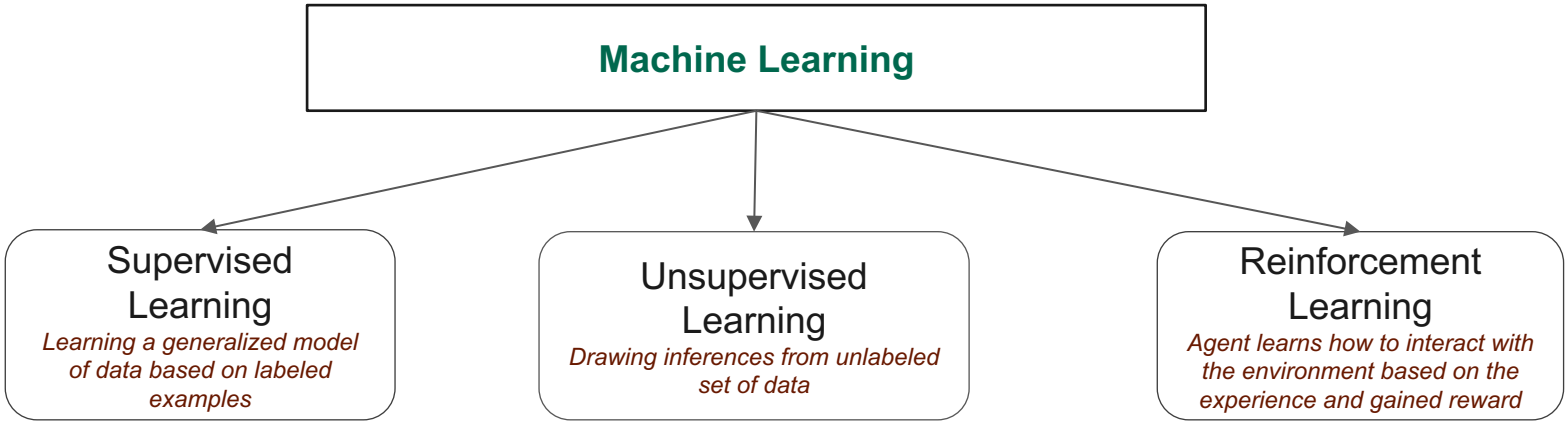
Introduction to Reinforcement Learning

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?

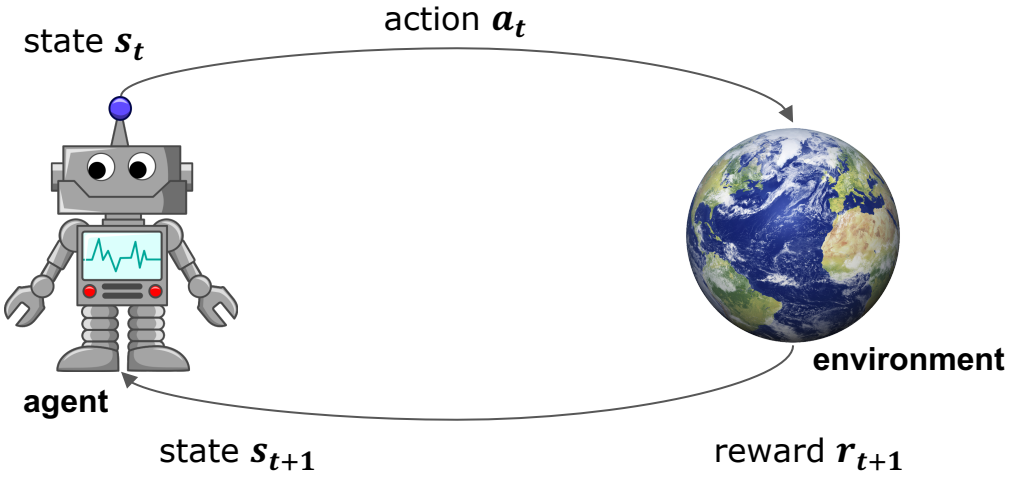
How RL is different from other types of machine learning?

- No labels or supervision, just a reward.
- It’s possible that the reward can be delayed.
- An agent takes 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

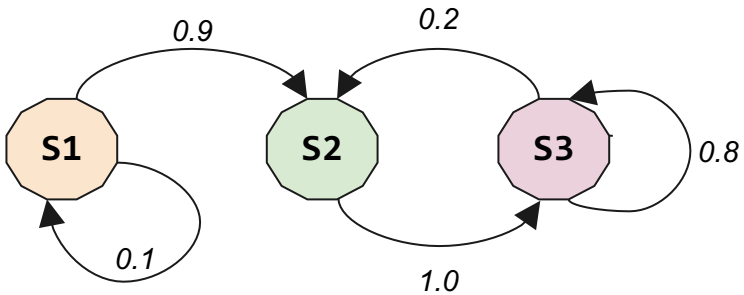
- **Model-based -** assumes that agent knows everything about the environment in which he operates, all consequences of any action at any state are known.

- **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

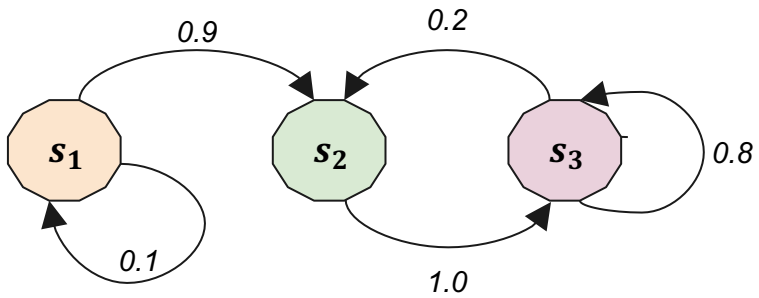
- **Markov Process** or **Markov Chain** is a *stochastic (random)* process that satisfies **Markov property**.

- **Markov Property** assumes *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

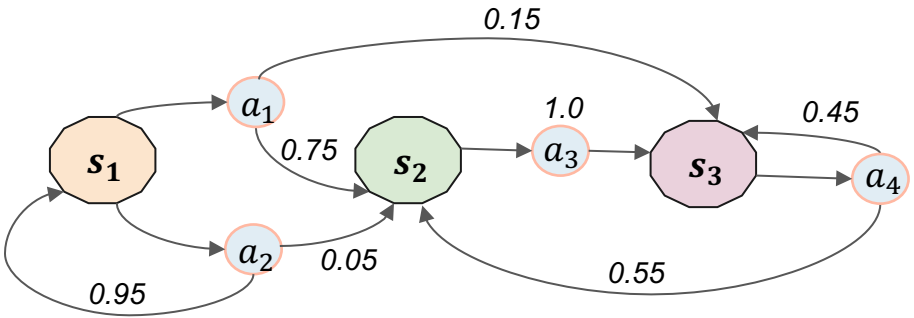
- **Markov Process** is characterized by:
  - a finite set of states S
  - transition probabilities from current state s to next state s'



s	s'	p
S1	S2	0.9
S1	S1	0.1
S2	S3	1.0
S3	S2	0.2
S3	S3	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:

$$\mathbf{s_0} \overset{\mathbf{a_0}}{\rightarrow} \overset{\mathbf{r_1}}{\rightarrow} \mathbf{s_1} \overset{\mathbf{a_1}}{\rightarrow} \overset{\mathbf{r_2}}{\rightarrow} \mathbf{s_2} \overset{\mathbf{a_2}}{\rightarrow} \overset{\mathbf{r_3}}{\rightarrow} \cdots$$

- Total reward can be represented as follows:

$$\mathbf{G_t} = \mathbf{R}(\boldsymbol{s_0}, \mathbf{a_1}) + \boldsymbol{\gamma}\mathbf{R}(\boldsymbol{s_1}, \boldsymbol{a_2}) + \boldsymbol{\gamma^2}\mathbf{R}(\boldsymbol{s_2}, \boldsymbol{a_3}) + \cdots$$

- Policy is any function that is mapping state to an action $\boldsymbol{\pi} : \mathbf{S} \longmapsto \mathbf{A}$

- Our main goal is to choose actions over time that will maximize the expected value of a total reward discounted by a factor of $\boldsymbol{\gamma^t}$ at time $\boldsymbol{t}$.

Policy

- A policy $\boldsymbol{\pi}$ defines agent's behaviour:

$$\boldsymbol{\pi(a \mid s) = \mathbb{P}(A_t = a \mid S_t = s)}$$

- 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

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

- Optimal state-value function can be represented as:

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

State-Action value function

- Tells us how good is for an agent to perform action $\boldsymbol{a}$ being in a state $\boldsymbol{s}$:

$$\mathbf{Q_{\pi}(s,a)} = \sum_{\mathbf{s' \in S}} \mathbf{P^{a}_{s,s'}} \left(\mathbf{R^{a}_{ss'}} + \boldsymbol{\gamma} \mathbf{V^{\pi}} \left(\mathbf{s'}\right)\right.$$

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

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

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 $\boldsymbol{\pi}$
- Repeat until $\boldsymbol{\pi}$ is unchanged:
  - Evaluate the policy by computing a state-value function:

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

  - Improve the policy:

$$\boldsymbol{\pi}(\mathbf{s}) \leftarrow \underset{\mathbf{a}}{\mathbf{arg\,max}} \sum_{\mathbf{s}' \in \mathbf{S}} \mathbf{T}(\mathbf{s}, \mathbf{a}, \mathbf{s}') \left[\mathbf{R}(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \boldsymbol{\gamma}\, \mathbf{V}(\mathbf{s}')\right]$$

Value iteration

- Initialize vector $\boldsymbol{V}$ with an arbitrary values (e.g. zeros)
- Repeat until Δ reach threshold (e.g. $\mathbf{10^{-6}}$):
    - For each $\boldsymbol{s} \in \boldsymbol{S}$:

$$v \leftarrow V(s)$$

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

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

    - Output optimal deterministic policy:

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

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:

$$\mathbf{V(s)} \leftarrow \mathbf{V(s)} + \ \boldsymbol{\alpha}\ (\mathbf{G_t}\ - \mathbf{V(s)})$$

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.

$$\mathbf{V}(\mathbf{s}) \leftarrow \mathbf{V}(\mathbf{s}) + \boldsymbol{\alpha}\,(\mathbf{r} + \mathbf{V}(\mathbf{s}') - \mathbf{V}(\mathbf{s}))$$

Q-Learning: off-policy TD control

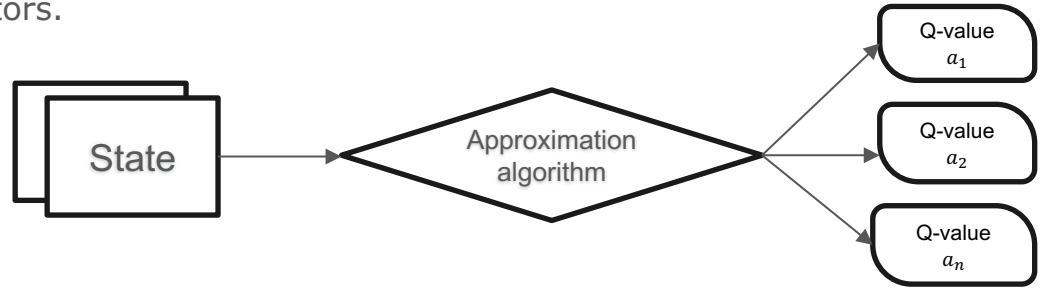
- Model-free RL algorithm used to find an optimal policy for a given MDP, introduced by Watkins in 1989.

- At each step t agent selects an action a_t, observes a reward r_t and new state s_{t+1} and used this information to updated $Q(s_t, a_t)$

$$\mathbf{Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot (r_t + \gamma \cdot \max_a Q(s_{t+1}, a))}$$

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.



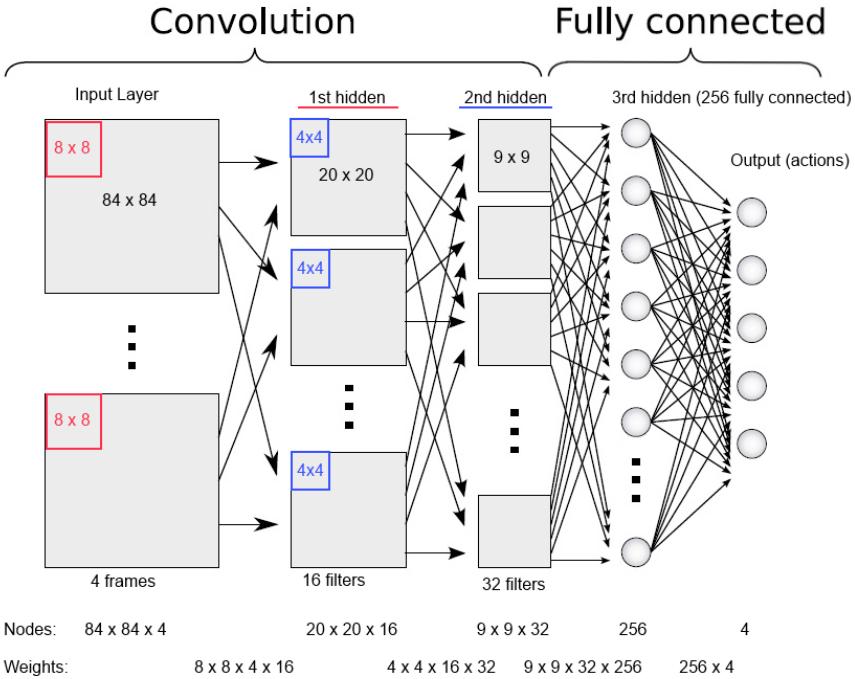
Exploration-Exploitation trade-off

- In Q-learning **Q-function** for each **state-action** pair is learned based on the agent's experience.
- Based on the Q-function agent selects the **best action** to perform in a current state.
- If the agent will select actions greedily based on the previous experience (**exploitation**) all the time it might be stuck with suboptimal policy.
- Though, **exploring** random action continuously will lead to wasting valuable knowledge.
- ε**-greedy**: exploit best actions with the probability $1 - \varepsilon$, take random actions otherwise

Deep Q-Learning

- Version of Q-Learning where Deep Neural Network is used 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



Source: http://robohub.org/artificial-general-intelligence-that-plays-atari-video-games-how-did-deepmind-do-it

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

OLX	letgo	Avito	dubizzle
global	global app-only	Russia	UAE

Real Estate Verticals

storia	global
property24	Africa and Philippines
Domofond.ru	Russia
IMOVIRTUAL	Portugal
otodom.	Poland

OLX Group

Other Verticals

shedd	fashion, global
DOMOZO	property finder, global
TRADUS	heavy machinery, global
fixly	services, global

Car Verticals

STRADIA	global
AUTOVIT.RO	Romania
STANDVIRTUAL	Portugal
OTOMOTO	Poland

We are hiring!

https://www.joinolx.com