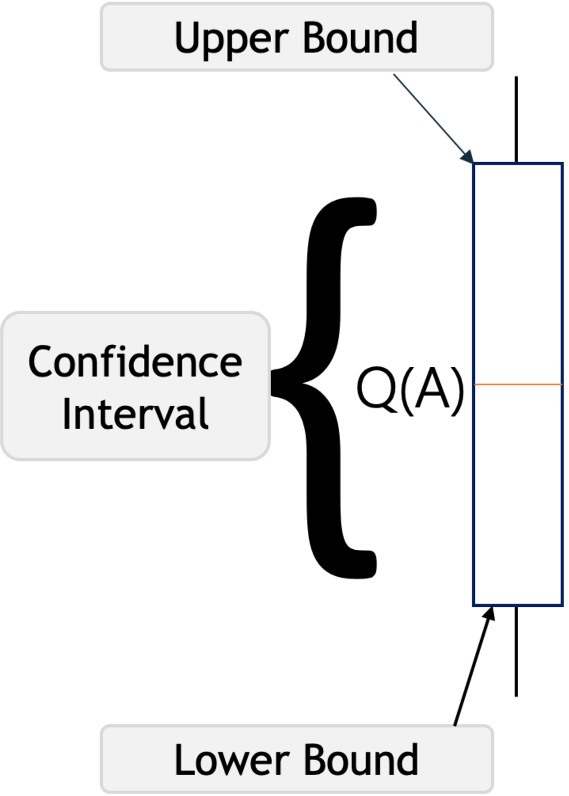
## UCB (Upper confidence bound)

- Optimism in the face of uncertainty
  - Exponentially decays as number of pulls increase
  - Boosts arms that have been explored less

$$UCB(a,t) = \sqrt{\frac{2\,log(t)}{k}}$$

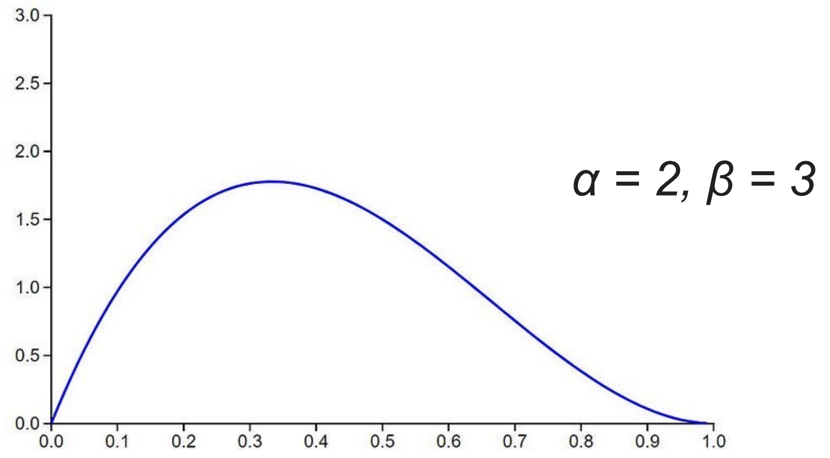
- Selection of best arm

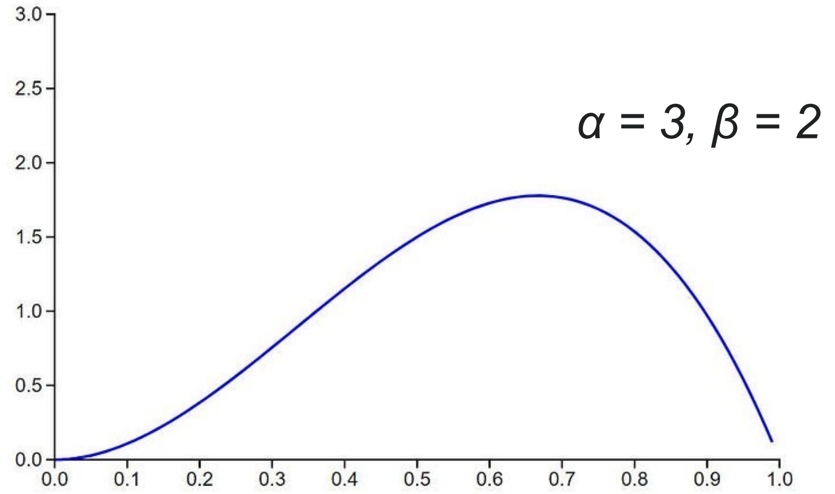
$$a_{greedy} = argmax_a\ (\underbrace{Q_k(a)}_{\text{EXPLOIT}} + \underbrace{UCB\ (a,t)}_{\text{EXPLORE}})$$



## Thompson Sampling

- Each arm is modeled as a beta distribution
  - $\alpha$ : # successful events
  - $\beta$: # unsuccessful events
- Each arm value is sampled from distribution
  - Sampling allows exploration-exploitation
- Selected arm is the one with highest value



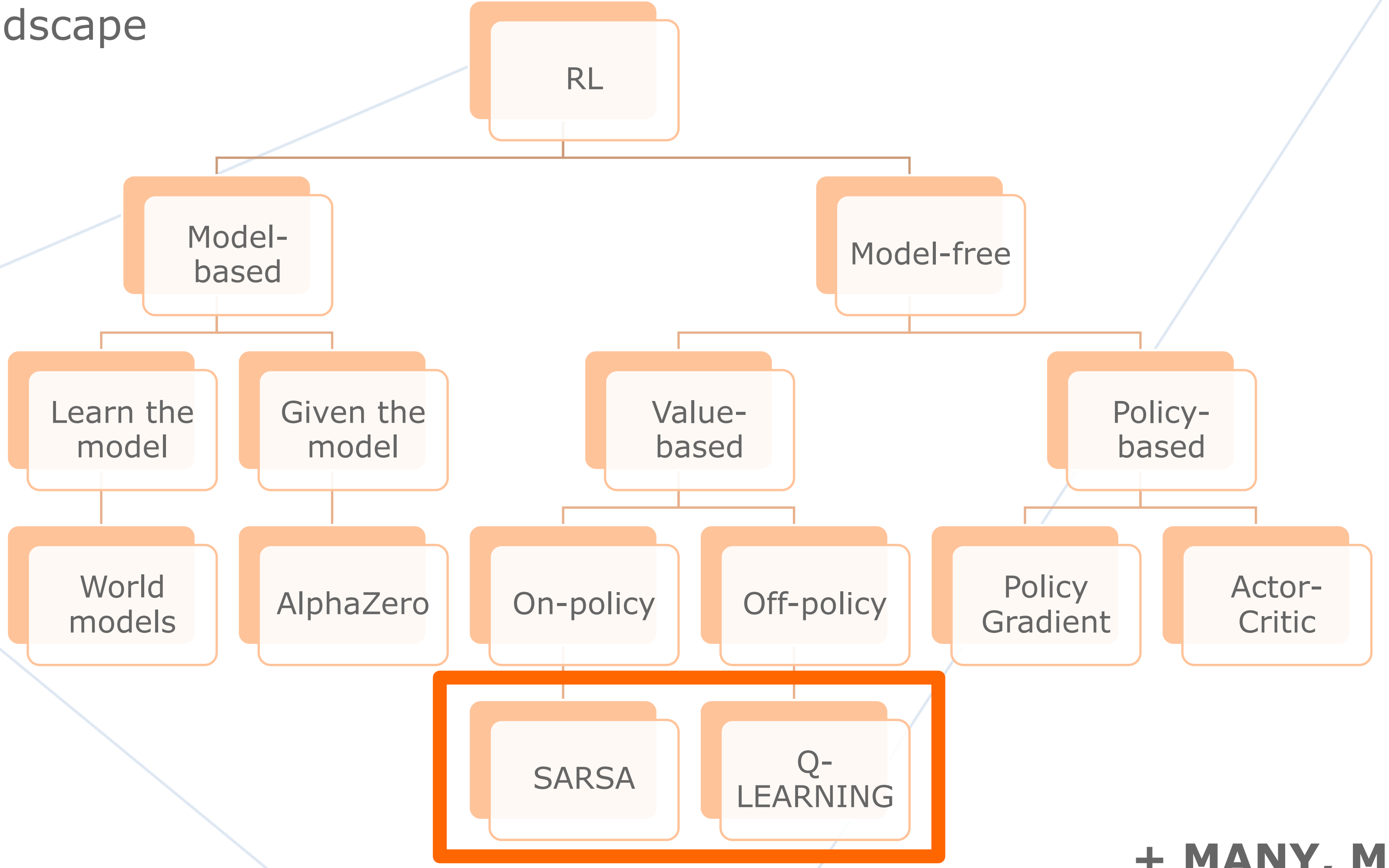
# Reinforcement Learning

- Demo:
  - https://cse442-17f.github.io/LinUCB/
- 2 bandits example:
  - https://colab.research.google.com/drive/18nRNu-fl4u0t6d0t7P9Zo_ZEmqgnLg8V
- 10 bandits example
  - Exercise:
    - https://colab.research.google.com/drive/1dQXVN7OF8IwSeEdKsZTB_v3mCPLTElGt
  - Solution:
    - https://colab.research.google.com/drive/1p4ufAC3puDT7M_zU11k9apZCym2NuLL9

# Full Reinforcement Learning methods

# Reinforcement Learning

Summary RL landscape




**+ MANY, MANY OTHERS**

Porto Business School / University of Porto

## Markov Decision Process (MDP)

- Stochastic decision-making process
  - Used in sequential decisions over time

- MDPs evaluate actions considering current environment
  - Probability of next state is the same whether dependency is of current state or all previous states

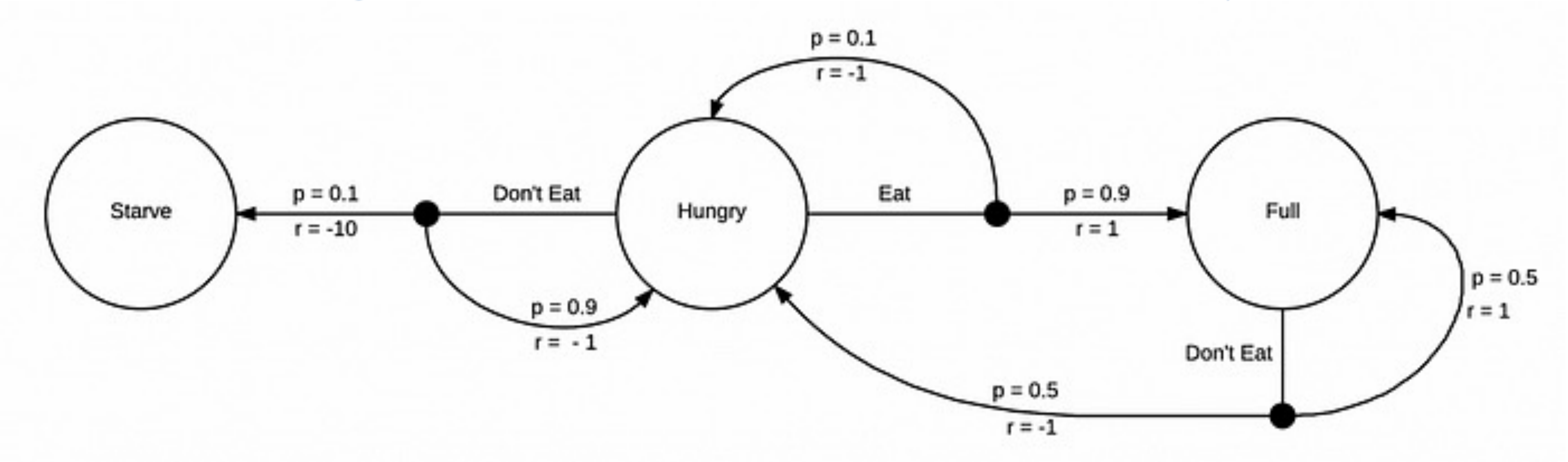
| | MDP=⟨S,A,T,R,γ⟩ |
|---|---|
| S | States |
| A | Actions |
| T | Transition probabilities |
| R | Rewards |
| γ | Discount factor |

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

BPI Grupo CaixaBank

Porto Business School / University of Porto

## Markov Decision Process (MDP) example



## Bellman Equations

- State-Value Function
  - Depends only on current state

$$V(s) = E[\underbrace{R_{t+1}}_{\text{EXPECTED VALUE OF IMMEDIATE REWARD}} + \underbrace{\gamma\, V(S_{t+1})}_{\text{DISCOUNTED VALUE OF NEXT STATE}} | S_t = s]$$

- Action-Value Function
  - Depends on current state and action

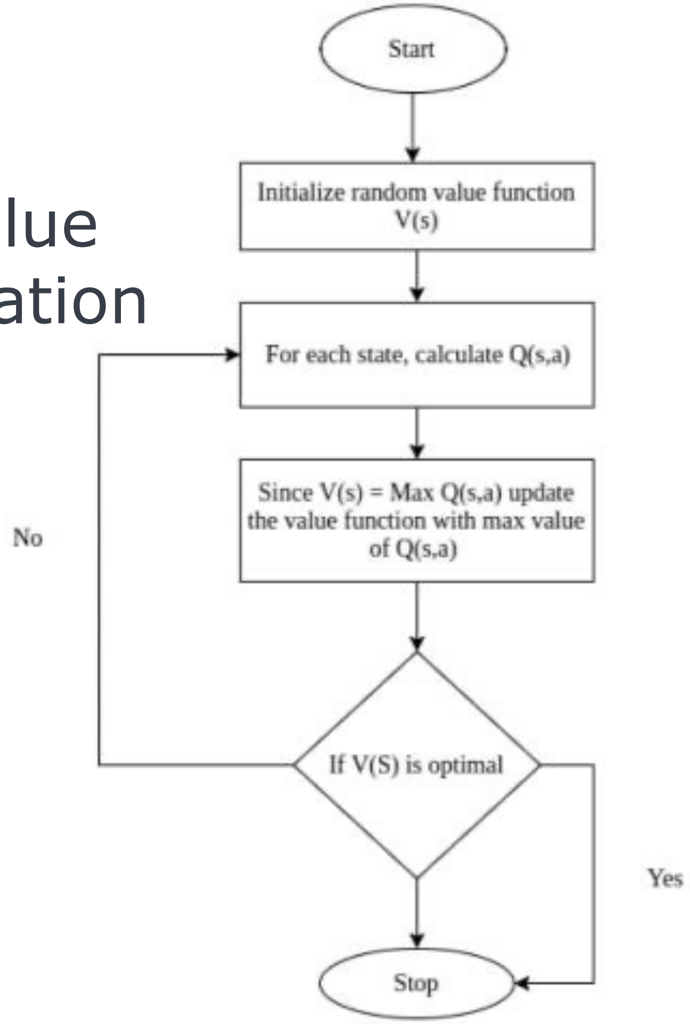
$$Q(s,a) = E[\underbrace{R_{t+1}}_{\text{EXPECTED VALUE OF IMMEDIATE REWARD}} + \underbrace{\gamma\, Q(S_{t+1}, A_{t+1})}_{\text{DISCOUNTED VALUE OF NEXT STATE-ACTION PAIR}} | S_t = s, A_t = a]$$

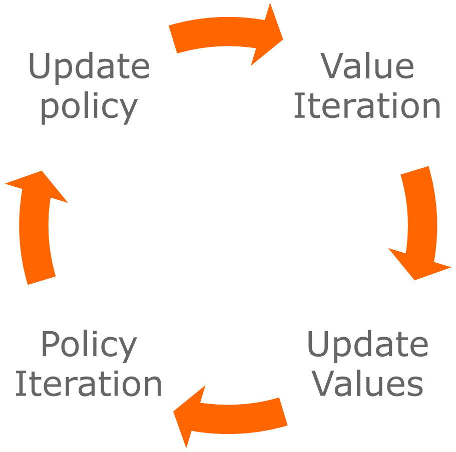
https://builtin.com/machine-learning/markov-decision-process

# Reinforcement Learning

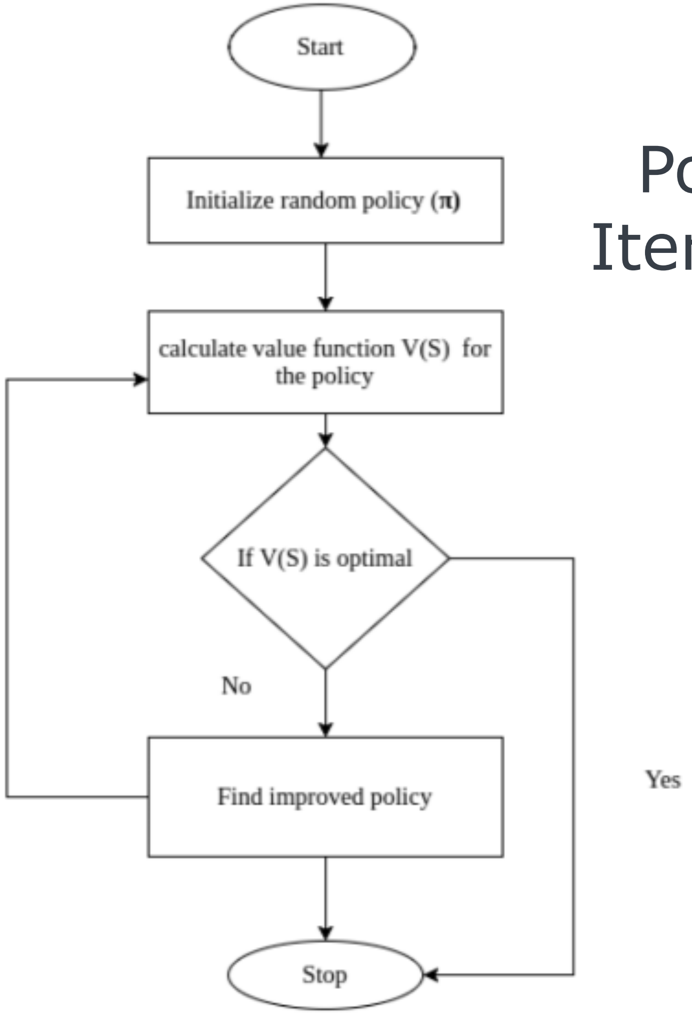
## Dynamic Programming

Value Iteration




Policy Iteration



# Reinforcement Learning

## Q-Learning vs SARSA

- State-Value Function

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

- Action-Value Function
  - SARSA

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \boxed{\gamma Q(S_{t+1}, a_{t+1})} - Q(S_t, A_t)$$

  - Q-Learning

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \boxed{\gamma\, max_a Q(S_{t+1}, a)} - Q(S_t, A_t)$$

# Reinforcement learning

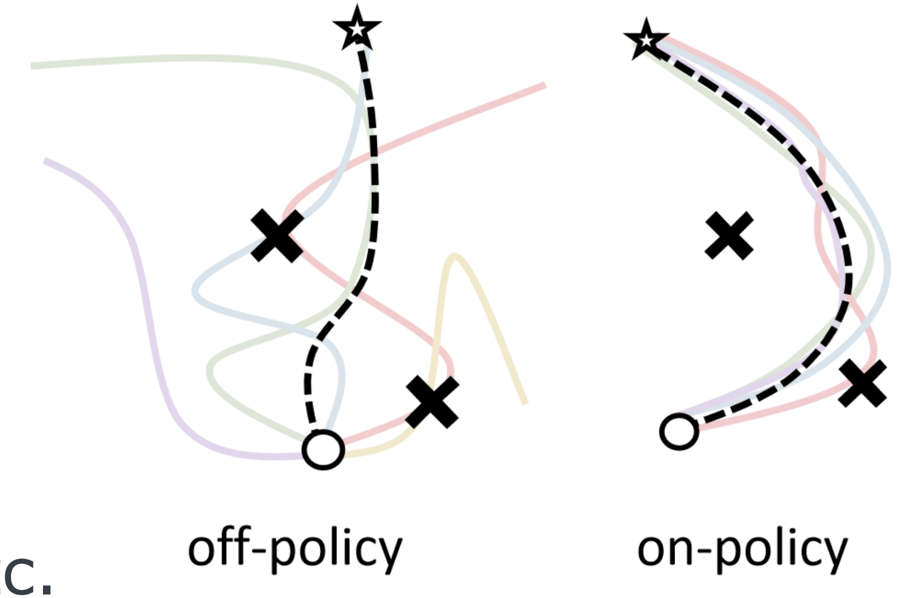
## On-policy vs off-policy

### On-policy

- $Q(s, a)$ function is learned from actions using the current policy
  - E.g. SARSA

### Off-policy

- $Q(s, a)$ function is learned from taking different actions, e.g. random, expert, etc.
  - Q-Learning



# Reinforcement Learning

| | On-Policy | Off-Policy |
|---|---|---|
| **Advantages** | • Learns safer strategy<br>• Often converges faster<br>• Often has better online performance | • More likely to find optimal policy<br>• Less likely to get stuck in local minimum<br>• Can utilize experience replay<br>• Data can be collected via various method |
| **Disadvantages** | • May become trapped in local minima<br>• Less likely to find optimal policy<br>• Data must be collected following current policy | • Policy learned may not be as safe<br>• May not perform as well online |

# Reinforcement Learning

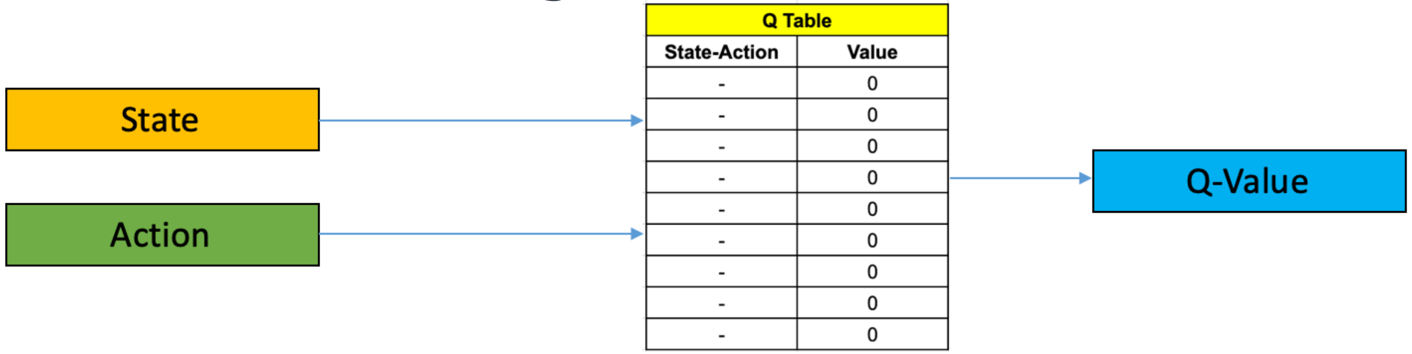
- Exercise:
  - https://colab.research.google.com/drive/12iWBgnfBSR0YHbxopq27TxwFzYrjwXS2

- Solution:
  - https://colab.research.google.com/drive/1m_HzekTsR5ZDzq_x2RLv8w4999KuJzCi

# Reinforcement Learning in the wild

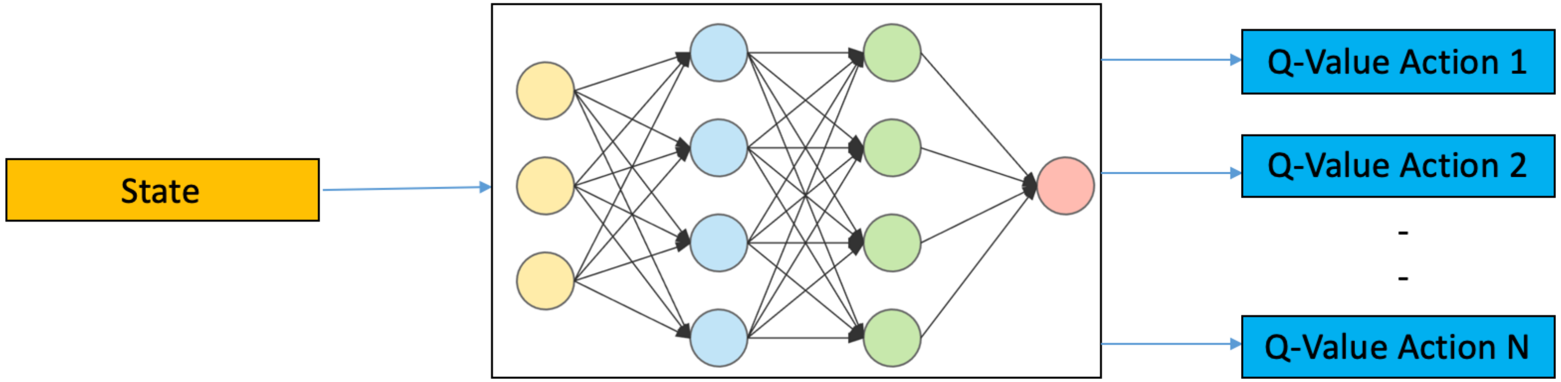
# Reinforcement Learning

## Deep Reinforcement Learning




| Q Table | |
|---|---|
| State-Action | Value |
| - | 0 |
| - | 0 |
| - | 0 |
| - | 0 |
| - | 0 |
| - | 0 |
| - | 0 |
| - | 0 |
| - | 0 |

Q Learning




Deep Q Learning

# Reinforcement Learning

## Self Driving Cars - Simulator

Porto Business School / University of Porto

## Games - Mario Brothers


BPI Grupo CaixaBank

# Reinforcement Learning

## Drone Flight Race

## More resources

- Sutton and Barto – Reinforcement Learning: An introduction
  http://incompleteideas.net/book/the-book-2nd.html
- Reinforcement Learning Specialization in Coursera
  https://www.coursera.org/specializations/reinforcement-learning
- Reinforcement Learning Lecture Series by Google DeepMind
  https://www.deepmind.com/learning-resources/reinforcement-learning-lecture-series-2021
- Deep Reinforcement Learning from UC Berkeley
  https://rail.eecs.berkeley.edu/deeprlcourse/

# Wrap-up

- ✓ Conclusion #1 – Reinforcement Learning uses agents and their interaction with the world to learn policies on how to solve a task
- ✓ Conclusion #2 – Multi-Armed bandits are simple agents which can learn the value of each action, disregarding the state information
- ✓ Conclusion #3 – Almost all reinforcement Learning problems can be formulated via a Markov Decision Process
- ✓ Conclusion #4 – Q-learning is a generic and powerful method to learn the optimal policy in MPDs
- ✓ Conclusion #5 – Deep Reinforcement Learning allows to bring Reinforcement Learning to the next level, by leveraging the power of Deep Learning

Make change happen