Reinforcement Learning

⇒ Its Elements : A Policy
Reward Signal
Value function
model of environment (optional)

1) Policy : 1) Policy defines learning agent's way of behaving at a given time.

2) It is a policy of mapping from perceived states of the env to actions to be taken when in those states.

2) Reward Signal : 1) Goal of RL.

2) on each time step, environment sends to RL agent a single no called the reward.

3) agent's sole objective is to maximise it in long run

3) Value Function : 1) rewards signal tells what is good in immediate sense, value function specifies good in long run.

4) Model of Environment : 1) Something that mimics the behaviour of environment i.e inferences to be made on how the model will behave.

→ model based methods : methods for solving RL problems that use models & planning

→ Model Free Methods : are expicitly trial & error methods.

RL had a history of 3 threads

1) Trial & error
2) optimal Control & its sol^n using value fun^n & dynamic program
3) Temporal difference methods

Side Note:

Bellman Eq^n: optimal Control was solved by using concepts of a dynamical System's State & value function, or "optimal Return function", to define functional eq^n known as Bellman eq^n "

---

ch-2 : MULTI-ARMED BANDITS (Pg 25-36)

2.1) k ARMED Bandit Problem:

1) Every k ~~optimal~~ actions has an expected reward given its selected we call it value of that action
2) Denote the action selected on time step t as A_t
3) Reward as R_t
4) Arbitary action as a
5) Expected Reward as $q_*(a)$

$$\therefore \quad q_*(a) \doteq E[R_t \mid A_t = a]$$

2.2) Action Value Methods: (estimate action value by sample average method)

~~$Q_t(a)$~~ 1) True value of an action is the mean reward when that action is selected.

2) we ~~don~~ don't know the value of each action but we can estimate. estimated value of action 'a' at time step t is $Q_t(a)$

$\therefore Q_t(a) \doteq \dfrac{\text{Sum of rewards when 'a' taken prior to } t}{\text{no. of times a taken prior to } t} = \dfrac{\sum_{i=1}^{t-1} R_i}{t-1}$

$$= \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i = a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i = a}}$$

Here, $\mathbb{1}_{\text{predicate}}$ = denotes random variable ~~that~~

i.e $\begin{cases} 1 & \text{if predicate is true} \\ 0 & \text{if it is not.} \end{cases}$

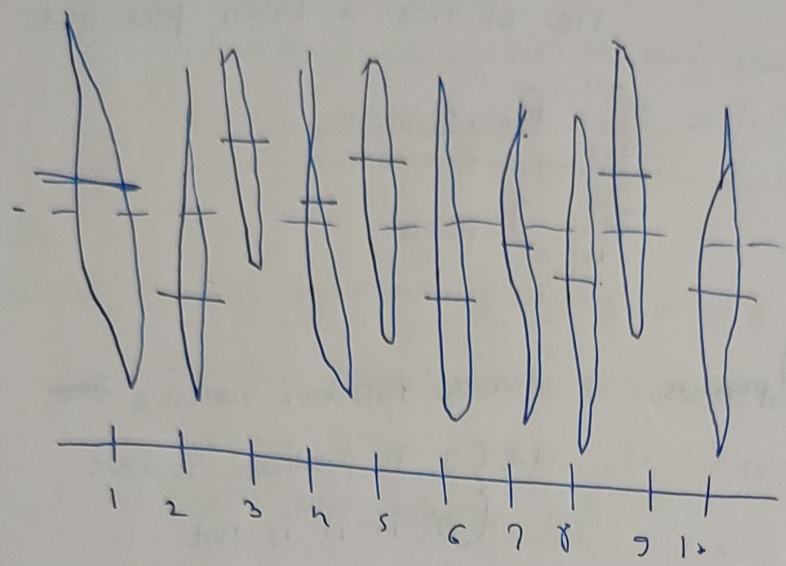
Simplest action Selection Rule to this is:

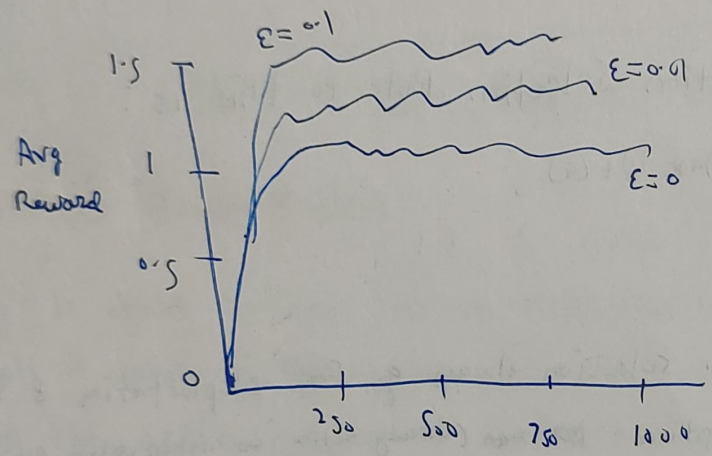
$A_t \doteq \underset{a}{\operatorname{argmax}}\, Q_t(a)$,

→ Greedy action selection always go for exploitation & rarely explore the options. ~~but~~ (Greedy action has highest value estimate)

→ So, Simple ~~behavior~~ alternative is to behave greedily most of the time, but select randomly from all actions with equal probability ε. This is called ε - greedy methods.

2.3 The 10-armed Testbed.





2.4 Incremental Implementation. (Estimating Actⁿ value Incrementally)

$$\boxed{Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i}$$

$$= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}\left(R_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} R_i\right)$$

$$= \frac{1}{n}\left(R_n + (n-1)\,Q_n\right) = \frac{1}{n}\left(R_n + nQ_n - Q_n\right)$$

$$= Q_n + \frac{1}{n}\left[R_n - Q_n\right]$$

$\Rightarrow \quad Q_{n+1} \leftarrow$

$\Rightarrow$ New Estimate $\leftarrow$ Old Estimate + Step Size $[\underbrace{\text{Target} - \text{Old Estimate}}_{\text{error in estimate}}]$

$\Rightarrow$ & step size parameter $= \frac{1}{n} = \alpha = \alpha_t(a)$

& $\alpha : [0, 1]$

Stationary – Distribution of rewards remains same

Non-Stationary – Distribution of rewards changes with time

* ~~How~~ we have eq^n $\rightarrow$ $Q_{n+1} = Q_n + \alpha_n [R_n - Q_n]$

Let Decay it to see what it depends upon.

$$\Rightarrow Q_{n+1} = Q_n + \alpha_n [R_n - Q_n]$$

$$= \underline{Q_n} + \alpha_n R_n - \underline{\alpha_n Q_n}$$

$$= \alpha R_n + (1-\alpha)\boxed{Q_n}$$

$$= \alpha R_n + (1-\alpha)[\alpha R_{n-1} + (1+\alpha) Q_{n-1}]$$

$$= \alpha R_n + (1-\alpha) R_{n-1} + (1-\alpha)^2 Q_{n-1}$$

$$= \alpha R_n + R_{n-1}(1-\alpha) + R_{n-2}(1-\alpha)^2 + \ldots\ldots$$

$$(1-\alpha)^{n-1} \alpha R_1 + (1-\alpha)^n Q_1$$

$$= \underbrace{(1-\alpha)^n Q_1}_{\text{initial value of } Q} + \underbrace{\sum_{i=1}^{n} \alpha (1-\alpha)^{n-i} R_i}_{\text{weighted sum of rewards over time}}$$

Q_1 = initial action value

* What is Trade Off ?

- Exploration Vs Exploitation

Exploration : Improve knowledge for long-term benefits

Exploitation : Exploit ~~Gr~~ knowledge for Short-term benefit

→ can't do both simultaneously, So we choose randomly.

Epsilon-Greedy ~~Set~~ Action Selection.

$$A_t \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\, Q_t(a) & \text{with probability } 1-\varepsilon \\ a \sim \text{Uniform}(\{a_1, \dots a_k\}) & \text{with probability } \varepsilon \end{cases}$$

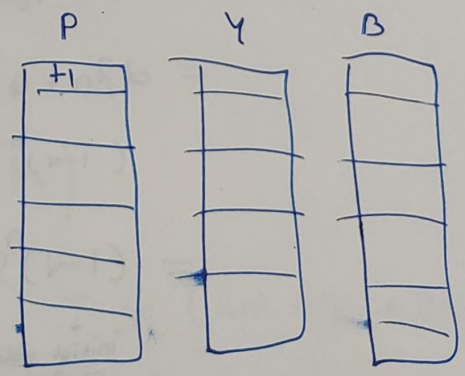
* Optimistic Initial Values

Have high initial value for action selection

A reward of 1 if treatment succeeds otherwise 0.

$Q_{n+1} \leftarrow Q_n + \alpha (R_n - Q_n)$

let $\alpha = 0.5$



$2 + 0.5(1-2) = 1.5$

$Q_1(P) = 2$ $Q_2(Y) = 2$ $Q_3(B) = 2$

$Q_4(P) = 0.75$ $Q_4(Y) = 0.75$ $Q_4(B) = 0.5$

Limitations: optimistic initial values only drive early exploration

They are not well suited for non-stationary problems

we may not know what optimistic initial value should be.

* Upper - Confidence Bound (UCB) Action Selection

↳ uses uncertainity in value estimate to balance exploration & exploitation.

user specified param control amt of exploration

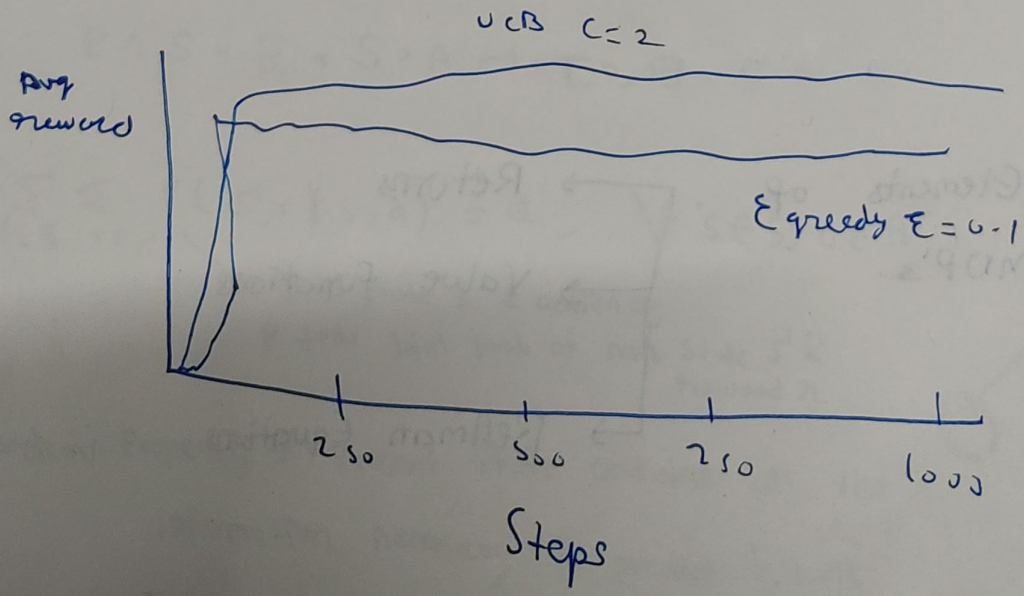
$$A_t \doteq \arg\max \left[Q_t(a) + c\sqrt{\frac{\ln T}{N_t(a)}} \right]$$

$Q_t(a)$ → Action that has highest estimate value (Exploit)

$c\sqrt{\frac{\ln T}{N_t(a)}}$ → UCB term (Exploration)

Here, $c\sqrt{\frac{\ln T}{N_t(a)}} = c\sqrt{\frac{\ln \text{timesteps}}{\text{time action taken}}}$

Performance of optimistic initial value on 10 armed testbed



Summary (Pg 42-43)

→ k armed Bandit

→ Sample avg method

→ Incremental update rule

→ Exploration vs exploitation

Can't do both so
- → Epsilon Greedy Action selection
  - ↳ optimal initial values
- → UCB Actn selection

FINITE MARKOV DECISION PROCESSES (Pg 47-56)

→ MDP's are classical formalization of sequential decision making, where actions influence not just immediate rewards but also subsequent ~~decision making~~ situations or states & through those future rewards.

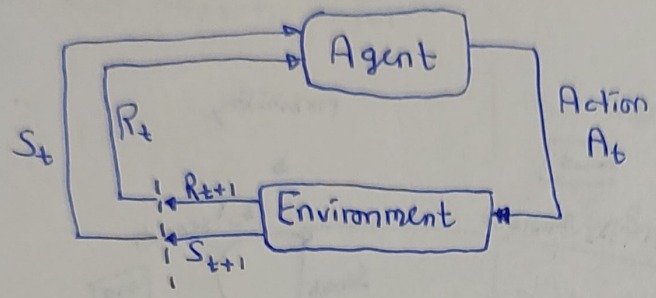
→ In bandit problem we just estimed value of each action but in MDP we will estimate value as well as state s at each time step.

Key Elements of MDP's
- → Returns
- → Value functions
- → Bellman Equations

3.1. The Agent Environment Variable

→ The learner or decision maker is called Agent

→ The thing it interacts with, comprising ~~of~~ everything outside the agent is environment.



[→ The agent & env interact at each of sequence of discrete time steps, $t = 0, 1, 2, 3, \ldots$

→ At each time steps ~~later, in part as consq~~, agent recieves some representat^n of env state, $S_t \in S$, & on that basis, ~~at~~ selects an action, $A_t \in A(S)$.

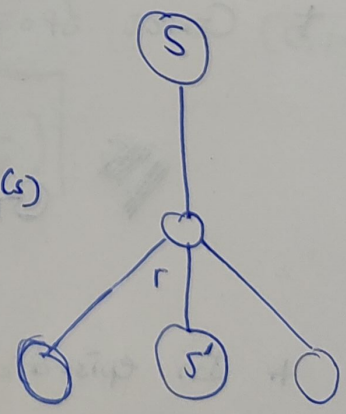
→ One time step later, as a consequence of its action, agent recieve num reward $R_{t+1} \in R$ & finds itself in new state S_{t+1}.

$\Rightarrow S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$

$$P : S \times R \times S \times A \rightarrow [0, 1]$$

$$\sum_{s' \in S} \sum_{r \in R} P(s', r \mid s, a) = 1, \quad \forall s \in S, a \in A(s)$$

given state s & action a, P tells joint prob of next state s' & reward r.

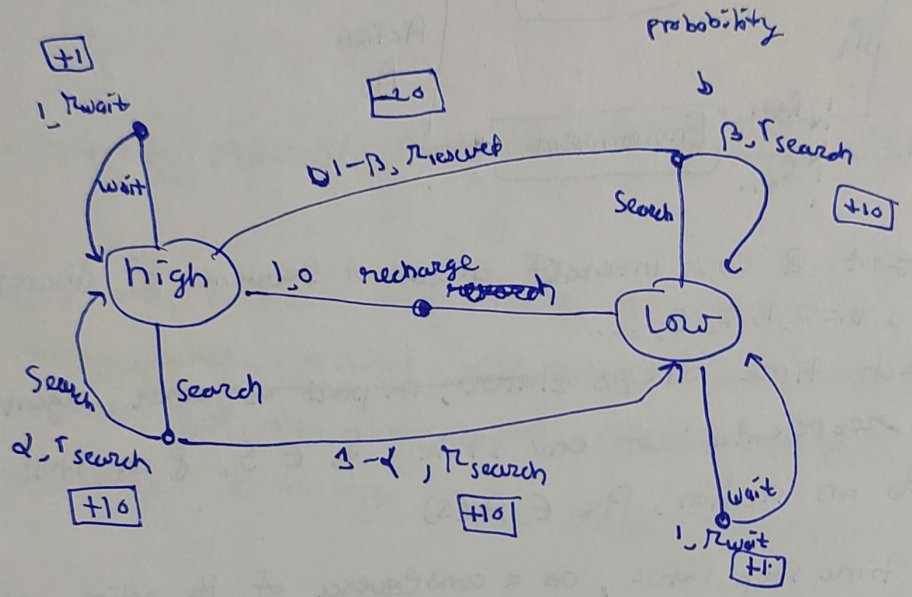


Markov Property → Present State contains all the information necessary to predict future.

A(low) = {search, wait, recharge}

A(high) = {search, wait}

Dynamics of the Recycling Robot



Goal & Episodic Tasks

* Goal of an Agent : Formal definition
  It is to maximize expected return

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \ldots$$

* In episodic tasks, agent-env interact^n breaks up into episodes

Episodic Tasks	Continuing Tasks
• Interaction breaks normally naturally into episodes	• Interaction goes on continually.
• Each episode ends in terminal state	No terminal states.
$G_t \doteq R_{t+1} + R_{t+2} + \dots + R_T$	
	eg: Thermostat
	$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \dots$ But this goes infinitely

* How to make sure G_t finite ?

⇒ Discount the rewards in future by γ where $0 \leq \gamma \leq 1$

$$G_t \doteq R_{t+1} + \gamma(R_{t+2}) + \gamma^2 R_{t+3} + \dots \gamma^{k-1} R_{t+k}$$

$$G_t \doteq \boxed{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}} = R_{max} \sum_{k=0}^{\infty} \gamma^k \longrightarrow (G.P)$$

$$= \boxed{R_{max}\left(\frac{1}{1-\gamma}\right)}$$

* effects

$\gamma = 0 \rightarrow G_t = R_{t+1}$

$\gamma = 1 \rightarrow$ Agent take future rewards into account more strongly.

* Recursive Nature of returns

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

$$= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \dots)$$

$$\boxed{= R_{t+1} + \gamma G_{t+1}}$$

3.5 Policies And Value Functions (Pg 58 - 67)

- Value Functions → Functions of states that estimate how good it is for the agent to be in given state (or how good it is to perform given action in given state)
- The notion of "how good" is defined in terms of future rewards that can be expected, or to be precise in terms of expected return
- These rewards are dependent upon what action it will take
- Value Functions are defined with respect to particular ways of acting, called policies

* • Policy is mapping from states to probabilities of selecting each possible action. (Policy = π)

* The Value function of state S under a policy π, denoted $V_\pi(s)$ is the ~~return~~ expected return when starting in S & following π

$$\boxed{\therefore V_\pi(s) \doteq \mathbb{E}[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right]}$$

expected value of random variable

$\forall s \in S$

Here, V_π = State-value function for policy π.

Now,

Value of taking action "a" in state s under policy "π" is denoted by:

$$q_\pi(s,a) \doteq \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right]$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Here, q_π = action-value function for policy π.

Types of Policies

Deterministic

$\pi(s) = a$

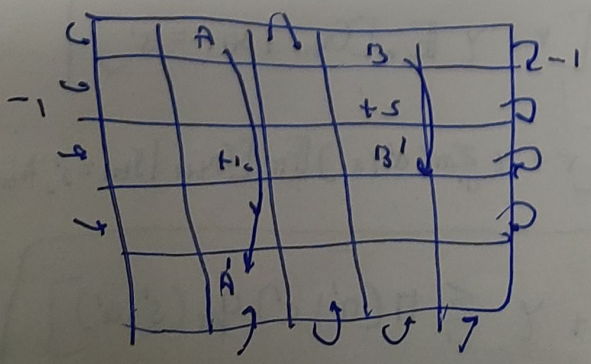
States	Actions
s_0	a_0
s_1	a_1
s_2	a_2

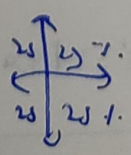
Stochastic

$\sum_{a \in A(s)} \pi(a|s) = 1$

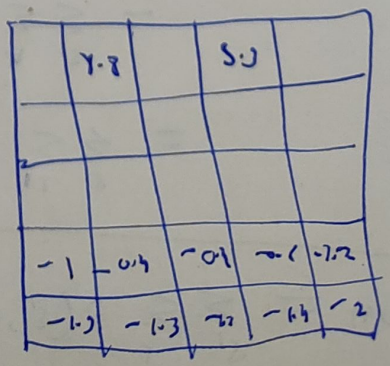
$\because$ ~~$\pi(s,a)$~~

$\pi(a|s) \geq 0$





Policy



→ State value function represent expected return from given state under specific policy

→ Action value funct^n represent expected return from given state after taking specif action, later following a specific policy.

* State-value Bellman equation

$$V_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\right]$$

$$\boxed{V_\pi(s) = \underbrace{\sum_a \pi(a|s)}_{\text{Policy selecting an action}} \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma V_\pi(s')\right]}$$

* Action Value Bellman equation

$$q_\pi(s,a) = \cancel{\sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s', a')\right]}$$

$$\doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

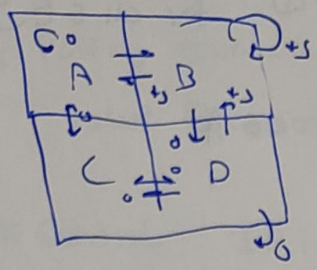
$$= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s']\right]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, \mathbb{E}_\pi[G_{t+1} \mid S_{t+1} = s', A_{t+1} = a']\right]$$

$$\boxed{q_\pi(s,a) \doteq \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q_\pi(s', a')\right]}$$

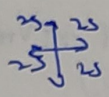
ex → grid world problem:

$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} P(s', r | s, a)\left[r + \gamma V_\pi(s')\right]$$



$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

$$V_\pi(B) = \frac{1}{2}(5 + 0.7V_\pi(B) + \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D)$$

π: 

$$V_\pi(C) = \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D) + \frac{1}{2}(0.7)V_\pi(C)$$

$$V_\pi(D) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(D)$$

Solve eq^{n}s & get $V_\pi(A, B, C, D)$.

Optimal Policy

→ An optimal policy is defined as policy with highest possible value function in all states

→ At least one optimal policies always exists, but there may be more than one

→ exponential no. of possible policies makes searching for optimal policy by brute force intractable

$\therefore$ $\pi_1 \geq \pi_2$ if and only if $V_{\pi_1}(s) \geq V_{\pi_2}(s)$ $\forall s \in S$

* Value function for optimal policy:

$$V_{\pi_*}(s) \doteq \mathbb{E}_{\pi_*}[G_t \mid S_t = s] = \max_{\pi} V_{\pi}(s) \quad \text{for all } s \in S$$

$$q_{\pi_*}(s,a) = \max_{\pi} q_{\pi}(s,a) \quad \text{for all } s \in S \ \& \ a \in A$$

→ ~~optimal~~ Bellman eqn

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

$$V_*(s) = \sum_a \pi_*(a|s) \sum_{s'} \sum_r p(s',r|s,a)[r + \gamma V_*(s')]$$

$$\boxed{V_*(s) = \max_a \sum_{s'} \sum_r p(s',r|s,a)[r + \gamma V_*(s')]}$$

Bellman optimality eqn for V_*

Similarly,

$$\boxed{q_*(s,a) = \sum_{s'} \sum_r p(s',r|s,a)[r + \gamma \max_a q_*(s',a')]}$$

→ The bellman optimality eqn relate value of a state to its possible successors under any optimal policy

4. DYNAMIC PROGRAMMING (Pg 73 - 88)

* Policy Evaluation : PE is the task of determining the value function for specific policy.

* Control : It is the task of finding policy to obtain as much reward as possible
  (ie finding policy which maximizes values function)

we have dynamic programming to solve both PE & control problems

* Policy Evaluation :

$\pi \rightarrow V_\pi$

Recall that

$$V_\pi(s) \doteq \mathbb{E}_\pi [G_t \mid S_t = s] \quad \because G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

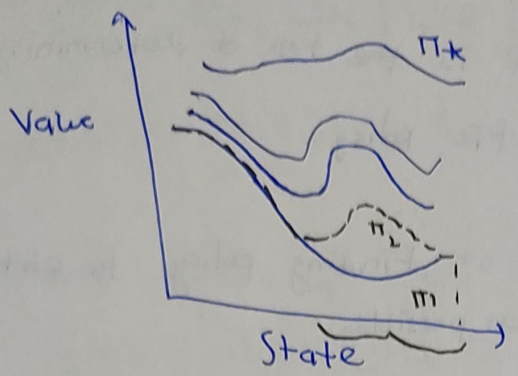
We have seen that how bellman eq^n reduces problem of V_π to system of linear eq^n for each state

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

Recall that : $\pi, P, \gamma \rightarrow$ Linear System Solver $\rightarrow V_\pi$

In practice, $\pi, P, \gamma \rightarrow$ Dynamic Programming $\rightarrow V_\pi$

* Control is the task of policy



π_2 is strictly better than π_1

$V_{\pi_2}(s) > V_{\pi_1}(s)$

How $\quad P, \gamma \longrightarrow \boxed{\text{Dynamic Programming}} \longrightarrow \pi_* \text{ Control}$

$\Rightarrow$ P.E is the task of determining the state-value function V_π, for particular policy π.

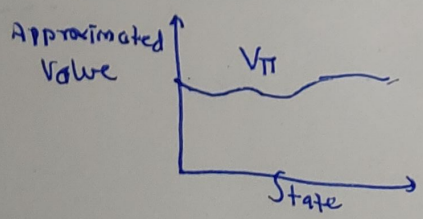
$\Rightarrow$ Control is task ~~for~~ of improving an existing policy.

$\Rightarrow$ D.P techniques can be used to solve both these tasks, if we have dynamics funct^n P.

* Iterative Policy Evaluation

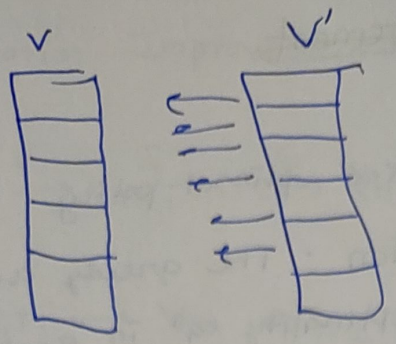
Bellman Eq^n $\Rightarrow$ $V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma V_\pi(s') \right]$

$$V_{k+1}(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma V_k(s') \right]$$



For any V_0

$\lim_{k \to \infty} V_k = V_\pi$



V → V stores current approximation value functions
V' → stores updated values

→ using 2 arrays, we can compute new value from old, one state at time without changing old in process
→ At end of full sweep, we can write all new values in V
→ then we do next iteration.

• Iterative Policy Evaluation, for estimating $V \approx V_\pi$

Input V_π, the policy to be evaluated.

$V \leftarrow \vec{0}$, $V' \leftarrow \vec{0}$

Loop:

$\Delta \leftarrow 0$ → approx value function close to zero

Loop for each $s \in S$:

$$V'(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$$

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

$$V \leftarrow V'$$

until $\Delta < \theta$ (a small positive number) → user specified cond

Output $V \approx V_\pi$

Policy Improvement

Give v_{star}, we can find optimal policy by choosing the greedy action. The greedy action maximizes the Bellman's optimality eqn in each state.

i.e $\pi_*(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} P(s',r|s,a)[r + \gamma v_*(s')]$

The New Policy is a strict improvement over π unless π is already optimal.

$\Rightarrow$ Policy Improvement Thm,

$$q_\pi(s, \pi'(s)) \geq q_\pi(s, \pi(s)) \quad \text{for all } s \in S$$

$$\rightarrow \pi' \geq \pi$$

we cannot find optimal policy so easily as this thm is only an improvement on the original.

Policy Iteration

Policy improvement Thm

$\hookrightarrow \pi'(s) \doteq \underset{a}{\operatorname{argmax}} \sum_{s'} \sum_{r} P(s',r|s,a)[r + \gamma v_\pi(s')]$

π' is strictly better than π

Evaluation Improvement:

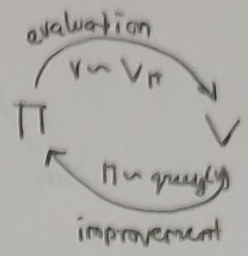
$$\pi_1 \xrightarrow{E} V_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} V_{\pi_2} \xrightarrow{I} \pi_3 \xrightarrow{E} V_{\pi_3} \xrightarrow{I} \pi_4 \xrightarrow{E} V_{\pi_4} \xrightarrow{I} \pi_*$$

$\pi_1 \longrightarrow V_{\pi_1} \rightarrow \pi_2$ (greedy w.r.t. V_{π_1})

$\pi_2 \rightarrow V_{\pi_2}$ (not greedy now) $\rightarrow \pi_3$

$\vdots$

$\pi_* \leftrightarrow V_*$



Pseudocode: (using iterative policy evaluation

1. Initialization

   $V(s) \in R$ and $\pi(s) \in A(s)$ arbitarily for all $s \in S$

2. Evaluation

   Loop:

   $\Delta \leftarrow 0$

   Loop for each $s \in S$:

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))[r + \gamma V(s')]$

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

   Until $\Delta < \theta$ (a small +ve no. determining accuracy of estimation)

3. Policy Improvement

   Policy-stable $\leftarrow$ true

   For each $s \in S$:

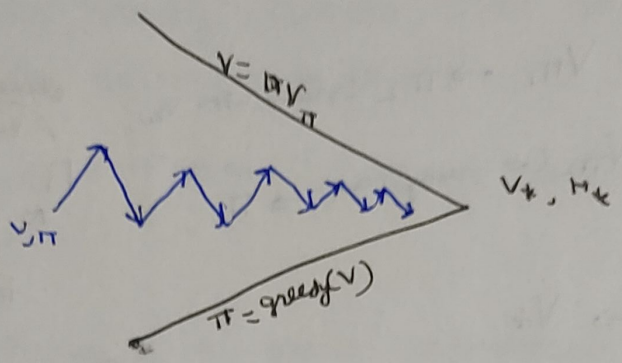
   old action $\leftarrow \pi(s)$

   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v'(s)]$

   If old action $\neq \pi(s)$, then policy-stable $\leftarrow$ false

   If policy-stable, then stop & return $V \simeq v_*$ & $\pi \simeq \pi_*$; else go to 2

Flexibility of Policy Iteration Framework



Pseudocode:

Algo parameter : a small threshold $\theta > 0$ determining accuracy of estimation.

Initialize $V(s)$ for all $s \in S^+$ arbitarily except that $V(terminal) = 0$

Loop :

$\Delta \leftarrow 0$

Loop for each $s \in S$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s',r} P(s',r|s,a)\,[r + \gamma V(s')]$

does not represent specific policy

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

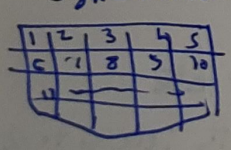
~~until $\Delta < \theta$~~

until $\Delta < \theta$

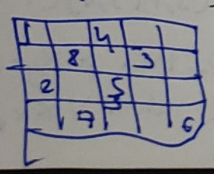
Output a deterministic policy, $\pi \approx \pi_*$, such that

$\pi(s) = \arg\max_a \sum_{s',r} P(s',r|s,a)\,[r + \gamma V(s')]$

Synchronous



Asynchronous



Efficiency of Dynamic Programming

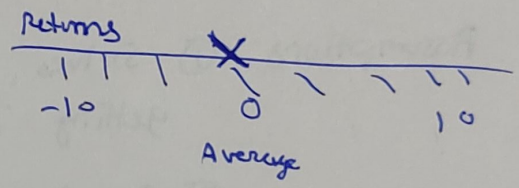
→ Iterative Policy Evaluation is the dynamic programming solution to the prediction or policy Evaluation problem.

Other Alternative:

1) A Sampling ~~Base~~ Alternative for P.E:

$$V_\pi(s) \doteq E_\pi [G_t \mid S_t = s]$$

Monte Carlo Method:



→ we need many returns to average out value for every state

→ In DP we get s' from s (older value estimates to get new)

→ The process of using value estimates of successor states to improve our current value estimate is known as Bootstrapping in DP.

2) Brute Force Search

$\pi(s_1)$	$\pi(s_2)$	$\pi(s_3)$	...	$\pi(s_{|S|})$

$$|A|^{|S|} \left\{ |A| * |A| * |A| * |A| \cdots |A| \right.$$

Research Paper

SOCIALLY - ATTENTIVE POLICY OPTIMISATION IN MULTI - AGENT SELF DRIVING SYSTEM

→ Self attention module

→ Paper focuses on figuring out current interacting vehicle in MASD (Multi Agent Self Driving)

→ Social Aware integration mechanism ~~to into~~

To integrate ~~the~~ objectives of ego AV & its interacting partner up updating their SP's from current observations

Assumptions: ① Solves the Decision making problem by getting correct data from sensors.

② Doesn't use global information for centralized control.

other papers : ① focuses on control physical & subjective observations.

② Doesn't use SP's (Social Preferences) for decision making.

③ Methods used are QMIX, MADDPG (SP's are fixed)

Platform : SMARTS

Methods

Q - LEARNING

$$Q(s,a) = Q(s,a) + \alpha\left[r + \gamma * \max(Q(s',a'))\right]$$

Algorithm:

Start with $Q_0(s,a)$ for all s, a

Get initial state s

For $k = 1, 2, 3, \ldots$ till convergence

Sample action a, get ~~state~~ next state s'

IF s' is terminal:

target $= R(s, a, s')$

Sample new initial state s'

else :

target $= R(s, a, s') + \gamma \max Q_k(s', a')$

$Q_{k+1}(s,a) = (1-\alpha) Q_k(s,a) + \alpha[\text{target}]$

$s \leftarrow s'$

Instead of table format, we use parameterised $Q(s,a)$ i.e. neural network $Q(s,a)$ to estimate or approximate the values.

Policy Gradient & Advantage Estimation

- Why Policy Optimisation?

→ often π can be simpler than Q or V.
eg: robotic grasp.

→ V: doesn't prescribe actions
↳ would need dynamics model

→ Q: need to able to efficiently solve $\arg\max Q_\theta(s,a) \to$ action

$$\boxed{\nabla U(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}; \theta)\, R(\tau^{(i)})} \to -b \to \text{baseline}$$

Gradient tries to:

→ increase prob of paths with +ve R.
→ decrease prob of paths with -ve R.

Decompose path into states & action

↳ $\nabla_\theta \log P(\tau^{(i)}; \theta) = \sum_{t=1}^{H} \nabla_\theta \log \pi_\theta(u_t^i \mid s_t^i)$

[no dynamics model required]

Safe RL

" Safe RL can be defined as process of learning policies that maximise expectation of ~~return~~ return in problems in which it is important to ensure reasonable system performance & ~~res~~respect safety constraints during learning & deployment process. "

- Constraints can be vel, angle, torque for robot OR AV must respect its kinematic constraints.

- Approach (2 ways):
  1) Change optimisation criteria.
  2) Change exploration process.

1) ⇒ a) Worst case Criteria: maximise worst case return

i.e ⇒ $\max_{\pi \in \Pi} \min_{w \in \Omega^{\pi}} E_{\pi,w}(R) = \max \min E\left(\sum_{t=0}^{\infty} \gamma^t R_t\right)$

b) Risk Sensitive ~~Prob~~ Criteria: include risk in long term reward maximisation objective

Risk: Variance of return

c) Constrained Criteria:

$\max_{\pi \in \Pi} E_{\pi}(R)$ subject to $C_i \in C$, $C_i = \{h_i \leq \alpha_i\}$

For eg: policy is updated if it is safe with certain confidence given constraints

2) Change Exploratⁿ process

↳ 2 ways

1. Incorporate external knowledge.
2. Risk Directed exploration

↳ use controllability as an exploration heuristic instead of boltzmann ~~expo~~ Exploration

(if we are avoiding risk in RL how will agent learn)

Constraints

- → Soft Constraint (Satisfy constraint avg over time may violate constraint occasionally during explore)
  - → Constrained Policy optimizat^n
  - → Safe, MARL for AD
- → Hard Constraint
  - → trial without error
  - → Safe Exploration in Continuous Action spaces.