Reinforcement Learning

Reinforcement learning solves the difficult problem of correlating immediate actions with the delayed returns they produce. Like humans, reinforcement learning algorithms sometimes have to wait a while to see the fruit of their decisions. They operate in a delayed return environment, where it can be difficult to understand which action leads to which outcome over many time steps. Reinforcement learning algorithms can be expected to perform better and better in more ambiguous, real-life environments while choosing from an arbitrary number of possible actions, rather than from the limited options of a video game. That is, with time we expect them to be valuable to achieve goals in the real world.

Reinforcement learning can be understood using the concepts of (**1**) **agents**, (**2**) **environments**, (**3**) **states**, (**4**) **actions** and (**5**) **rewards.**

1. The algorithm is the **agent**. In real life you (we) are the agent.
2. **Environment** is the world through which the agent moves. The environment takes the agent’s current state and action as input, and returns as output the agent’s reward and its next state. If you are the agent, the environment could be the laws of physics and the rules of society that process your actions and determine the consequences of them.
3. A **state** is an immediate situation in which the agent finds itself in, an instantaneous configuration that puts the agent in relation to other significant things such as tools, obstacles, enemies or prizes. It can the current situation returned by the environment, or any future situation.
4. A is the set of all possible moves the agent can make. Agents choose among a list of possible **actions**. In video games, the list might include running right or left, jumping high or low, crouching or standing still. In the stock markets, the list might include buying, selling or holding any one of an array of securities and their derivatives. When handling aerial drones, alternatives would include many different velocities and accelerations in 3D space.
5. A **reward** is the feedback by which we measure the success or failure of an agent’s actions. For example, in a video game, when Mario touches a coin, he wins points. From any given state, an agent sends output in the form of actions to the environment, and the environment returns the agent’s new state (which resulted from acting on the previous state) as well as rewards, if there are any. Rewards can be immediate or delayed. They effectively evaluate the agent’s action.

Other important terms:

1. **Discount factor**: The **discount factor**is multiplied by future rewards as discovered by the agent in order to dampen these rewards’ effect on the agent’s choice of action. Why? It is designed to make future rewards worth less than immediate rewards; i.e. it enforces a kind of short-term hedonism in the agent. Often expressed with the lower-case Greek letter gamma: γ. If γ is .8, and there’s a reward of 10 points after 3 time steps, the present value of that reward is 0.8³ x 10. A discount factor of 1 would make future rewards worth just as much as immediate rewards. We’re fighting against [delayed gratification](https://en.wikipedia.org/wiki/Stanford_marshmallow_experiment) here.
2. **Policy (π):** The **policy** is the strategy that the agent employs to determine the next action based on the current state. It maps states to actions, the actions that promise the highest reward.
3. **Value (V):** The expected long-term return with discount, as opposed to the short-term reward R. Vπ(s) is defined as the expected long-term return of the current state under policy π. We discount rewards, or lower their estimated value, the further into the future they occur. See discount factor. And remember Keynes: “In the long run, we are all dead.” That’s why you discount future rewards
4. **Q-value** or **action-value (Q): Q-value** is similar to Value, except that it takes an extra parameter, the current action a. Qπ(s, a) refers to the long-term return of the current state s, taking action a under policy π. Q maps state-action pairs to rewards. Note the difference between Q and policy
5. **Trajectory:** A sequence of states and actions that influence those states. From the Latin “to throw across.” The life of an agent is but a ball tossed high and arching through space-time.

Reinforcement learning is a machine learning technique for solving problems by a feedback system (rewards and penalties) applied on an **agent** which operates in an **environment** and needs to move through a series of states in order to reach a pre-defined final state. A classical example is a rat (agent) which is trying to find the shortest route from a starting cell to a target cheese cell in a maze (environment). The agent is **experimenting** and **exploiting** past experiences (**episodes**) in order to achieve its goal. It may fail again and again, but hopefully, after lots of trial and error (rewards and penalties) it will arrive to the solution of the problem. The solution will be reached if the agent finds the optimal sequence of states in which the **accumulated sum of rewards** is maximal (in short, we lure the agent to accumulate a maximal reward, and while doing so, he actually solves our problem). Note that it may happen that in order to reach the goal, the agent will have to endure many penalties (negative rewards) on its way. (https://www.samyzaf.com/ML/rl/qmaze.html)