# Reinforcement Learning Notes

## The Agent-Environment Interface

Markov Decision Process is solving the problem of learning from interaction to achieve a goal. The learner and decision maker is called an **agent**. An **agent** interacts with its **environment**. These two entities interact continuously – the agent selecting an action and the environment and the environment responding to these actions presenting new situations to the agent. The environment gives rise to **rewards**, special numerical values that the agent seeks to maximize over time through its choice of values.

A diagram of a agent and environment

Description automatically generated with low confidence

Figure: the agent-environment interaction in a Markov decision process

More specifically, the agent and environment interact at each of a sequence of discrete time steps,

Notes: 1) In *Optimal Control Theory* the terms **controller**, **controlled system**, and **control signal** are used instead of agent, environment and action.

2) We are going to restrict our attention to discrete time steps to keep the analysis as simple as possible. The framework can be extended to continuous time at the expense of analytical simplicity (see *Bertsekas, Reinforcement Learning and Optimal Control*).

At each time step the agent receives a representation of the environment’s **state** and on this basis selects an **action**, . One time step later, in part of consequence of its action, the agent receives a numerical **reward**, , and finds itself in a new state, . The MDP and the agent thereby give rise to a sequence or **trajectory** that begins like this:

In a finite MDP, the sets of states, actions, and rewards () all have finite number of elements. In this case, the random variables and have well defined probability distributions dependent only on the preceding state and action. That is, for particular values of these random variables, and , there is a probability of thse values occurring at time t, given particular values of the preceding state and action:

## Policies and Value Functions

Almost all reinforcement learning algorithms involve estimating **value functions** – functions of state (or state-action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). The notion of “how good” is defined in terms of future rewards that can be expected, or to be precise, in terms of expected return. Of course, the reward the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular ways of acting, called **policies**.

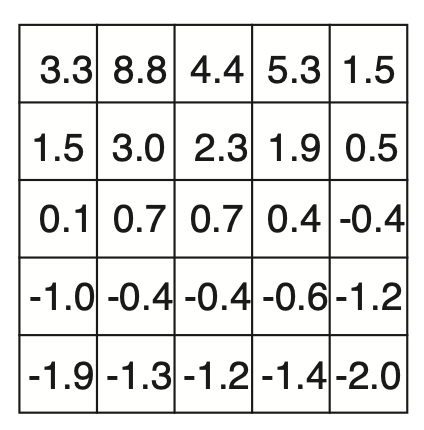
Formally, a **policy** is a mapping from states to probabilities of selecting each possible action. If an agent is following policy at time , then is the probability that if . Like , is an ordinary function which defines a probability distribution.

***Gridworld***

Gridworld is an example illustrating the use of a finite Markov Decision Process.

The cells of the grid correspond to the states of the environment. At each cell, four actions are possible: ***north***, ***south***, ***east***, and ***west***, which deterministically cause the agent to move one cell in the respective direction on the grid. Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of -1.

Other actions result in a reward of 0, except those that move the agent out of the special states and . From state , all four actions yield a reward of +10 and take the agent to . From state B



A picture containing symbol, font, white, graphics

Description automatically generatedA picture containing diagram, line

Description automatically generated

//TODO: finish the discussion on Markov decision processes

# Bibliography

Berteskas, D. P. (2019). *Reinforcement Learning and Optimal Control.* Belmont, Massachusetts: Athena Scientific.

Richard S. Sutton, A. G. (2020). *Reinforcement Learning: An Introduction.* Cambridge, Massachusetts: The MIT Press.