RL - unit 3

## Reinforcement Learning Problem

It is about training an agent to make a sequence of decision in an environment to maximize a cumulative reward.

### Key Elements

- Agent.
- Environment.
- State.
- Action.
- Reward
- policy
- value function.
- Q-value

Agent

1. observe state

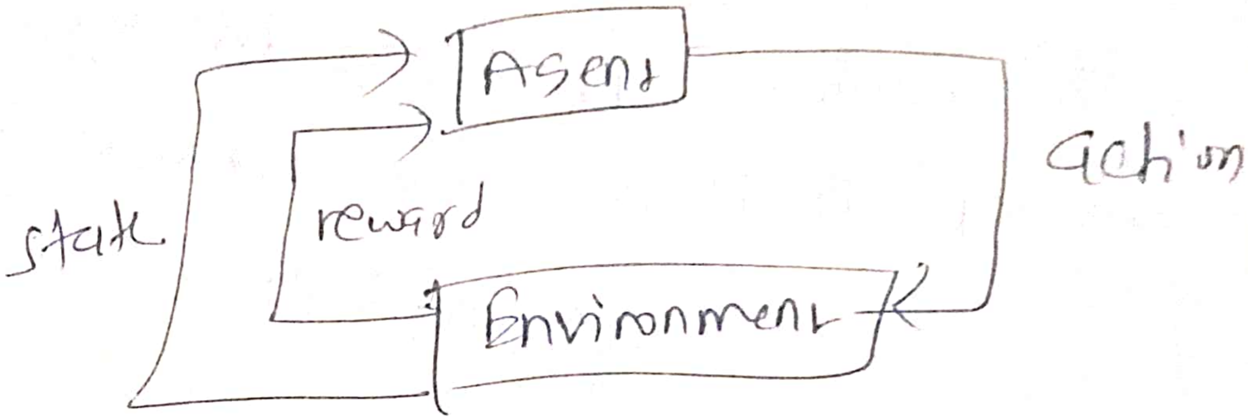
2. choose Action

Action →

Environment

3. Provide Reward

4. New state




## Applications

- → Game playing
- → Robotics
- → self driving cars
- → Recommendation systems.

# Prediction and control problems

RL is a subfied of ML where an asent learns to make decision by interactions with an environment. The two fundamental problems in RL are prediction and control.

- RL methods
  - Prediction problems
    - model-Based
      - Dynamic programming
    - model-free
      - Temporal Difference learning
      - Monte Carlo methods
  - control problems
    - model based
      - Policy iteration
      - value iteration
    - model free
      - Q-learn
      - SARSA
      - Ex SARSA

## Prediction Problem

estimation the value function or the expected return for a given policy. The goal is to evaluate how good a particular policy is by predictions the feature reward, the agent will receive.

fixed policy $\pi$

↓

Estimate value function $v(s)$

↓

compute Expeted Returns

goal is to evaluate Perform-ance of Existing Policy

Ex:-

- weather forecasting
- stock market
- Demand forecasting in economics.

## Control problem

in RL involve finding the optimal policy $\pi^*$ that maximizes the expected cumulative reward over time. The goal is to improve the policy iteratively to acheive the best possible performance.

Policy improvement

↓

maximize cumulative Reward

↓

find optimal policy $\pi^*$

Goal is to Discover Best policy to maximize Rewards.

Ex :-
→ Autonomous vechile navigation
→ industrial Process optimization.

| Prediction modeling | Control system modeling |
| --- | --- |
| General form:- | Feedback control Equation :- |
| $Y = f(x) + \epsilon$ | $u(t) = k[r(t) - y(t)]$ |
| $Y$ = predicted outcome | $u(t) \rightarrow$ control input |
| $x$ = input features | $r(t) \rightarrow$ Desired reference state |
| $\epsilon \rightarrow$ Error term | $y(t) \rightarrow$ current system state |
| | $k \rightarrow$ feedback gain. |

# Model based algorithms

This are class of methods that rely on learning a model of the environment to make decisions. unlike model-free algorithms, which directly learn a policy or value function without explicitly modeling the environment.

## Key components

1. → model learning

It learns a model of the environment,

→ transistion model

↳ Predicts the next state given the current state and action

$P(s'/s, a)$.

Reward model

↳ predicts the reward for a given state-action pair, R(s,a).

→ The model is learned using supervised learning techniques, such as regression or neural networks.

② <u>planning</u>

once the model is learned, the agent uses it to simulate trajectories or plan actions.

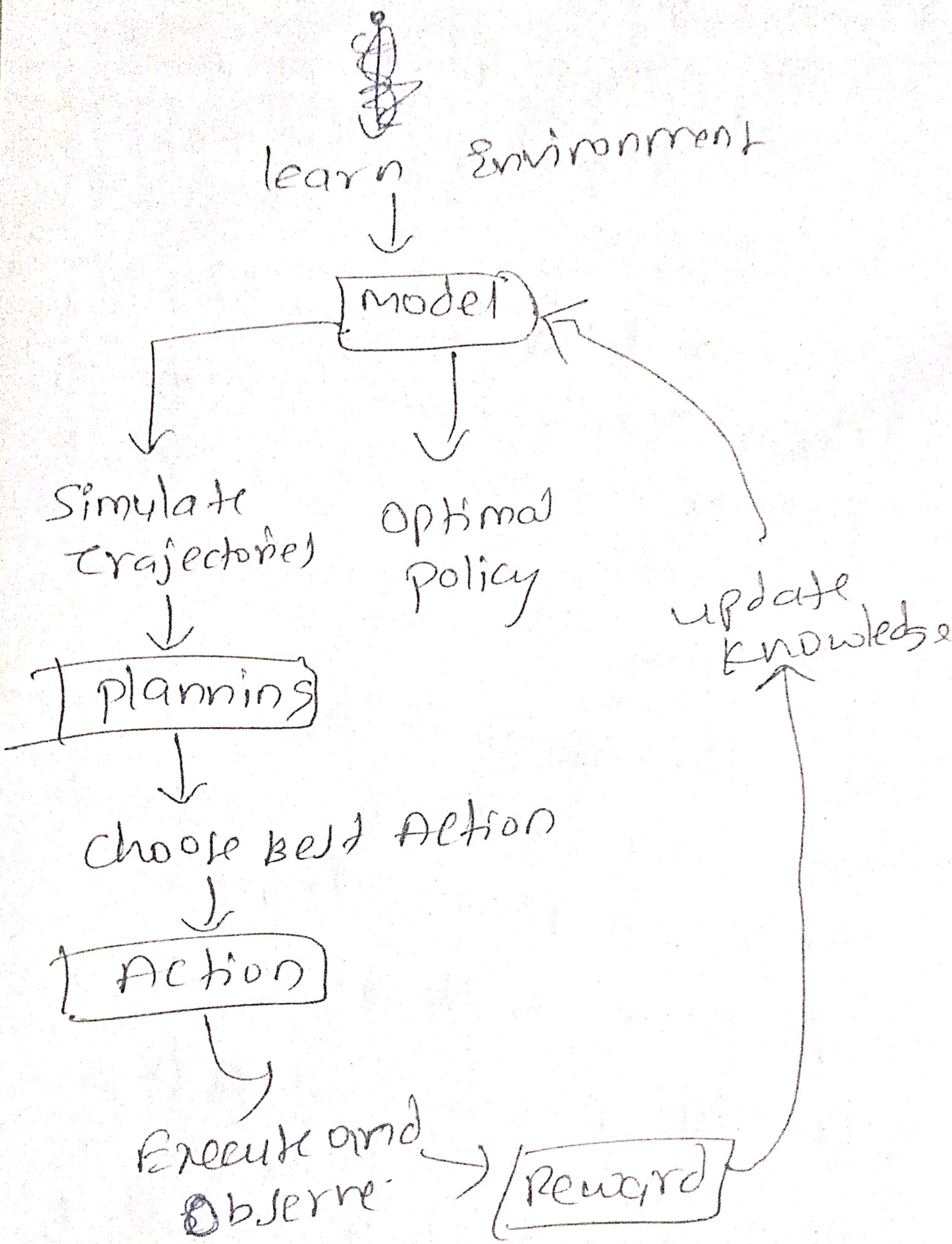
→ planning methods include

↳ value Iteration or policy iteration

↳ Model predictive control.

③ <u>policy improvement</u>

the agent improves its policy based on the planned actions.


learn Environment
Model
Simulate trajectories
Optimal policy
update knowledge
planning
choose best Action
Action
Execute and Observe
Reward

Advantages

→ sample Efficiency
→ flexibility

Disadvantages

→ computational cost
→ complexity.

Ex:-

→ Dyna-Q.
→ model-Based policy optimiza-tion (MBPO).
→ world models.

Monte carlo Methods for Prediction

→ this algorithms in RL is used for prediction and control. They relay on sampling ~~comput~~ complete episodes of interaction with environment to estimate value functions or policies.

→ unlike TD methods, Monte Carlo require complete episode to compute returns, making them ~~interleved~~ inherently episodic.

Key concepts

① Episode.

Monte Carlo methods operate on complete episodes. which are sequen of state, action, Rewards from initial to terminal state.

$S_0, A_0, R_1, S_1, A_1, R_2, \ldots S_T$

→ Episode should must termi-nate for monte carlo method work

## 2. Return

The Return $G_t$ is the total discounted reward from time step $t$ onward.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

## 3. value function Estimation

The value function $v_\pi(s)$ is the expected return when starting in state $s$ and following policy $\pi$.

$$v_\pi(s) = E[G_t | S_t = s]$$

## Monte carlo prediction Alg

The goal is to estimate $v_\pi(s)$ for a given Policy $\pi$.

① Initialize.

$V(S) \rightarrow$ A table to store the value function estimates for all states $s$.

+ Returns (s): ~~All s Ahtst of~~

↳ A list to store the returns observed for each state s.

② Generate Episodes

↳ use the policy π to interact with the environment and generate episodes.

③ Compute Returns

for each episode, compute the return $G_t$ for each state $S_t$ visited in the episode.

④ update value functions

For each state $S_t$ in the episode append the return $G_t$ to Return($S_t$)

→ update the value estimate $V(S_t)$ as the average of all returns in Returns($S_t$).

(5) Repeat

Repeat the process until the value estimate converge.

Generate Episode

↓

Episode Generation

↓

Compute Return

↓

Return Calculation

↓

update value Estimates

↓

value Estimation

↓

Refine Policy

↓

policy improvement

↓

converge to optimal value.

## Online implementation of Monte carlo policy evaluation

Monte carlo Policy evaluation is a method used to estimate the value function $v^{\pi}(s)$ for a given policy $\pi$ by averaging the returns observed from sampled episodes.

→ The online implementation of monte carlo policy evaluation updates the value function incrementally as new episodes are generated, rather than waiting process all episodes in a batch

Key Steps

① Initialization

→ initialize the value function V(s) for all state s arbitrarily
$V(s) = 0.$

→ Initialize a counter N(s) for each state s to keep track of the number of times the state has been visited.

② Generate Episodes

↳ simulate episodes using the policy π, each episode is a sequence of states, actions, rewards

③ Compute Returns

for each state $s_t$ in episode, compute the return $G_t$, which is the cumulative discounted reward from time t onward.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T$$

④ update value function

For each state $S_t$ in the episode, update the value function incrementally using the observed return $G_t$:

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} \left(G_t - V(S_t)\right)$$

⑤ Repeat

Repeat the process for multiple episodes until the value function $V(S)$ converges to the true value function $V^{\pi}(S)$.

| First-visit Monte Carlo | Every-visit Monte Carlo |
|---|---|
| * updates each state once per episode | * updates each state every time it appears. |
| * unbiased, higher variance | * slightly biased, lower variance. |
| * lower (fewer updates) | * Higher (more updates). |

Ex:-

consider an episode with sequence

$S_0, S_1, S_2, S_1, S_3$

First-visit MC → Update $V(S_1)$ only for the first occurrence of $S_1$

Every-visit MC → Update $V(S_1)$ for both occurrences of $S_1$.

Start Episode

↓

$\boxed{\text{Episode}}$

↓

Collect Returns

↓

$\boxed{\text{Returns}}$

↓

update v(s)

↓

$\boxed{\text{values}}$

↓

improve

↓

$\boxed{\text{policy}}$

↓

converse