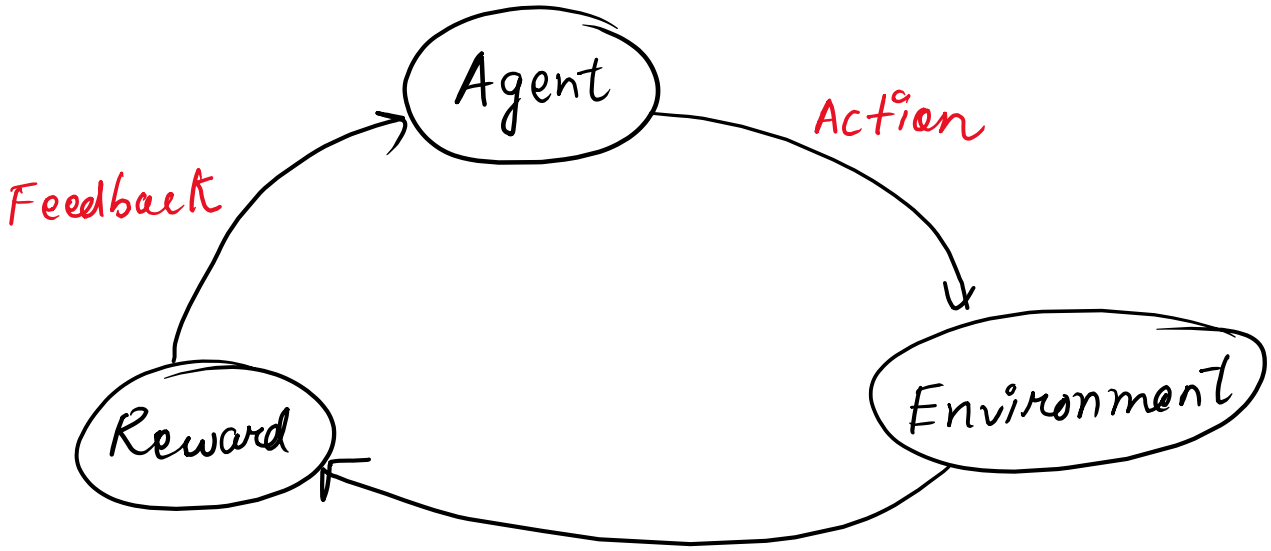
Lecture 1

- Total Lectures on RL : 3
- First Lecture :
  - Introduction to RL
  - Markov Decision Process
  - Terminologies
  - Basic Algorithms
- Second Lecture :
  - Q learning
  - SARSA
  - Function approximation
- Third Lecture :
  - Actor Critic Algorithms
  - Deep Q-learning
  - Model based RL

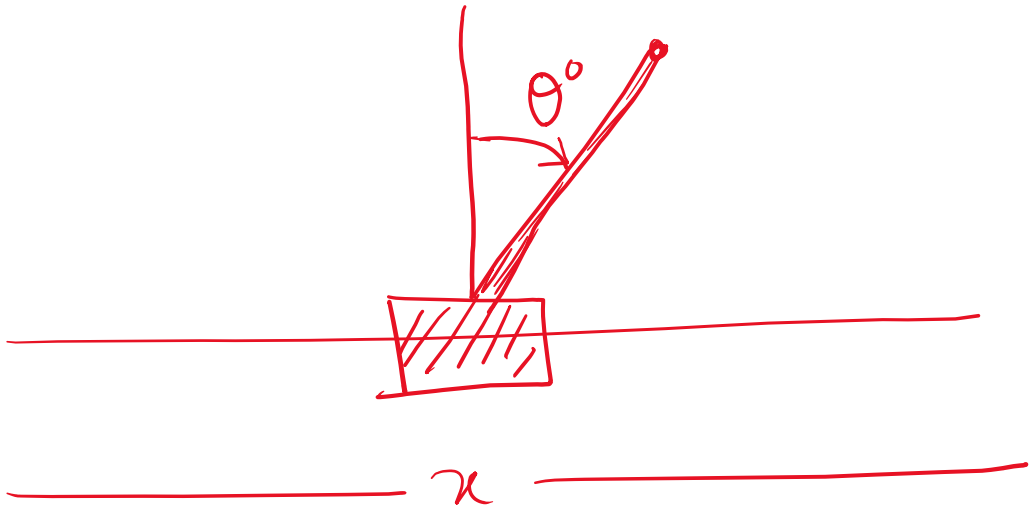
1.1 Introduction to Reinforcement Learning



- Agent takes action. Environment provides feedback and agent improves itself

- In supervised learning, the feedback would be the exact action that should be taken by agent.
- In RL, agent recieves feedback in terms of reward. Reward gives indication whether the action taken was good or bad.

Consider the example of cartpole balance task.



1) Agent applies force to the cart

2) Cart moves in one direction

3) Agent receives reward +1

if $|\theta| < \theta_{thres}$

If we use supervised learning, we need to have data regarding force required for a few values of angle, position, velocity and angular velocity.

1.2 Markov Decision Process (MDP)

An RL problem is described using the framework of MDP. An MDP is described by the following elements:

1) S (states): Set of different situations in an enviroment

2) A (action): Set of actions that can be taken by the agent

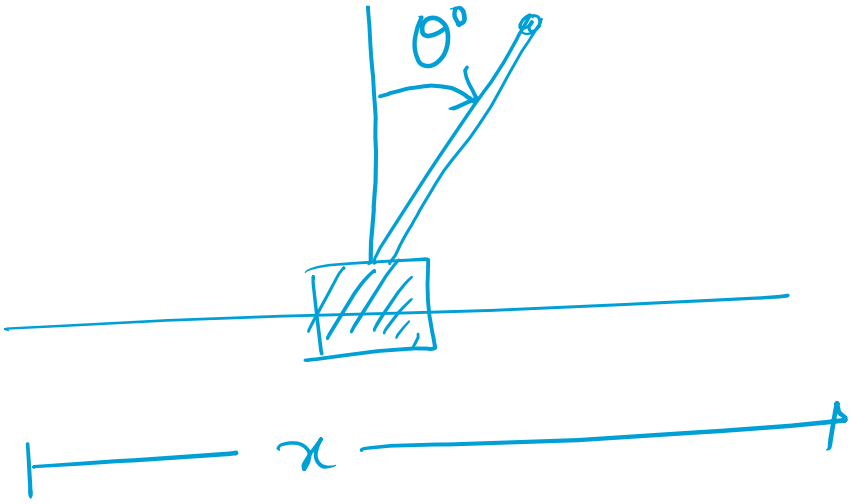
3) R (reward): A function defined as

$$R : (s, a, s') \rightarrow \mathbb{R} \text{ (Real number)}$$

- s: Current state
- a: Action
- s': Next state

4) P (transition dynamics): A probability distribution over states given current state and action

MDP for Cartpole - Balance



State: $(\theta, \omega, x, v) = s$

- θ: Angle
- ω: Angular velocity
- x: Position
- v: Linear velocity

Action: Force applied on the cart. (F)

Reward: $R(s) = +1$ if $|\theta| \leq \theta_{thres}$

Transition Dynamics: Mostly deterministic

$$\frac{\partial^2 x(t)}{\partial t^2} = f(x(t), \theta(t), F)$$

$$\frac{\partial^2 \theta(t)}{\partial t^2} = g(x(t), \theta(t), F)$$

$$x_{t+1} = x_t + \tau v_t \quad \text{---(1)}$$

$$v_{t+1} = v_t + \tau f(x_t, \theta_t, F) \quad \text{---(2)}$$

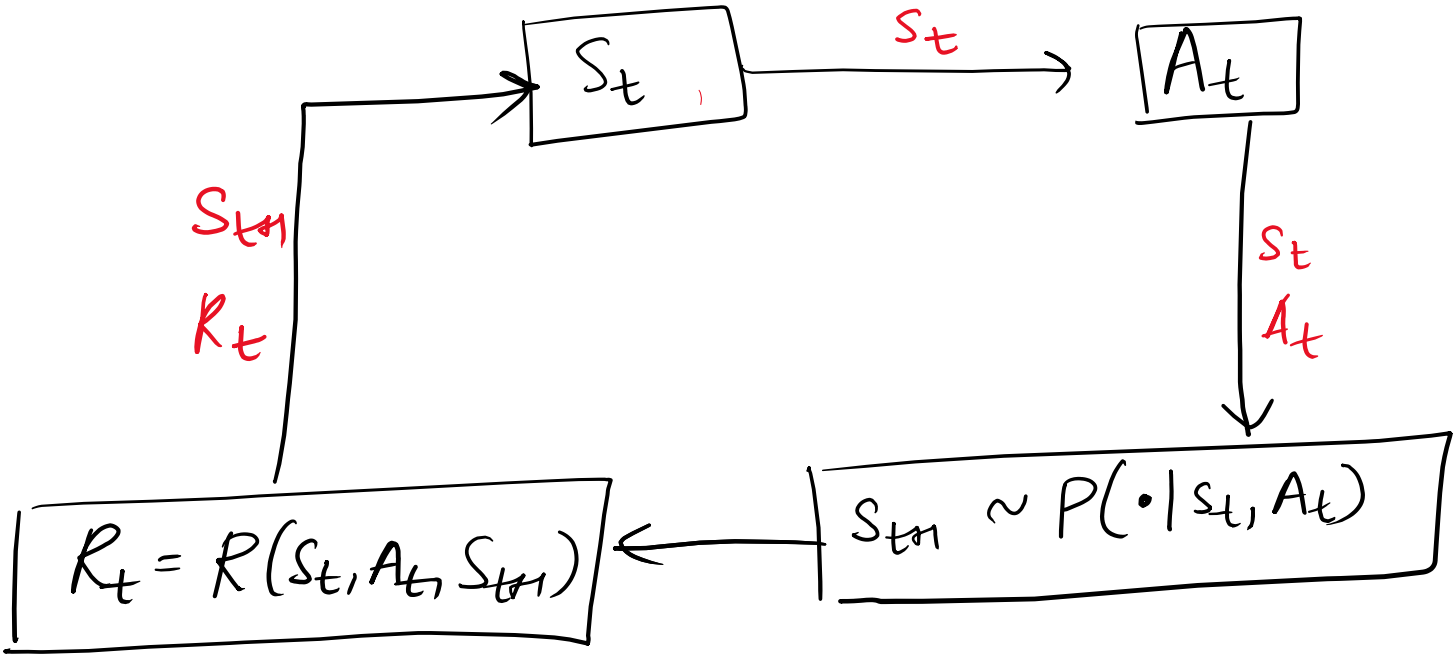
$$\theta_{t+1} = \theta_t + \tau \omega_t \quad \text{---(3)}$$

$$\omega_{t+1} = \omega_t + \tau g(x_t, \theta_t, F) \quad \text{---(4)}$$

- Eq (1)–(4) represents deterministic transition dynamics.
- However, addition of noise to eq (1)–(4) will make the transition dynamics stochastic.
- In case of stochastic transition dynamics next state will be sampled from transition probability distribution

$$(\theta_{t+1}, \omega_{t+1}, x_{t+1}, v_{t+1}) \sim P(\cdot \mid \theta_t, \omega_t, x_t, v_t, a_t)$$

Flow Diagram of MDP



Q) How to select A_t ?

→ RL algorithm is used to learn a decision rule / policy.

→ A policy is either a function of current state S_t or a conditional probability distribution over action space/set.

→ $A_t = \pi(S_t)$ or $A_t \sim \pi(\cdot | S_t)$

→ Policy decides A_t using S_t.

Categories of RL Algorithms

1. Finite horizon
   vs
   Infinite horizon

2. Model based
   vs
   Model free

3. Off-Policy
   vs
   On-Policy

4. Discounted Reward
   vs
   Average Reward
   vs
   Total Reward → finite horizon

   } Infinite horizon only

5. Stationary Policy
   vs
   Non stationary Policy

6. Deterministic Policy
   vs
   Stochastic Policy

Dynamic Programming

- We are in finite horizon and finite state and action space setting.
- Finite horizon problem refers to the task that last for finite number of time steps.
- The set S of states and the set A of action has finite cardinality.
- We have to optimize the following objective function.

$$\forall s \quad \max_{\pi} \; E\left[\sum_{t=0}^{N-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = s\right]$$

where $\pi = [\pi_0, \pi_1, \ldots \pi_{N-1}]$

$\pi_i : S \to A$ (mapping from state space to action space)

Let us define value function $V_0(s)$ as

$$V_0(s) = \max_{\pi} E\left[\sum_{t=0}^{N-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = s\right]$$

$$V_K(s) = \max_{\pi} E\left[\sum_{t=K}^{N-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_K = s\right]$$

- Here we have policy as $\pi = [\pi_0, \pi_1 \cdots \pi_{N-1}]$. The policy changes every time step and hence it is called non stationary.
- If $\pi_0 = \pi_1 = \cdots = \pi_{N-1}$, the policy is called stationary policy.

Let us consider a task with $N = 1$

$$\forall S \quad \max_{\pi} E\left[R(S_0, A_0, S_1) \mid S_0 = S \right]$$

Here we need to select policy such that: $\pi_0(s) = \arg\max_a \sum P(s' \mid s, a) R(s, a, s')$

Let $N = 2$

$$\forall S \quad \max_{\pi} E\left[R(S_0, A_0, S_1) + R(S_1, A_1, S_2) \mid S_0 = S \right]$$

$$V_0(S) = \max_{\pi_0 \pi_1} E\left[R(S_0, A_0, S_1) + R(S_1, A_1, S_2) \mid S_0 = S \right]$$

$$= \max_a \max_{\pi_1} \sum_{s'} P(s'|s,a)\Big(R(s,a,s') + E[R(s_1,A_1,s_2)|s_1=s']\Big)$$

$$= \max_a \sum P(s'|s,a)\Big(R(s,a,s') + \max_{\pi_1} E[R(s_1,A_1,s_2)|s_1=s']\Big)$$

$$= \max_a \sum P(s'|s,a)\Big(R(s,a,s') + V_1(s')\Big)$$

$$V_0(s) = \max_a \sum P(s'|s,a)\Big(R(s,a,s') + V_1(s')\Big)$$

$$\pi_0(s) = \arg\max_a \sum P(s'|s,a)\Big(R(s,a,s') + V_1(s')\Big)$$

Similarly for any N:

$$\forall s \; V_0(s) = \max_a \sum p(s'|s,a) \left(R(s,a,s') + V_1(s')\right)$$

$$\forall s \; \pi_0(s) = \operatorname{argmax}_a \sum p(s'|s,a) \left(R(s,a,s') + V_1(s')\right)$$

$$\forall s \; V_K(s) = \max_a \sum p(s'|s,a) \left(R(s,a,s') + V_{K+1}(s')\right)$$

$$\forall s \; \pi_K(s) = \operatorname{argmax}_a \sum p(s'|s,a) \left(R(s,a,s') + V_{K+1}(s')\right)$$

$$\forall s \; V_{N-1}(s) = \max_a \sum p(s'|s,a) R(s,a,s')$$

$$\forall s \; \pi_{N-1}(s) = \operatorname{argmax}_a \sum p(s'|s,a) R(s,a,s')$$

→ We calculate value functions in the order $V_{N-1}, V_{N-2} \dots V_0$ for all states and policies in the order $\pi_{N-1}, \pi_{N-2}, \dots \pi_0$.

→ We are using solution for a smaller problem ($V_{K+1}(s)$) to find solution of larger problem ($V_K(s)$). Hence paradigm of dynamic programming is being used.

→ Important point to note over here is that access to $R(s, a, s')$ and $P(s'|s, a)$ is assumed. Hence dynamic programming based method is model based.

Infinite Horizon Methods

- In infinite horizon the value of timesteps starts from 0 and goes upto ∞.
- The following is the objective function used for infinite horizon task:

$$\forall s \quad \max_{\pi} \; E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = S\right]$$

- Here π is stationary. Therefore $\pi_0 = \pi_1 = \pi_2 \ldots$.
- Further we use a discounting factor with the reward function ($\gamma < 1$)
- Let us assume $R(s, a, s') \leq L$.

$$E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = S\right] \leq \sum_{t=0}^{\infty} \gamma^t L$$

$$\leq \frac{L}{1-\gamma} < \infty$$

- Therefore because of discounting the objective function is bounded!

• we define value function as below:

$$V(s) = \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t, S_{t+1}) \,\middle|\, S_0 = S\right]$$

$$V(s) = \max_{\pi} \sum_{s'} P(s'|s, \pi(s)) \Big(R(s, \pi(s), s') + \gamma E\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_1 = s'\right] \Big)$$

$$= \max_{a} \sum P(s'|s,a) \Big(R(s,a,s') + \gamma \max_{\pi} E\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(S_t, A_t, S_{t+1}) \,\middle|\, S_1 = s'\right] \Big)$$

$$V(s) = \max_{a} \sum P(s'|s,a) \Big(R(s,a,s') + \gamma V(s') \Big)$$

$$\pi(s) = \operatorname*{argmax}_{a} \sum P(s'|s,a) \Big(R(s,a,s') + \gamma V(s') \Big)$$

→ This is the bellman equation.

- We will discuss two algorithms for infinite horizon task.

  1) Value Iteration

  2) Policy Iteration

Value Iteration Algorithm

- We take an initial guess of value function $V_0(s)$ (for all states) and use the iterative algorithm to arrive at the value function which satisfies bellman equation
- Algorithm:

1) Initialize the value function as

$V_0(s)$

2) Perform the following updates until convergence

$$V_{K+1}(s) = \max_a \sum p(s'|s,a)\left(R(s,a,s') + \gamma V_K(s') \right)$$

3) Optimal policy is given as follows:

$$\pi^*(s) = \operatorname{argmax}_a \sum p(s'|s,a)\left(R(s,a,s') + \gamma V^*(s') \right)$$

Comments:

- Optimal value function follows the bellman equation i.e.:

$$\forall s \; V^*(s) = \max_a \sum P(s'|s,a) \left(R(s,a,s') + \gamma V^*(s') \right)$$

- The policy obtained is deterministic and stationary.
- Access to reward function and transition probability is assumed. Hence value iteration is model based algorithm.

Policy Iteration Algorithm

1. Initialize an arbitrary function as $V_{-1}(s)$ for all the states.

Obtain initial policy π_0 as follows:

$$\forall s \quad \pi_0(s) = \underset{a}{\operatorname{argmax}} \sum p(s'|s,a) \left(R(s,a,s') + \gamma V_{-1}(s') \right)$$

2. Let $V_{k-1} = V_{k-1,0}$

$$\forall s \quad V_{k-1,1}(s) = \sum p(s'|s, \pi_k(s)) \left(R(s, \pi_k(s), s') + \gamma V_{k-1,0}(s) \right)$$

→ The above operation is performed until convergence.

→ At convergence we have

$$\forall s \quad V_{k-1,\infty}(s) = \sum p(s'|s, \pi_k(s)) \left(R(s, \pi_k(s), s') + \gamma V_{k-1,\infty}(s') \right)$$

→ Let $V_{k-1,\infty} = V_k$

→ Step 2 is called policy evaluation step. We obtained the value function V_k corresponding to the policy π_k

(3)

$$\forall s \quad \pi_{k+1}(s) = \arg\max_a \sum p(s'|s,a)\left(R(s,a,s') + \gamma V_k(s')\right)$$

→ This step is called policy improvement step.

(4) Repeat step 2 and 3 in alternation untill there exist state s such that $\pi_k(s) \neq \pi_{k+1}(s)$.

Comments:

1) Upon convergence the policy iteration algorithm obtains optimal policy

2) State and action space is finite

3) Policy is deterministic and stationary

4) Policy iteration is model based algorithm.

Conclusions

1) Introduction about RL framework.

2) Description of Markov Decision Process with example of cartpole balance task

3) Dynamic Programming based algorithm for finite horizon

4) Value iteration and policy iteration algorithm for infinite horizon

5) Stationary vs non stationary policy

6) Model based v/s model free algorithm

To be covered next

1) Model free algorithms based on policy iteration and value iteration

2) Infinite space algorithms

3) Off-policy v/s on-policy algorithms