Reinforcement Learning

Chapter 5

Rajan Adhikari

Introduction

- Reinforcement learning (RL) is a subfield of machine learning and optimal control concerned with how an intelligent agent ought to take actions in a dynamic environment in order to maximize the cumulative reward.

- Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. l

- Instead the focus is on finding a balance between exploration (of uncharted solution) and exploitation (of current knowledge).

- Reinforcement learning is studied across multiple discipline like game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, and statistics.

Introduction

- Reinforcement Learning is the subfield of machine learning where AI driven system i.e. agent learns from trial and error. It is the science of decision making.
- Agent learn from its mistakes such that for every right action agent is provided a positive feedback and for every wrong action agent is provided a negative feedback.
- The agent decides itself what to do to perform the task correctly.
- Thus, in RL, agent learns and optimal behavior in an environment to obtain maximum reward.
- The agent interacts with the environment and explore the environment by itself guided by continuous feedback mechanism.
- In any environment, agent takes action, may or may not change its state and get feedback.
- Ultimately, the agent learns that what action leads to positive rewards and what action leads to negative rewards.
- RL solves a very specific category of problem where decision making is sequential, and a goal is long-term such as game playing, robotics etc.

Applications of RL

- Self Driving Car
- Industry Automation
- Trading and Finance
- Natural Language Processing
- Physical Sciences
- Gaming etc.

10 Real-Life Applications of Reinforcement Learning (neptune.ai)

Elements of Reinforcement Learning

There are six elements of Reinforcement learning.

- Agent
- Environment
- Policy
- Reward Signal
- Value Function
- Model

Elements of Reinforcement Learning

Agent

- An agent is an entity capable of perceiving and acting in environment with ab ability to learn from its action based on feedback provided.
- A learning agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state.
- The agent must also have some goal or goals relating to the state of the environment.
- The agent must also be able to learn from its own experience.

Elements of Reinforcement Learning

Environment

- Environment refers to the world context in which the RL agent operates.
- It's a model or simulation that the agent interacts with by taking actions, receiving rewards or penalties, and moving to new states based on those actions.
- Reinforcement Learning Environments are essential for training RL agents. They provide the necessary feedback loop, allowing the agent to learn from its actions and improve its policy over time.
- The complexity and diversity of these environments can greatly influence the learning process and the performance of the trained agent.
- Examples of Reinforcement Learning Environments include game simulations (*like Chess, Go, or video games like Atari or StarCraft*), robotic control simulations (like OpenAI's Gym), or even real-world environments (like self-driving cars or automated trading systems).

Elements of Reinforcement Learning

Policy

- A policy defines the learning agent's way of behaving at a given time.
- Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states.
- This is similar to, in psychology, a set of stimulus-response rules.
- In some case, a policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as search process.
- The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior.
- In general, policies may be stochastic, specifying probabilities for each action.

Elements of Reinforcement Learning

Reward Signal

- A reward signal defines the goal of reinforcement learning problem.
- On each time step, the environment sends to the reinforcement learning agent a single number called the reward.
- The agent's sole objective is to maximize the total reward it receives over the long run. The reward signal thus defines what are the good and bad events for the agent.
- The reward signal is the primary basis for altering the policy; if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future.
- Thus, in general, the reward signals may be stochastic functions of the state of the environment and the actions taken.

Elements of Reinforcement Learning

Value

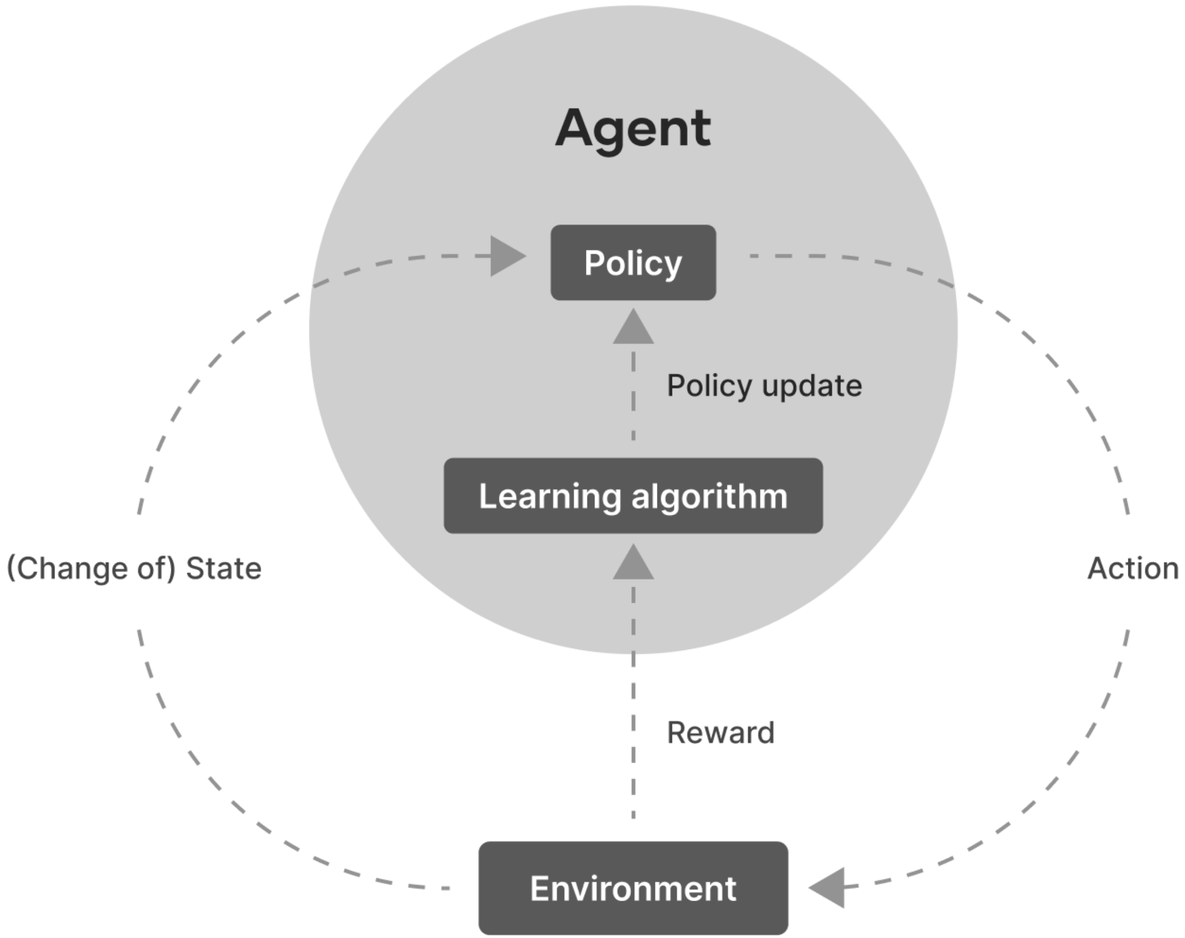
- As the reward signal indicates what is good in an immediate sense, a value function specifies what is good in long run.
- Value indicate the long term desirability of states after taking into account the states that are likely to follow and the rewards available in those states.
- Thus, roughly speaking, the value of a state is total amount of reward an agent can expect to accumulate over the future, starting from that state.
- For example, a state might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards.

Elements of Reinforcement Learning

Model

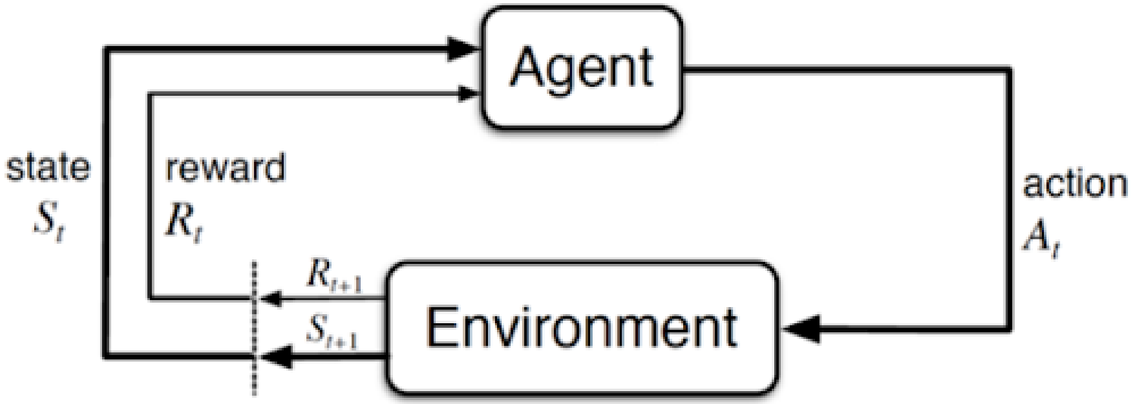
- The final element of the reinforcement learning systems is a model of the environment.
- This is something that mimics the behavior of environment and allows inferences to be made about how the environment will behave.
- For example, given a state and action, the model might predict the resultant next state and next reward.
- Models are used for planning, which means any way of deciding on a course of action by considering possible future situations before they are actually experienced.
- Methods for solving reinforcement learning problems that use models and planning are called model-based methods as opposed to simple model-free methods that are explicitly trial-and-error learners.

General Framework of Reinforcement Learning



Markov Decision Process (MDP)

- A Markov decision process (MDP) is defined as a stochastic decision-making process that uses a mathematical framework to model the decision-making of a dynamic system.
- It is a framework for describing sequential decision making problems.
- A Markov Decision Process (MDP) contains (S, A, P, R, γ) :
  - S represents the set of all states.
  - A represents the set of possible actions.
  - P represents the transition probabilities.
  - R represents the rewards
  - And, γ is known as the discount factor.



Markov Decision Process (MDP)…

In MDP, we start in some state s_0, and get to choose some action $a_0 \in \mathrm{A}$ to take. The result would be MDP randomly transitioning to some successor state s_1.

From the state s_1, another action a_1 is taken and transit to new random state s_2. And then we pick another action a_2 and so on.

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$$

Upon visiting the sequence of states $s_0, s_1, \ldots$ with actions $a_0, a_1, \ldots$ then the total payoff is given as

$$G_t = R_{t+1}(s_0) + \gamma R_{t+2}(s_1) + \gamma^2 R_{t+3}(s_2) + \cdots = \sum_{k=1}^{\infty} \gamma^k R_{t+k+1}$$

where γ is discount rate; $0 \leq \gamma \leq 1$. The discount rate determines the present value of future rewards; a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately.

This means that the reward at time step t is discounted by a factor of γ^t.

Thus in order to achieve highest reward possible, the agent must omit the negative rewards and attain to achieve the positive rewards.

Markov Decision Process (MDP)

Markov Property

The MDP model uses the $Markov\ Property$, which states that the future can be determined only from the present state that encapsulates all the necessary information from the past.

The Markov Property can be evaluated by using this equation:

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, s_2, s_3 \dots \dots s_t]$$

According to this equation, the probability of the next state ($P[s_{t+1}]$) given the present state (s_t) is given by the next state's probability ($P[s_{t+1}]$) considering all the previous states $(s_1, s_2, s_3 \dots \dots s_t)$.

This implies that MDP uses only the present/current state to evaluate the next actions without any dependencies on previous states or actions.

Value and Policy Functions

Policy Function

- A policy function is any function Π that maps from the state to action.

$$i.e. \prod : S \rightarrow A$$

- We say that we are executing some policy Π if, whenever we are in state s, we take action $a = \prod(s)$.

- To determine the best policy, it is essential to define the returns that reveal the agent's rewards at every state.

- We use the discounted infinite-horizon method to find the best policy which basically means if γ has values that are closer to zero, then the immediate rewards are prioritized. Subsequently, if γ reveals values closer to one, then the focus shifts to long-term rewards.

Value and Policy Functions

Value Function

- Value function is the functions of states that estimate how good it is for the agent to be in a given state (how good it is to perform a given action in a given state).
- The notion of how good here is defined in terms of future rewards that can be expected, or , to be precise, in terms of expected rewards.
- It's obvious that the rewards the agent can expect to receive in the future depends on what actions it will take.
- Accordingly, value functions are defined with respect to particular policies. Thus, a value of a state s under a policy π, denoted by $v_\pi(s)$, is the expected return when starting in s and following π thereafter.

Value and Policy Functions

Value Function

Formally,

$$v_\pi(s) = E[G_t|S_t = s] = E_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]$$

We call the function v_π the state-value function for policy π.

Similarly, we define the value of taking action a in state s under a policy π, denoted $q_\pi(s, a)$, as the expected return starting from s, taking the action a, and thereafter following policy π:

$$q_\pi(s, a) = E[G_t|S_t = s, A_t = a] = E_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

We call q_π the action-value function policy π. The value functions v_π and q_π can be estimated from experience..

Value Iteration

- Value iteration is an algorithm for calculating a value function V, from which a policy can be extracted using policy extraction.

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$

Output: Value function V

Set V to arbitrary value function; e.g., $V(s) = 0$ for all s

Repeat

  $\Delta \leftarrow 0$

  For each $s \in S$

    $V'(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} P_a(s' \mid s) \left[r(s, a, s') + \gamma V(s') \right]$ (Bellman equation)

    $\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$

  $V \leftarrow V'$

Until $\Delta \leq \theta$

Value iteration converges to the optimal policy as iterations continue: $V \rightarrow V * \text{ as } i \rightarrow \infty$, where i is the number of iterations. So, given an infinite amount of iterations, it will be optimal.

Value Iteration

- Value iteration converges to the optimal value function asymptotically, but in practice, the algorithm is stopped when the residual $\triangle$ reaches some pre-determined threshold θ – that is, when the largest change in the values between iterations is “small enough”.

Policy Evaluation

- Policy evaluation is an evaluation of the expected reward of a policy.
- The expected reward of policy π from s, $V_\pi(s)$, is the weighted average of reward of the possible state sequences defined by that policy times their probability given π.

$$V_\pi(s) = \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma v_\pi(s')]$$

- Here, $V_\pi(s)$ is not the value of best action, but instead just as the value of $\pi(s)$ but is the value of the action that would be chosen in s by the policy π.

Policy Improvement

- Our reason for computing the value function for a policy is to help find better policies.
- Suppose we have determined the value function v_w for a arbitrary deterministic policy π. For some state s we would like to know whether or not we should change the policy to deterministically choose an action $a \neq \pi(s)$.
- We know how good it is to follow the current policy from s – i.e. $v_\pi(s)$ - but would it be better or worse to change to the new policy?
- Say we select action a in s and thereafter following the existing policy, π then the value of this would be

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}|S_t = s, A_t = 1] = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$$

Policy Improvement…

- So, the key criterion here is whether this value is greater than or less than $v_\pi(s)$ .If it is greater then it would be wise to select a every time s is encountered than thereafter follow π compared to following π all the time.

- This is true is special case of ***policy improvement theorem.***

- The process of making a new policy that improves on an original policy, by making it greedy with respect to the value function of the original policy, is called policy improvement.

- Let π and π' be any pair or deterministic policies such that for all $s \in S$,

$$q_{s,\pi'(s)} \geq v_\pi(s)$$

Policy Improvement…

Then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in S$:

$$v_{\pi'}(s) \geq v_{\pi}(s)$$

This result applies only when

$$\pi'(s) = a \neq \pi(s)$$

Policy Iteration

- Once a policy, π, has been improved using v_π to yield a better policy, π', we can then compute $v_{\pi'}$ and improve it again to yield an even better π''.
- We thus obtain a sequence of monotonically improving policies and value functions:

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

Where, ->E denotes a policy evaluation and ->I denotes a policy improvement. Each policy is guaranteed to be a strict improvement over the previous one.

Policy Iteration

Algorithm

- The policy iteration algorithm finishes with an optimal π after a finite number of iterations, because the number of policies is finite, unlike value iteration, which can theoretically require infinite iterations.

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$
Output: Policy π

Set V^π to arbitrary value function; e.g., $V^\pi(s) = 0$ for all s.

Set π to arbitrary policy; e.g. $\pi(s) = a$ for all s, where $a \in A$ is an arbitrary action.

Repeat
    Compute $V^\pi(s)$ for all s using policy evaluation
    For each $s \in S$
        $\pi(s) \leftarrow \mathbf{argmax}_{a \in A(s)} Q^\pi(s, a)$
Until π does not change

Generalization to Continuous State

Common Reinforcement Algorithms

- **State-action-reward-state-action:** This reinforcement learning algorithm starts by giving the agent what's known as a policy. Determining the optimal policy-based approach requires looking at the probability of certain actions resulting in rewards, or beneficial states, to guide its decision-making.

- **Q-learning:** This approach to reinforcement learning takes the opposite approach. The agent receives no policy and learns about an action's value based on exploration of its environment. This approach isn't model-based but instead is more self-directed.

- **Deep Q-networks**. Combined with deep Q-learning, these algorithms use neural networks in addition to reinforcement learning techniques. They are also referred to as deep reinforcement learning and use reinforcement learning's self-directed environment exploration approach. As part of the learning process, these networks base future actions on a random sample of past beneficial actions.