MACHINE LEARNING

Reinforcement Learning

Dr G.Kalyani

Department of Information Technology

Velagapudi Ramakrishna Siddhartha Engineering College

Topics

- Introduction
- Learning Tasks
- Q-Learning

Introduction to RL

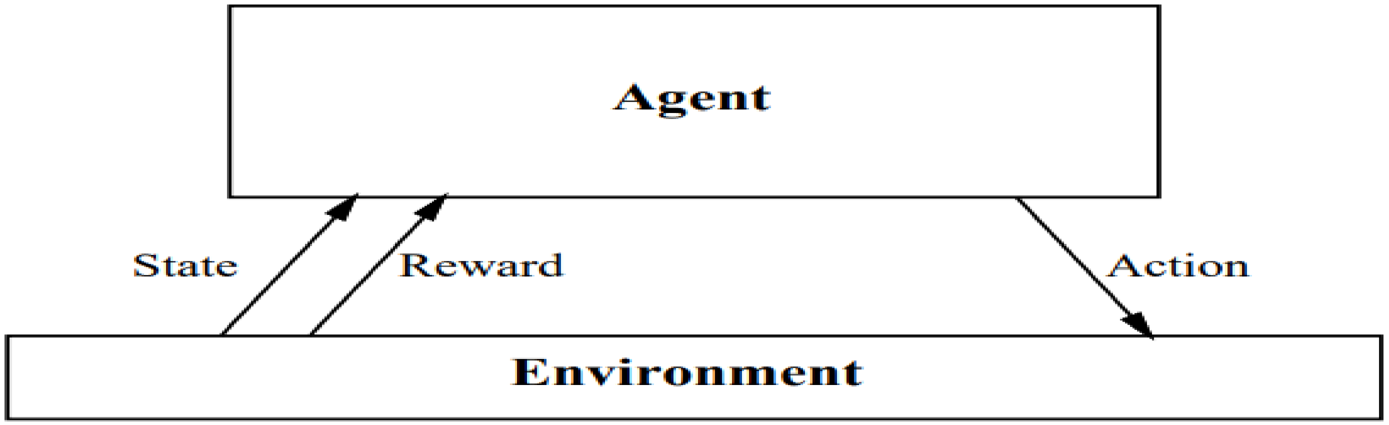
- Reinforcement learning addresses the question of **how an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals.**

- This very generic problem covers tasks such as learning to
  - control a mobile robot,
  - learning to optimize operations in factories, and
  - learning to play board games.

- Each time the agent performs an action in its environment, a trainer may provide a reward or penalty to indicate the desirability of the resulting state.

- The task of the agent is to learn from this indirect, delayed reward, to choose sequences of actions that produce the greatest cumulative reward.

Introduction to RL



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \quad , \text{ where } 0 \leq \gamma < 1$$

Introduction to RL

- An agent interacting with its environment. The agent exists in an environment described by some set of possible states S.
- It can perform any of a set of possible actions A.
- Each time it performs an action a, in some state st the agent receives a real-valued reward r, that indicates the immediate value of this state-action transition.
- This produces a sequence of states si, actions ai, and immediate rewards ri.
- The agent's task is to learn a control policy, $\Pi : S \rightarrow A$, that maximizes the expected sum of these rewards, with future rewards discounted exponentially by their delay.

Characteristics of RL

- **Delayed reward:**
  - The task of the agent is to learn a target function n that maps from the current state s to the optimal action a = n(s).
  - In earlier learnings when learning some target function such as n, each training example would be a pair of the form (s, n(s)).
  - In reinforcement learning, however, training information is not available in this form.
  - Instead, the trainer provides only a sequence of immediate reward values as the agent executes its sequence of actions.
  - The agent, therefore, faces the problem of temporal credit assignment: determining which of the actions in its sequence are to be credited with producing the eventual rewards.

Characteristics of RL

- **Exploration:**
  - In reinforcement learning, the agent influences the distribution of training examples by the action sequence it chooses.
  - This raises the question of which experimentation strategy produces most effective learning.
  - The learner faces a tradeoff in choosing whether to favor exploration of unknown states and actions (to gather new information), or exploitation of states and actions that it has already learned will yield high reward (to maximize its cumulative reward).

Characteristics of RL

- **Partially observable states:**
  - Although it is convenient to assume that the agent's sensors can perceive the entire state of the environment at each time step, in many practical situations sensors provide only partial information.
  - For example, a robot with a forward-pointing camera cannot see what is behind it.
  - In such cases, it may be necessary for the agent to consider its previous observations together with its current sensor data when choosing actions, and the best policy may be one that chooses actions specifically to improve the observability of the environment.

Characteristics of RL

- **Life-long learning:**
  - Unlike isolated function approximation tasks, robot learning often requires that the robot learn several related tasks within the same environment, using the same sensors.
  - For example, a mobile robot may need to learn how to dock on its battery charger, how to navigate through narrow corridors, and how to pick up output from laser printers.
  - This setting raises the possibility of using previously obtained experience or knowledge to reduce sample complexity when learning new tasks.

Topics

- Introduction
- Learning Tasks
- Q-Learning

Learning Task as MDP

- General formulation of the problem will be done based on Markov Decision Process (MDP).
- In a Markov decision process (MDP) the agent can perceive
  - a set ***S*** of distinct states of its environment and has a set ***A*** of actions that it can perform.
  - At each discrete time step *t*, the agent senses the current state s_t , chooses a current action a_t and performs it.
  - The environment responds by giving the agent a reward $r_t = r(s_t, a_t)$ and by producing the succeeding state $s_{t+1} = \delta(s_t, a_t)$.
- The functions δ and r are part of the environment and are not necessarily known to the agent.
- In an MDP, the functions $\delta(s_t, a_t)$ and $r(s_t, a_t)$ depend only on the current state and action, and not on earlier states or actions.

Learning Task as MDP

- The task of the agent is to learn a ***policy,*** $\prod : S \rightarrow A$, for selecting its next action a_t, based on the current observed state s_t; that is, $\prod(s_t) = a_t$.
- How to specify precisely which policy the agent to learn?
- One obvious approach is to require the policy that produces the greatest possible cumulative reward for the agent over time.

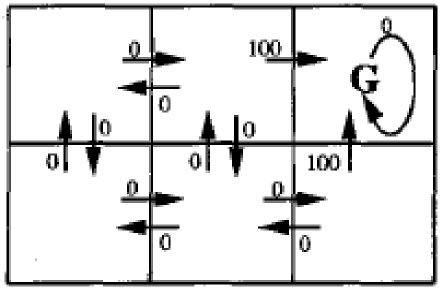
$$
\begin{aligned}
V^{\pi}(s_t) &\equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \\
&\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}
\end{aligned}
$$

- where the sequence of rewards r_{t+i} is generated by beginning at state s_i and by repeatedly using the policy to select actions as described above
- Here $0 \leq \gamma < 1$ is a constant that determines the relative value of delayed versus immediate rewards.
- if we set $\gamma = 0$, only the immediate reward is considered. if we set γ closer to 1, future rewards are given greater emphasis relative to the immediate reward.

- We require that the agent learn a policy $\prod$ that maximizes $V^{\Pi}(s_t)$ for all states s.
- We will call such a policy an optimal policy and denote it by $\prod$*.

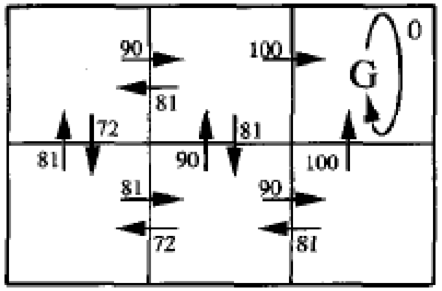
$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$

A simple Example

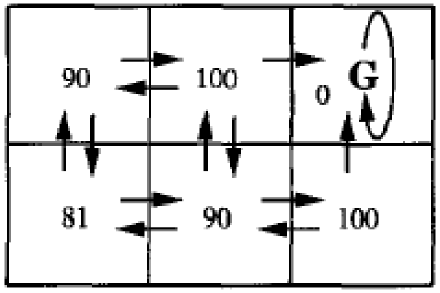


$r(s, a)$ (immediate reward) values

- Each grid square represents a distinct state, each arrow a distinct action. The immediate reward function, ***r(s,*** a) gives reward 100 for actions entering the goal state G, and zero otherwise.

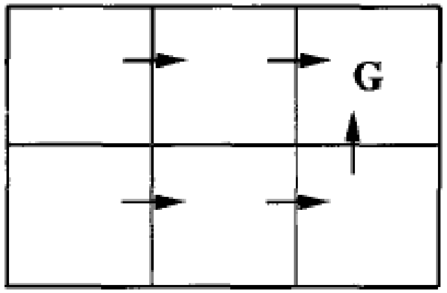


$Q(s, a)$ values



$V^*(s)$ values

- Values of ***V*(s)*** and ***Q(s, a)*** follow from ***r(s,*** a), and the discount factor γ = 0.9.



One optimal policy

- An optimal policy, corresponding to actions with maximal Q values, is also shown.

Topics

- Introduction
- Learning Tasks
- Q-Learning

Q-Learning

- **Included a separate document for Q-Learning topic with solved example**

Topics

- Introduction
- Learning Tasks
- Q-Learning