Source: Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto

1. Introduction:

Learning- interacting with environment

Cause effect- from interactions

1.1. Reinforcement learning:

Learning what to do- how to map situations to actions to maximize reward

Learner has to discover what action yields the most reward by trying them

Trial and error search and delayed reward

Markov decision process: three aspects—sensation, action, and goal

Different from supervised learning

Different from unsupervised learning also

Trade-off between exploration and exploitation

it explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment

Agents and environments

Can be used in different fields

1.2. Examples:

Master chess player makes a move

Real-time control of a parameter of a petroleum refinery: cost quality trade-off

A newborn calf tries to stand, walk and run

Mobile robot makes a decision like entering a room in search of something, considering its own battery level

Phill makes his breakfast considering many things like hunger, preference, availability of a particular item, and many things

All these examples show that, agent can do something using the info from experience.

1.3. Elements of reinforcement learning:

one can identify four main subelements of a reinforcement learning system: a policy, a reward signal, a value function, and, optionally, a model of the environment.

A policy defines the learning agent's way of behaving at a given time.

A reward signal defines the goal in a reinforcement learning problem.

A value function specifies what is good in the long run.

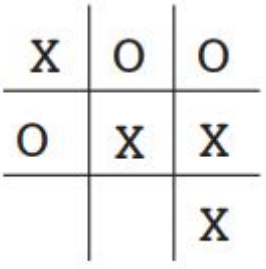
A model of the environment specifies the way how environment will react after a move is made by the agent.

1.4. Limitations and scope:

Think of the state as whatever information is available to the agent about its environment.

Most of the reinforcement learning methods we consider in this book are structured around estimating value functions.

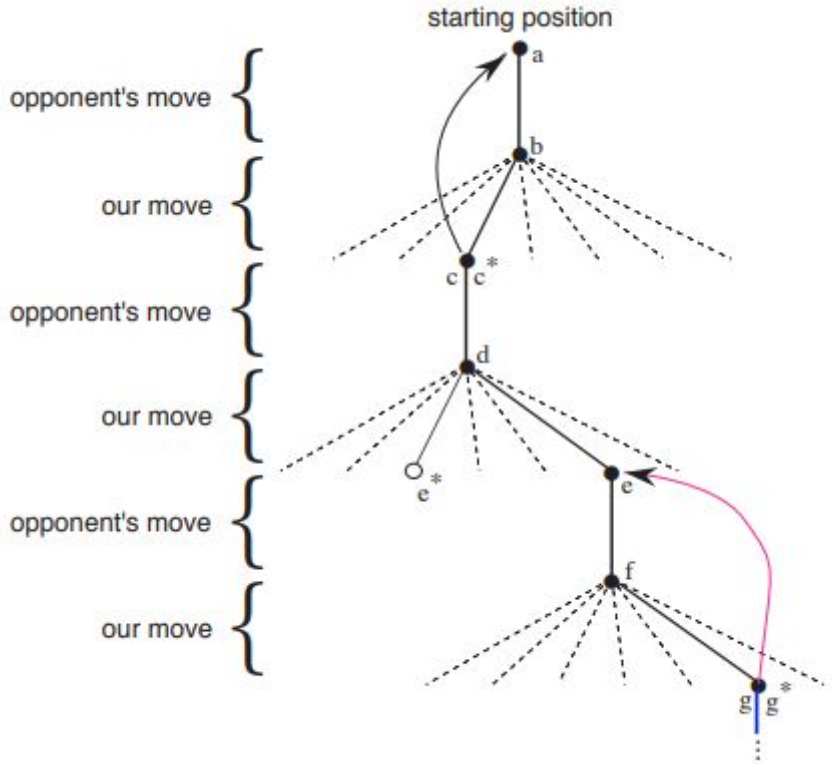
1.5. An extended example: tic-tac-toe



An evolutionary method applied to this problem would directly search the space of possible policies for one with a high probability of winning against the opponent. Here, a policy is a rule that tells the player what move to make for every state of the game—every possible configuration of Xs and Os on the three-by-three board. For each policy considered, an estimate of its winning probability would be obtained by

playing some number of games against the opponent. This evaluation would then direct which policy or policies were considered next. A typical evolutionary method would hill-climb in policy space, successively generating and evaluating policies in an attempt to obtain incremental improvements. Or, perhaps, a genetic-style algorithm could be used that would maintain and evaluate a population of policies. Literally hundreds of different optimization methods could be applied.

Here is how the tic-tac-toe problem would be approached with a method making use of a value function. First we set up a table of numbers, one for each possible state of the game. Each number will be the latest estimate of the probability of our winning from that state. We treat this estimate as the state's value, and the whole table is the learned value function. State A has higher value than state B, or is considered "better" than state B, if the current estimate of the probability of our winning from A is higher than it is from B. Assuming we always play Xs, then for all states with three Xs in a row the probability of winning is 1, because we have already won. Similarly, for all states with three Os in a row, or that are "filled up," the correct probability is 0, as we cannot win from them. We set the initial values of all the other states to 0.5, representing a guess that we have a 50% chance of winning.



after the move, then the update to the estimated value of s, denoted $V(s)$, can be written as

$$V(s) \leftarrow V(s) + \alpha\Big[V(s') - V(s)\Big],$$

where α is a small positive fraction called the *step-size parameter*, which influences the rate of learning. This update rule is an example of a *temporal-difference* learning method, so called because its changes are based on a difference, $V(s') - V(s)$, between estimates at two different times.

1.6. Summary:

Reinforcement learning uses the formal framework of Markov decision processes to define the interaction between a learning agent and its environment in terms of states, actions, and rewards.

1.7. Early history of reinforcement learning:

The early history of reinforcement learning has two main threads, both long and rich, that were pursued independently before intertwining in modern reinforcement learning. One thread concerns learning by trial and error that started in the psychology of animal learning. This thread runs through some of the earliest work in artificial intelligence and led to the revival of reinforcement learning in the early 1980s. The other thread concerns the problem of optimal control and its solution using value functions and dynamic programming. For the most part, this thread did not involve learning. Although the two threads have been largely independent, the exceptions revolve around a third, less distinct thread concerning temporal-difference methods such as the one used in the tic-tac-toe example in this chapter. All three threads came together in the late 1980s to produce the modern field of reinforcement learning as we present it in this book.