KIOS Research and Innovation Center of Excellence

MSc on Intelligent Critical Infrastructure Systems

Machine Learning

Lecture 11 and 12

Marios Polycarpou

Director, KIOS Research and Innovation Center of Excellence

Professor, Electrical and Computer Engineering

University of Cyprus

University of Cyprus

Imperial College London

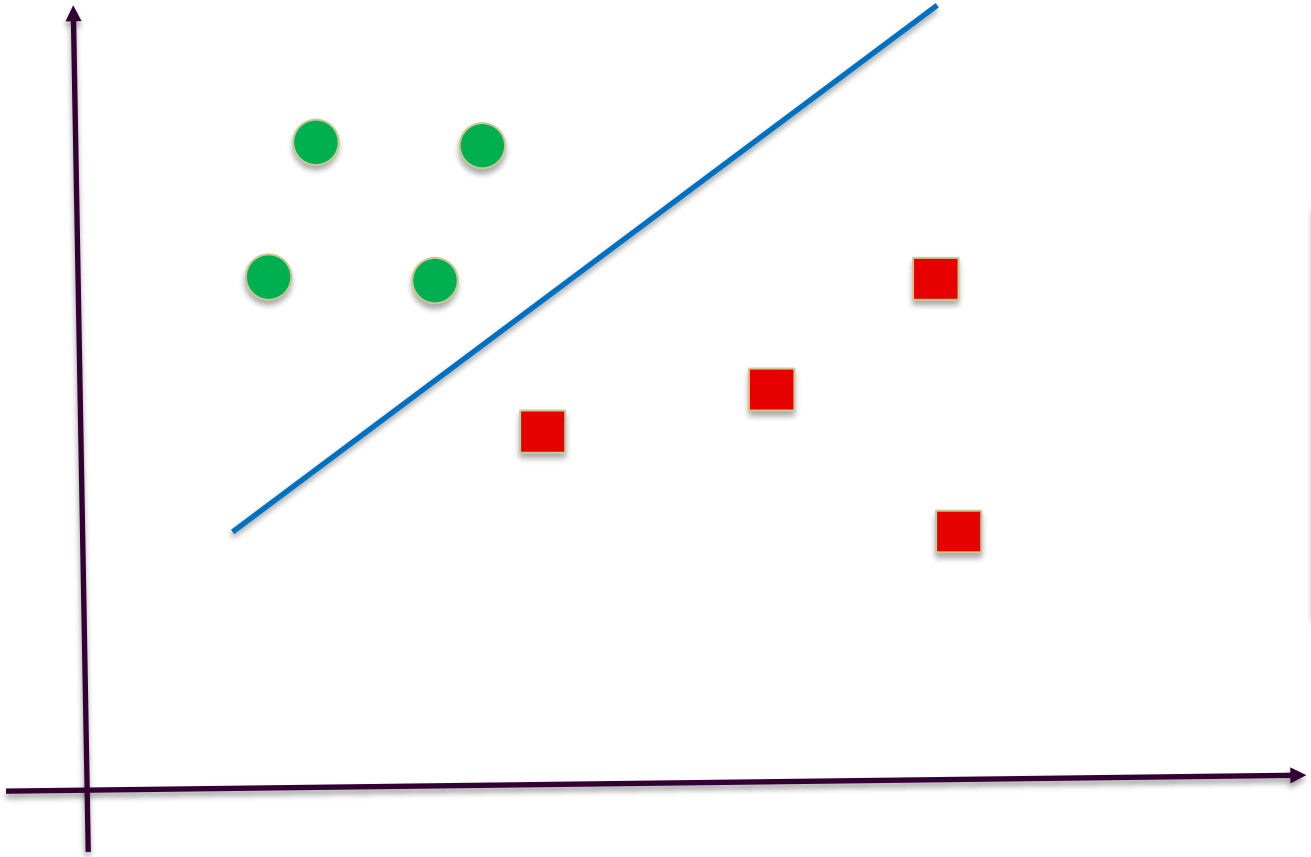
funded by:

Outline

Reinforcement Learning

- Key characteristics for reinforcement learning
- Sequential decision making
- Q-Learning
- Discussion of reinforcement learning algorithms
- Relation to other areas

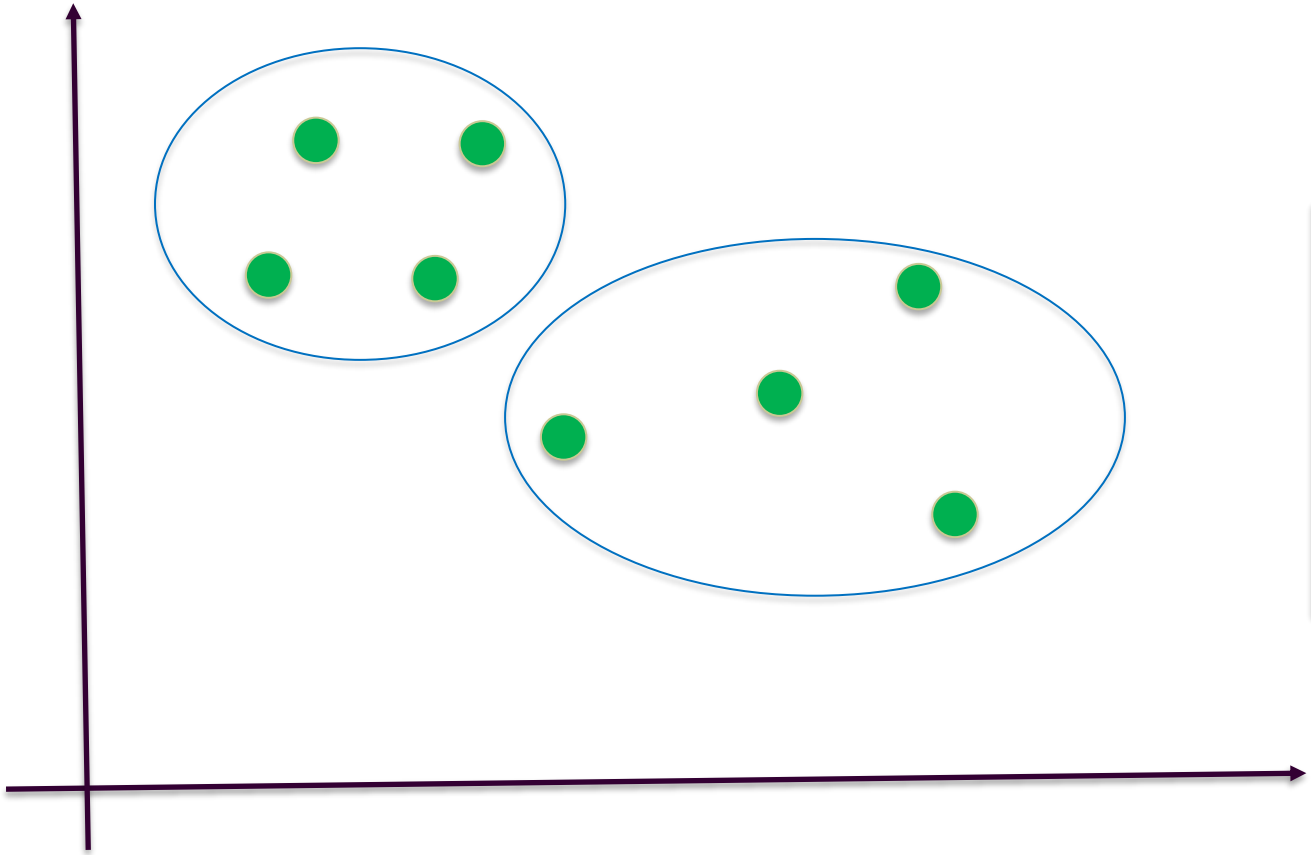
Supervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\ldots (x_N, y_N)\}$

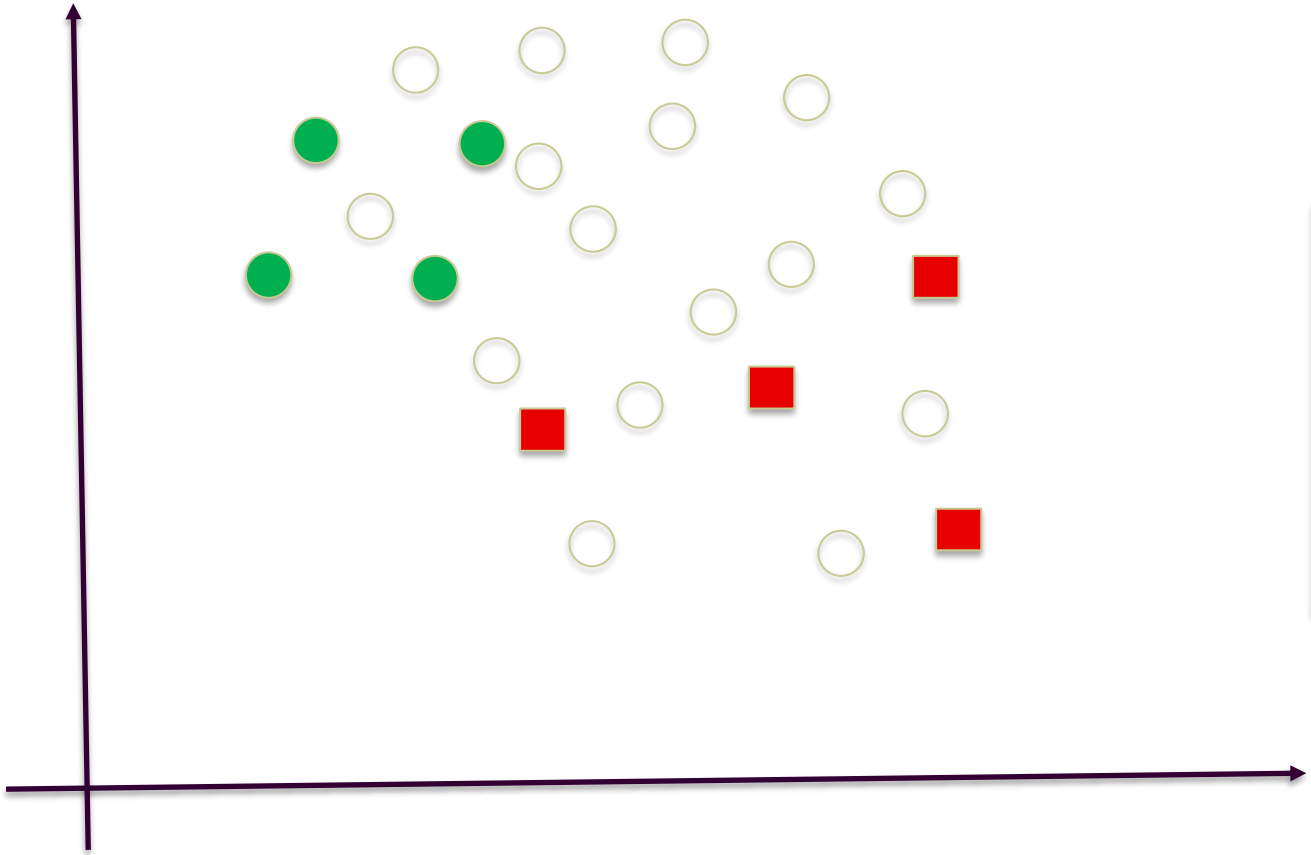
Unsupervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{x_1, x_2, x_3, \ldots x_N\}$

Semi-supervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\ldots (x_L, y_L), x_{L+1}, x_{L+2}, x_{L+3}, \ldots x_N\}$

Reinforcement Learning: Key Characteristics

- There are no labels (correct output) provided. Only reward signals
- There is a delayed reward (not instantaneous feedback)
- Sequential decision making. Agents actions affect the subsequent data it receives (similar to control problems)
- Includes exploration

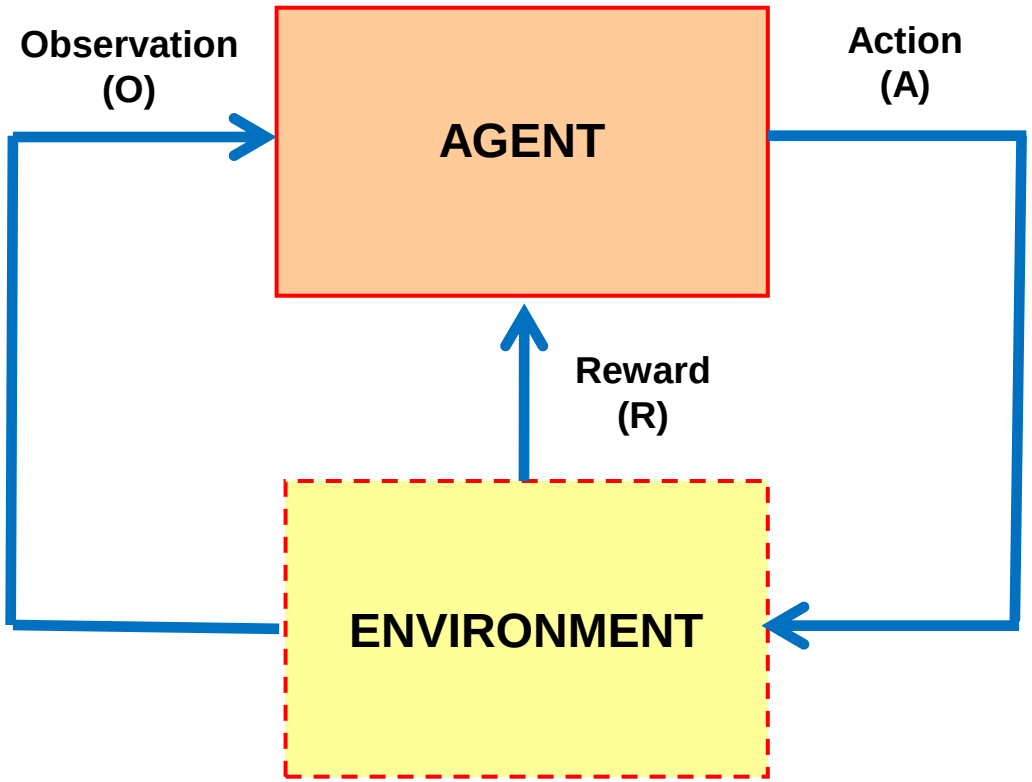
Reinforcement Learning: Examples

- Mobile robot deciding which action to take
- Develop a computer chess master
- Manage an investment portfolio
- Make a humanoid robot walk
- Resource management in computer clusters
- Traffic light control
- Optimize chemical reactions

Reinforcement Learning: Key Elements

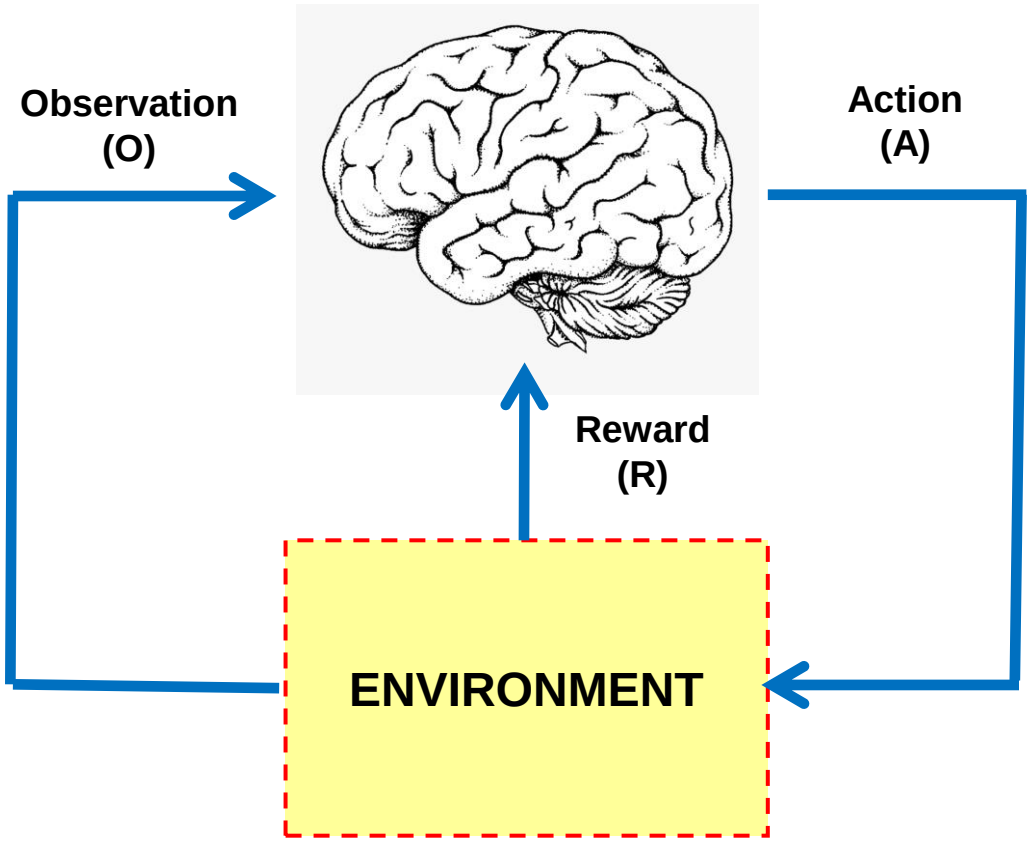
- Policy
- Reward signal
- Value function
- Model of the environment

RL: Sequential Decision Making



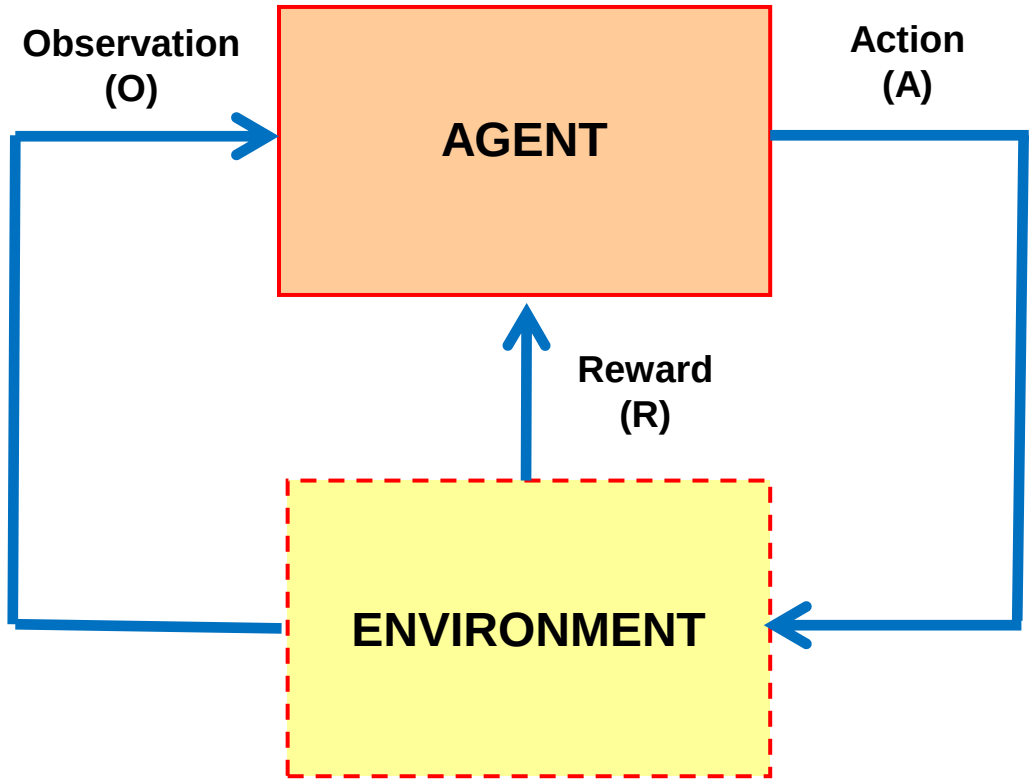
$$O_1 \xrightarrow[R_1]{A_1} O_2 \xrightarrow[R_2]{A_2} O_3 \xrightarrow[R_3]{A_3} \cdots$$

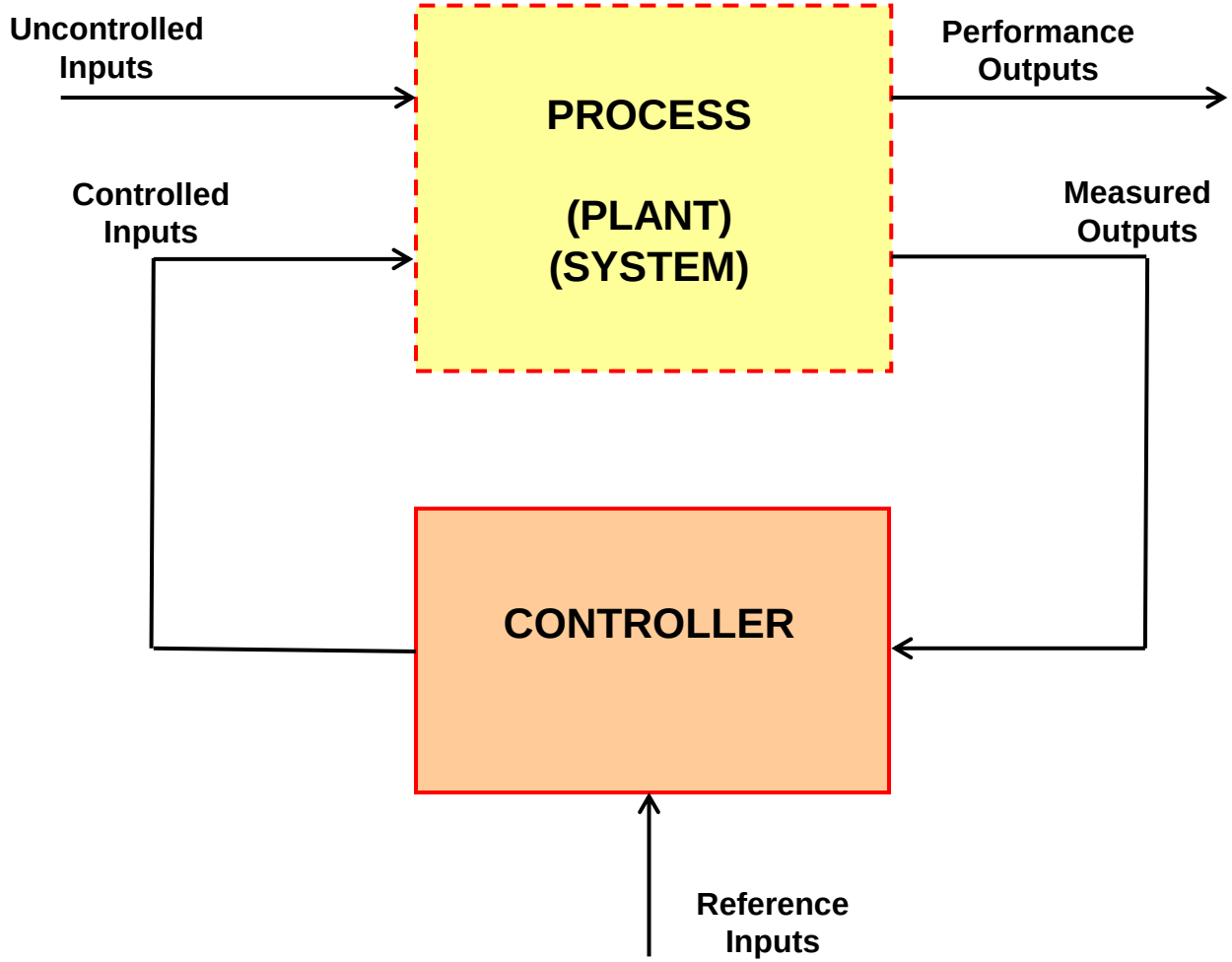
RL: Sequential Decision Making



$$O_1 \xrightarrow[R_1]{A_1} O_2 \xrightarrow[R_2]{A_2} O_3 \xrightarrow[R_3]{A_3} \cdots$$

Reinforcement Learning vs. Feedback Control





RL: Sequential Decision Making

- GOAL: select actions to maximize total future reward (value)
- Current actions may have long term consequences
- Reward may be delayed
- It may be better to sacrifice immediate reward to gain long term value
    - Examples?

History and State

- The history at time t is the sequence of observations, actions and rewards up to time t: $H_t = O_1, R_1, A_1, \ldots\ldots A_{t-1}, O_t, R_t$
- Based on the history, the AGENT selects the next Action
- Based on the history, the ENVIRONMENT selects the next Observation/Reward
- The State is a function of the history, i.e., $S_t = f(H_t)$, which determines what happens next.
    - How does that compare with Control Theory?

Markov Decision Processes

A state S_t is said to be a Markov State if

$$\Pr\left[S_{t+1} \mid S_t\right] = \Pr\left[S_{t+1} \mid S_1, S_2, \ldots S_t\right]$$

- The future is independent of the past given the present
- Once the state is known, the history may be discarded
- Markov state is also known as information state
- If the Agent/Environment State is a Markov State then this is known as **Markov Decision Process**

Markov Decision Processes

- By definition, Markov Decision Process (MDP) is a fully observable environment; i.e., the agent directly observes the environment state
- In a Partially Observable Markov Decision Process (POMDP) the agent does not directly observe the environment, but it may do that indirectly.
  - A mobile robot may not know its absolute location
  - A poker playing agent only observes public cards

Partially Observable Markov Decision Processes

- In a Partially Observable Markov Decision Process the agent must construct its own state representation (model of the environment)
- We can use a neural network (for example) to approximate it

Reinforcement Learning: Policy

- The task of the agent is to learn a policy $\pi : S \mapsto A$ for selecting the next action based on the current observed state
- The policy characterizes the agent's behavior
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a \mid s) = \Pr\left[A_t = a \mid S_t = s \right]$

Reinforcement Learning: Value

- The value function is a prediction of future reward
- Policy is selected in a way that maximizes the value
- The cumulative value $V_\pi(s_t)$ is defined what is achieved by following a policy π from an initial state s_t

$$V_\pi(s_t) = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$$

$$= \sum_{i=0}^{\infty} \gamma^i R_{t+i} \qquad 0 \le \gamma < 1$$

Discount factor
(tradeoffs?)

Reinforcement Learning: Optimal Policy

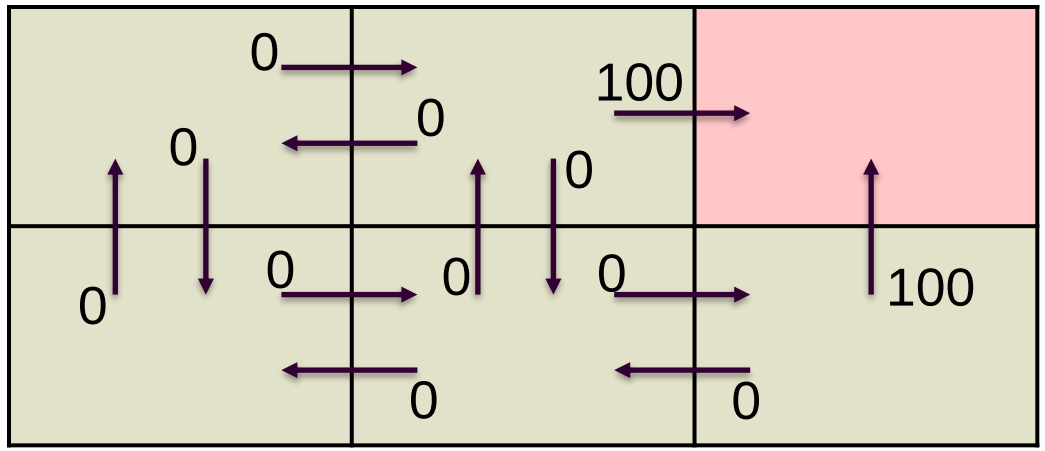
- The optimal policy for each state s is defined as:

$$\pi^*(s) = \arg\max_{\pi} \{V_\pi(s)\}$$

$$\begin{aligned} V_\pi(s_t) &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots \\ &= R_t + \gamma(R_{t+1} + \gamma^2 R_{t+2} + \dots) \\ &= R_t + \gamma V_\pi(s_{t+1}) \end{aligned}$$

- There are other options for $V_\pi(s_t)$

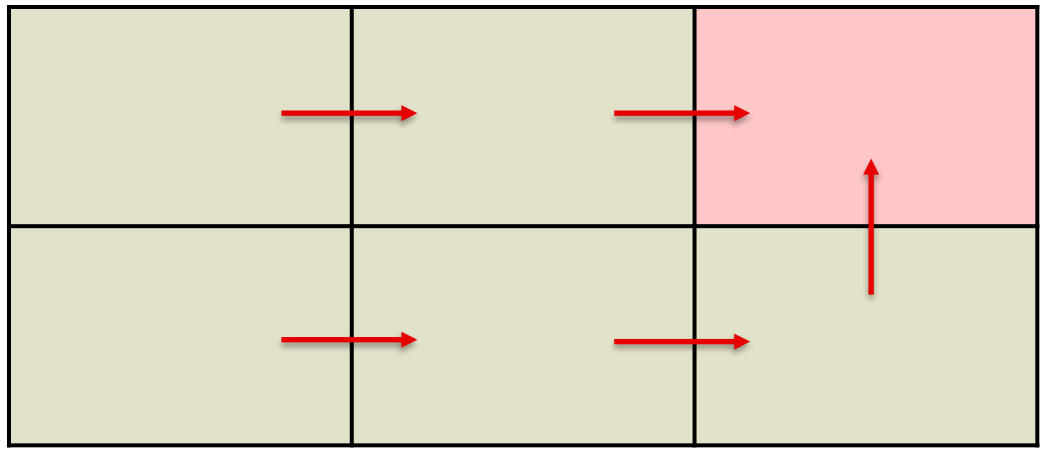
Reinforcement Learning: Example



R(s,a) : Immediate reward values

- Each cell represents a state
- The goal is to reach the red cell
- Once it reaches the red cell it remains there (absorbing state)
- Discount factor: $\gamma = 0.9$

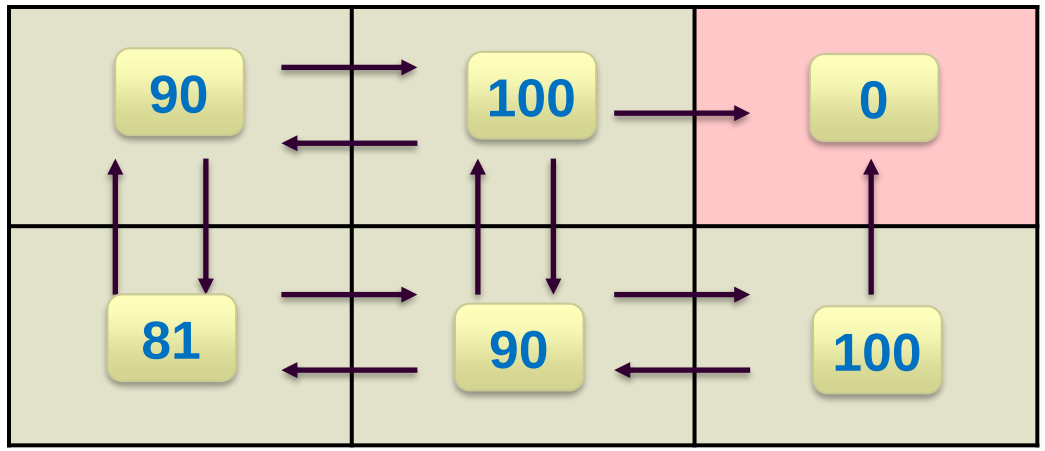
Reinforcement Learning: Example



One optimal policy (not unique)

- Each cell represents a state
- The goal is to reach the red cell
- Once it reaches the red cell it remains there (absorbing state)
- Discount factor: $\gamma = 0.9$

Reinforcement Learning: Example



V*(s) values

$$V^*(s) = \arg\max_{\pi} \{V_\pi(s)\}$$

$$V_\pi(s_t) = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$$

$$= \sum_{i=0}^{\infty} \gamma^i R_{t+i}$$

Reinforcement Learning: Q-Learning

- Learning an optimal policy for an arbitrary environment is very difficult
- The training data does nor provide examples of the form (s, a)

$$\pi^*(s) = \arg\max_{\pi} \{R_t + \gamma V_\pi(s_{t+1})\}$$

$$Q^*(s,a) = R(s,a) + \gamma V^*(\delta(s,a))$$

$$\pi^*(s) = \arg\max_{a} \{Q^*(s,a)\}$$

Reinforcement Learning: Q-Learning

$$Q^*(s,a) = R(s,a) + \gamma V^*(\delta(s,a))$$

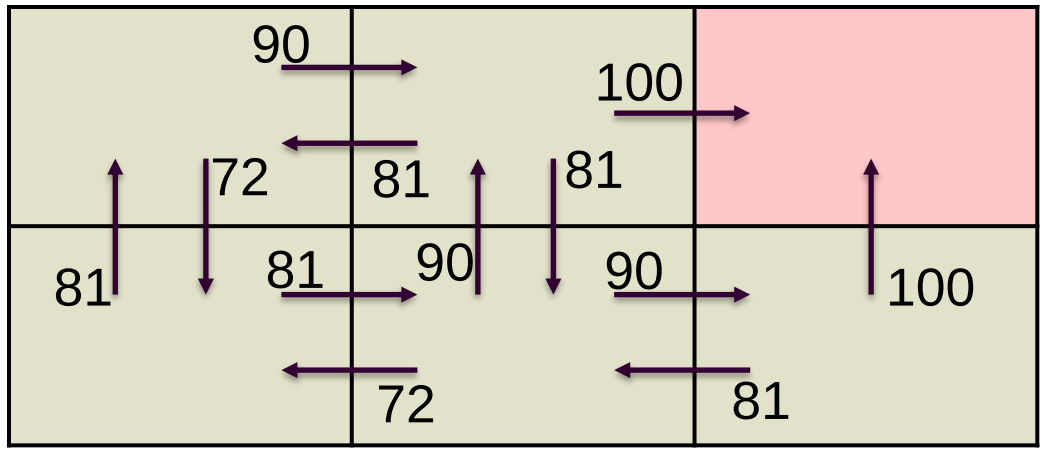
$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} Q^*(\delta(s,a),a')$$

$$Q^{NEW}(s_t,a_t) = R_t + \gamma \max_a Q(s_{t+1},a)$$

Non-recursive Q-Learning

(simpler version)

Reinforcement Learning: Example

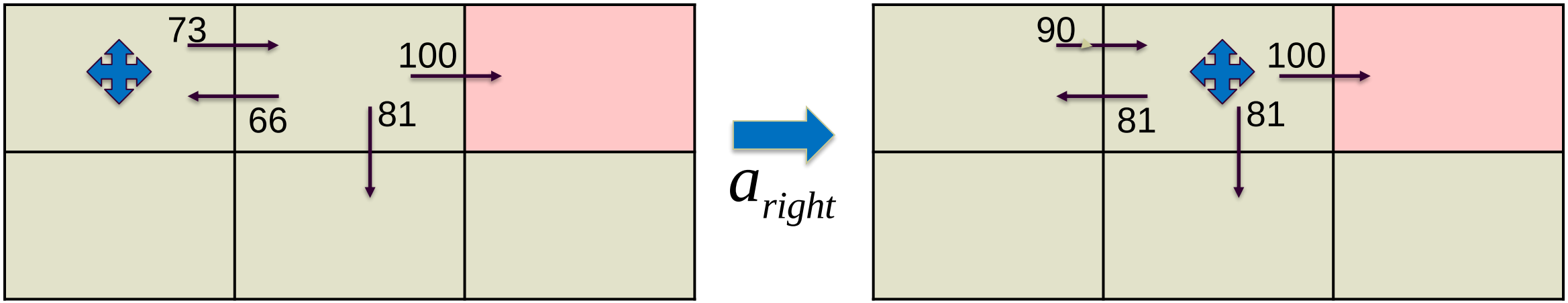


Q*(s,a) values

$$Q^*(s,a) = R(s,a) + \gamma V^*(\delta(s,a))$$

$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} Q^*(\delta(s,a),a')$$

Reinforcement Learning: Q-Learning Example



$$Q^{NEW}(s_t, a_{right}) = R_t + \gamma \max_a Q(s_{t+1}, a)$$
$$= 0 + 0.9 \max\{66, 81, 100\}$$
$$= 90$$

Reinforcement Learning: Q-Learning

$$Q^{NEW}(s_t, a_t) = Q(s_t, a_t) + \alpha \left[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Old Q value

learning rate (step-size)

Immediate reward

Discount factor

Estimate of optimal future value

Old Q value

Reinforcement Learning: Q-Learning

$$Q^{NEW}(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

Learning rate (0, 1] : determines to what degree new information overrides old information. If α = 1 then it causes the agent to consider only the most recent information. In fully deterministic environments, α = 1 is optimal. Usually chosen small (α = 0.1). Learning rate can be designed to be time-varying.

Discount factor [0, 1) : determines the importance of future rewards. If γ = 0 then agent short-sighted, considering only immediate rewards. If it approaches 1 then agent pays attention to long-term rewards. If γ = 1 or close to 1 then it may cause instability. Sometimes it helps the learning to start the discount factor low and then to increase it towards its final value.

Reinforcement Learning: Q-Learning

$$Q^{NEW}(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

Initial Conditions: typically, chosen arbitrary, except for the values associated with the terminal state, which need to be zero. There are many suggested techniques for selecting the initial conditions to improve learning. If initial values are high, that encourages exploration.

Convergence: convergence of Q to its optimal value can be proven under certain conditions. The conditions involve visiting every distinct state infinitely often – this is related to the persistency of excitation condition in parameter estimation.

Reinforcement Learning: SARSA vs. Q-Learning

$$Q^{NEW}(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

$$Q^{NEW}(s_t, a_t) = Q(s_t, a_t) + \alpha\left[R_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right]$$

SARSA (state-action-reward-state-action): on-policy control compared to Q-Learning, which is an off-policy algorithm.

Reinforcement Learning Algorithms

- Q-Learning and SARSA are examples of reinforcement learning algorithms that rely on Temporal-Difference (TD) Learning
- Model-Free reinforcement learning vs. Model-Based reinforcement learning
- On-policy algorithms vs. off-policy algorithms
- Discrete-time action vs. continuous-time action
- Discrete-time state space vs. continuous-time state space

Exploration/Exploitation Dilemma

- Try a new restaurant or go to your favorite restaurant!
- To receive high rewards, the RL algorithm should prefer actions that it has tried in the past and found to be effective in producing high rewards.
- However, to discover such actions, it needs to try actions that it has not selected before.
- Need to find a balance between exploiting known actions and exploring new actions.

RL with Function Approximation

- How can we solve large-scale problems?
  - Backgammon: 10^{20} states
  - Chess: 10^{47} states
  - Computer GO (19X19): 10^{170} states
  - Mobile robot: continuous state space.

RL with Function Approximation

- So far, we have represented the Q function or the V value as a lookup table with an entry for every state/action (s, a)
- Problem: Too many states and/or actions to store in memory
- Problem: Too slow to learn the value for each state and/or action
- Solution: use function approximation to estimate the value function (or Q function), and generalize from seen states to unseen states.
- Neural networks (and other type of approximators) can be used.
- Require training/learning of the neural network

Reinforcement Learning discussion

- Relationship to Dynamic Programming and Optimal Control
- Relationship to neuroscience and psychology
- Learning and Planning

Current Trends in Reinforcement Learning

- Reinforcement Learning is a hot field
- Deep Reinforcement Learning
- Multi Agent Reinforcement Learning
- DeepMind (U.K.) bought by Google for $500M in 2014 has very strong technologies in RL

Some Bibliography on Reinforcement Learning

- R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd Edition, The MIT Press, 2018.
- D. Bertsekas, *Reinforcement Learning and Optimal Control*, Athena Scientific, 2019.
- C. Szepesvari, *Algorithms for Reinforcement Learning*, Morgan and Claypool, 2010.
- David Silver Lectures on RL: https://www.davidsilver.uk/teaching