REINFORCEMENT LEARNING

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.

INTRODUCTION

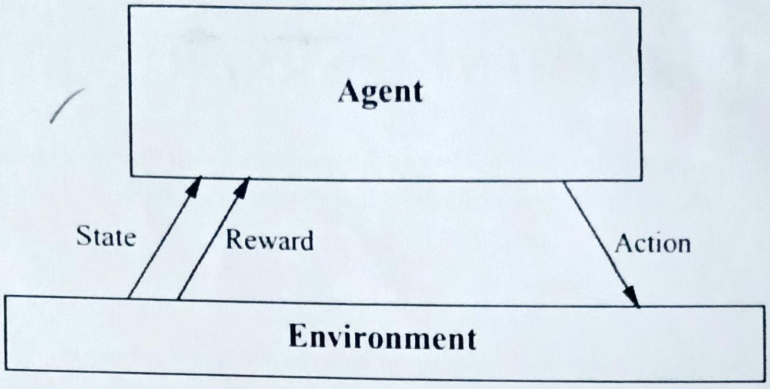
- Considerbuildinga**learningrobot**.Therobot,or***agent***,hasasetofsensorstoobserve the state of its environment, and a set of actions it can perform to alter thisstate.
- Itstaskistolearnacontrolstrategy,or***policy***,forchoosingactionsthatachieveitsgoals.
- The goals of the agent can be defined by a ***reward function*** that assigns a numerical value to each distinct action the agent may take from each distinctstate.
- This reward function may be built into the robot, or known only to an external teacher who provides the reward value for each action performed by therobot.
- The **task** of the robot is to perform sequences of actions, observe their consequences, and learn a controlpolicy.
- The control policy is one that, from any initial state, chooses actions that maximize the reward accumulated over time by theagent.

Example:

- A mobile robot may have sensors such as a camera and sonars, and actions such as "move forward" and"turn."
- Therobotmayhaveagoalofdockingontoitsbatterychargerwheneveritsbatterylevel islow.
- The goal of docking to the battery charger can be captured by assigning a positive reward (Eg., +100) to state-action transitions that immediately result in a connection to the charger and a reward of zero to every other state-actiontransition.

Reinforcement Learning Problem

- Anagentinteractingwithitsenvironment.Theagentexistsinanenvironmentdescribed by some set of possible statesS.
- Agent perform any of a set of possible actions A. Each time it performs an action a, in somestates,theagentreceivesareal-valuedrewardr,thatindicatestheimmediatevalue of this state-action transition. This produces a sequence of states s_i, actions a_i, and immediate rewards r_i as shown in thefigure.
- Theagent'staskistolearnacontrolpolicy,□:S→A,thatmaximizestheexpectedsum of these rewards, with future rewards discounted exponentially by theirdelay.



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

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{ , where } 0 \le \gamma < 1$$

Reinforcement learning problem characteristics

1. **Delayedreward**:Thetaskoftheagentistolearnatargetfunction π thatmapsfromthe current state s to the optimal action a = π (s). In reinforcement learning, training informationisnotavailablein(s, π(s)).Instead,thetrainerprovidesonlyasequenceof 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 eventualrewards.

2. **Exploration:**Inreinforcementlearning,theagentinfluencesthedistributionoftraining examples by the action sequence it chooses. This raises the question of which experimentationstrategyproducesmosteffectivelearning.Thelearnerfacesatrade-off in choosing whether to favor exploration of unknown states and actions, orexploitation of states and actions that it has already learned will yield highreward.

3. **Partially observable states:** 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.Insuchcases.theagentneedstoconsideritspreviousobservationstogether with its current sensor data when choosing actions. and the best policy may be one that chooses actions specifically to improve the observability of theenvironment.

4. **Life-long learning:** Robot requires to 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 newtasks.

THE LEARNING TASK

- Consider Markov decision process (MDP) where the agent can perceive a set S of distinct states of its environment and has a set A of actions that it canperform.
- Ateachdiscretetimestept,theagentsensesthecurrentstates_t,choosesacurrentaction a_t, and performsit.
- The environment responds by giving the agent a reward $r_t = r(s_t, a_t)$ and by producing the succeeding state $s_{t+1} = \delta(s_t, a_t)$. Here 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 oractions.

Thetaskoftheagentistolearnapolicy, $\pi: S \rightarrow A$, forselectingitsnextactiona,basedonthe current observed state s_t; that is, $\pi(s_t) = a_t$.

How shall we specify precisely which policy π we would like the agent to learn?

1. Oneapproachistorequirethepolicythatproducesthegreatestpossible***cumulativereward*** for the robot over time.
   - Tostatethisrequirementmoreprecisely,definethecumulativevalue$V^{\pi}(s_t)$achieved by following an arbitrary policy π from an arbitrary initial state s_t as follows:

$$V^{\pi}(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i} \qquad \textbf{equ (1)}$$

   - Where, the sequence of rewards r_{t+i} is generated by beginning at state s_t andby repeatedly using the policy π to select actions.
   - Here $0 \leq \gamma \leq 1$ is a constant that determines the relative value of delayed versus immediate rewards. if we set $\gamma = 0$, only the immediate reward is considered. As weset γcloserto1.futurerewardsaregivengreateremphasisrelativetotheimmediatereward.
   - The quantity $V^{\pi}(s_t)$ is called the ***discounted cumulative reward*** achieved by policy π from initial state *s*. It is reasonable to discount future rewards relative to immediate rewards because, in many cases, we prefer to obtain the reward sooner rather thanlater.

2. Other definitions of total reward is ***finite horizonreward,***

$$\sum_{i=0}^{h} r_{t+i}$$

Considers the undiscounted sum of rewards over a finite number ***h*** of steps

3. Another approach is ***averagereward***

$$\lim_{h\to\infty} \frac{1}{h} \sum_{i=0}^{h} r_{t+i}$$

Considers the average reward per time step over the entire lifetime of the agent.

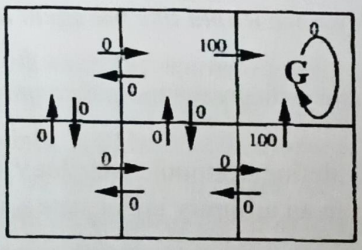
We require that the agent learn a policy π that maximizes $V^{\pi}(s_t)$ for all states s. such a policy is called an ***optimal policy*** and denote it by π^*

$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s) \qquad \textbf{equ (2)}$$

Refer the value function $V^{\pi}*(s)$ an optimal policy as $V^*(s)$. $V^*(s)$ gives the maximum discounted cumulative reward that the agent can obtain starting from state *s*.

Example:

A simple grid-world environment is depicted in the diagram

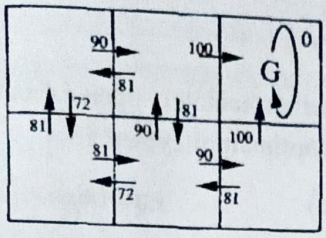


$r(s, a)$ (immediate reward) values

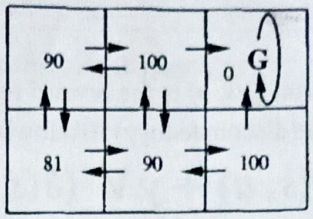
- The six grid squares in this diagram represent six possible states, or locations, for the agent.
- Each arrow in the diagram represents a possible action the agent can take to movefrom one state toanother.
- The number associated with each arrow represents the immediate reward ***r(s, a)*** the agent receives if it executes the corresponding state-actiontransition
- The immediate reward in this environment is defined to be zero for all state-action transitions except for those leading into the state labelled G. The state G as the goal state. and the agent can receive reward by entering thisstate.

Once the states. actions. and immediate rewards are defined. choose a value for the discount factor γ. determine the optimal policy π^* and its value function $V^*(s)$.

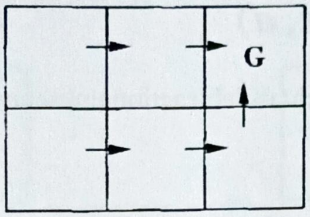
et'schooseγ=0.9.Thediagramatthebottomofthefigureshowsoneoptimalpolicyforthis setting.



$Q(s, a)$ values



$V^*(s)$ values



One optimal policy

Values of V*(s) and Q(s, a) follow from r(s, a), and the discount factor $\gamma = 0.9$. An optimal policy, corresponding to actions with maximal Q values, is also shown.

The discounted future reward from the bottom centre state is

$$0 + \gamma\, 100 + \gamma^2 0 + \gamma^3 0 + \ldots = 90$$

Q LEARNING

How can an agent learn an optimal policy π * for an arbitrary environment?

Thetraininginformationavailabletothelearneristhesequenceofimmediaterewardsr(s_i,a_i) for i = 0, 1,2, ... Given this kind of training information it is easier to learn a numerical evaluation function defined over states and actions, then implement the optimal policy in terms of this evaluationfunction.

What evaluation function should the agent attempt to learn?

One obvious choice is V*. The agent should prefer state s_1 over state s_2 whenever $V^*(s_1) > V^*(s_2)$, because the cumulative future reward will be greater from s_1

Theoptimalactioninstatesistheaction*a*thatmaximizesthesumoftheimmediatereward r(s, a) plus the value V* of the immediate successor state, discounted byγ.

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s, a) + \gamma V^*(\delta(s, a))] \qquad \text{equ (3)}$$

The Q Function

The value of Evaluation function ***Q(s, a)*** is the reward received immediately upon executing actionafromstates,plusthevalue(discountedbyγ)offollowingtheoptimalpolicythereafter

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a)) \qquad \textbf{equ (4)}$$

Rewrite Equation (3) in terms of ***Q(s, a)*** as

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a) \qquad \textbf{equ (5)}$$

Equation(5)makesclear,itneedonlyconsidereachavailableaction***a***initscurrentstate***s***and choose the action that maximizes ***Q(s, a)***.

An Algorithm for Learning Q

- Learning the Q function corresponds to learning the **optimalpolicy**.
- The key problem is finding a reliable way to estimate training values for ***Q***, given only a sequence of immediate rewards ***r*** spread out over time. This can be accomplished through ***iterativeapproximation***

$$V^*(s) = \max_{a'} Q(s,a')$$

Rewriting Equation

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a')$$

- **Q learningalgorithm:**

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s,a)$ to zero.
Observe the current state s
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

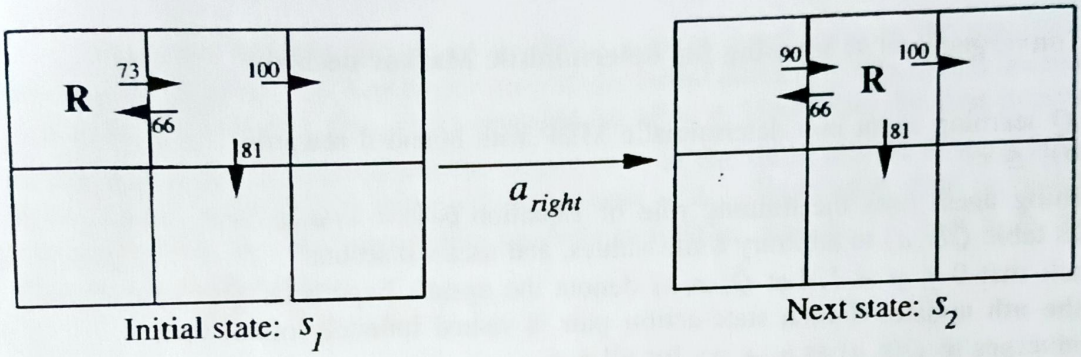
$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- $s \leftarrow s'$

- Q learning algorithm assuming deterministic rewards and actions. The discountfactor γ may be any constant such that $0 \leq \gamma < 1$
- torefertothelearner'sestimate,orhypothesis,oftheactualQfunction

An Illustrative Example

- To illustrate the operation of the Q learning algorithm, consider a single action taken byanagent,andthecorrespondingrefinementto shownbelowfigure



- The agent moves one cell to the right in its grid world and receives an immediate reward of zero for thistransition.
- Apply the training rule ofEquation

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

to refine its estimate Q for the state-action transition it just executed.

- Accordingtothetrainingrule,thenew estimateforthistransitionisthesumofthe receivedreward(zero)andthehighest valueassociatedwiththeresultingstate (100), discounted by γ(.9).

$$\begin{aligned} \hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &\leftarrow 0 + 0.9 \max\{66, 81, 100\} \\ &\leftarrow 90 \end{aligned}$$