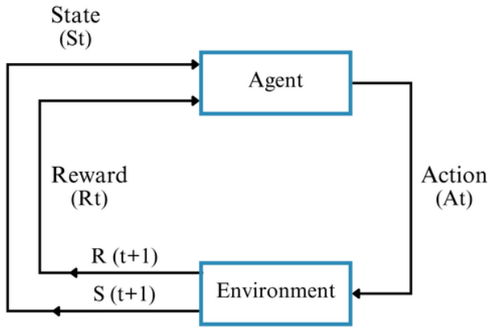
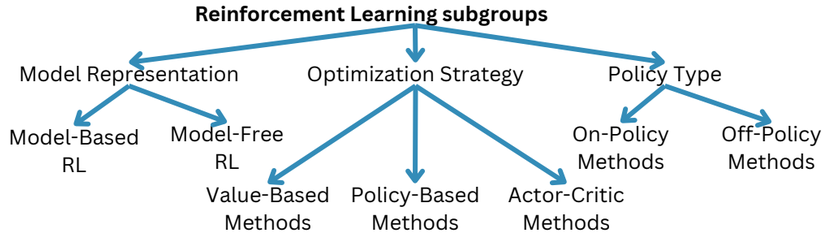
Reinforcement Learning Cheat Sheet



Reinforcement Learning consists of a system where a certain agent (or multi-agents) interacts with a static or dynamic environment using their experience to optimize its decision-making process with a final goal of getting more points (rewards). These agents will make decisions based on trial-and-error starting by exploring the environment and then exploiting what they already know to find the best path or the best actions to maximize the cumulative rewards from the environment.
Known terms when defining a RL problem are:

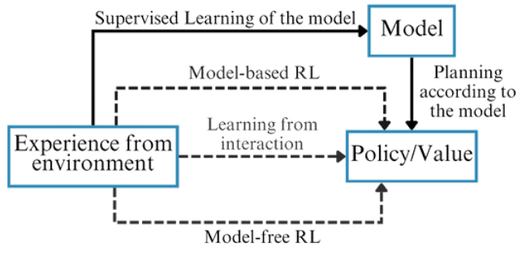
- **Action space:** a set of possible actions. It could be discrete or continuous (or both).
- **Environment**: a simulator with rules or mechanisms that allows an agent to interact with the observations gathered by acting with possible actions available. It can be fully observable or partially observable referring to the agent's point of view.
- **Reward:** a value or points returned by the environment corresponding to the agent's action.
- **Episode:** a sequence of interactions between an agent and the environment.
- **Policy:** a function that determines which action will the agent take in a certain state. The policy is trained during experiments in order to achieve the maximum possible cumulative reward (maximum rewards in an episode).



Model-Based vs. Model-Free

Model-Based RL: The agent builds a model of the environment (i.e., it tries to understand the dynamics of how states change or rewards are given) and uses this model to plan and make decisions.

Model-Free RL: The agent doesn't try to build an internal model of the environment and learns purely through interactions (direct trial-and-error learning).



Value-Based vs. Policy-Based vs. Actor-Critic (applies to Model-Free RL)

Value-Based Methods: The agent learns a value function (state value or state-action value) to guide its actions. The policy is implicitly derived from the value function.

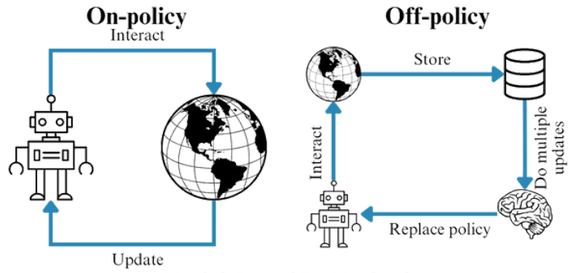
Policy-Based Methods: The agent learns the policy directly by optimizing it without estimating value functions. It works directly on finding the best policy to maximize rewards.

Actor-Critic Methods: A hybrid between value-based and policy-based. The actor updates the policy (like policy-based methods), and the critic evaluates how good the action taken by the actor was (like value-based methods).

On-Policy vs. Off-Policy

On-Policy: The agent learns about and improves the policy it is currently using. It updates its knowledge based on actions it actually took according to its policy.

Off-Policy: The agent learns about a policy that is different from the one it is currently using to interact with the environment. This allows more flexibility, such as learning from past experiences stored in a buffer (experience replay).



Model-Based RL Methods

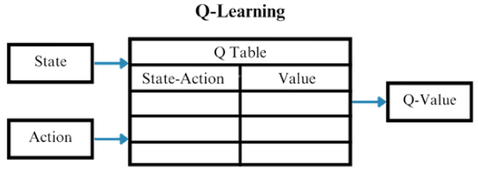
- **Value Iteration:** Iteratively updates the value function for each state by finding the maximum expected value over all possible actions. The policy is derived once the value function converges.
- **Policy Iteration:** Alternates between policy evaluation (computing the value of a fixed policy) and policy improvement (updating the policy based on the current value function) until convergence.

Model-Free, On-Policy

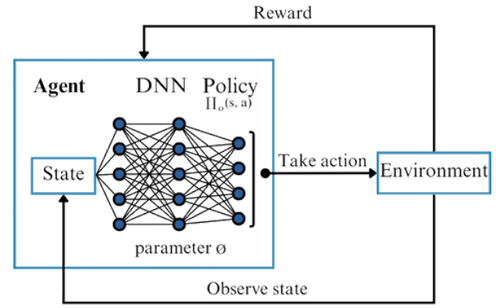
- **Monte Carlo Methods:** A value-based method that estimates value functions by averaging returns over multiple episodes. Requires full episodes to calculate the value function, and is purely on-policy (learns directly from the policy it is following).
- **Sarsa:** A value-based temporal-difference method that updates the action-value function using the action selected by the current policy (on-policy). It updates based on the sequence: state, action, reward, next state, next action, instead of waiting before an episode is finished.
- **n-Step Sarsa:** Extends Sarsa by considering rewards over multiple steps (n steps), updating the action-value function based on cumulative rewards from several future steps. This approach provides a balance between the bias of one-step Sarsa and the variance of Monte Carlo methods.
- **Expected Sarsa:** Similar to Sarsa, but instead of using the next action's actual value, it updates using the expected value of the next action, averaged over the current policy's action distribution.

Model-Free, Off-Policy

Q-Learning: A value-based, off-policy temporal-difference algorithm that learns the optimal action-value function by updating based on the maximum future action value, regardless of the policy being followed.

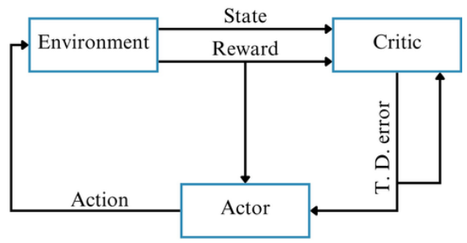


- **Double Q-Learning:** A variation of Q-learning that mitigates overestimation bias by maintaining two Q-functions and updating them alternately, using one to select actions and the other to evaluate them.
- **Deep Q-Learning (DQN):** Combines Q-Learning with deep neural networks to handle large, high-dimensional state spaces. It uses experience replay and a target network to stabilize training.



Actor-Critic Methods (AC)

- **Actor-Critic:** Combines policy-based (on and off) and value-based methods. The actor updates the policy based on feedback from the critic, which estimates the value function. This allows for more stable learning than purely policy-based methods.



- **A2C (Advantage Actor-Critic):** An on-policy algorithm where multiple workers synchronously sample data using the current policy. The actor is updated based on the advantage function (difference between the action's value and average value).
- **A3C (Asynchronous Advantage Actor-Critic):** An on-policy algorithm where workers asynchronously interact with the environment, each using the current policy to collect data. The asynchronous nature of A3C speeds up learning and prevents the need for experience replay, as used in A2C.
- **Proximal Policy Optimization (PPO):** A popular on-policy gradient method that improves Actor-Critic by using a clipped objective function to ensure policy updates remain stable. PPO is simpler and more computationally efficient than TRPO.
- **Trust Region Policy Optimization (TRPO):** An on-policy algorithm that optimizes the policy by ensuring updates remain within a "trust region," preventing large and unstable changes, which improves the stability of training.
- **Deep Deterministic Policy Gradient (DDPG):** An off-policy actor-critic algorithm designed for continuous action spaces. It uses a deterministic policy (actor) and a Q-function (critic) with experience replay and a target network.
- **Soft Actor-Critic (SAC):** An off-policy actor-critic method that adds an entropy term to the reward, encouraging exploration by maximizing both the expected return and entropy. This provides a balance between exploration and exploitation being efficient in high-dimensional continuous action spaces.

Rui Fernandes
rpfernandes@ipca.pt