Machine Learning Assignment

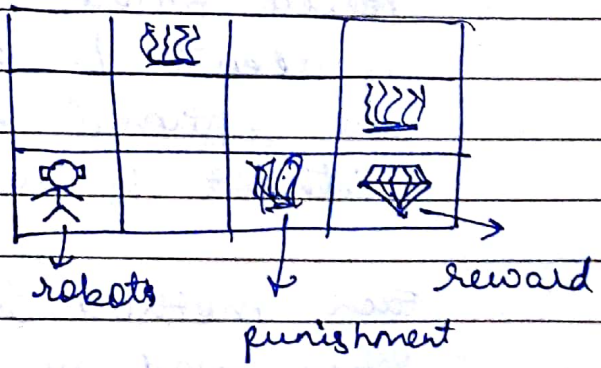
(1) Reinforcement Learning

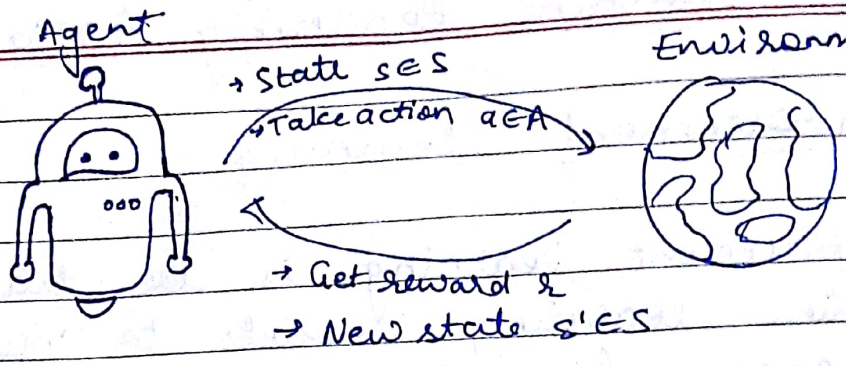
Reinforcement learning is the training of machine learning models to "make sequence of decisions".
The agent learns to achieve a goal in an uncertain, potentially complex environment. In reinforcement learning an artificial intelligence faces a game-like situation. The computer employs trial & error to come up with a solut^n to the problem. To get the machine to do what the programmer wants, the artificial intelligence either gets rewards / penalties for the action its performs. The goal is to maximize the total reward.

Applications of Reinforcement Learning

→ In robotics for industrial automat^n
→ In machine learning & data processing
→ In games

Ex- we have an agent & a reward with hurdles (fire). The agent is to find the best possible path to find the reward





An agent interacts with environment, trying to make smart decisions to maximize rewards.

---

(2) Actor - Critic Model

As we know there are two main types of RL -

a) Value Based :-
It tries to find or approximate the optimal value function, which is a mapping b/w an action & a value. The higher the value, the better the action. The most famous algo is Q-learning & its enhancements like Deep Q-Networks, Double Dueling Q etc.

b) Policy Based :-
Policy based algo based like Policy Gradients & Reinforce try to find the optimal value policy directly without Q-value as middleman

Each method has their advantages. Policy-based are better for continuous & stochastic environments have faster convergence

while value based are more sample efficient & steady.

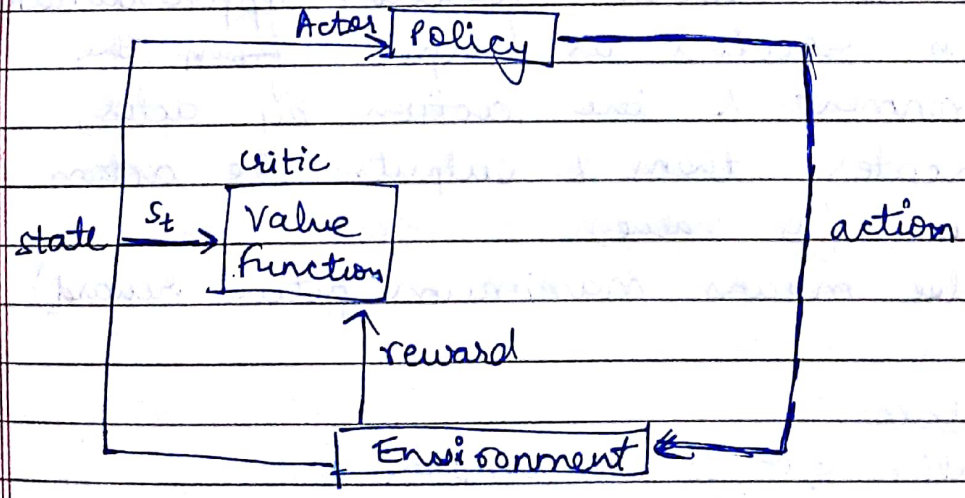
Actor - Critic model merges both the two.

Principal Idea

To split the model into two:

→ one for computing an action based on state & another one to produce the Q values of the action

The actor takes as input the state & outputs the best action. It essentially controls how the agent behaves by "learning the optimal policy" (policy-based). The critic on the other hand "evaluates the action by computing the value function" (value-based). Both two get better over-time.



- Actor:- Decides which action to take
- Critic:- Tells the actor how good its action was & how it should adjust.

Analogy of actor - critic model

Actor (Child)
Critic (Mother)

The child constantly tries new things - eats its toys, touches hot oven, bangs his head on wall. while the mother watches him & either criticizes or compliments him. The child listens to what his mother says & adjusts his behaviour. As child grows, he learns them all.

Actor can be function approximator like neural networks & its task is to produce the best action for a given state.

Critic is another function approximator which receives as input from the environment & the action by actor, concates them & outputs the action value (Q value)
(Q value means maximum future reward).

Algorithm

1. Initialize s, θ, w at random
2. for $t = 1$ to T :-
   a) sample reward r_t & next state s'
   b) Then sample next action
   c) Update policy parameters (θ)

d) Compute the correctness of action-value at time t.
e) Update $a \leftarrow a'$ & $s \leftarrow s'$.

Improvements of Actor-critic :-

a) Advantage Actor-Critic (A2C)
Instead of having critic to learn the Q values, we make him learn the advantage values.

b) ~~The Adv~~. Asynchronous Advantage Actor critic (A3C)
A3C consists of multiple independent agents with their own weights, who interacts with a different copy of the environment in parallel. Thus, they can explore a bigger part of the state-action space in much less time.

---

(3) Q Learning
Q learning is a basic form of RL which uses Q-values (also called action values) to iteratively improve the behaviour of the learning agent.

Algo :- In QL, we initialize the Q, we choose an action & perform it, we evaluate it by measuring the reward & we update the Q accordingly.
As The agent explores the environment, the algo will find the best Q value for each state & action.

Q-values / Action-values

Q values are defined for states & actions. $Q(S,A)$ is an estimate of how good it is to take the action. This estimation of $Q(S,A)$ will be iteratively computed using TD-update rule.

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

S : current state of agent

A : current action picked according to some policy

S' : Next state where the agent ends up

A' : Next action to be picked using current Q-value estimation (maximum)

R : current reward observed from the environment in response of the current action.

γ (>0 & $<=1$) : Discounting factor for future rewards. future reward are less value than current rewards.

α : step length taken to update the estimation of $Q(S,A)$

Q values is a matrix with states as rows & actions as columns. We initialize Q-values randomly, the agent starts to interact with environment & measures rewards for each action. It then computes observed Q values & updates the matrix

1. SARSA

SARSA is an on-policy algorithm where in the current state S, an action A is taken, & the agent gets a reward R, & ends up in next state S1 & takes action A1 in S1. Therefore, the tuple (S, A, R, S1, A1) stands for the acronym SARSA.

It is an on-policy algo because it updates the policy based on actions taken

Diff. b/w SARSA & Q-learning

SARSA chooses an action following the same current policy & updates the Q value.

Q learning chooses a greedy action, that is the action that gives maximum Q-value for the state, that is follows an optimal policy.

Q learning (off policy algo):

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, a_t)]$$

SARSA (on policy algo):

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha (R_{t+1} + \gamma Q(S_{t+1}, a_{t+1}) - Q(S_t, a_t))$$

Algorithm

a) Initialize $Q(s,a)$ $\forall s \in S, a \in A$, arbitrarily & $Q = 0$

b) Repeat (for each episode):

i) Initialize S

ii) choose A from s using policy derived from Q

iii) Repeat (for each step of episode):

1) Take action A, observe R, S'

2) choose A' from S' using policy derived from Q

$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$

$S \leftarrow S'$

$A \leftarrow A'$

until S is terminal.