



Introduction to Reinforcement Learning

Vitalii Duk
dubizzle

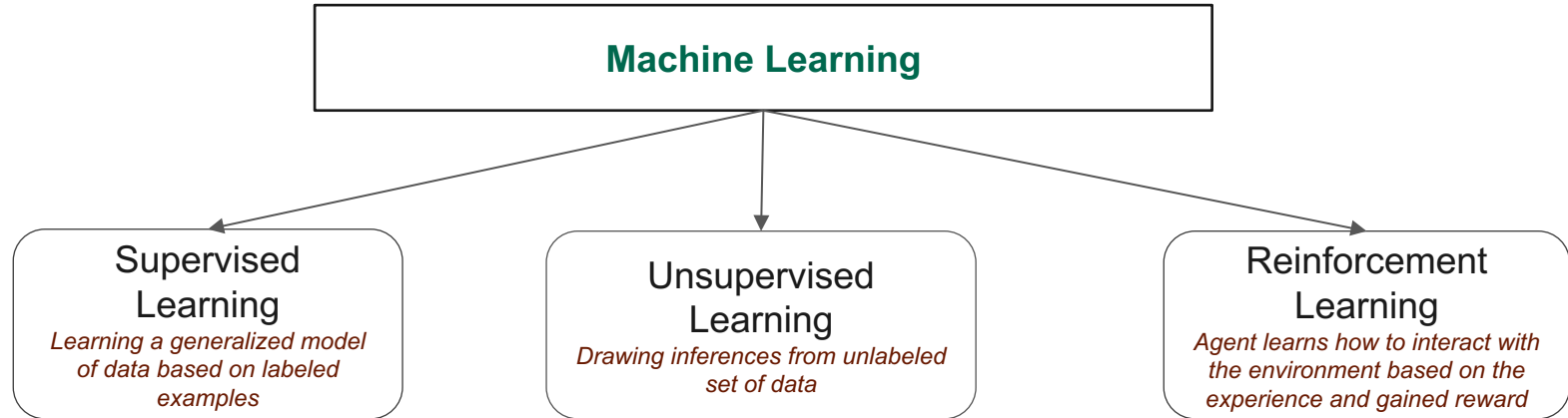


Agenda

- What is Reinforcement Learning?
- Markov Chains and Markov Decision Process
- Solving MDP
- Q-learning and Deep Q-learning
- Python examples



What is Reinforcement Learning?





What is Reinforcement Learning?

How RL is different from other types of machine learning?

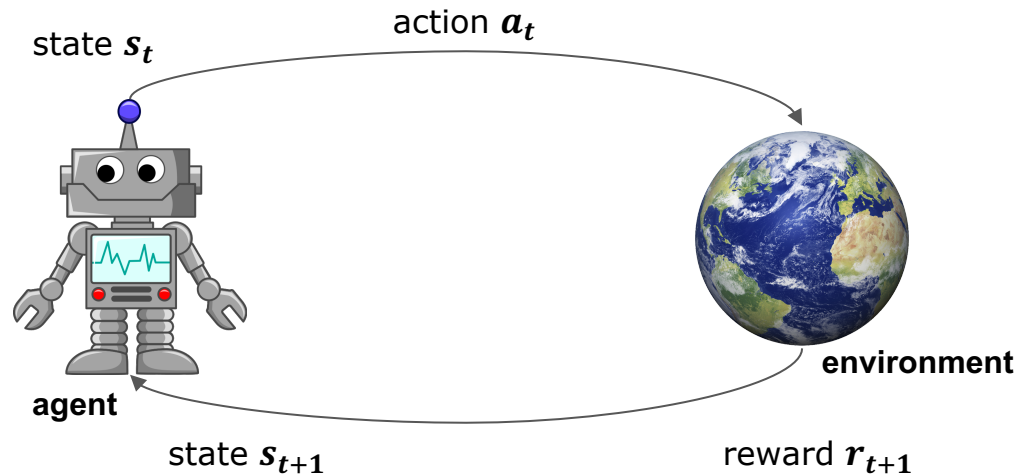
- No labels or supervision, just a reward.
- It's possible that the reward can be delayed.
- Agent take actions which will affect the future reward.
- Observed data is sequential, and order/time matters (non *i.i.d*).



What is Reinforcement Learning?

- **Engineering**
 - *Optimal Control*
- **Mathematics**
 - *Operations Research*
- **Psychology**
 - *Classical Conditioning*
- **Neuroscience**
 - *Reward System*
- **Economics**
 - *Bounded Rationality*

What is Reinforcement Learning?



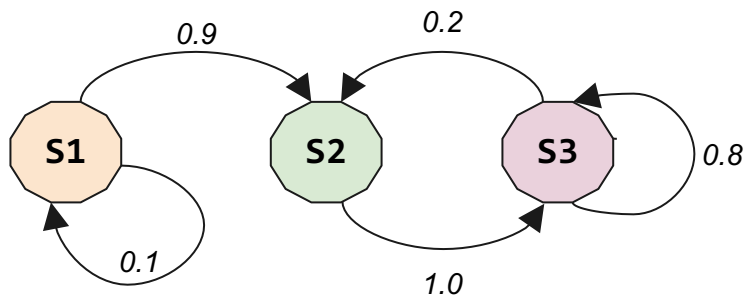


Types of RL environments

- **Model-based** - assumes that agent knows everything about the environment in which he operates, all consequences of any action at any state are known.
- **Model-free** - means that explicit knowledge about the environment dynamics is missing, and it can be learned only during the interaction with the environment.

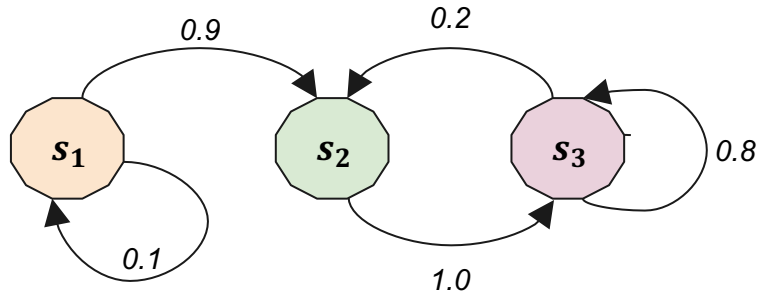
Markov Process

- **Markov Process** or **Markov Chain** is a *stochastic (random)* process that satisfies **Markov property**.
- **Markov Property** assume *memorylessness*, which means that predictions about the future of the process can be made based only on the current state, without any knowledge about the historical states.



Markov Process

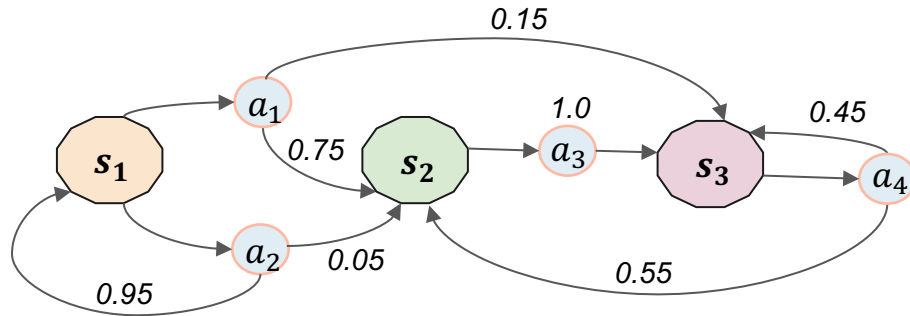
- **Markov Process** is characterized by:
 - finite set of states \mathcal{S}
 - transition probabilities from current state s to next state s'



s	s'	p
s_1	s_2	0.9
s_1	s_1	0.1
s_2	s_3	1.0
s_3	s_2	0.2
s_3	s_3	0.8

Markov Decision Process

- **Markov Process** is characterized by:
 - Finite set of states S and finite set of actions A
 - Reward $R(s, a, s')$ received after transitioning from state s to s' , due to action a
 - Probability $Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ that action a in state s will lead to state s'
 - Discount factor $\gamma \in [0, 1)$





Markov Decision Process

- MDP can be represented as follows:

$$s_0 \xrightarrow{a_0} \xrightarrow{r_1} s_1 \xrightarrow{a_1} \xrightarrow{r_2} s_2 \xrightarrow{a_2} \xrightarrow{r_3} \dots$$

- Total reward can be represented as follows:

$$G_t = R(s_0, a_1) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$$

- Policy is any function that is mapping state to an action $\pi : \mathcal{S} \mapsto \mathcal{A}$
- Our main goal is to choose actions over time that will maximize the expected value of a total reward discounted by a factor of γ^t at time t .



Policy

- A policy π defines agent's behaviour:

$$\pi(\mathbf{a} \mid \mathbf{s}) = \mathbb{P}(\mathbf{A}_t = \mathbf{a} \mid \mathbf{S}_t = \mathbf{s})$$

- Due to a **Markov Property** policy depends only on a current state (*time-independent*).
- In a simple words policy is mapping that says which action should be performed at each state.



State value function

- Tells us how good it is for an agent to be in a particular state with respect to a particular policy

$$V_{\pi}(\mathbf{s}) = \sum_{\mathbf{a} \in \mathcal{A}} \pi(\mathbf{s}, \mathbf{a}) \sum_{\mathbf{s}' \in \mathcal{S}} \mathbf{P}_{\mathbf{s}, \mathbf{s}'}^{\mathbf{a}} (\mathbf{R}_{\mathbf{s}, \mathbf{s}'}^{\mathbf{a}} + \gamma V^{\pi}(\mathbf{s}'))$$

- Optimal state-value function can be represented as:

$$\mathbf{V}^*(\mathbf{s}) = \max_{\mathbf{a} \in \mathcal{A}} \sum_{\mathbf{s}' \in \mathcal{S}} \mathbf{P}_{\mathbf{s}, \mathbf{s}'}^{\mathbf{a}} (\mathbf{R}_{\mathbf{s}, \mathbf{s}'}^{\mathbf{a}} + \gamma \mathbf{V}^*(\mathbf{s}'))$$



State-Action value function

- Tells us how good is for an agent to perform action a being in a state s :

$$Q_{\pi}(s, a) = \sum_{s' \in S} P_{s,s'}^a (R_{ss'}^a + \gamma V^{\pi}(s'))$$

- Optimal state-action value function can be represented as:

$$Q^*(s, a) = \sum_{s' \in S} P_{s,s'}^a (R_{ss'}^a + \max_{a' \in \mathcal{A}} Q^*(s', a'))$$



Bellman Equation & Optimality Principle

- Bellman's Principle of Optimality:
 - *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (Bellman, 1957)*
- Bellman equation is a necessary condition for optimality in dynamic programming.



Ways to solve MDP

- **Dynamic Programming**
 - Policy iteration
 - Value iteration
- **Monte Carlo methods**
- **Temporal Difference**
 - SARSA
 - Q-Learning



Policy iteration

- Pick an arbitrary policy π
- Repeat until π is unchanged:
 - Evaluate the policy by computing a state-value function:

$$V(s) = \sum_{s' \in S} T(s, a, s') [R(s, \pi(s), s') + \gamma V(s')]$$

- Improve the policy:

$$\pi(s) \leftarrow \arg \max_a \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma V(s')]$$



Value iteration

- Initialize vector V with an arbitrary values (e.g. zeros)
- Repeat until Δ reach threshold (e.g. 10^{-6}):

- For each $s \in \mathcal{S}$:

$$v \leftarrow V(s)$$

$$V(s) = \max_a \sum_{s' \in \mathcal{S}} T(s, a, s') [R(s, a, s') + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

- Output optimal deterministic policy:

$$\pi(s) \leftarrow \arg \max_a \sum_{s' \in \mathcal{S}} T(s, a, s') [R(s, a, s') + \gamma V(s')]$$



Monte Carlo Methods

- Can learn purely from experience and don't need a complete knowledge of the environment.
- Require only sample sequences of states, actions and rewards obtained during the interaction with an environment.
- Can be used to predict ***state-value*** or ***action-value*** functions.
- Value estimation can be written as:

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



Temporal-Difference Learning

- Can be used for episodic learning, no need to reach the end of the “game”.
- Make an update of a value function at each step after observing immediate reward.

$$V(s) \leftarrow V(s) + \alpha (r + V(s') - V(s))$$



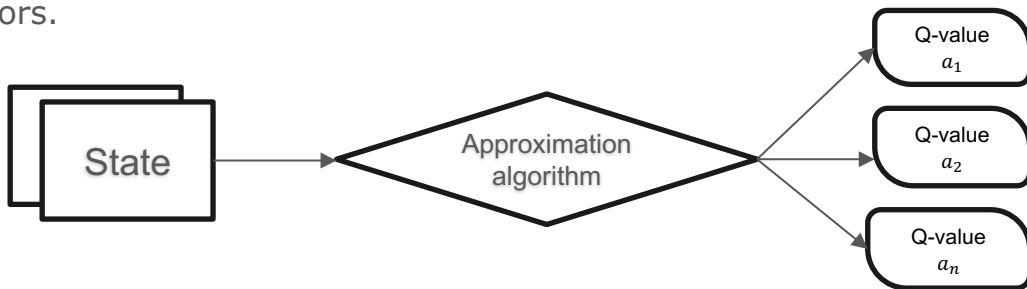
Q-Learning: off-policy TD control

- Model-free RL algorithm used to find an optimal policy for a given MDP, introduced by Watkins in 1989.
- At each step t agent selects an action \mathbf{a}_t , observes a reward r_t and new state \mathbf{s}_{t+1} and used this information to update $Q(\mathbf{s}_t, \mathbf{a}_t)$

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow (1 - \alpha) \cdot Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \cdot (r_t + \gamma \cdot \max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}))$$

Value function approximation

- If a state space is huge (*e.g. position of the helicopter*) than it's computationally infeasible to construct and store Q-table directly.
- Function approximation approaches can be used to estimate Q-value based on the state representation. We assume that similar states have similar Q-values.
- **Neural Networks, Linear Regression, Gradient Boosting** can be used as a Q-value approximators.

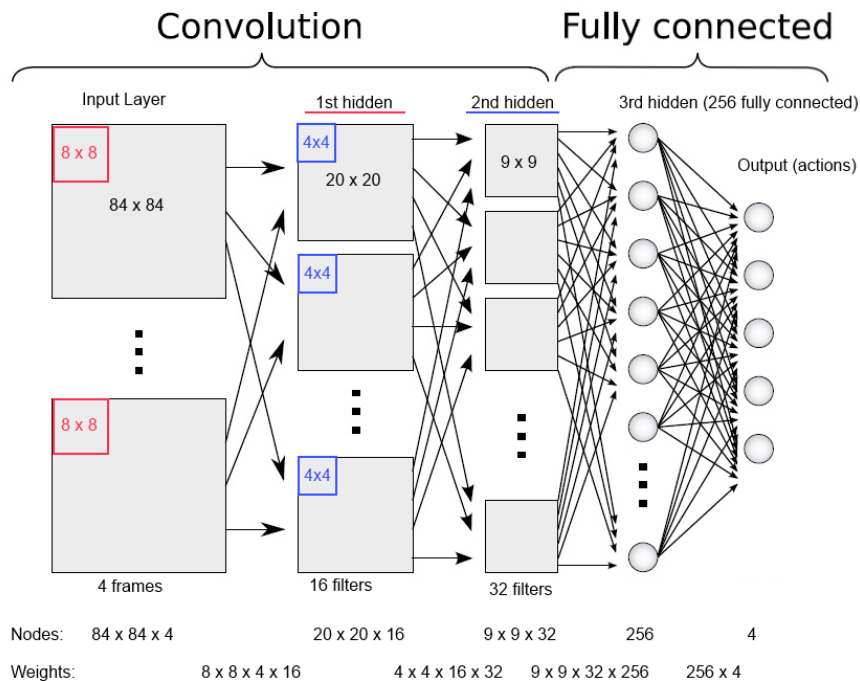




Deep Q-Learning

- Version of Q-Learning where Deep Neural Network is used as a state-action value estimator.
- Raw pixels can be used as an input for Value Function Approximation.
- First proposed in DeepMind's paper in Nature (2013): <https://arxiv.org/abs/1312.5602>.
- Patented by Google: <https://www.google.com/patents/US20150100530>.

Playing Atari with Deep Reinforcement Learning





Recommended resources

- **Reinforcement Learning course by David Silver from DeepMind:**
 - Video-lectures: <https://www.youtube.com/watch?v=2pWv7GOvuf0>
 - Slides: <http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html>
- **Reinforcement Learning course by Georgia Tech on Udacity:**
 - CS 8803: <https://www.udacity.com/course/reinforcement-learning--ud600>
- **Deep Reinforcement Learning course by UC Berkley:**
 - CS 294: <http://rll.berkeley.edu/deeprlcourse>
- **Deep Reinforcement Learning and Control course by CMU:**
 - CMU 10703: <https://katefvision.github.io>
- **R. Sutton, A. Barto, Reinforcement Learning: An introduction:**
 - 1st edition, 1998: <http://incompleteideas.net/sutton/book/the-book-1st.html>
 - 2nd edition, 2017: <http://incompleteideas.net/sutton/book/the-book-2nd.html>



Examples

<https://github.com/root-ua/rl-intro>

OLX Group today: the world's #1 classifieds business

Horizontals



global



global app-only



Russia



UAE

Real Estate Verticals



global



Africa and
Philippines



Russia



Portugal



Poland



Other Verticals



fashion, global



property finder, global



heavy machinery, global



services, global

Car Verticals



global



Romania



Portugal



Poland



We are hiring!

<https://www.joinolx.com>