Lec-0 | 05-08-2025

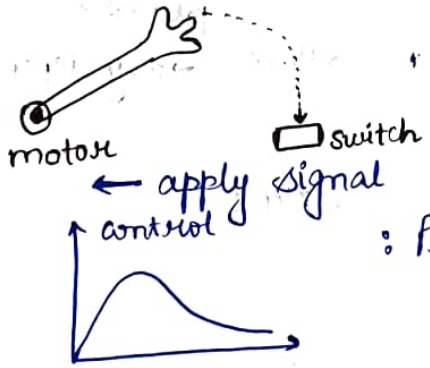
Reinforcement Learning (RL) in Practice

→ Reward-based learning
→ Successful completion of a task
→ Positive feedback
→ Machine moves on to the next track
(a kind of trajectory tracking)

Usage:

LLMs - Large Language Models
DeepSeek - RLHF
DeepMind - AlphaZero, Alpha star, Alpha Fold } read about
Boston Dynamics - robotics application

Think about a robot.
How to make a robot do a "task"?



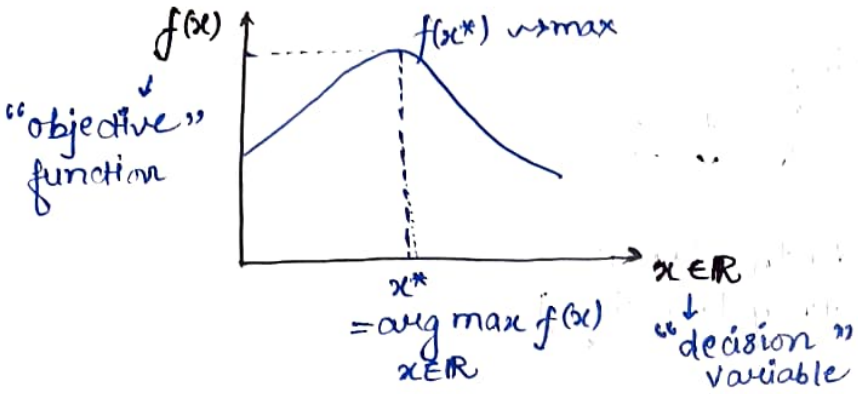
: Programming

"Method to make this robot learn how to move its arm to achieve a task via reward feedback."

Formal definition:

Achieving an objective is finding maximum of a function
↳ Optimization problem

Optimization

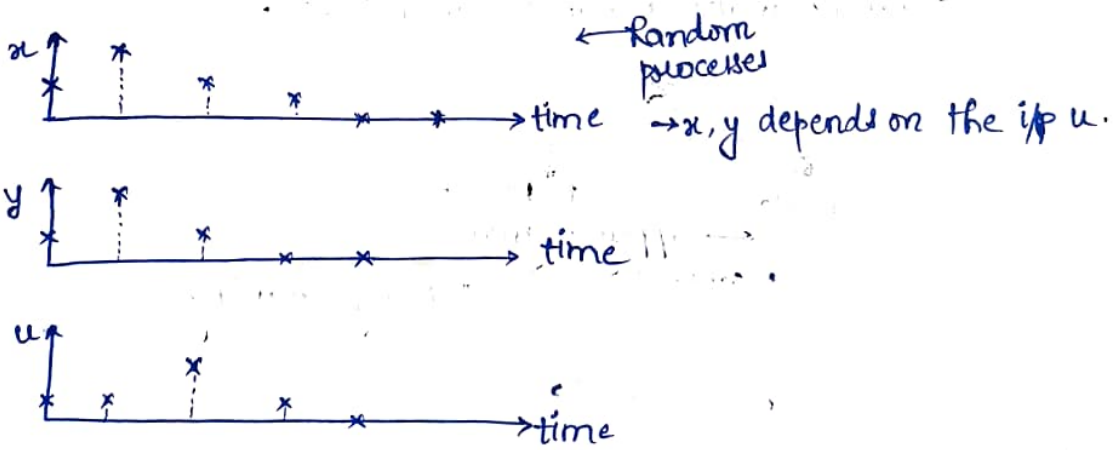


Eg. Maximize $2x_1 + 3x_2$,
$x_1 \geq 0,\ x_2 \geq 0$
$x_1 + x_2 \leq 2$
} A type of optimization problem

→ RL is a method to solve sequential optimization problems.

Sequential Optimization (Decision) Problems ⊆ Optimization problems

↓ notion of time



← Random processes

→ x, y depends on the i/p u.

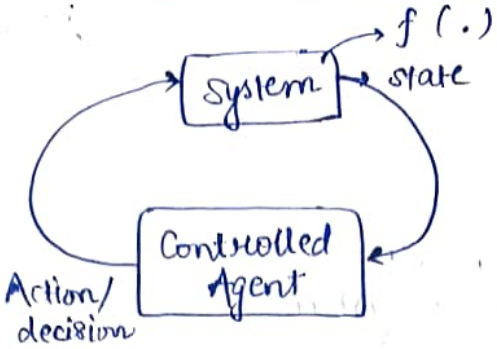
max f (state evolution)

seq. decision

$$\frac{dx}{dt} = g(x, u)$$

Proportional (P) controller : $\frac{dx}{dt} = g(x) + K \cdot x$

initial state



Optimal control theory

R.L: Mathematical framework of optimizing a system without knowing what the system is.

↳ Learns the properties of system.

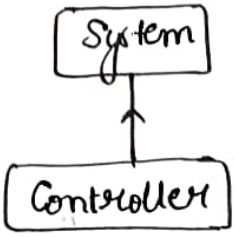
↳ Uses machine learning and deep learning.

Course: 40 % theory + 60 % Hands-on [From control perspective]

Prerequisites: - Probability and random processes
- Programming (Python)

Evaluation: - No written exams (No endsem exams!)
- Programming lab exam (no internet!)
- Class tests
- Project (RL implementation)

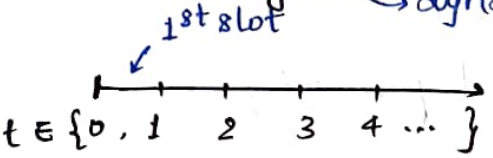
Reinforcement Learning



Learn how to control a system.

Discrete time systems:

↳ dynamic



Dynamical Systems

↳ Objects having state which is a function of time.

→ Discrete-time system: discrete time, t, function.

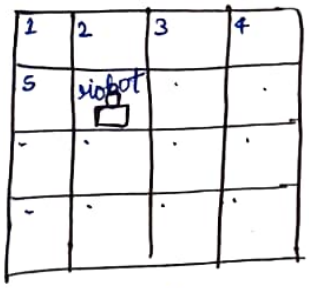
→ state
↳ vector $\in \mathbb{R}^d$

S_t → notation

$S_t \in S$ → state space
↳ fixed

S_t, $S(t)$
↓ discrete ↓ continuous

Eg. ⓐ Grid world



$A = \{n, s, e, w\}$

$S = \{1, \ldots, 16\}$

ⓑ

$\mathbb{R}^2 = S$

How does the state change with time?

Control inputs:

At every time t, apply some control input or action, A_t.

$A_t \in \mathcal{A}$ (action space)

Model: Evolution of state (time-variant evolution)

$$S_{t+1} = f(S_t, A_t),$$

f: transition function

↳ can be represented as table

S_t	A_t	S_{t+1}
1	e	2
1	n	1

[Gridworld]

Eg. Bike moving along x-direction.

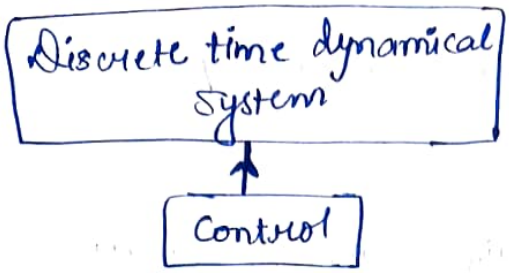
$$S_t = \begin{pmatrix} x_t \\ v_t \\ a_t \end{pmatrix}, \quad F, m$$

$$S_{t+1} = \begin{pmatrix} x_{t+1} \\ v_{t+1} \\ a_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + v_t + \frac{1}{2}a_t \\ v_t + a_t \\ a_t + \frac{F}{m} \end{pmatrix}$$

$$S_{t+1} = \underbrace{\begin{pmatrix} 1 & 1 & 1/2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_t \\ v_t \\ a_t \end{pmatrix}}_{S_t} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ F/m \end{pmatrix}}_{A_t}$$

$\therefore$ $\boxed{S_{t+1} = AS_t + A_t}$

Review



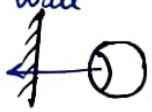
State - $S_t \in S$

Action - $A_t \in A$

Transitions - $S_{t+1} = f(S_t, A_t)$

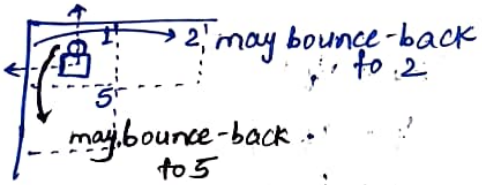
↑ transition fn

Eg



may not stop in time & bounce back.

Gridworld:



Eg $S_{t+1} = \hat{A} \cdot S_t + A_t$

↓ diff. matrix

$S_{t+1} = A \cdot S_t + A_t + \epsilon_t$

↳ error
↳ non-deterministic

In many scenarios,

S_{t+1} is not deterministic, given S_t, A_t.

Model S_{t+1} as a random variable - dependent on S_t & A_t.

Start from a S_0.

S_1, S_2, etc are all RVs.

For grid world,

Suppose $S_0 = 1$, $A_0 = n$.

S_1 has a conditional pmf with

$$P_r\{S_1 = i \mid S_0 = 1, A_0 = n\}$$

[$P_r\{S_1 = i \mid S_0 = 1, A_0 = w\}$ is a different pmf.]

Transitions:

$$S_{t+1} = F(S_t, A_t)$$

↓ not deterministic

[change of notation]

For now, we will assume that S, A are discrete (finite) sets.

$$S_{t+1} = F(S_t, A_t)$$

$$\equiv P_r\{S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t\}$$

→ can change with time or for simplicity take as time-invariant.

- Time invariant PMF does not change with time.

Policy: → Tells how to control the system.

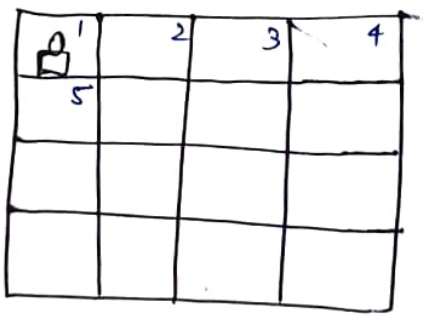
Sequence of control actions $(A_0, A_1, A_2, \ldots)$.

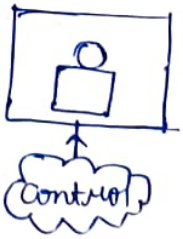
↳ A function that looks at the current state and decide what action to take. [A ~~kin~~ type of policy – this one is the usual.]

$$\pi(S_t) = A_t$$

↳ May not depend on any state, or may depend on past actions.

Eg.





Policy eg.,

S_t	$\pi(S_t) = A_t$
1	e
2	e
3	e
4	s

→ deterministic fn of current state

Random (stochastic) policies :

$$P_r\{A_t = a \mid S_t = s\}$$

We have two kinds of prob. : Probability of deciding an action → then Probability of deciding the next stage.

Trajectory: Set of states and actions.

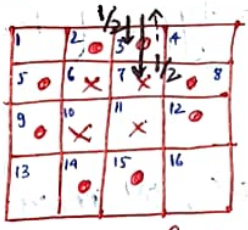
Horizon: stopping condition

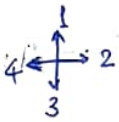
$S_0, A_0, S_1, A_1, S_2, A_2$ → infinite horizon

$\pi(S_0)$ $\pi(S_1)$

S_{T-1} (T-horizon) ⇝ stop at (T-1).
↳ finite

H.W Simulate a trajectory of a robot moving in the grid-world.





Transition probability model

π-policy

$$P_r\{A = d \mid S_t = s\} = 1/4$$

$S_0 = 1$; current state $= S_0$

for {

current A = sample an action based on $P_r\{A = a \mid$ current state $\}$

next state = sample from $P_r\{S_{t+1} = s_{t+1} \mid S_t =$ current state, $A_t =$ current $A\}$

current state = next state

} store in array to generate trajectory

Recall

Discrete time dynamical system:

State - S_t, $t \in \{0, 1, \dots\}$

State space - $\mathcal{S}$

Action - A_t

Action space - $\mathcal{A}$

Evolutional Transition:

$$S_{t+1} = F(S_t, A_t)$$

$$P_r\{S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t\}$$

Policy:

deterministic: $A_t = \pi(S_t)$

stochastic: A_t

$$P_r\{A_t = a_t \mid S_t = s_t\}$$

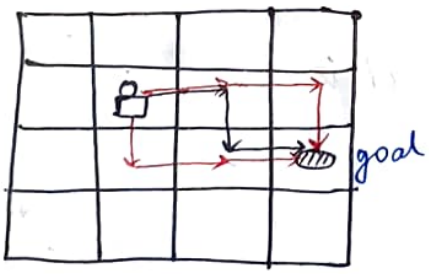
Trajectory:

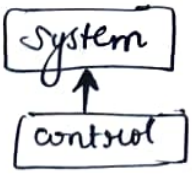
$S_0, A_0, S_1, A_1, \dots, S_{T-1}$: finite horizon

$\dots \infty$: infinite horizon

Sample Path: Another term for trajectory.

Rewards and Costs





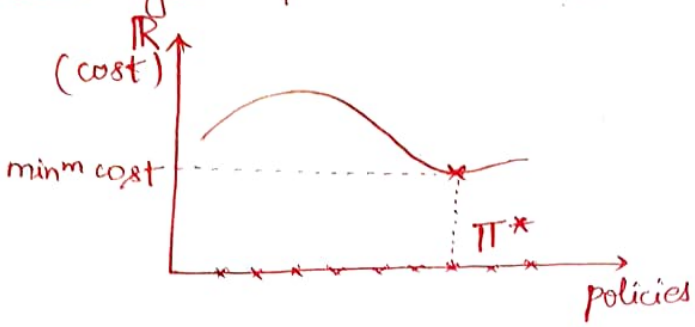
Task/Objective/Goal:

Example: Reach the goal in minimum no. of steps.

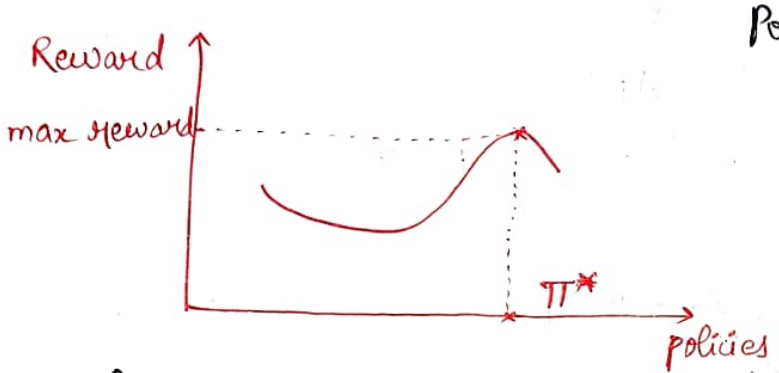
↳ Which policy is the best?

↳ Mathematical model?

Ranking the policies (although they're complex mathematical objects):



Policy with lower value is better: cost



Policy with higher value is better: Reward

Duality of cost & Reward: Minm cost occurs at max reward.

↳ Negative of cost is reward.

↳ Usually reward is used in RL.

Stage-wise Rewards:

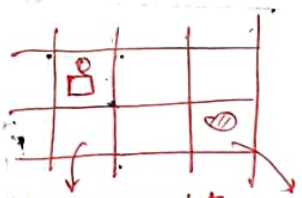
Example for grid-world:

At every time 't', there is a reward.

$$R_t = R(S_t, A_t), \quad \ldots$$

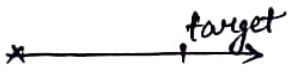
and for the policy, reward is

$$\sum_{t=0}^{} R_t$$



Eg. forevery step other than the target, there is a reward of '-1'. '0' reward for target

Eg.



$$S_{t+1} = AS_t + A_t$$

$$R_t = -c_t = -|S_t - S_{target}|^2$$

Trajectory:

$S_0, A_0, R_0, S_1, A_1, R_1, \ldots$ → all are random variables

π ↑

Rewards will be random (in general) even when R fn is deterministic.

In general,

$R \rightarrow$ stochastic

$$Pr\{R_t = r_t \mid S_t = s_t, A_t = a_t\}$$

$\sum_{t=0} R_t \rightarrow$ scalar but random (sum of RVs)

To rank the policies, we need a single value, so take average value of reward (expectation):

$$\mathbb{E} \sum_{t=0} R_t$$

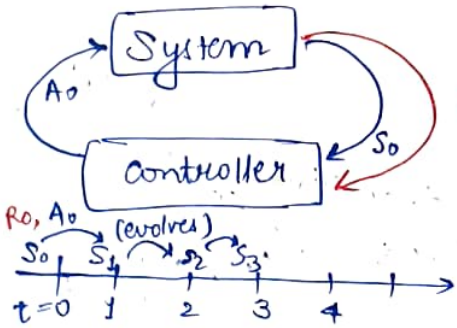
for infinite horizon,

$\sum R_t$ may be infinite, so not possible to rank two policies.

- Discounted rewards:

$$\mathbb{E}\left[\sum_{t=0} \gamma^t R_t\right], \quad \gamma \in (0,1).$$

Lec-4 | 19-08-2025



$A_t = \pi(S_t)$

$Pr\{A_t = a \mid S_t = s\}$

$\boxed{\pi(a|s)}$

$S_{t+1} = F(S_t, A_t)$

↑

$Pr\{S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t\}$

$\boxed{P(S_{t+1} \mid s_t, a_t)}$ → shortcut

$R_t = R(S_t, A_t)$

Trajectory: $(S_0, A_0, R_0, S_1, A_1, R_1, \dots)$

$$\begin{pmatrix} S_0 & S_1 & \\ A_0 & A_1 & \dots \\ R_0 & R_1 & \end{pmatrix}$$

Return: Total rewards along a trajectory

↳ $\sum_{t=0} R_t$ ← random

$\mathbb{E}\left(\sum_{t=0} R_t\right)$: expected return

Discounted return: For infinite horizon

$\sum_{t=0}^{\infty} \gamma^t R_t$, γ: discount factor $\in (0, 1]$

Given some policy π, $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$ can be computed.

Find the policy π with the largest value of the above expected return.

Markov decision process (MDP)

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot R_t\right]$$

such that

$R_t = R(S_t, A_t)$

$S_t \in \mathcal{S}$, $A_t \in A$

$P = Pr\{S_{t+1} = s_{t+1} \mid S_t = s_t, A_t = a_t\}$, ← transition prob/ fn ← state evolution

$\gamma \in [0, 1]$.

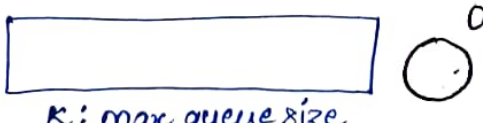
Optimization problem:

$\max_{\bar{x}} f(\bar{x})$ → vector

Look for some algo. for the type of fn f.

$\bar{x} \in D$

eg.



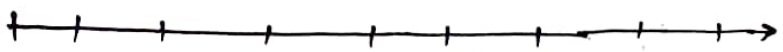
K: max queue size

Counter (eg, bank counter)

Two persons servicing:

$\mu_1 < \mu_2$ → more experienced

$c_1 < c_2$ → payments



In every slot t, a person comes into this system with probability λ.

In every slot, if a person is served, she finishes service with probability μ.

State = num. in queue

S_t :

$\pi(S_t) \in \{\mu_1, \mu_2\}$

$R(S_t) = -(S_t + f \cdot C_t)$ → To accomodate the money paid (additional cost)

To reduce the cost

f ↳ multiplying factor ↳ to make unit compatible

$$C_t = \begin{cases} C_1, & \text{if } \pi(S_t) = \mu_1 \\ C_2, & \text{if } \pi(S_t) = \mu_2 \end{cases}$$

If $S_t = 0$, $A_t \in \{\mu_1, \mu_2\}$, [Special case]

$S_{t+1} = 0$, w.p. $(1-\lambda)$ (no one comes)

$= 1$, w.p. λ. (one person comes)

If $S_t = K$, $A_t = \mu_1$,

$S_{t+1} = K-1$, w.p. $\mu_1(1-\lambda)$ (no one comes, one served)

$= K$, w.p. $(1-\lambda)(1-\mu_1) + \lambda\mu_1$.

$= K+1$, w.p. $(1-\mu_1)\lambda$

(no one comes, no one served; one came, one served)

If $S_t = j$, $A_t = \mu_1$,

$S_{t+1} = j-1$, w.p. $(1-\lambda)\cdot\mu_1$

j, w.p. $(1-\mu_1)(1-\lambda) + \mu_1\lambda$

$j+1$, w.p. $\lambda(1-\mu_1)$.

$MDP = (S, A, P, R, \gamma)$ → P → transition fn

Read ch-1

- Markov ↳ due to the conditional prob. that next state depends upon S_t, A_t
- Episodic & Continuity

[Book: Mathematical model of RL]

Markov decision process:

(S, A, P, R, γ)

Eg Machine repair problem

$S = \{G, B\}^5$, no. of G (good) components.

$A = \{run, repair\}$

$P_\gamma = \{S_{t+1} = g' \mid S_t = g, A_t = run\}$

$g' = g - \Delta$,

$\Delta \in \{0, \ldots, g\}$

$P_\gamma\{\Delta = \delta\} = \binom{g}{\delta} p^\delta (1-p)^{g-\delta}$

$P_\gamma\{S_{t+1} = g - \delta \mid S_t = g, A_t = run\}$

$= \binom{g}{\delta} \cdot p^\delta (1-p)^{g-\delta}$

$P_\gamma\{S_{t+1} = 5 \mid S_t = g, A_t = repair\} = 1$ [everything is good]

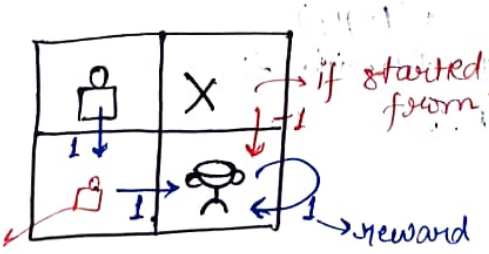
Reward $R(S_t, run) = R$

$R(S_t, repair) = -C$

$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]$ → to maximise this

state value → $\mathbb{E}\left[\sum_{t=0} \gamma^t R_t \mid S_0 = s_0\right]$: Metric is dependent on the initial state.

Eg



if started from here

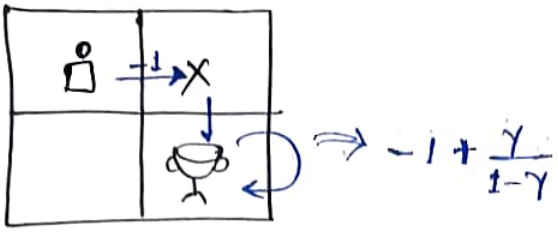
What is the metric for this policy?

↳ $\frac{1}{1-\gamma}$, $\frac{1}{1-\gamma}$, $-1 + \frac{\gamma}{1-\gamma}$

if started from

if started from ×

Eg.

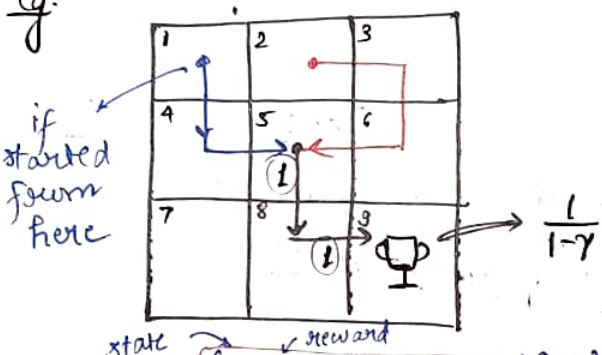


$\Rightarrow -1 + \frac{\gamma}{1-\gamma}$

$\left(-1+\frac{\gamma}{1-\gamma}\right)$ OR $\left(\frac{1}{1-\gamma}\right)$ → which is better?

↓ compare the policy

Eg.



state → (5,1), (8,1) → (9,1) ... ← reward

(1,1), (4,1), (5,1), (8,1) → (9,1), ...

(2,1), (3,1), (6,1), (5,1), (8,1) → (9,1) ...

$$\mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} R_\tau \,\middle|\, S_t = s\right]$$

↓ expected return from time 't'

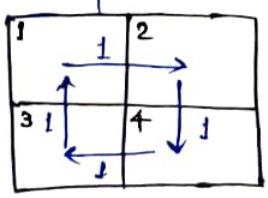
$$= \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} R_\tau \,\middle|\, S_0 = s\right] = V^{\pi}(s).$$

↳ doesn't depend on time; depends on the initial state.

→ The value of the state does not depend on the time when we started.

Find the policy that maximises $V^{\pi}(s)$.

→ How to compute $V^{\pi}(s)$?



$1 + \gamma + \gamma^2 + \dots = \frac{1}{1-\gamma}$

$V^\pi(1) = V^\pi(2) = V^\pi(3) = V^\pi(4) = \frac{1}{1-\gamma}$

$$
\begin{aligned}
V^\pi(1) &= 1 + \gamma V^\pi(2) \\
V^\pi(2) &= 1 + \gamma V^\pi(3) \\
V^\pi(3) &= 1 + \gamma V^\pi(4) \\
V^\pi(4) &= 1 + \gamma V^\pi(1)
\end{aligned}
$$

↓ solve for these

↓ same quantities

: Bootstrapping

↓

V^π written in-terms of itself.

Read ch-2

Lec-6 | 26-08-2025

$V^\pi(s)$: expected return (sum of discounted return) for the policy π starting from state 's'.

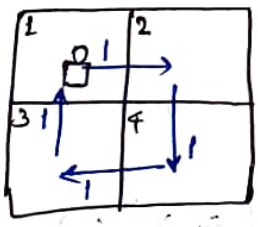
$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t^{\pi} \mid S_0 = s\right]$$

↓

defined $\forall\, s \in S$

↳ Also called State value function or State function.

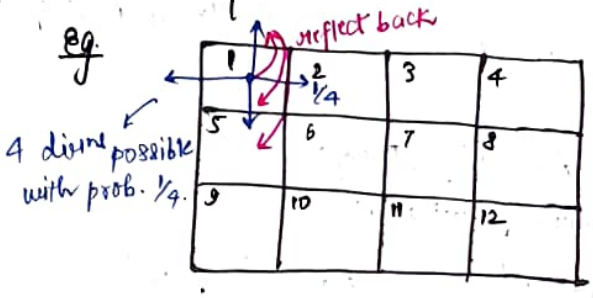
Eg.



$$
\begin{bmatrix} 1 & -\gamma & 0 & 0 \\ 0 & 1 & -\gamma & 0 \\ 0 & 0 & 1 & -\gamma \\ -\gamma & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} V^\pi(1) \\ V^\pi(2) \\ V^\pi(3) \\ V^\pi(4) \end{bmatrix}
=
\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
$$

How to express $V^\pi(s)$ (in terms of $V^\pi(s), \forall\, s \in S$) in general?

eg.



$\pi(a|s)$: probability that action is 'a', given current state is 's'. ↳ policy

$p'(s'|s,a)$: pmf of $s_{t+1} = s'$, state is 's', action is 'a'.

$p(r|s,a)$: distribution of reward. ↳ Transition fn.

$$V^\pi(s) = \mathbb{E}[R_0 | S_0 = s] + \gamma \cdot \mathbb{E}\left[\underbrace{\sum_{t=1}^{\infty} \gamma^{t-1} R_t}_{Z} \,\middle|\, S_0 = s\right]$$

If we had $\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t' \,|\, S_1 = s_1\right] = V^\pi(s_1)$.

$$\mathbb{E}[Z | S_0 = s_0] = \sum_z Pr\{Z = z \mid S_0 = s_0\} \cdot z$$

$$= \sum_{s_1} \sum_z Pr\{Z = z_0, S_1 = s_1 \mid S_0 = s_0\} \cdot z \quad : \text{Marginal of joint distn.}$$

$$= \sum_{s_1} \sum_z Pr\{Z = z \mid S_1 = s_1, S_0 = s_0\} \cdot Pr\{S_1 \mid S_0\} \cdot z$$

$$= \sum_{s_1} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \cdot R_t \mid S_0 = s_0, S_1 = s_1\right] \cdot Pr\{S_1 = s_1 \mid S_0 = s_0\}$$

$$= \sum_{s_1} V^\pi(s_1) \cdot Pr\{S_1 = s_1 \mid S_0 = s_0\}$$

$$V^\pi(s) = \underbrace{\mathbb{E}[R_0 | S_0 = s]}_{\text{Immediate reward}} + \underbrace{\gamma \cdot \sum_{s_1} P(S_1 = s_1 | S_0 = s_0) \cdot V^\pi(s_1)}_{\text{Average of rewards for next states}}$$

$$= \sum_r P(r|s_0) \cdot r + \gamma \sum_{s_1} P(S_1 = s_1 | S_0 = s_0) \cdot V^\pi(s_1)$$

↓ marginalize over actions

$$\Rightarrow V^\pi(s) = \sum_a \pi(a|s_0) \cdot \sum_r P(r|s_0, a) \cdot r + \gamma \sum_a \pi(a|s_0) \cdot \sum_{s_1} P(S_1 = s_1 | S_0 = s_0, A_0 = a) \cdot V^\pi(s_1)$$

(avg. over actions; avg. over rewards)

Bellman's Equation

How to compute $V^\pi(s)$?

Read about Richard Bellman (Wikipedia)

$$\mathcal{H}^\pi(s_0) = \sum_a \pi(a|s_0) \cdot \sum_{\mathcal{H}} P(\mathcal{H}|s_0, a_0)\, \mathcal{H}$$

$$\sum_a \pi(a|s_0) \cdot \sum_{s_1} P(s_1 = s_1 \mid s_0 = s_0, A_0 = a) \cdot V^\pi(s_1)$$

prob. of action · prob. of next state

$$= \sum_{s_1} \left(\sum_a \pi(a|s_0) \cdot P(s_1 = s_1 \mid s_0 = s_0, A_0 = a) \right) \cdot V^\pi(s_1)$$

$$\left[\equiv \sum a_{s_1} \cdot b_{s_1} \rightarrow \text{dot product of 2 quantities} \right]$$

$$= \begin{pmatrix} P^\pi(1|s_0) & P^\pi(2|s_0) \dots \end{pmatrix} \begin{pmatrix} V^\pi(1) \\ V^\pi(2) \\ \vdots \\ V^\pi(s) \end{pmatrix}$$

$$V^\pi(s_0) = \mathcal{H}^\pi(s_0) + \gamma \begin{pmatrix} P^\pi(1|s_0) & P^\pi(2|s_0) \dots \end{pmatrix} \begin{pmatrix} V^\pi(1) \\ \vdots \\ V^\pi(s) \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} V^\pi(1) \\ V^\pi(2) \\ V^\pi(3) \\ \vdots \\ V^\pi(s) \end{pmatrix} = \begin{pmatrix} \mathcal{H}^\pi(1) \\ \mathcal{H}^\pi(2) \\ \vdots \\ \mathcal{H}^\pi(s) \end{pmatrix} + \gamma \begin{pmatrix} P^\pi(1|1) & P^\pi(2|1) \dots \\ P^\pi(1|2) & P^\pi(2|2) \dots \\ \vdots & \vdots \\ & \end{pmatrix} \begin{pmatrix} V^\pi(1) \\ V^\pi(2) \\ \vdots \\ V^\pi(s) \end{pmatrix}$$

$\bar{V}^\pi$ $\quad$ $\bar{\mathcal{H}}^\pi$ $\quad$ $\bar{P}^\pi$ $\quad$ $\bar{V}^\pi$

$$\Rightarrow \bar{V}^\pi = \bar{\mathcal{H}}^\pi + \gamma \bar{P}^\pi \bar{V}^\pi$$

$$\Rightarrow \bar{V}^\pi = (I - \gamma \bar{P}^\pi)^{-1} \bar{\mathcal{H}}^\pi$$

Read section 2.7 for invertibility

↓

Matrix inverse exists

↳ computationally very difficult

⇓

Iterative algorithm

State value : $v^{\pi}(s)$

Metric for evaluating / ranking policies ?

$V^{\pi}(\cdot) \rightarrow$ state value function

$\bar{v}^{\pi} = (V^{\pi}(1) \dots V^{\pi}(s)) \rightarrow$ vector

Bellmann's equation:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{r} p(r|s,a)\, r + \gamma \sum_{a} \pi(a|s) \sum_{s'} p(s'|s,a)\, V^{\pi}(s')$$

In metric form,

$$\bar{V}^{\pi} = \bar{r}^{\pi} + \gamma\, P^{\pi} \bar{V}^{\pi}$$

$$\bar{V}^{\pi} = (I - \gamma P^{\pi})^{-1} \bar{r}^{\pi}$$

Claim: $\bar{V}^{\pi}$ – unique soln exists.

Iterative Algorithm for Finding $\bar{V}^{\pi}$

Start with $\bar{V}_0 = (V_0(1), V_0(2), \dots V_0(s))^T$ [or with all zero]

Compute $\bar{V}_{k+1}^{\pi} = \bar{r}^{\pi} + \gamma P^{\pi} \bar{V}_k^{\pi}$, with $\bar{V}_k^{\pi} = \bar{V}_0$ ~~at start~~ to start with.

Repeat.

$\lim_{k \to \infty} \bar{V}_k^{\pi} = \bar{V}^{\pi}$

Stop when a close ~~one~~ enough value is achieved.

$\max_{s} |V_{k+1}(s) - V_k(s)| < \epsilon V_k(s)$,

then stop these iterations.

$V_k(s) \approx V^{\pi}(s)$.

Assignment-①: Prove that $\lim_{k \to \infty} \bar{V}_k^{\pi} = \bar{V}^{\pi}$.

[Read about "contraction maps" (wiki). Apply contraction mapping theorem.]

Assignment-②: Implement Value iteration for policy evaluation in Python.

Eg. $|S|$, $|A|$

0.1 0.9
0.5 0.5 } Prob. for 2 states

Reward → deterministic
↳ 1
2
3
4

Transition prob.: 0.1 0.9
↓
$P(s'|s,a)$

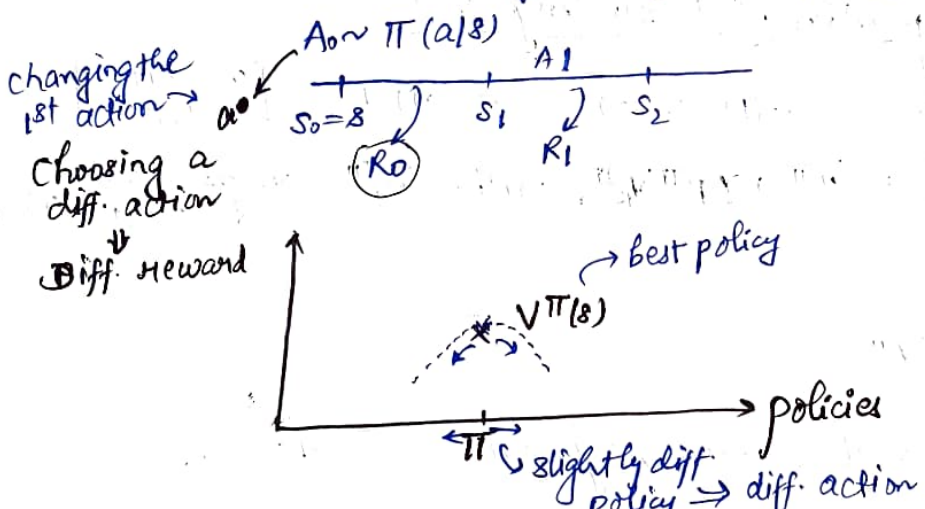
State-action value function:

Q-learning

Recall:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \,|\, S_0 = s\right]$$

↳ Fn of state given a policy.



$$q^{\pi}(s,a) = \mathbb{E}[R(s,a)] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t R_t \,|\, S_0 = s, A_0 = a\right]$$

↳ Q-function
↳ Diff. value at the 1st instant
↳ Further values depend on the action taken.
↳ Further behaviour will depend on the policy:
same from $V^{\pi}(s)$ & $q^{\pi}(s,a)$.

$$q^{\pi}(s,a) = \mathbb{E}[R(s,a)] + \gamma \underbrace{\mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid S_0 = s_0, A_0 = a\right]}_{\sum_{s_1} P(s_1 \mid s_0, a).\, V^{\pi}(s_1)}$$

$$\Rightarrow \boxed{q^{\pi}(s,a) = \mathbb{E}[R(s,a)] + \gamma \sum_{s_1} P(s_1 \mid s_0, a).\, V^{\pi}(s_1)}$$

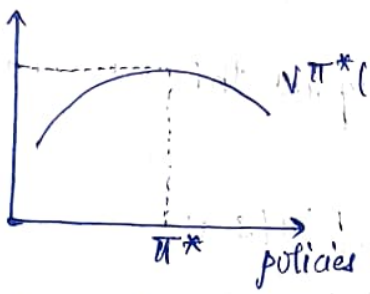
$$V^{\pi}(s) = \sum_{a} \pi(a \mid s) \left(\mathbb{E}[R(s,a)] + \gamma \sum_{i} p(s_i \mid s, a).\, V^{\pi}(s_i) \right)$$

Read section 2.8

$$\Rightarrow \boxed{V^{\pi}(s) = \sum_{a} \pi(a \mid s).\, q^{\pi}(s,a)}$$

L8 01-09-2025

Optimal policies : $MDP = (S, A, P, R, \gamma)$



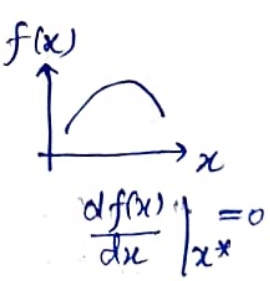
What we have now:

Given a policy π, $V^{\pi}(s)$, $\forall s \in S$,

or, $\overline{V}^{\pi}$ and $q^{\pi}(s,a)$ $\forall$

Defn: A policy π^* is optimal if $V^{\pi^*}(s) \geq V^{\pi}(s)$ for every $s \in S$ and any policy π.

$V^{\pi^*}(s)$: optimal state value



Ranking: Condition for policies

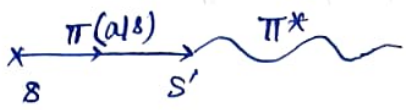
π_1 is better than π_2, if $V^{\pi_1}(s) \geq V^{\pi_2}(s)$, $\forall s \in S$.

$\pi^* \longrightarrow \pi^*(a|s)$ → conditional pmf
$V^{\pi^*}(s)$

} Given.
↳ Actually unknown

We know,

$$V^{\pi}(s) = \sum_a \pi^*(a|s) \cdot \sum_r p(r|s,a) \cdot r + \gamma \sum_a \pi^*(a|s) \cdot \sum_{s'} p(s'|s,a) \cdot V^{\pi^*}(s')$$



For $\pi(a|s)$, only (first) control: choose π.

eg. $\pi(a|s) = (0.1, 0.8, 0.1)$

$$\max_{\pi} \left(\sum_a \pi(a|s) \sum_r p(r|a,s) \cdot r + \gamma \cdot \sum_a \pi(a|s) \cdot \sum_{s'} p(s'|s,a) \cdot V^{\pi^*}(s') \right) = V^{\pi^*}(s)$$

→ Max^m/optimal after 1st step

↳ What is this? (Interpret this total reward)

$$V^{\pi^*}(s) = \max_{\pi} \left\{ \sum_a \pi(a|s) \sum p(r|s,a) \cdot r + \gamma \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \cdot V^{\pi^*}(s') \right\}$$

Bellman's Optimality Equation (BOE)

↳ Necessary condition and sufficient condition

$\pi^* \longrightarrow \max V^{\pi}(s)$

$V^{\pi^*}(s) \geq V^{\pi}(s)$

: Digression : Tradeoff b/w immediate rewards & future value

$r_0^* + \gamma r_1^* + \gamma^2 r_2^* + \ldots \geq r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ → may not be true

} only total reward dominates; not individual rewards

$$\rightarrow V(s) = \max_{\pi} \left\{ \sum_a \pi(a|s) \sum p(r|s,a) \cdot r + \gamma \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \cdot V(s') \right\}$$

↳ jointly maximise for both π and $V(s)$.

then $V(s)$ is $V^{\pi^*}(s)$.
The argmax is $\pi^*(s)$.

$$\# \max_{\pi_0, \pi_1, \pi_2} \left\{ \mathbb{E} \sum_{t=0}^{\infty} \gamma^t . R_t \mid S_0 = s \right\}$$

$$= \max_{\pi_0, \pi_1, \pi_2} \left\{ \mathbb{E}[R_0 \mid S_0 = s] + \gamma . \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} . R_t \mid S_0 = s \right] \right\}$$

$$= \max_{\pi_0, \pi_1, \pi_2} \left\{ \overbrace{\sum_a \pi_0(a|s) . \sum_r p(r|s,a) . r}^{\pi_0 \text{ controls this}} + \gamma . \sum_a \pi_0(a|s) . \sum_{s'} p(s'|s,a) . \overbrace{\left(\mathbb{E} \sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid S_1 = s' \right)}^{\pi_1, \pi_2, \ldots \text{ control this}} \right.$$

$$= \max_{\pi_0} \left\{ \sum_a \pi_0(a|s) \sum_r p(r|s,a) . r + \gamma \sum_a \pi_0(a|s) . \sum_{s'} p(s'|s,a) . \underbrace{\max_{\pi_1, \pi_2} \mathbb{E} \sum_{t=1}^{\infty} \gamma^{t-1} R_t \mid S_1 = s'}_{\text{original problem (with different time)}} \right\}$$

↳ ~~π_0, π_1, π~~

$\pi_0 = \pi_1 = \pi_2$ maximizes this.

Read Ch. 3 & 4

Lec-9

Recall $v_{\pi^*}(s) \geq v_{\pi}(s)$, $\forall s \in S$, and any other π.

To find π^* and $v_{\pi^*}(s)$,
find any solution to the Bellman's Optimality equation.

$$V(s) = \left(\max_{\pi} \right)^{\text{argmax}} \left\{ \sum_a \pi(a|s) \sum_r p(r|s,a) . r + \gamma \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) . V(s') \right\}$$

then a soln,

$V(s) = v_{\pi^*}(s)$

argmax $\pi = \pi^*(s)$ ⇝ Optimal policy

2 unknowns: V, argmax

How to solve? + Some important points regarding this eqn:

- Compare with Bellman's equation for a particular policy π.
  ↳ Here there are two unknowns : V and argmax.
  ↳ max

- $\max_{\pi} \left\{ \sum_a \pi(a|s) \left\{ \underbrace{\sum_r p(r|s,a)\, r + \gamma \sum_{s'} p(s'|s,a).\, V(s')}_{\text{like } q(s,a)} \right\} \right\}$ ↳ fn of a, s.

$$= \max_{\substack{\pi(a|s) \\ \{\sum_a \pi(a|s)=1, \\ \pi(a|s) \geq 0\}}} \left\{ \sum_a \pi(a|s) \cdot q(s,a) \right\}$$

[Eg $a \in \{1,2,3\}$
$q(s,1) = 2$
$q(s,2) = 10$
$q(s,3) = 11$ → give max. weight to the max of these
$\pi(3|s) = 1$]

$$= \max_a q(s,a)$$

⇒ Optimal policy is deterministic.

$$V(s) = \max_a \left\{ \sum_r p(r|s,a)\, r + \gamma \sum_{s'} p(s'|s,a).\, V(s') \right\}$$

argmax of 'a' gives $\pi^*(s)$.

For MDPs (S, A, P, R, γ), it is enough to look for deterministic policy.

- $\sum_r p(r|s,a)\, r$: Expectation of rewards $= \bar{r}(s,a)$
- $\gamma \sum_{s'} p(s'|s,a) \cdot V(s')$ → dot product

$= \gamma \langle \_\_\_ , \bar{V}(\cdot) \rangle$

$s \bullet \begin{pmatrix} p(s'|s,a) \\ \longleftrightarrow \end{pmatrix}$, if the action is a

↳ matrix of row vectors ↳ tells prob. of evolving from s to s'

Machine eg.

	0	1	2
0	0	0	1
1	0	0	1
2	0	0	1

	0	1	2	
0	(1−...)	$2p(1-p)$	p	← run
1	0	$1-p$	p	
2				

repair

Solve

$$\bar{V} = \max_{\pi(s)} \{ \bar{R}(\cdot, \pi(s)) + \gamma P \bar{V} \} \longrightarrow \text{Generalization}$$

↳ How to choose the row depends on $\pi(s)$.

↳ This is a contraction map.

$V(1) = \max_{a_1} \{ \bar{R}(1, a_1) + \gamma < P(1|1,a_1), P(2|1,a_1) \begin{pmatrix} V(1) \\ V(2) \end{pmatrix} > \}$

$V(2) = \max_{a_2} \{ \bar{R}(2, a_2) + \gamma < P(1|2,a_2), P(2|2,a_2) \begin{pmatrix} V(1) \\ V(2) \end{pmatrix} > \}$

↓ compiling [only 2-states, here]

$$\bar{V} = \begin{pmatrix} V(1) \\ V(2) \end{pmatrix} = \max_{\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \pi(1) \\ \pi(2) \end{pmatrix}} \left\{ \bar{R}(\cdot, \begin{matrix} a_1 \\ a_2 \end{matrix}) + \gamma \begin{pmatrix} P(1|1,a) P(2|1,a) \\ P(1|2,a) P(2|2,a) \end{pmatrix} \begin{pmatrix} V(1) \\ V(2) \end{pmatrix} \right\}$$

$\bar{V} = T(\bar{V})$ ⟶ Transformation ⟶ Unique solution

↳ But not unique optimum policy; just atleast one deterministic optimum policy.

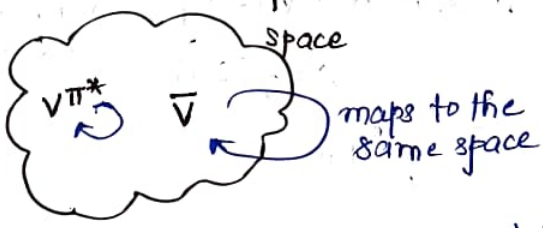
We can compare with policy evaluation

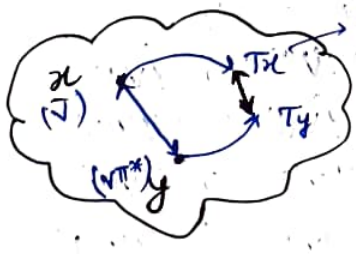
$$\bar{V}^{\pi} = \bar{R}^{\pi} + \gamma P^{\pi} \bar{V}^{\pi}$$

If we get a soln to this, it is called a fixed point.

T → function / mapping

↳ contraction map

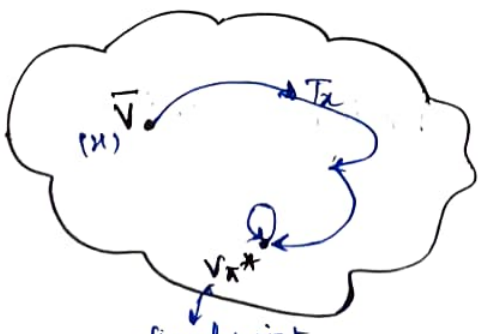




Transformed x vector

Separation b/w Tx & Ty is closer.

↳ Contraction



fixed point ↳ doesn't move on transformation

$\bar{V}$ → moves closer to V^{π^*}.

Procedure for solving: Value Iteration

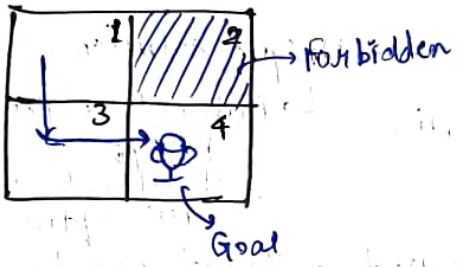
Start with an arbitrary $\bar{V}_0$.

Put $\bar{V}_0$ in the map$^{\wedge}_{(\bar{V}_K)}$ to get $\bar{V}_1$ and so on.

- $\bar{V}_{K+1} = \max_{\pi(s)} \{ \bar{r}(\cdot, \pi(s)) + \gamma P \cdot \bar{V}_K \}$
- $\lim_{K \to \infty} \bar{V}_K = V^{\pi^*}$.

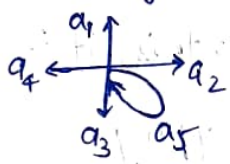
[$\bar{V}_{K+1}$ → Not value^{n} of some policy
↳ Just iterates]

Example:



→ forbidden

go outside = −1 (bounce back)
forbidden = −1
goal = +1



$$V(s) = \max_{\{a_1 \dots a_5\}} \{ \bar{r}(s,a) + \gamma \bar{V}(s') \}$$ → no distribution ↳ deterministic

$s' = f(s,a)$

$q(s|a)$

	a_1	a_2	a_3	a_4	a_5
1	$-1+\gamma V(1)$	$-1+\gamma V(2)$	$0+\gamma V(3)$	$-1+\gamma V(1)$	$0+\gamma V(1)$
2	$-1+\gamma V(2)$	$-1+\gamma V(2)$	$1+\gamma V(4)$	$0+\gamma V(1)$	$-1+\gamma V(2)$
3	$0+\gamma V(1)$	$1+\gamma V(4)$	$-1+\gamma V(3)$	$-1+\gamma V(3)$	$0+\gamma V(3)$
4	$-1+\gamma V(2)$	$-1+\gamma V(4)$	$-1+\gamma V(4)$	$0+\gamma V(3)$	$1+\gamma V(4)$

Start with $\bar{V}_0 = (0,0,0,0)^T$

Take row-wise max^{n} after putting all $V(i) = 0$.

$\therefore \bar{V}_1 = (0,1,1,1)^T$

Rewrite the table.

-1	$-1+\gamma$	(γ)	-1	0
$-1+\gamma$	$-1+\gamma$	$(1+\gamma)$	0	$-1+\gamma$
0	$(1+\gamma)$	$-1+\gamma$	$-1+\gamma$	γ
$-1+\gamma$	$-1+\gamma$	$-1+\gamma$	γ	$(1+\gamma)$

$\bar{V}_2 = (\gamma, 1+\gamma, 1+\gamma, 1+\gamma)^T$

Repeat.

Value Iteration Algorithm:

MDP $= (S, A, P, R, \gamma)$, ϵ ↘ tolerance ↳ i/p

Initialize $\bar{V}_0 (=0)$.

$\bar{V}_{new} = 5\epsilon$ (eg.)

while $\| \bar{V}_{new} - \bar{V}_0 \| > \epsilon$

$V_0 = V_{new}$

for s in S

$$V_{new}(s) = \max_a \{ \bar{R}(s,a) + \gamma \sum p(s'|s,a)\, V_0(s') \}.$$

Lec-10 | 02-09-2025

Obtaining Optimal Policies

Recall:

If we find a soln to

$$V(s) = \max_a \{ \bar{R}(s,a) + \gamma \sum_{s'} p(s'|s,a)\, V(s') \}$$

then $V^{\pi^*}(s) = V(s)$

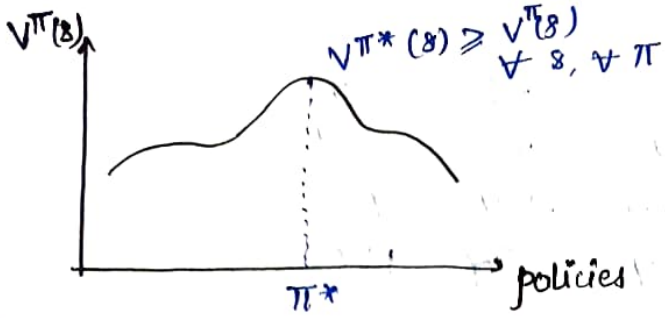
and $\pi^*(s) = \text{argmax}$ for every s.

Value Iteration Procedure: (VI) → like $q_K(s,a)$

$$V_{k+1}(s) = \max_a \{ \bar{R}(s,a) + \gamma \sum_{s'} p(s'|s,a)\, V_k(s') \},$$

with $V_0(s) = 0$

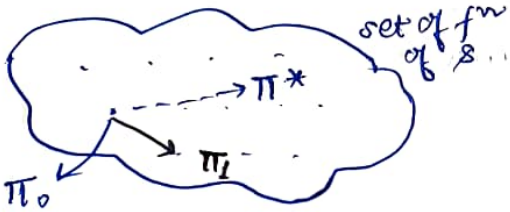
Terminate if $\| \bar{V}_{k+1} - \bar{V}_k \| < \epsilon$.



Lecture notes:
Siva Theja
Maguluri
Georgia Tech

At every step,

value function ⟶ policy (we get when taking argmax) ⟶ Value update



Eg.

State	action
1	a_1
	a_2
2	a_3

$\pi(1) \in \{a_1, a_2, a_3\}$

$\pi(2) \in \{a_1, a_2, a_3\}$

↳ Total 9

Policy Iteration:

Start with π_0, $\boxed{V^{\pi_0}(s)}$ ← policy evaluation ①

Find $q^{\pi_0}(s,a) = r(s,a) + \gamma \sum p(s'|s,a) V^{\pi_0}(s')$

Suppose $\exists\ a$ such that $q^{\pi_0}(s,a) > V^{\pi_0}(s)$.

Get a time-independent policy π_1

$$\pi_1(s) = \arg\max_a q^{\pi_0}(s,a)$$

↳ ② Policy Improvement Theorem (Ref. text)

Policy Iteration Algorithm:

Initialize π_0.

do {

evaluate π_k to get $V^{\pi_k}(s)$ and $q^{\pi_k}(s,a)$ using policy evaluation.

$\pi_{k+1}(s) = \arg\max_a q^{\pi_k}(s,a)$.

} while $\| \bar{V}^{\pi_{k+1}} - \bar{V}^{\pi_k} \| \geq \epsilon$.

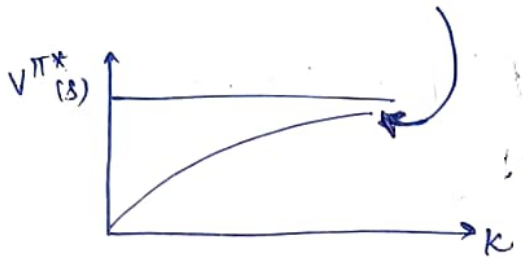
Why should policy iteration converge?

$\pi_k \rightarrow \pi_{k+1}$

$V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s)$

$\bar{R}(s,a) \leq R_{max}$

$V^{\pi^*}(s) \leq \frac{R_{max}}{1-\gamma}$



→ Increasing sequence
- diverges ✗
- oscillate ✗

⇒ converge ✓

Greedy policy
(Greedy myopic policy)
↳ Just look at the current reward

Multi-armed Bandits

$$\left[R_0 + \gamma R_1 + \gamma^2 R_2 + \dots \right]$$
$$\leq \quad \leq \quad \leq$$
$$R_{max} \quad R_{max} \quad R_{max}$$

Policy iteration → faster than value iteration
↳ Requires policy evaluation
↳ value iteration

→ <u>Truncated policy iteration</u>: Do VI for policy evaluation for j steps.

↳ with just 1 step, it is value iteration

1 step ⟶ ∞ step
pure VI pure PI

Lec-11 | 08-09-2025

Review

Markov Decision Processes (MDPs)

↳ Formulate an optimization problem

- Objective: control system using a policy π
- $\max_{\pi} V^{\pi}(s)$ - (S, A, P, R, γ)
- $V(s) = \max_{a} \{ r(s,a) + \gamma \sum p(s'|s,a) . V(s') \}$

$\pi^*(s) = \arg\max_{a} \{ \downarrow \}$

VI: $V_{k+1}(s) = \max_{a} \{ r(s,a) + \sum p(s'|s,a) V_k(s) \}$

As $k \to \infty$, $V_k(s) \longrightarrow V^{\pi^*}(s)$

- $\pi_k \longrightarrow V^{\pi_k^*}(s) \longrightarrow q^{\pi_k}(s,a)$

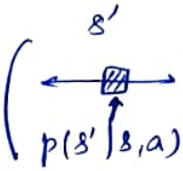
$\pi_{k+1} = \arg\max q^{\pi_k}(s,a)$

As $k \to \infty$

- Truncated PI.

→ Structural Properties of π^* : Book : Bertsekas (Dynamic Programming and optimal control)

Transition probability:

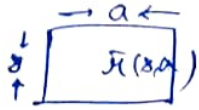
$p(s'|s,a) \rightarrow$ 

$s = (e, \#, !)$

↓ map to no.

$s = (0, 1, 2)$

We need $\sum_{s'} p(s'|s,a) . r = \bar{r}(s,a)$: Average reward



Stochastic matrix → Rows sum to 1.

Doubly stochastic matrices → Rows & columns sum to 1.

$\sum_{s'} p(s'|s,a) \cdot r(s',s,a) = \bar{r}(s,a)$ → Reward depending on the next state
↳ More general.

Span of matrix = max − min

Average Reward: $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} R_t$ → Relative value Iteration method

Lec-12 09-09-2025

MDP Toolbox

↳ Python variant : pymdptoolbox

↳ Eg. Forest management
Clinical decision making (Puterman)

MDP Examples:

Ref: MDPToolbox (Forest)
Puterman (for other examples).

Steps: → finite discrete

① MDP = (S, A, P, R, γ).

② Solve using VI, PI (or variants).

Example-1:

Forest management

Trees ∈ forest

Every year, decide cut or wait

Age of the forest: 0, 1, 2, ..., A

If we wait, a fire will destroy the forest with prob. p.

$S = \{0, 1, \ldots, A\}$

$A = \{c, w\}$ → cut, wait

| # example.forest() from mdptoolbox

P^c:

$$\begin{array}{c} \\ A+1 \end{array} \begin{array}{c} \overset{A+1}{} \\ \begin{pmatrix} 1 & & \\ 1 & & \\ \vdots & & \\ 1 & & \end{pmatrix} \end{array}$$

P^w:

$$\begin{array}{c} \uparrow \\ A+1 \\ \downarrow \\ a \end{array} \begin{array}{c} \overset{\leftarrow\ A+1\ \rightarrow}{} \\ \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ p & 1-p & 0 & \cdots & 0 \\ p & 0 & 1-p & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ p & 0 & \cdots & 0 & 1-p \end{pmatrix} \end{array}$$

(columns: 0, 1, ..., a+1, ..., A)

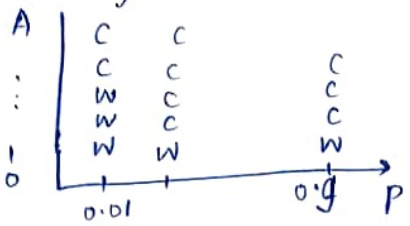
Reward:

$R(s, a)$

$$\begin{array}{c} \\ 0 \\ \vdots \\ A \end{array} \begin{array}{c} \begin{array}{cc} c & w \end{array} \\ \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 2 & 4 \end{pmatrix} \end{array}$$

Choose

Assign: To solve this problem.
How optimal policies behave as a function of p and γ?



Example-2:

Decision making for organ transplants (liver):

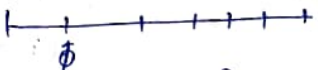
health = $\{1 \ldots H\} \cup \{\Delta\}$ ← best health, ← lowest health, → patient is dead.

organ = $\{1 \ldots L\}$ ← best, ← worst organ health

Every day, the health changes.

Every day, the patient gets an organ with one of the health states or not get an organ with prob. ϕ. → organ

decisions = {reject, accept}



State space: Includes both patient and organ healths.

$\{1 \ldots H\} \cup \{\Delta\} \times \{1 \ldots L\} \cup \{\phi\}$

eg. $s_0 = (h, \phi)$
$s_0 = (h, l)$

Reject action:

health transition prob.: $p(h'|h)$
dying prob.: $\delta(h)$
prob. of getting organ: $\beta(l|h)$
prob. of not getting organ: $\phi(h)$

Reward:

$r(s, a, j) = 0$ → Δ (died)

If you accept: $R(h, l)$ → depends on patient & organ healths

Uncertainty Equivalence

Lec-14 | 15-09-2025

Reinforcement Learning:

Recall

MDP = (S, A, P, R, γ)

Obtain a policy π to

$$\max_{\pi} V^{\pi}(s), \quad \forall s \in S$$

Methods: VI, PI and variants → Model Based → Known R and P

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

Bellman operator acts on $V(s)$.

Model-free algorithm

Objective of finding

$$\pi^* = \arg\max_{\pi} V^{\pi}(s)$$

without knowledge of

$R(s,a)$ and $P(s'|s,a)$

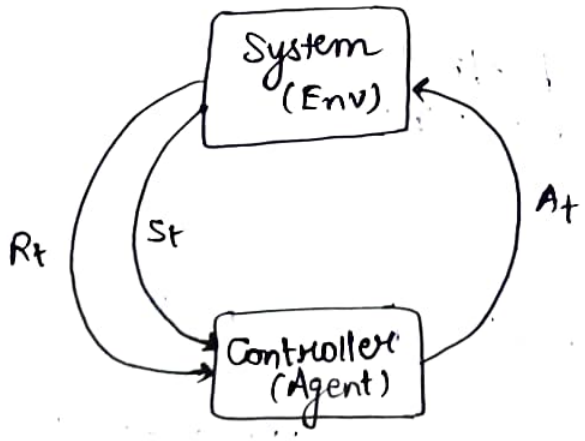
$\forall s, a$ $\forall s', s, a$

There is still a system (no change in system) i.e., $V^{\pi}(s)$ but controller does not know it.

Still S and A are assumed to be known.

π_0 : initial policy

$\arg\max_{a} q^{\pi}(s,a) \longrightarrow \pi_1$



Generate $(S_0, A_0, R_0, S_1, A_1, R_1, \ldots)$

Recall

- MDP Model : (S, A, P, R, γ)

Bellman Operation :

$$\underset{\text{or } a}{\operatorname{argmax}} \; \max_a \left\{ R(s,a) + \gamma \sum p(s'|s,a)\, V(s') \right\}$$

Model-based methods : VI, PI
Use Bellman operation
Uses knowledge of R, $p(s'|s,a)$.

RL ⇒ MDP model
(Mode-free) but controller does not know R, p
Still wants to $\max_\pi V^\pi(s)$.

π_0
↓ policy evaluation ⇒ How to do this model-free?
$q^{\pi_0}(s,a)$
↓ policy improvement
$\pi_1(s) = \underset{a}{\operatorname{argmax}}\, q^{\pi_0}(s,a)$
$V^{\pi_1}(s) \geq V^{\pi_0}(s)$

$$q^\pi(s,a) = \mathbb{E}\left\{ R(s,a) + \gamma R(S_1, A_1) + \gamma^2 R(S_2, A_2) + \gamma^3 R(S_3, A_3) + \dots \right\}$$

$A_1 \sim \pi(S_1)$, $A_2 \sim \pi(S_2)$

↳ Using π

$\{\cdot\}$
↓
G_0 → Random return (total discounted reward)

$$= \mathbb{E}\left[G_0 \mid S_0 = s, A_0 = a \right]$$

In general,
X, $\mathbb{E}X = \int f_X(x).x\,dx$
How to estimate $\mathbb{E}X$ from samples of the R.V. X.

Eg. $X = \{1, 2, 3\}$

$P_X = \{0.2, 0.4, 0.4\}$

$x_1, x_2, \ldots, x_N$

$$\frac{1}{N}\sum_{i=1}^{N} x_i \approx \mathbb{E}X$$

($\frac{1}{N}\sum_{i=1}^{N} x_i$: R.V.; $\mathbb{E}X$: constant if $N \to \infty$, otherwise R.V.)

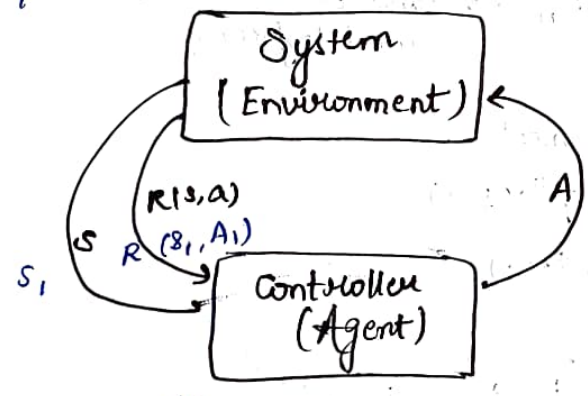
Monte Carlo (MC) Estimate

$$\lim_{N\to\infty} \sum_{i=1}^{N} x_i = \mathbb{E}X \quad : \text{Law of Large Numbers}$$

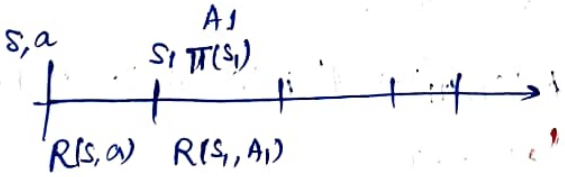
How can we estimate $q^{\pi}(s,a)$

$$= \mathbb{E}\left[G_0 \mid S_0 = s,\ A_0 = a\right]$$

Samples of the return G_0:



- Transition happens a/c to the model that is assumed for the system, which is not known to the controller.



$S, a, R_0, S_1, A_1, R_1, S_2, A_2, R_2, \ldots$: Trajectory

↳ Truncate : Episode

Samples of G_0:

$G_0^1 = R_0^1 + \gamma R_1^1 + \gamma R_2^1 + \ldots$ | episode

$G_0^2 = R_0^2 + \gamma R_1^2 + \gamma R_2^2 + \ldots$ |

N episodes :

$$q^{\pi}(s,a) \approx \frac{1}{N}\sum_{i=1}^{N} G_0^i$$

- Interact with the system for total $|S| \times |A| \times |N|$ episodes.

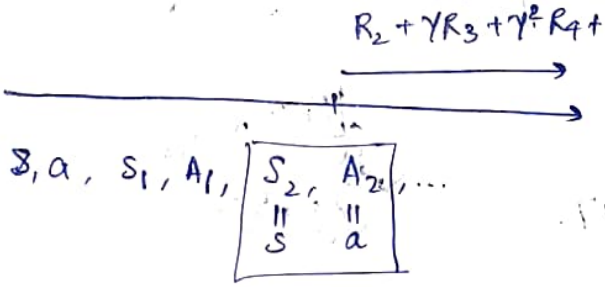
- Estimate the $q^{\pi}(s,a)$ for every s and a.
It should be possible to start the system in any s:
Exploring starts

↳ First Visit Monte Carlo Algorithm : Starting in state 's'. for policy evaluation

↳ Sample-inefficient.

To measure the quality of estimate : • Variance
• Bias → how far the mean of estimate is from the actual value.

(Get zero bias and low variance.)



$$q^{\pi}(s,a) = \mathbb{E}[G_t \mid S_t = s, A_t = a]$$

Every Visit Monte Carlo : Every time we visit a the state 's' and take action 'a', the expectation we get is same if we had started with state 's' and action 'a'.

Once we have estimated $q^{\pi_k}(s,a)$

$$\pi_{k+1} = \text{argmax}\ q^{\pi_k}(s,a).$$

Lec-16 | 22-09-2025

Review:

Model-free methods:

$MDP = (S, A, P, R, \gamma)$

$\pi^* = \text{argmax}_\pi V^\pi(s)$.

- Motivated by PI

$\pi_0 \longrightarrow q^{\pi_0}(s,a)$

π_1 ← improve $\pi_1(s) = \text{argmax}_a q^{\pi_0}(s,a)$

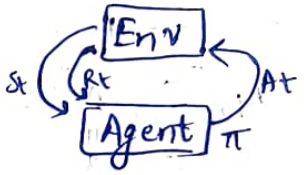
- Last class

Estimate $q^\pi(s,a)$ – from a sample of the return

- what is return?

R_0, R_1, R_2

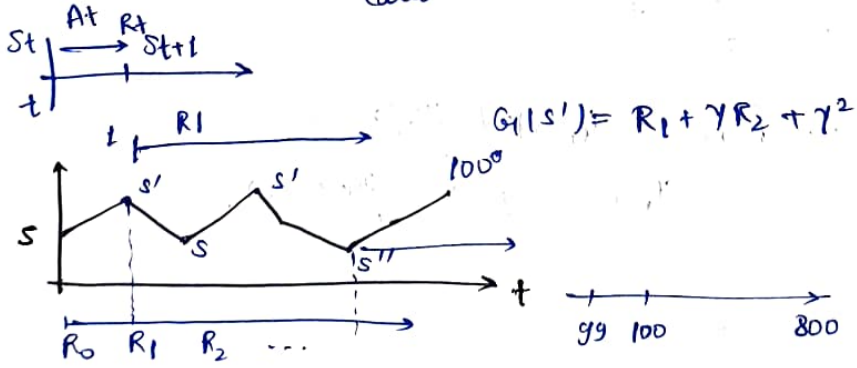
$G_0 = R_0 + \gamma R_1 + \gamma^2 R_2$



- Multiple episodes

$G_0^{(1)}, G_0^{(2)}, \ldots$

$q^\pi(s,a) = \mathbb{E} G_0 \underset{\text{Monte carlo}}{\approx} \frac{1}{N} \sum_{i=1}^{N} G_0^{(i)}$ ⇝ First visit Estimate



$G_0 = R_0 + \gamma R_1 + \gamma^2 R_2 + \ldots$

$\bar{X}_N = \frac{1}{N} \left(\sum_{i=1}^{N} x_i \right)$

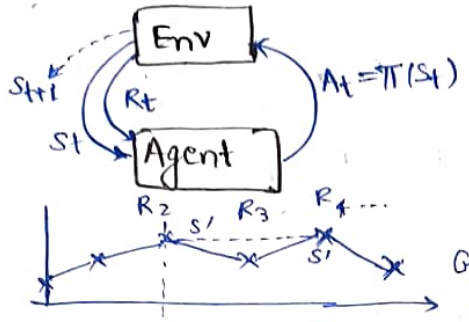
$\bar{X}_{N+1} = \frac{1}{N+1} \sum_{i=1}^{N+1} x_i$

$x_{N+1}, \quad \frac{\bar{X}_N N + x_{N+1}}{N+1}$.

Lec-17 | 23-09-2025

FVMC for policy evaluation - Implementation

$q^\pi(s,a)$ or $v^\pi(s)$ — agent does not know $(\mathbb{P}, R)$



$R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$

↳ avg. to obtain estimates of q^π and v^π

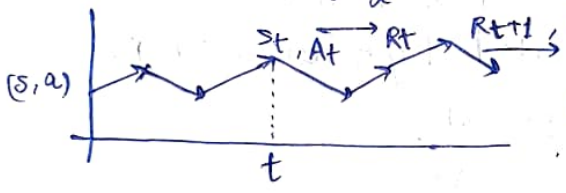
$G_{12} = R_2 + \gamma R_3 + \gamma^2 R_4 + \dots$

$G_{14} = R_4 + \gamma R_5 + \gamma^2 R_6 + \dots$

$x_1, x_2, \dots : \frac{1}{N} \Sigma x_i \to \mu, \frac{\sigma}{\sqrt{N}}$

$x_1, x_2, \dots : \Sigma x_i \to x_1$

$q^\pi(s,a) \Longrightarrow \underset{a}{\text{argmax}}\, q^\pi(s,a) = \pi_{new}(s)$

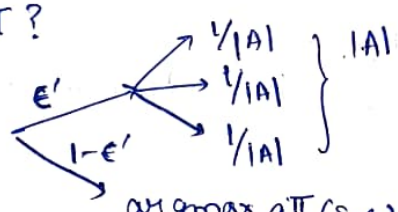


$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2}$

↓

$q^\pi(S_t, A_t)$

π ?



$\pi(a|s) = \epsilon'/|A|$ $\forall a \neq$ argmax

$a =$ argmax

$\pi(a|s) = \epsilon'/|A| + (1-\epsilon')$

How to estimate $q^\pi(s,a)$ $\forall s, \forall G$ in the FVMC code (H/W)

$\pi = [0\ 0\ 0\ 1\ 0\ 1]$ ⟶ $q^\pi(s, \neq 0)$

$S_t = (0, 1, 1, 2, \dots 0)$

$A_t = (0, 0, 0, 0, \dots 0)$

π_0 ⟷ $q^{\pi_0}(s,a)$

$\pi_1(s) =$ argmax $q^{\pi_0}(s,a)$

$q^{\pi_1}(s, \neq \pi_1(s))$.

Recall:

EVMC based control

In the first step,

S_0 S_1
A_0 A_1 ...
R_0 R_1

(S_t, A_t) —return sample→ $\forall s$

t

$\pi_0 \longrightarrow \hat{q}^{\pi_0}(s,a)$

At from a randomised policy

ϵ' → A uniform random
$1-\epsilon'$ → $\text{argmax}\ q^{\pi}(s,a)$

What will be the policy in the 2nd episode if randomization is not there?

$\pi_1(s) = \text{argmax}\ \hat{q}^{\pi_0}(s,a)$ ↔ why is this a good idea?
$\pi_1(s) = \text{argmax}\ q^{\pi_0}(s,a)$ [PI variant]

Work with estimate of $q^{\pi_0}(s,a)$ and will get policy improvement (motivated by truncated PI)

We have $\pi_1(s)$ – is this optimal?

How should we improve $\pi_1(s)$?

Get $q^{\pi_1}(s,a)$ or its estimate.

If no randomization, then $\pi_1(s)$ is deterministic.

In the $\epsilon=1$ case:

$S_0=s$ S_1 $S_2=s$ → a' ... $S_{60}=s$

$A_t \to \pi_1(s_0)$ $\pi_1(s_1)$ $\pi_1(s_2)$ ...

R_0 R_1 R_2

$q^{\pi_1}(s,\pi(s))$

$q^{\pi_1}(s,a')$

On-~~line~~ and off policy algorithms

π → $1/|A|$, $1/|A|$, ... → $q^{\pi}(s,a)$ → $\text{argmax}\ q^{\pi}(s,a)$

Exploratory policy

Env → R_t → Agent → S_t → Returns to evaluate $q^{\pi}(s,a)$ for a target policy π.

Importance sampling:

Monte-carlo estimates: $X \to x_1, x_2, x_3, \ldots$

$$\mathbb{E}X = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Suppose we get samples of another random variable Y:

$$Y_1, Y_2, \ldots \to \frac{1}{N}\sum_{i=1}^{N} Y_i \approx \mathbb{E}X$$

$$\mathbb{E}X = \sum p_X(x).x$$

$$\frac{1}{N}\sum y_i \leftarrow \mathbb{E}y = \sum p_Y(y).y = \sum p_Y(x).x = \sum p_Y(x) \cdot \frac{p_X(x).x}{p_X(x)} = \sum_i p_X(x_i)\left(\frac{p_Y(x_i)}{p_X(x_i)}x_i\right)$$

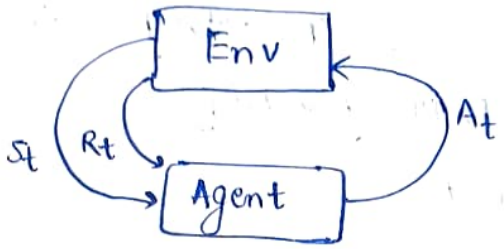
$$f(\cdot) \to \frac{1}{N}\sum_{i=1}^{N} f(x_i) \approx \mathbb{E}f(X) = \mathbb{E}_X f(x)\,\frac{p_Y(x)}{p_X(x)}$$

Lec-20 | 30-09-2025

Temporal Difference Method

Chap-7

Recall



$S_0, S_1, S_2, \ldots$

$R_0, R_1, R_2, \ldots$

$A_0, A_1, A_2, \ldots$

Value function for a policy π:

$A_t = \pi(S_t)$ or $\sim \pi(a|S_t)$

$$V^\pi(s) = \mathbb{E}(G_0) \approx \frac{1}{N}\sum_{i=1}^{N} G_0^i \quad \Big\} \text{MC}$$

- Extended to $q^\pi(s,a)$.

Another way of estimating $V^\pi(s)$:

$$V^\pi(s) = \mathcal{R}(s,a) + \gamma \sum p(s'|s,a)\, V^\pi(s')$$

$$= \mathbb{E}\left[\underbrace{R_0 + \gamma V^\pi(S_1)}_{\text{samples}}\right]$$

$R_0 + \gamma V^\pi(S_1)$

is like a sample of G_0.

$R_t + \gamma V^\pi(S_{t+1})$

is like a sample of G_t.

So, in TD, we use

$R_t + \gamma \hat{V}^\pi(S_{t+1})$.

$$\hat{V}^{\pi}_{k+1}(s_t) = \hat{V}^{\pi}_{k}(s_t) + \alpha_t\left(R_t + \gamma \hat{V}^{\pi}_{k}(s_{t+1}) - \hat{V}_k(s_t)\right)$$

↳ can be time-dependent

$$\hat{V}^{\pi}_{t+1}(s_t) = \hat{V}^{\pi}_{t}(s_t) + \alpha\left(R_t + \gamma \hat{V}^{\pi}_{t}(s_{t+1}) - \hat{V}_t(s_t)\right)$$

... "Bootstrap Estimate"

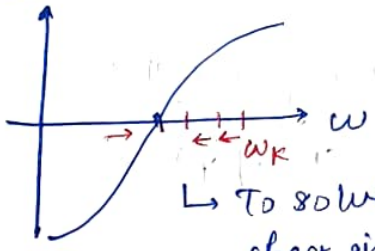
$R_t + \gamma \hat{V}^{\pi}_t(s_{t+1}) - \hat{V}_t(s_t)$: TD (Temporal Difference) error

$R_t + \gamma \hat{V}^{\pi}_t(s_{t+1})$ • : TD Target

we are trying to get $\hat{V}^{\pi}_t \to V^{\pi}$

Digression (Chap-6)

$g(w) = 0$



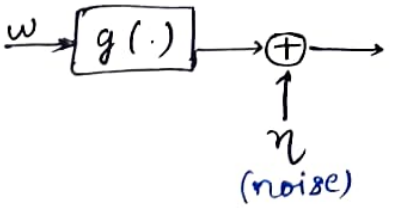
↳ To solve $g(w)=0$, take two values, check sign and reduce the interval.

[To solve $\frac{dJ(w)}{dw} = 0$: optimization problem

$$w_{K+1} = w_K - \alpha_K \frac{dJ(w_K)}{dw}$$]

$$w_{K+1} = w_K - \alpha_K\, g(w_K)$$

Take small α_K : $0 < \alpha_K < 1$



↳ Noisy

↳ Random (Not deterministic)

SGD : Stochastic Gradient Descent :

$$w_{K+1} = w_K - \alpha \tilde{g}(w_K, n_K)$$: Robbin-Munro Algorithm

↳ Used to solve $g(w)=0$ when the observation is noisy.

$$v^{\pi}(s) = R(s, \pi(s)) + \gamma \sum p(s'|s,a) \cdot v^{\pi}(s')$$

$$\Rightarrow 0 = R(s, \pi(s)) + \gamma \sum p(s'|s,a) \cdot v^{\pi}(s') - v^{\pi}(s)$$

$$\Rightarrow \vec{0} = \vec{R} + M\vec{v}^{\pi} \quad [\text{In Matrix form}]$$

$$\vec{0} = \boxed{g(\vec{v}^{\pi})} \longrightarrow \text{we don't know this}$$

↳ [Random reward, Random/sampled M]

RM Algorithm:

$$\bar{v}_{k+1} = \bar{v}_k - \alpha_k \tilde{g}(\bar{v}, \eta) \quad \left[\text{similar to } (R_t + \gamma \hat{v}_k^{\pi}(s_{t+1}) - \hat{v}_k(s_t)) \text{ in TD}\right]$$

Conditions: $\sum \alpha_k = \infty$

$\sum \alpha_k^2 < \infty$

How can we estimate $q^{\pi}(s,a)$?

- How to do this for Monte-Carlo & FVMC?

$s_0, s_1, \ldots$

$R_0, R_1, \ldots$ → first reward a/c to action a

→ a $(R_0^{a} + \gamma R_1 + \ldots)$

Choose all actions for 1st stage (not a/c to policy)

G_0 – return is still used.

$s \rightarrow a \quad R_0 + \gamma \hat{q}^{\pi}(s_{t+1}, A_{t+1})$

a/c to policy

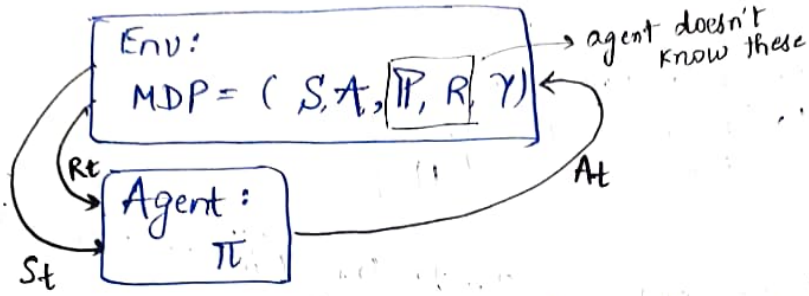
or

$\gamma \hat{v}^{\pi}(s_{t+1})$

$$\hat{v}^{\pi}(s) = \sum_a \frac{1}{|A|} \hat{q}(s,a)$$

Temporal difference Algorithm

Recall:



$$V^{\pi}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s' \mid s, \pi(s))\, V^{\pi}(s')$$

$$V^{*}(s) = \max_{a} \left\{ r(s,a) + \gamma \sum_{s'} p(s' \mid s, a)\, V^{*}(s') \right\}$$

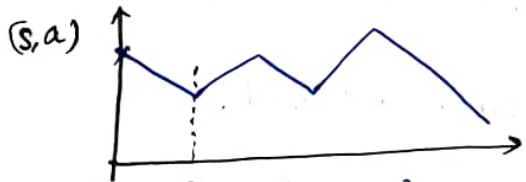
Value iteration

Policy improvement and iteration

$q^{\pi}(s,a) \longrightarrow \underset{a}{\text{argmax}}\, q^{\pi}(s,a)$

(If we know the model (MDP))

Estimate $q^{\pi}(s,a)$ from samples.



$$G_0 = R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots$$
$$= R_0 + \gamma \underbrace{(R_1 + \gamma R_2 + \cdots)}$$

↓ Replace by some other estimate (current estimate), $\hat{V}(S_1)$

TD Algorithm for policy evaluation

π (Known), $V^{\pi}(s)$ (estimate for every state) $= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s\right]$ [Here, $A_t = \pi(S_t)$]

$$= \mathbb{E}\left[G_0 \mid S_0 = s\right] \approx \frac{1}{N} \sum_{i} G_0^{(i)}$$

$$\hat{V}_N(s) = \frac{1}{N} \sum_{i} G_0^{(i)}$$

$$\hat{V}_{N+1}(s) = \hat{V}_N(s) + \alpha_N \left(G_0^{(N+1)} - \hat{V}_N(s)\right)$$

$G_0^{(N+1)} \rightarrow R_0 + \gamma \hat{V}_N(S_1)$ [Replace/Approx. G_0 by an estimate]

$$\hat{V}_{N+1}(s) = \hat{V}_N(s) - \alpha_N\left(\hat{V}_N(s) - (R_0 + \gamma \hat{V}_N(s_1))\right)$$ ... TD update equation

$\alpha_N = \frac{1}{N+1}$ or something else.

$\sum \alpha_N = \infty$

$\sum \alpha_N^2 < \infty$.

For any state,

$$V^\pi(s) \approx \frac{1}{N} \sum_i G_t^{(i)}$$

$$G_t = R_t + \gamma(R_{t+1} + \gamma R_{t+2} + \dots)$$

$$\therefore \hat{V}_{N+1}(s_t) = \hat{V}_N(s_t) - \alpha_N\left(\hat{V}_N(s_t) - (R_0 + \gamma \hat{V}_N(s_{t+1}))\right)$$

Disadvantage: We only know random R_0, for G_0, we estimate $\gamma(R_1 + \gamma R_2 + \dots)$ which may not be correct.

↳ Biased.

$\pi \rightarrow V^\pi$ or its estimate $\hat{V}$

For controls, we need Q^π.

TD for $Q^\pi(s,a)$:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s_0 = s, A_0 = a\right]$$

$\forall\, t > 0,\ A_t = \pi(s_t)$

$$= \mathbb{E}\left[G_0 \mid s_0 = s, A_0 = a\right]$$

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum p(s'|s,a)\, V^\pi(s')$$

$\Downarrow$ $\hat{V}$

$(s,a) \longrightarrow$ $R_0^{(1)}, R_0^{(2)}, R_0^{(3)}$ + $\hat{V}(s_1^{(1)}), \hat{V}(s_1^{(2)}), \hat{V}(s_1^{(3)})$

$\downarrow$ $\downarrow$

$$\mathcal{R}(s,a) + \gamma \sum p(s'|s,a)\, V^\pi(s') \rightarrow Q^\pi(s,a)$$

$\downarrow$

[we have to look every episode]

Estimate $Q^{\pi}(s,a)$ [something similar to $\hat{v}_N$]:

$$Q^{\pi}(s,a) = \mathbb{E}\left[G_0 \mid s_0 = s, A_0 = a\right]$$

$$\hat{Q}_N(s,a) \approx \frac{1}{N} \sum_i G_0^{(i)}$$

$$\hat{Q}_{N+1}(s,a) = \hat{Q}_N(s,a) + \alpha_N \left(G_0^{(N+1)} - \hat{Q}_N(s,a) \right)$$

$$G_0^{(N+1)} \approx R_0 + \gamma \hat{Q}_N(s_1, A_1)$$

$\mathbb{E}(R_1 + \gamma R_2 + \gamma R_3 + \dots)$ → is this some Q value?

$\downarrow$

$Q^{\pi}(s_1, A_1)$

we do not know this!

$\downarrow\downarrow$

$\hat{Q}_N(s_1, A_1)$

$$\therefore \hat{Q}_{N+1}(s,a) = \hat{Q}_N(s,a) - \alpha_N \left(\hat{Q}_N(s,a) - (R_0 + \gamma \hat{Q}_N(s_1, A_1)) \right)$$

$$\hat{Q}_{N+1}(S_t, A_t) = \hat{Q}_N(S_t, A_t) - \alpha_N \left(Q_N(S_t, A_t) - (R_t + \gamma \hat{Q}_N(S_{t+1}, A_{t+1})) \right)$$

... Q-function TD update equation

... $S_t A_t R_{t+1} S_{t+1} A_{t+1}$

- s, a are the state and action at time 't'. [$s = S_t$, $a = A_t$]

[N is taken as t]

Lec-22 | 21-10-2025

Temporal Difference (TD) algorithms for RL:

$\pi_0 \rightarrow q^{\pi_0}(s,a) \xrightarrow{\text{argmax}} \pi_1 \rightarrow \cdots \quad \pi^* = \underset{\pi}{\text{argmax}}\, V^{\pi}(s)$

MDP Model $\equiv (S, A, P, R, \gamma)$

(s,a) : $G_0^{\pi_0} = R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots$

$q^{\pi_0}(s,a) = \mathbb{E}\left[G_0^{\pi} \mid S_0 = s, A_0 = a\right]$

$\approx \underbrace{\frac{1}{N} \sum_{j=1}^{N} G_0^j}$

$\hat{q}_{N+1}(s,a) = \hat{q}_N(s,a) + \alpha\left(G^{N+1} - \hat{q}_N(s,a)\right)$

$G^{N+1} \rightarrow R + \gamma\, \hat{q}_N(S_1, A_1)$

$\hat{q}_{N+1}(s,a) = \hat{q}_N(s,a) + \alpha\left(G_t^{N+1} - \hat{q}_N(s,a)\right)$ [N is actually 't']

$t \quad S_t \; A_t \qquad t \quad S_t \; A_t \qquad t \quad S_t \; A_t$

① $\hat{q}_{t+1}(S_t, A_t) = \hat{q}_t(S_t, A_t) + \alpha\left(R_t + \gamma \hat{q}_t(S_{t+1}, A_{t+1}) - \hat{q}_t(S_t, A_t)\right)$

we need $S_t, A_t, R_t, S_{t+1}, A_{t+1}$: SRSA

At t, update (S_t, A_t).

② Others remain the same.

$\hat{q}_{t+1}(s', a') = \hat{q}_t(s', a')$,

$(s', a') \neq (S_t, A_t)$

H-W: we get a $V^{\pi}(s) \approx$ MDP soln.

SARSA Variants

- n-step SARSA:

$$R_t + \gamma \hat{q}_t(S_{t+1}, A_{t+1}) \longrightarrow R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots + \gamma^n R_{t+n} + \gamma^{n+1} \hat{q}(S_{t+n}, A_{t+n})$$

- Expected SARSA:

$$R_t + \gamma \hat{q}_t(S_{t+1}, A_{t+1}) \longrightarrow R_t + \gamma \underbrace{\mathbb{E}}_{\text{can be calculated}} \hat{q}(S_{t+1}, \underbrace{\bar{A}_{t+1}}_{\text{we know the distribution}})$$

Q-learning Algorithm

Recall:

$\pi_0 \rightarrow \hat{q}_{\pi_0}(s,a) \rightarrow \text{argmax} \rightarrow \pi_1$

$$\hat{q}_{t+1}(S_t, A_t) = \hat{q}_t(S_t, A_t) + \alpha_t \{ (R_t + \gamma \overbrace{\hat{q}_t(S_{t+1}, A_{t+1})}^{\text{estimation}}) - \hat{q}(S_t, A_t) \}$$

Main idea in the code:

Use a randomized policy
- $\epsilon \rightarrow$ uniform at random $a \in A$
- $1-\epsilon \rightarrow \underset{a}{\text{argmax}}\ \hat{q}(S_t, a)$

- Estimate $\hat{q}$.
- do policy improvement.

→ SRSA
+ variants – n-step expected SRSA

We also saw

Q-learning

$$\hat{q}_{t+1}(S_t, A_t) = \hat{q}_t(S_t, A_t) + \alpha_t \{ R_t + \gamma \max_a \hat{q}_t(S_{t+1}, a) - \hat{q}_t(S_t, A_t) \}$$

Observation:

$\hat{q} \approx$ MDP result →

$\underset{a}{\text{argmax}}\ \hat{q}(s,a) =$ opt. policy

In Q-learning:

Directly solve the Bellman's Optimality Equation.

$$v^{\pi^*} = \max_a \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v^{\pi^*}(s') \}$$

$$q^{\pi^*}(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \cdot \max_{a'} q^{\pi^*}(s',a')$$

$$v^{\pi^*}(s) = \max_a q^{\pi^*}(s,a)$$

If we ~~know~~ find a $q(\cdot)$ s.t.

$$q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \cdot \max_{a'} q(s',a')$$

To understand Q-learning,
let's start with the model-based procedure for q.

Start with $q_0(s,a)$.

$$q_{k+1}(s,a) = \underbrace{\mathbb{E}R(s,a)}_{r(s,a)} + \gamma \underbrace{\sum p(s'|s,a) \max_{a'} q_k(s',a')}_{\mathbb{E}\left[\max_{a'} q_k(S,a')\right]}$$

$k \to \infty$

S: random variable with prob. $p(s'|s,a)$

What we would have seen is

$$V_{k+1}(s) = \max_a \{ r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_k(s') \}$$

$$V_0(s) = \max_a q_0(s,a)$$

$$q_1(s,a) = r(s,a) + \gamma \sum p(s'|s,a) V_0(s')$$

$$V_1(s) = \max_a q_1(s,a)$$

$$q_{k+1}(s,a) = \mathbb{E}R(s,a) + \gamma \mathop{\mathbb{E}}_{s,a} \max_{a'} q_k(S,a')$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left(R^{(i)}(s,a) + \gamma \max_{a'} q_k(s^{(i)},a') \right)$$

$R^{(i)}(s,a)$: ↑ ; q_k: ~ arbitrary for initialised with some value.

What if we used an incremental update for this?

- Incremental on 'i'.

$$q_{(k+1,i)}(s,a) = q_{k+1,i-1}(s,a) + \alpha_{ki} \left(R^{(i)}(s,a) + \gamma \max_{a'} q_{k,i-1}(s^{(i)},a') - q_{k+1,i-1}(s,a) \right)$$

$(k+1,i) \to t$; $(s,a) \to s_t, A_t$

At every time steps t (combination of k & i, $(k+1,i)$), update q, each associated with an (s_t, A_t)

$$q_{t+1}(s_t,A_t) = q_t(s_t,A_t) + \alpha_t \left(R_t + \gamma \max_{a'} q_t(s_t,a) - q_t(s_t,A_t) \right)$$

- Unlike SRSA, we need to estimate

$R^{(i)}(s,a)$, $q_k(s^{(i)},a')$ } Neither depend on the policy

Types of policies for Q-learning:

- Behaviour policy: Used to run the system.
  - ↳ generates these $R^{(i)}$ and $S^{(i)}$ samples from the unknown $R, \mathbb{P}$.
- Target policy.

Q-learning = off policy algorithm
SARSA = on-policy algorithm.
} online learning

- Q-learning can be used on datasets, called offline RL.

| Ch-7

Lec-24 | 27-10-2025

Value function Approx. Methods

| Ch-8 of the book

Recall -

$MDP = (S, A, \mathbb{P}, R, \gamma)$

finite discrete

$V^{\pi}(s)$ or $q^{\pi}(s, a)$ → tables / arrays

Other RL problems:

These assumptions do not hold.

Motivating problems:

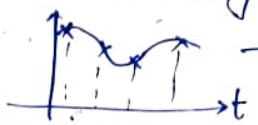
(a) Machine Repair Problem

Temp. of a particular component
$\in, S \subseteq [t_{min}, t_{max}]$

(b) Lunar Lander

real valued (x, y, z)
Orientation (θ, ϕ)

like continuous signal



→ can only store samples in computers.

Additionally, control variable

$a \in [a_{min}, a_{max}]$

↳ Continuous action space.

© Uncountable / discrete

eg. queue problem → [Sennatt]

$\mathcal{S} = \{0, 1, 2, 3, \dots\}$

state - no. of people in the queue.

Action space:

service rate $\in [\mu_{min}, \mu_{max}]$

Generalization:

$\mathcal{S}, \mathcal{A}$ → Continuous infinite

$\mathbb{P}$: $f_{s'|s,a}(s')$ → pdf

(PMF) $F_{s'|s,a}(s')$ → CDF

R : $f_{R|s,a}(r)$

$\int r \cdot f_{R|s,a}(r)\, dr = \bar{r}(s,a)$ → average reward

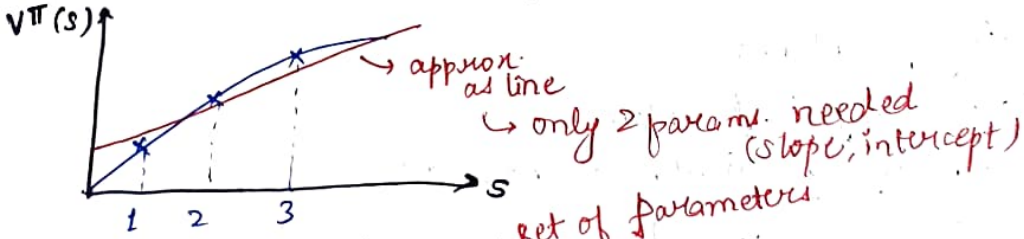
π : $\pi_{a|s} = \mathcal{N}(a | \mu_s, \sigma_s^2)$

Book: Bertsekas

- For continuous problems, Bellman optimality equation may not hold. This needs to be verified.

How to obtain $v^\pi(s)$, $s \in \mathcal{S}$ (General Space)

$s \in \mathbb{R}$



$v^\pi(s) \approx \hat{v}(s : \underline{w})$ → parametrized fn (set of parameters)

$= \sum \phi_i(s) \cdot w_i$ "Linear Function Approx. (LFA)"

Suppose we approximate $v^\pi(s)$ using a 'line'.

$w_0 + w_1 s$

$\phi_0(s) = 1$

$\phi_1(s) = s$

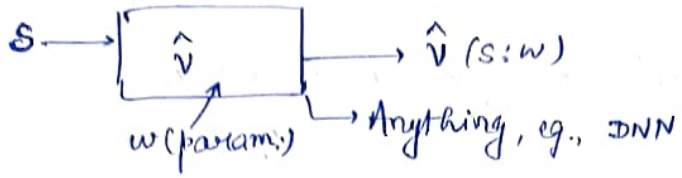
If $w_0 + w_1 s + w_2 s^2$

$\phi_0(s) = 1$

$\phi_1(s) = s, \quad \phi_2(s) = s^2$.

"linear in params. (may not be linear)

[Linear Regression]



$v^{\pi}(s) \approx \hat{v}(s:w)$

$= \phi(s)^T \cdot w$ [dot-product]

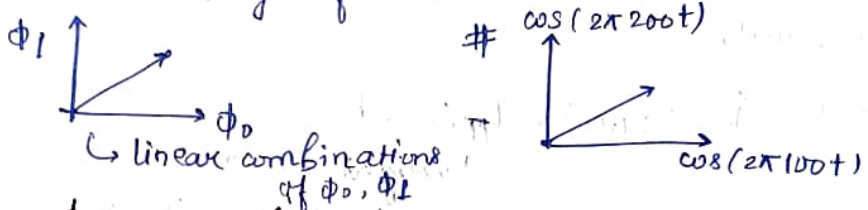
$w = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \end{pmatrix}$, $\phi(s) = \begin{pmatrix} \phi_0(s) \\ \phi_1(s) \\ \vdots \end{pmatrix}$

Question to which ϕ_i-s to use?

↳ Depends on ~~#~~ problem-to-problem → Guess (may not be possible)

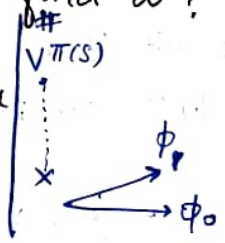
↳ Look into simple form of the problem

↳ Use DNN to get features.



Suppose ϕ_i-s are given/known — 'how to find $\vec{w}$?'

$\min_w \int \underbrace{(v^{\pi}(s) - \hat{v}(s,w))^2}_{\text{fn of } s} ds$ → total error (single value)



Suppose S is finite and discrete.

$\sum_{j=1}^{N} (v^{\pi}(s_j) - \hat{v}(s_j:w))^2$ ← find 'w' that minimizes this

To get exact approx., use impulse basis functions. [Finite, discrete]

$\phi_0(s) = \delta(s_0)$

$\phi_1(s) = \delta(s_1)$

$\phi_2(s) = \delta(s_2)$

$w_0 = v^{\pi}(s_0)$

$w_1 = v^{\pi}(s_1)$

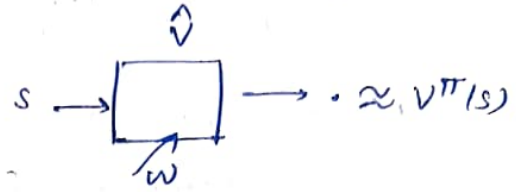
$w_2 = v^{\pi}(s_2)$

Lec-25 | 28-10-2025

Value function approximation → General $\mathcal{S}$ and $\mathcal{A}$ spaces.

Recall

$$v^{\pi}(s) \approx \hat{v}(s;w) = \sum_{i=0}^{M-1} \phi_i(s) \cdot w_i$$



w such that $\int (v^{\pi}(s) - \hat{v}(s:w))^2 ds$ ← not proper!

$$\sum_{i=1}^{N} (v^{\pi}(s_i) - \hat{v}[s_i;w))^2 = J(w) : \text{objective function.}$$

Find w to min $J(w)$.

Objective $J(w)$:

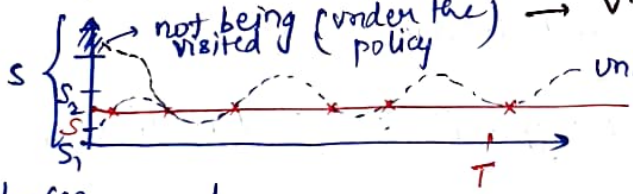
$$\int (v^{\pi}(s) - \hat{v}(s;w))^2 \, d_{\pi}(s) \, ds$$

↳ weight factor
↳ decays with 's'
↳ to make the integral converge (finite)

Write it as expectation which can be estimated as average.

$d_{\pi}(s)$ can be thought of as some pdf (of some state variable varying in $\mathcal{S}$).

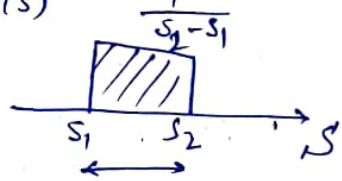
$$= \mathbb{E}(v^{\pi}(s) - \hat{v}(s;w))$$



$v^{\pi}(s)$ is defined for the policy under π } something like visited once

$\hat{v}(s;w) \approx v^{\pi}(s)$
↳ approx. may not be good for the one being visited once or few times.

Eg. $d_{\pi}(s)$



It means the approx. state is good for (s_1, s_2).
↳ Give high weight to the once being visited again & again.

s → being visited more-and-more.
↳ fraction of time ↑ being visited (for discrete spaces) ⇒ Eg. for $t = 1000$, it is being visited 100 times ⇓ frac. = $\frac{1}{10}$.
↳ distribution ↑
↳ Pdf ↑

Define $d_\pi(s)$ as the fraction of time a state is visited.

Suppose [Genie] gives $V^\pi(s)$

$$\frac{1}{N}\sum_{t=1}^{N}\left(V^\pi(S_t) - \hat{v}(S_t; w)\right)^2 = \mathbb{E}\left(V^\pi(s) - v(s; w)\right)$$

($V^\pi(S_t)$: at time 't')

d_π : dist. of the state for the policy

↳ Stationary probability distribution. [Read !]

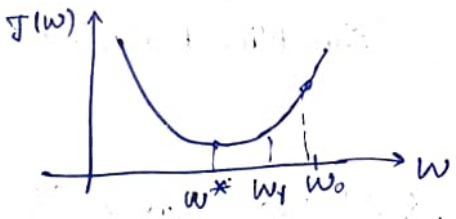
→ States get picked a/c to $d_\pi(s)$.

$$J(w) = \mathbb{E}_{d_\pi}\left[V^\pi(s) - \hat{v}(s; w)\right]^2 \rightarrow \text{minimize}$$

using gradient descent.

Start with parameter $w_0 \rightarrow w_1 \cdots w_K \rightarrow w_{K+1}$

$\hat{v}(s; w_K)$



$$w_{K+1} = w_K - \alpha_K \cdot \left(\nabla_w J(w_K)\right)$$

Gradient,

$$\nabla_w J(w) = \nabla_w \mathbb{E}\left(V^\pi(s) - \hat{v}(s; w)\right)^2$$

$$= \begin{pmatrix} \frac{\partial}{\partial w_0}\mathbb{E}\left(V^\pi(s) - \hat{v}(s; w)\right)^2 \\ \frac{\partial}{\partial w_1}\mathbb{E}\left(V^\pi(s) - \hat{v}(s; w)\right)^2 \\ \vdots \end{pmatrix}$$

$$= \mathbb{E}\,\frac{\partial}{\partial w_0}\left(V^\pi(s) - \hat{v}(s; w)\right)^2$$

$$= -2\mathbb{E}\left[\left(V^\pi(s) - \hat{v}(s; w)\right)\frac{\partial}{\partial w_0}\hat{v}(s; w_0)\right]$$

$$\text{vector} = \begin{bmatrix} -2\mathbb{E}\left(\underbrace{V^\pi(s) - \hat{v}(s; w)}_{\text{scalar}}\right)\frac{\partial \hat{v}}{\partial w_0}(s; w_0) \\ -2\mathbb{E}\left(V^\pi(s) - \hat{v}(s; w)\right)\frac{\partial \hat{v}}{\partial w_1}(s; w_1) \\ \vdots \end{bmatrix} \rightarrow \text{vector}$$

$$= -2\mathbb{E}\underbrace{\left(V^\pi(s) - \hat{v}(s; w)\right)}_{\text{scalar}}\cdot\underbrace{\nabla_w \hat{v}(s; w)}_{\text{vector (fn of vector } w)}$$

$$w_{k+1} = w_k + \alpha_k \cdot \mathbb{E}_{s \sim \pi}\left[\left(v^{\pi}(s) - \hat{v}(s;w)\right) \nabla_w \hat{v}(s;w)\right]$$

['2' absorbed in α_k]

$v^{\pi}(s)$: unknown $\Rightarrow$ cannot run!

Variant of G.D: Stochastic Gradient Descent (SGD).

- Problem to minimize objective fn: $\mathbb{E}_X f(x;\theta)$ → fn of θ
  ↳ min. w.r.t. θ

$$\theta_{k+1} = \theta_k - \alpha_k \nabla_\theta \mathbb{E}_x f(x;\theta)$$

$$\theta_{k+1} = \theta_k - \alpha_k \nabla_\theta f(x_k;\theta) \rightarrow \text{Expectation replaced by the value.}$$

↳ Expectation not needed, only needs samples.

[Batch method vs. sample method]

[Mini-batch → take few samples & average]

Change the GD step to SGD:

$$w_{k+1} = w_k + \alpha_k \left(v^{\pi}(s_k) \cdot \hat{v}(s_k;w)\right) \nabla_w \hat{v}(s_k;w)$$

↳ k^{th} step

Lec-26 | 03-11-2025

Recall

General state and action spaces, S, A

Approximation $\longrightarrow V^\pi(s)$

$\hat{v}(s; w) \approx v^\pi(s)$

$\hat{v}(s; w) = \sum_i w_i \cdot \phi_i(s)$ → Basis fn

Objective fn:

$$J(w) = \int_S \left(\hat{v}(s; w) - v^\pi(s)\right)^2 d_\pi(s)\, ds.$$

→ pdf = stationary distribution

$$= \mathbb{E}\left(\hat{v}(s; w) - v^\pi(s)\right)^2$$

Policy evaluation → Choose w so that $J(w)$ is minimized.

$\min_w J(w)$ using G.D.

$$w_{k+1} = w_k - \alpha_k \cdot \nabla_w J(w)$$

$$= w_k - \tilde{\alpha}_k \mathbb{E}\left[\left(\hat{v}(s; w_k) - v^\pi(s)\right) \cdot \nabla_w \hat{v}(s; w_k)\right] \quad (\because \tilde{\alpha} = 2\alpha_k)$$

↳ GD with expectation

↳ we can use stochastic GD

$$w_{k+1} = w_k + \alpha_k \left(v^\pi(s_k) - \hat{v}(s_k; w_k)\right) \cdot \nabla_w \hat{v}(s_k; w_k)$$

↳ for doing update, requires v^π ⇒ Not implementable.

↳ If w_k is known ⇒ $\hat{v}$ can be evaluated, but v^π is unknown.

Estimate v^π. Approximate from return.

$v^\pi(s_k^t) = \mathbb{E}\, G_t$

$v^\pi(s_t) = \mathbb{E}\, G_t$

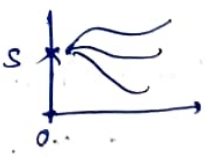
$$w_{k+1} = w_k + \alpha_k \left(\square \leftarrow \text{estimate}\right) \nabla \hat{v}(s; w_k)$$

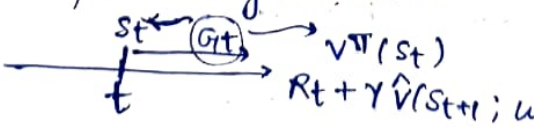
$\hat{v}(s; w_k)$

for every state s.

Monte Carlo:

$$v^\pi(s) \approx \frac{1}{N} \sum_i G_0^{(i)}(s)$$





MC: $w_{t+1} = w_t + \alpha_t \left(G_t - \hat{v}(s_t; w_t)\right) \nabla \hat{v}(s_t; w_t)$

↳ Implementable.

TD: $w_{t+1} = w_t + \alpha_t \left(R_t + \gamma \hat{v}(s_{t+1}; w_t) - \hat{v}(s_t; w_t) \right) \cdot \nabla \hat{v}(s_t; w_t)$

OpenAI's gym.

$\hat{v}(s, w) = \sum w_i \phi_i$

$\nabla_w (\hat{v}(s, w)) = \begin{pmatrix} \phi(0) \\ \phi(1) \\ \vdots \end{pmatrix}$

SARSA:

$v^\pi(s, a) \longrightarrow q^\pi(s, a)$ policy improvement

Lec-27 | 04-11-2025

Recall

$v^\pi(s) \longrightarrow \hat{v}(s; w)$

$q^\pi(s, a) \longrightarrow \hat{q}(s, a; w)$

linear fn approx.
$\hat{v}(s) = \sum w_i \phi_i(s)$

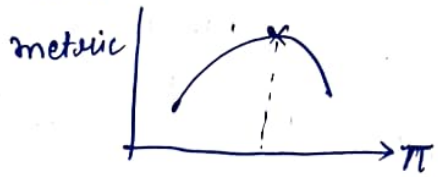
SGD update rules:

$w_{t+1} = w_t + \alpha_t \left(TD/MC - \hat{v}(s; w) \right) \nabla_w \hat{v}(s_t; w_t)$

$= w_t + \alpha_t \left(TD/MC - \hat{q}(s_t, A_t, w) \right) \nabla_w \hat{q}(s_t, A_t, w_t)$

$\hat{q}^\pi \approx \hat{q} \rightarrow$ policy improvement.

Policy gradient (Ch, 9)

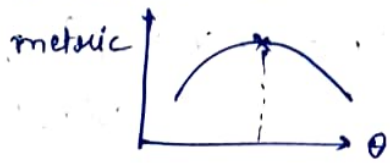


How do we represent policies?

$\pi(s) = \sum \theta_i \cdot \psi_i(s)$

parameters; different basis fn.

Optimize some metric with θ.



- Not always deterministic; we also do exploration.
- Represent the policy as distribution.

$$\pi(a|s) = \mathcal{N}\left(a : \mu(s,\theta)\ \sigma^2(s,\theta)\right) \rightarrow \text{Randomized policy}$$

↳ mean & variance for each state

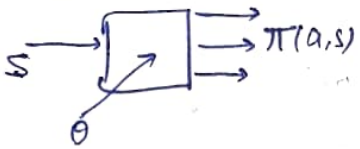
↳ used when A is continuous.

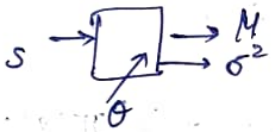
When A is discrete,

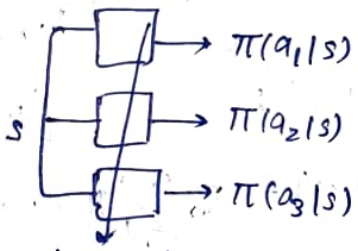
$$\pi(a,s) = \frac{e^{h(a;s,\theta)}}{\sum_{a'} e^{h(a';s,\theta)}}$$: soft-max function

(→ some fn of a)

Implementations :

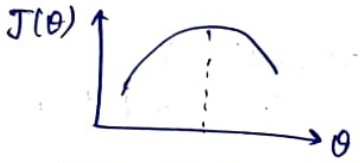






Optimize the policy with the parameter 'θ'.

Metric → ?



$\max_{\theta} J(\theta)$

→ argmax = θ^*

$\pi(a|s;\theta^*)$

What is J(θ) ?

- $V^{\pi}(s_0) \rightarrow \underset{\pi}{\text{argmax}}\ V^{\pi}(s_0)$
- $\mathbb{E}_d V^{\pi}(s)$ → denote as $\bar{V}^{\pi}$ ✓
  ↳ average with some distribution 'd'
  ↳ to choose initial state.
- $\mathbb{E}_d r(s, \pi(s))$ → like average reward

We will use

$\mathbb{E}_d\, v^{\pi}(s)$ - denote $v^{\pi_\theta} \rightarrow J(\theta)$.

↳ parameterized policy

How to maximize?

↳ Gradient Ascent

$$\bar{\theta}_{k+1} = \bar{\theta}_k + \alpha_k \nabla_\theta J(\theta)$$

↳ vector

Gradient $\nabla_\theta J(\theta)$?

$$\nabla_\theta \bar{V}^{\pi}_{\theta} = \nabla_\theta \left(\sum d(s)\, v^{\pi_\theta}(s) \right)$$

↳ pdf

$\nabla_\theta (v^{\pi_\theta}(s))$: $[\theta \rightarrow \text{vector}]$

$$\nabla_\theta \left(v^{\pi_\theta}(s)\right) = \nabla_\theta \left(\sum_a \pi(a|s;\theta).\, q^{\pi_\theta}(s,a) \right)$$

$$= \sum_a \boxed{\nabla_\theta \pi(a|s;\theta).\, q^{\pi_\theta}(s,a)} + \sum \pi(a|s,a).\nabla_\theta q^{\pi_\theta}(s,a)$$

A Special-case: Finite horizon with only 1 time-step

$$q^{\pi_\theta}(s,a) = r(s,a)$$

$$\Rightarrow \nabla_\theta (v^{\pi_\theta}(s)) = \nabla_\theta \pi(a|s;\theta).\, r(s,a)$$

(∵ second term → zero)

$$\nabla_\theta\, q^{\pi_\theta}(s,a) = \nabla_\theta \left(r(s,a) + \gamma \sum_{s'} p(s'|s,a)\, v^{\pi_\theta}(s') \right)$$

(∇_θ of $r(s,a)$ → 0)

$$= \gamma \sum_{s'} p(s'|s,a).\, \nabla_\theta v^{\pi_\theta}(s')$$

∴ we obtain

$$\nabla_\theta (v^{\pi_\theta}(s)) = \sum_a \nabla_\theta \pi(a|s;\theta).\, q^{\pi_\theta}(s,a)$$

$$+ \gamma \sum_a \pi(a|s,\theta). \sum_{s'} p(s'|s,a) : \nabla_\theta v^{\pi_\theta}(s').$$

↳ vector eqn for a single 's'

Take dimension $(\theta) = 1$, for simplicity.

$$\frac{d}{d\theta} v^{\pi_\theta}(s) = \sum_a \frac{d}{d\theta} \pi(a|s;\theta) \cdot q^{\pi_\theta}(s,a) + \gamma \left(\sum_a \pi(a|s,\theta) \cdot \sum_{s'} p(s'|s,a) \right) \cdot \frac{d}{d\theta} v^{\pi_\theta}(s') \quad \forall s$$

(same thing on both side; $\sum_{s'} p(s'|s,a) = p^{\pi_\theta}(s'|s)$)

$$\Rightarrow \left(\frac{dv^{\pi_\theta}}{d\theta}(\cdot) \right) = \left(u(s) \right) + \gamma p^{\pi} \left(\frac{d}{d\theta} v^{\pi_\theta}(\cdot) \right)$$

$$\Rightarrow \left(\frac{d}{d\theta} v^{\pi_\theta}(\cdot) \right) - \left(I_{|s|} - \gamma p^{\pi} \right) = \left(u(s) \right)$$

$$\Rightarrow \left(I_{|s|} - \gamma p^{\pi} \right) \left(\frac{d}{d\theta} v^{\pi_\theta}(\cdot) \right) = \left(u(s) \right)$$

($I_{|s|} - \gamma p^{\pi}$ is invertible)

$$\Rightarrow \left(\frac{d}{d\theta} v^{\pi_\theta}(\cdot) \right) = \left(I_{|s|} - \gamma p^{\pi} \right)^{-1} \left(u(s) \right)$$

($I_{|s| \times |s|}$) → independent of q^{π}

↳ something like $\nabla_\theta \pi(a|s,\theta) \cdot q^{\pi_\theta}(s,a)$.

Lec-28

Now,

$$\left(\frac{d}{d\theta} v^{\pi_\theta}(\cdot) \right) = \underbrace{(I - \gamma p^{\pi})^{-1}}_{I + \gamma p^{\pi} + \gamma^2 (p^{\pi})^2 + \gamma^3 (p^{\pi})^3 + \dots} \left(\sum_a \nabla_\theta \pi(a|s;\theta) \cdot q^{\pi_\theta}(s,a) \right)$$

For a particular 's',

$$\frac{d}{d\theta} v^{\pi_\theta}(s) = \sum_a \nabla_\theta \cdot \pi(a|s,\theta) \cdot q^{\pi_\theta}(s,a) + \gamma \sum_{s'} p^{\pi}(s'|s) \cdot \sum_a \nabla_\theta \cdot \pi(a|s',a) \cdot q^{\pi_\theta}(s',a) + \gamma^2 \sum_{s'} (p^{\pi}(s'|s))^2 \sum_a \nabla_\theta \cdot \pi(a|s',\theta) \cdot q^{\pi_\theta}(s',a) + \dots$$

$$= \sum_a \nabla_\theta \cdot \pi(a|s,\theta) \cdot q^{\pi_\theta}(s,a) + \sum_a \sum_{s'} \left(\gamma p^{\pi}(s'|s) + \gamma^2 (p^{\pi}(s'|s))^2 + \dots \right) \cdot \nabla_\theta \pi(a|s',\theta) \, q^{\pi_\theta}(s',a)$$

About value function approximation :

$d^\pi = d^\pi \cdot P^\pi$ $\Rightarrow \sum_s \underbrace{P_r\{s\}}_{d^\pi = f^\pi} \cdot P_r(s'|s) = \underbrace{P_r(s')}_{d^\pi}$

↓

Stationary distribution

↓

Represent it by f instead.

$$\sum f(s) \frac{d}{d\theta} v^{\pi_\theta}(s) = \sum_s f(s) \Big(\sum_a \nabla_\theta \cdot \pi(a|s;\theta) \cdot q^{\pi_\theta}(s,a) + \sum_a \sum_{s'} \big(\gamma P^\pi(s'|s) + \gamma^2 (P^\pi(s'|s))^2 + \dots \big) \cdot \nabla_\theta \pi(a|s',\theta) \cdot q^{\pi_\theta}(s',a) \Big)$$

$$\sum_a \sum_{s'} \sum_s f(s) \big(\gamma P^\pi(s'|s) + \gamma^2 P^\pi(s'|s)^2 + \dots \big) \cdot \nabla_\theta \pi(a|s',\theta) \cdot q^{\pi_\theta}(s',a)$$

$$\sum_a \sum_{s'} \big(\gamma f(s') + \gamma^2 f(s') + \dots \big) \nabla_\theta \pi(a|s',\theta) \cdot q^{\pi_\theta}(s',a)$$

$$= \sum_a \sum_{s'} f(s') \frac{\gamma}{1-\gamma} \cdot \nabla_\theta \pi(a|s',\theta) \cdot q^{\pi_\theta}(s',a)$$

$$= \sum_a \sum_s \boxed{\frac{f(s)}{1-\gamma}} \cdot \boxed{\nabla_\theta \pi(a|s,\theta)} \cdot q^{\pi_\theta}(s,a) \rightsquigarrow \text{No } \nabla_\theta q^{\pi_\theta}$$

$$\therefore \nabla_\theta J(\theta) = \sum_a \sum_s \underbrace{\frac{f(s)}{1-\gamma}} \cdot \nabla_\theta \pi(a|s;\theta) \, q^{\pi_\theta}(s,a)$$

→ No gradient of q^{π_θ}; only of π.

↓ Not a pdf; doesn't sum to 1;

$\frac{1}{1-\gamma}$ → constant → can be taken with α. $[\alpha \nabla_\theta J(\theta)]$

Log-trick :

$$\nabla_\theta \ln \pi(a|s;\theta) = \frac{\nabla_\theta \pi(a|s;\theta)}{\pi(a|s;\theta)}$$

$$\Rightarrow \pi(a|s;\theta) \cdot \nabla_\theta \ln \pi(a|s;\theta) = \nabla_\theta \pi(a|s;\theta)$$

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{S\sim J(S)} \; \mathop{\mathbb{E}}_{A\sim\pi(a|s;\theta)} \left(\nabla_\theta \ln \pi(A|s;\theta)\right) \cdot q^{\pi_\theta}(S,A)$$

S_t, A_t → at a time 't' are samples from these distributions.

$\nabla_\theta \ln \pi(\theta_t | S_t; \theta) \cdot q^{\pi_\theta}(S_t, A_t)$

$\theta_{t+1} = \theta_t + \alpha_t \left(\nabla_\theta \ln \pi(A_t|S_t;\theta) \cdot q^{\pi_\theta}(S_t, A_t)\right)$ → Update equation

↳ Not implementable as q is not known.

Estimate 'q' by MC or TD.

Monte-Carlo estimate: Replace by G_t.

$$\theta_{t+1} = \theta_t + \alpha_t \left(\nabla_\theta \ln \pi(A_t|S_t;\theta)\right) G_t$$ ... REINFORCE algorithm

How to compute $\nabla_\theta \ln \pi(a|s;\theta)$ for a discrete A.

$$\pi(a|s;\theta) = \frac{e^{\beta h(a|s;\theta)}}{\sum_{a'} e^{\beta h(a'|s;\theta)}} \quad (\because \text{softmax})$$

↳ to control overflow if h is large. (e^h → much larger)

↳ (Temperature)

$$\Rightarrow \ln \pi(a|s;\theta) = \beta \cdot h(a|s;\theta) - \ln \sum_{a'} e^{\beta h(a'|s;\theta)}$$

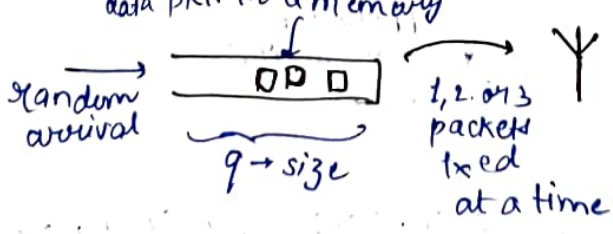
$$\Rightarrow \nabla_\theta \ln \pi(a|s;\theta) = \beta \nabla_\theta h(a|s;\theta) - \frac{\sum_{a'} \left(e^{\beta h(a'|s;a)} \nabla_\theta h(a'|s;a)\right)}{\sum_{a'} e^{\beta h(a'|s;a)}}$$

$$h(a|s;\theta) = \sum \theta_i \psi_i(a,s)$$

$$\Rightarrow \nabla_\theta h(a|s;\theta) = \left(\psi_i(a,s)\right)$$

↳ vector

Queue Control Environment:



Lec-29 | 06-11-2025

Actor-Critic Algorithms

Recall:

Policies: Parameterised functions $\pi(s)$, $\pi(a|s)$

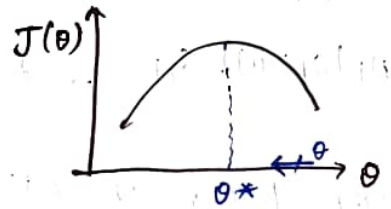
$\pi(s;\theta)$ $\pi(a|s;\theta)$

$$\pi(a|s;\theta) = \frac{e^{h(s,a;\theta)}}{\sum_{a'} e^{h(s,a';\theta)}} \quad \text{(softmax)},$$

↳ $a \in$ discrete

$$h(s,a;\theta) = \sum_i \phi_i \psi_i(s,a)$$

$$\pi(a|s;\theta) = \mathcal{N}(a \mid \mu_s(\theta), \sigma_s^2(\theta)) \rightarrow \text{Gaussian (continuous)}$$



$$J(\theta) = \mathbb{E}_d\, v^{\pi_\theta}(s)$$

Update: $\theta_{k+1} = \theta_k + \alpha_k \nabla_\theta J(\theta)\big|_{\theta=\theta_k}$

Gradient of $J(\theta)$,

$$\nabla_\theta J(\theta) = \mathbb{E}_d\left(\nabla_\theta v^{\pi_\theta}(s)\right)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_S\, \mathbb{E}_{A\sim\pi(a|s;\theta)}\left(\nabla_\theta \ln \pi(A|s;\theta)\right). q^{\pi_\theta}(S,A).$$

↓ a sample

$$\nabla_\theta \ln \pi(A_t|S_t;\theta).\, q^{\pi_\theta}(S_t,A_t)$$

↙

SGD: $\boxed{\theta_{t+1} = \theta_k + \alpha_t \nabla_\theta \ln \pi(A_t|S_t;\theta).\, q^{\pi_\theta}(S_t,A_t)}$

unknown

↙

Reinforce:

Replace by G_t → MC-return

Approximate $q^{\pi_\theta}(S_t,A_t) \approx \hat{q}(S_t,A_t)$ by TD method:

↓ approx. by a parametrised q

$\hat{q}(s,a,w)$

SGD update:

$$\theta_{t+1} = \theta_k + \alpha_t^\theta \left(\nabla_\theta \ln \pi(A_t|S_t;\theta)\right).\hat{q}(S_t,A_t;w_t)$$

↳ Actor update

But we don't know w_t^*. So update 'w' also side-by-side to make correct approx. to 'q'.

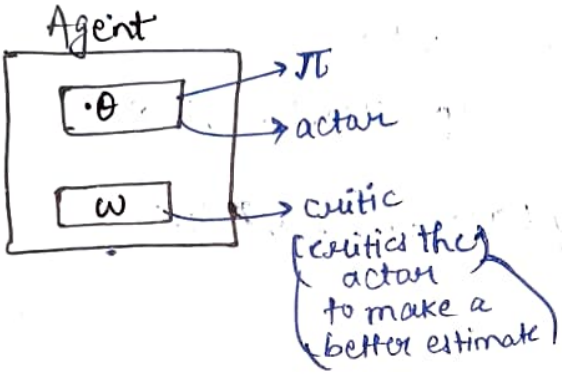
$$w_{t+1} = w_t + \alpha_t^w \cdot \left[\left(R_t + \gamma \cdot \hat{q}(S_{t+1}, A_{t+1}; w_t)\right) - \hat{q}(S_t, A_t; w_t)\right] \cdot \nabla_w \hat{q}(S_t, A_t; w_t)$$

↳ critic-update ! $q(S_t, A_t)$

↳ "Actor-critic algorithm" (Q-actor critic) [QAC]

~~[Actor-update]~~

↳ Requires correct q-estimate



QAC with a small modification:

A2C — Advantage Action Critic

$$\mathbb{E}_{A\sim\pi(A|S,\theta)}\left(\nabla_\theta \ln \pi(a|s;\theta)\right)\left(q^\pi(s,a;w) - b(s)\right)$$

(b(s): some fn, choose as $V^\pi(s)$)

= same-gradient as before

$$q^\pi(s,a;w) \approx R_t + \gamma\, \hat{q}(S_{t+1}, A_{t+1}; w_t)$$

$$V^\pi(s) \approx \text{avg}(R_t) + \gamma \cdot \text{avg}\left(\hat{q}(S_{t+1}, A_{t+1}; w_t)\right)$$

↳ cancels

$$\mathbb{E}_{A\sim\pi(a|s,\theta)}\left(\nabla_\theta \ln \pi(a|s;\theta)\right)\left(\hat{q}(s,a;w) - \hat{v}(s;w)\right)$$

ideally ↓ small variance

$\hat{q}(s,a) - V^\pi(s)$: Advantage functin

$A^\pi(s,a)$ ↳ A2C uses this fn instead of q

$$\therefore \mathbb{E}_{A\sim\pi(a|s;\theta)}\left(\nabla_\theta \ln \pi(a|s;\theta)\right)\left(A^\pi(s,a)\right)$$

$$= \mathbb{E}_{A\sim\pi(a|s;\theta)} \nabla_\theta \ln\left(\pi(a|s;\theta)\right) \cdot q^\pi(s,a)$$

$$\mathbb{E}_{A\sim\pi(a|s;\theta)} \nabla_\theta \ln \pi(a|s;\theta)\ V^\pi(s,a)$$

$$= \sum_a V^\pi(s)\ \nabla_\theta \Big(\sum_a \pi(a|s;\theta) \Big) = 0$$

$$\therefore \theta_{t+1} = \theta_t + \alpha_{\theta_t} + \nabla_\theta \ln \pi(A_t|s_t;\theta_t) \cdot \hat{A}(s_t, A_t; \theta_t)$$

$$A^\pi(s,a) = q^\pi(s,a) - V^\pi(s)$$
$$= r(s,a) + \gamma \mathbb{E}\, V^\pi(s_t) - V^\pi(s)$$
(estimate using) $\rightarrow R_t + \gamma \hat{V}^\pi(s_{t+1}) - \hat{V}^\pi(s_t)$

Final update:

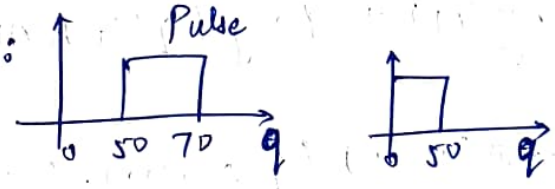
$$\theta_{t+1} = \theta_t + \alpha^\theta_t \cdot \nabla_\theta \ln \pi(\theta_t|s_t;\theta_t) \cdot \big[R_t + \gamma \hat{V}(s_{t+1};w_t) - \hat{V}(s_t;w_t) \big]$$
(Actor)

$$w_{t+1} = w_t + \alpha^w_t \cdot \big(R_t + \gamma \hat{V}(s_{t+1};w_t) - \hat{V}(s_t;w_t) \big) \cdot \nabla_w \hat{V}(s_t;w_t)$$
(Critic)

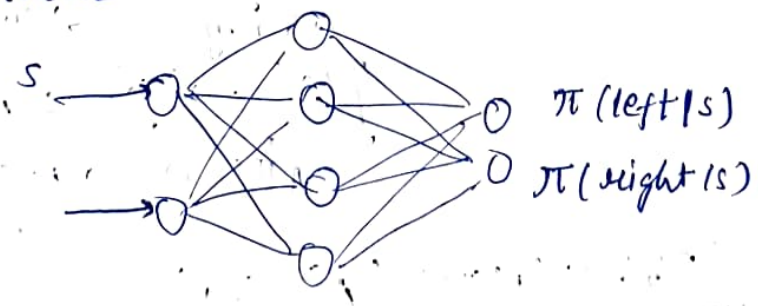
Implementations:

Basis fn's (ψ):

ψ_0's:



Minimal RL



Model-Based RL methods:

Upto now, model free

→ $x_{t+1} = \boxed{A} x_t + \boxed{B} a_t + N_t$

(A, B: unknown)

Let's estimate A and B.

→ $R_t = -(x_t Q x_t + u_t R u_t)$ ← LQR

$a_t = K x_t$

DYNA algorithm:

↳ Uses experience samples

↳ we see (S_t, A_t, R_t, S_{t+1}).

- Actual model: $\mathcal{H}(s,a)$, P
- Sample-based model: Train the system/weights on the model (if sufficient memory is present)

S	A	R	S^+
0	1	5	2
1	0	−2	1

Lec-30 | 10-11-2025

Recall

Model-free and Model-based RL algorithms

- Value
  - SARSA
  - Q-learning
- Policy gradient

↳ DYNA (samples to build a sample-based model)

... Build a model → solve an MDP

Function approximations: $\hat{v}(s;w)$, $\hat{q}(s,a;w)$, $\pi(a|s;\theta)$

Linear function approx.: $\sum w_i \phi_i(s)$ or $\sum w_i \phi_i(s,a)$

$\sum \theta_i \psi_i(s)$

Basis function: Appropriate functions - very difficult!

REINFORCE: $\pi(a|s,\theta)$

Neural Network:

$n : \mathbb{R}^2 \to \mathbb{R}$

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$ ← bias

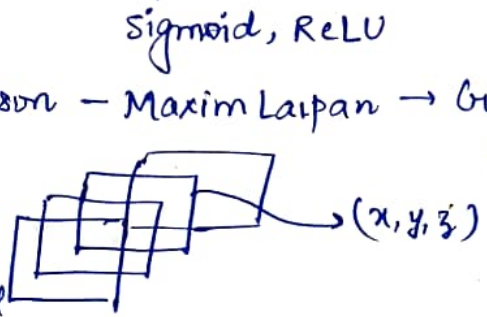
↓

Sigmoid, ReLU

→ Deep RL Handson – Maxim Lapan → Github (Chap-3)

Tensors:

↳ multi-dimensional array

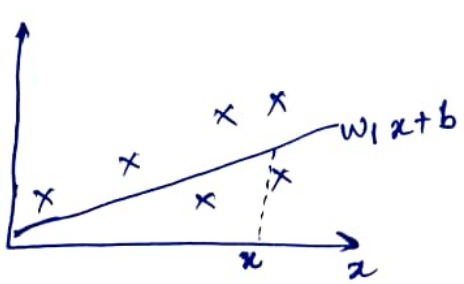


| Scalar, vector, metrices

$n: w_1x_1 + w_2x_2 + w_3x_3 + b$

$n: w_1x + b$

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix}$$



$$w_{k+1} = w_k - \alpha_k \left(\frac{\partial L}{\partial w} \Big|_{w_k} \right)$$

$$b_{k+1} = b_k - \alpha_k \left(\frac{\partial L}{\partial w} \Big|_{b_k} \right)$$

∇

$$\theta_{k+1} = \theta_k - \alpha_k \nabla L$$

n: $w_1 x + b$

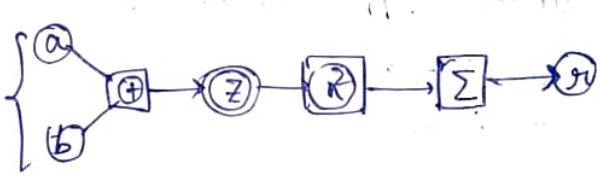
$(w_1 + \Delta w_1) x + b$

$\nabla = \begin{pmatrix} x \\ 1 \end{pmatrix}$

$f = f_2(f_1(x))$

$$\left[\frac{df}{dx} = \frac{df}{df_2} \cdot \frac{df_2}{df_1} \cdot \frac{df_1}{dx} \right]$$

↔ Micrograd implementation: [YouTube : Karpathy]



Minimal RL codes

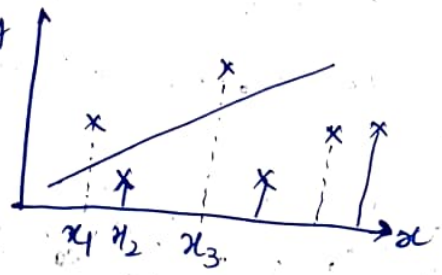
Recall:

Lec-31 | 11-11-2025

Tensor: multi-dimensional array

- Functions
- Autograd — optimization

$\nabla f(x)$

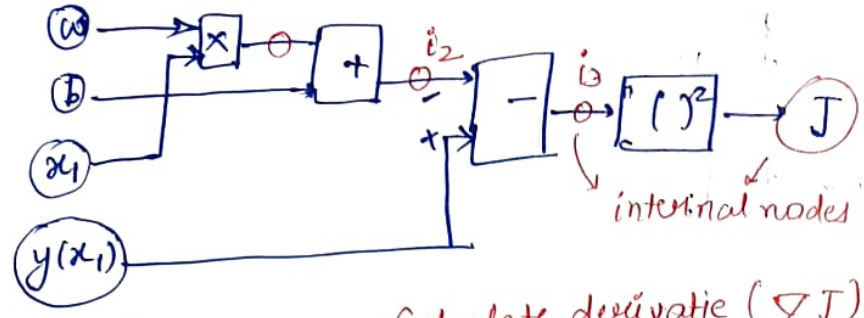


$y = wx + b$

$$\sum_{i=1}^{N} (y(x_i) - (wx_i + b))^2 = J\begin{pmatrix} w \\ b \end{pmatrix}$$

$$\begin{pmatrix} w \\ b \end{pmatrix}_{k+1} = \begin{pmatrix} w \\ b \end{pmatrix}_k - \alpha_k \cdot \nabla J\begin{pmatrix} w \\ b \end{pmatrix}$$

$\begin{matrix} w \\ b \end{matrix}$ } Two variables



→ computation graph (in pytorch)

Nodes (leaf nodes)

Calculate derivatie (∇J) by chain-rule.

Required-grad → The required gradients to be calculated.

$$\frac{\partial J}{\partial b} = (2i_3)(-1)(1) = -2i_3$$
$$= -2(y(x_1) - (w(x_1) + b))$$

↳ .backward()

Neural Network: Function approximator

$(w^T x + b) \longrightarrow \sigma(\cdot) \longrightarrow$ (non-linear)

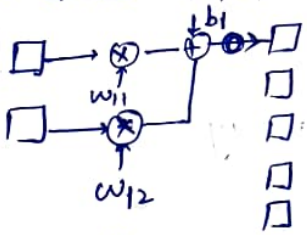
nn.Linear(,)

↓

tensor.nn

→ ip/op dimensions

ℓ = nn.Linear(2,5)



(15 parameters)

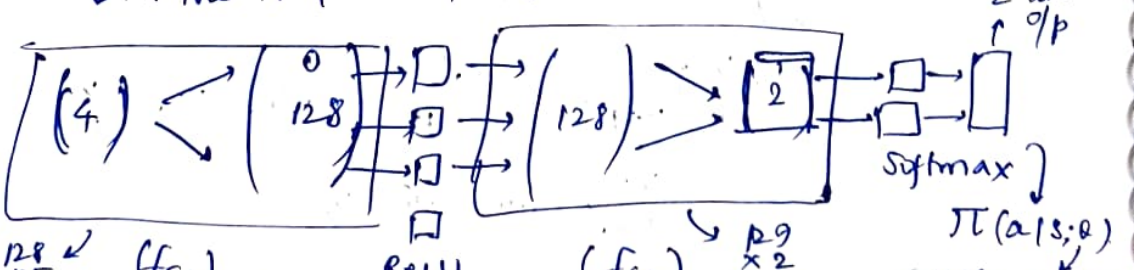
- Zero the gradient → .zero_grad()
- Cascade the blocks → .nn.sequential(, ,)

Recall REINFORCE algorithm: policy gradient theorem

$\pi(a|s;\theta)$ ↳ ~~$\mathbb{E}_S \mathbb{E}_{A\sim\pi(s)}$~~ $\nabla \ln(\pi(A|s,\theta)) \cdot q(s,A)$ → G_t

$= \nabla_\theta \ln \pi(A_t | s_t, \theta_t) \cdot G_t$

In the implementation



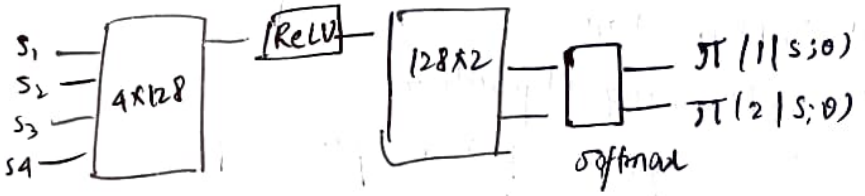
Optimizers: Helps in local minimas

Eg. Adam (gradient smoothener)

Lec-32 | 12-11-2025

REINFORCE

$\pi(a|s;\theta)$



$J(\theta) = \mathbb{E} V^{\pi}(s)$

$\nabla J(\theta) = \mathbb{E}_s \mathbb{E}_{A\sim\pi(.|s,\theta)} (\nabla_\theta \pi(A|s;\theta))\, q^{\pi}(s_t, A_t)$

$\nabla J(\theta) = \nabla_\theta \pi(A_t|s_t,\theta_t)\, G_t$

$R_1, R_2, \ldots R_T$

$\pi(A_0|s_0) \ldots \pi(A_T|s_T;\theta)$

DQN (Deep Q-learning):

Q-learning: Directly learn optimal q (Tabular) by solving Bellman optimality equation.

- Chapter-8 → sec. 8.4
- Book: Lapan

$$q_{t+1}(s_t,A_t) = q_t(s_t,A_t) + \alpha_t\left(R_t + \gamma \max_a q(s_{t+1},a) - q(s_t,A_t)\right)$$

Continuous: Parameterized by weight 'w'.

$q(s,a,w)$

Update w: $w_{t+1} = w_t + \alpha_t (R_t + \gamma \max_a q(s_{t+1},a;w_t) - q(s_t,A_t;w_t)) \nabla_w q(s_t,A_t;w_t)$

Minimize the error: $(R_t + \gamma \max_a \hat{q}(s_{t+1},a,w) - \hat{q}(s_t,A_t,w))^2$ (target)

- Error is also parameterised by w; whi b/c of which it is no longer ∇q.
- Introduce one more parametrization, q', q'^2 or, q^{target}, q, and q^{target} should not or slowly change.

Experience Replay.

Lec-33 | 17-11-2025

DQN Implementation

↳ Q-learning algorithm

↳ finding $q(s;a)$ for an optimal policy.

In SARSA, at t, we we are at S_t, A_t,

$R_t + \gamma \hat{q}(S_{t+1}, A_{t+1})$ → update the current estimate of S_t, A_t

↑ for a particular policy

Q-learning target:

$$R_t + \gamma \max_a \hat{q}(S_{t+1}, a)$$

$$\hat{q}_{t+1}(S_t, A_t) = \hat{q}_t(S_t, A_t) + \alpha_t(R_t + \gamma \max_a \hat{q}(S_{t+1}, a) - \hat{q}(S_t, A_t))$$ ... Tabular update

↓

$\hat{q}(S_t, A_t; w_t)$

↳ update w_t so that q-estimate and q-target error is minimized.

Error is measured:

$$(R_t + \gamma \max_a \hat{q}(S_{t+1}, a; w_t) - \hat{q}(S_t, A_t; w_t))^2$$

$$w_{t+1} = w_t + \alpha_t(R_t + \gamma \max_a \hat{q}(S_{t+1}, a; w_t) - \hat{q}(S_t, A_t; w_t)) \times \nabla_w \hat{q}(S_t, A_t; w_t))$$

target

update this

Recall

① $V(s;w) \xrightarrow{NN}$ change w s.t. $V(s;w) \approx$ target

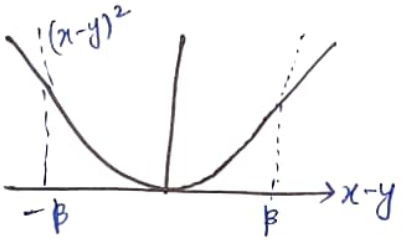
$V(s;w)$ → NN

target → TD → fixed target

(2 NNs)

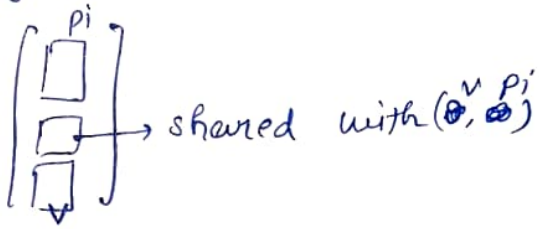
② Experience replay buffer.

Smooth L1 loss :



loss (x, y)

Parameter vector:



computation graph :