Genetic Algo

Population based Algo.
Extremely slow convergence

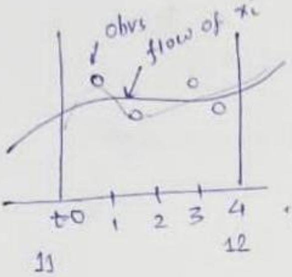
* Derivative-free methods

(a) 4D - Data Assimilation problem:

$$\Psi(x_0) = \|x_0 - x_0^b\|_{B^{-1}} + \sum_{i=1}^{n} \|H_i(x_i) - y_i\|^2_{R^{-1}} \equiv \begin{pmatrix}\text{initial} \\ \text{value} \\ \text{PDE}\end{pmatrix}$$

obj fn

st $x_i = M_{i,i-1}(x_{i-1})$ - forward model



x_0 = initial value

y_i = observation

M = governing eqn. (phy sys).

(phy space)

H = mapping from obs. loc to entire domain. (obs space)

reqd ∵ obvs at all pts mayn't be possible

x_0 = decision variable (to be determined)

⇒ Reverse engg problem to determine x_0 & match based on final values it with obs value.

$$\Psi(x_0) = \|x_0 - x_0^b\|^2_{B^{-1}} + \sum_{i=1}^{n} \|H_i(x_i) - y_i\|^2_{R^{-1}} + \sum_{i=1}^{n} \lambda_i (x_i - M_{i,i-1}(x_{i-1}))$$

Applying SGD, we need $\nabla\Psi(x_0)$

$$\nabla\Psi(x_0) = B^{-1}x_0 + \boxed{\frac{\partial H_i(x_i)}{\partial x_0}} \times \ldots$$

$$\frac{\partial H_i(x_i)}{\partial x_i} \cdot \frac{\partial x_i}{\partial x_{i-1}} \frac{\partial x_{i-1}}{\partial x_{i-2}} \cdots \frac{\partial x_1}{\partial x_0}$$

to be done for $i = 1$ to n

This to be done for each iter.

Hence computational cost is ↑ in cal. the derivatives

Even storing the derivatives is expensive as it will take a lot of storage (2000 × 2000 grid pts)

(b) Derivative can't be cal.

Ex in LASSO regression

$$L + \sum_{i=1}^{n} \lambda_i |\theta_i| \hookrightarrow \text{can't be differentiated}$$

(c) ~~No. of sepo~~ hyperparameter tuning. :

can't be learnt from data, done by grid search method. No fn relates the hyperparameter to loss fn. No direct reln

Derivative free Methods $\rightarrow$ local search - Gen. pattern search

Derivative free Methods $\rightarrow$ Global opti - Evolutionary Algo - Genetic Algo.

global optima due to stochasticity involved

local search 4 all algo are learnt till now

Evol. Algo

1. Generating the population
2. Recombination
3. New population for next iteration } Depends on method

Genetic Algo.

(1). Generating the population (candidate soln)

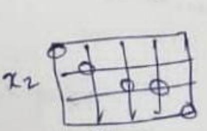
Randomly generate samples from domain

samples = 4 × # variables (gen)

Any random dist method (Normal / Uniform) to generate samples.

They should span the entire set. well.

(Adv: Latin Hypercube sampling).



only diagonal is chosen.

Not good sampling.

(2) Evaluate fitness score.

min $f(x) = f(x_1, \cdots x_4)$. (unconstrained)

Suppose we have 8 samples

Fitness score (FS)		x_1	x_2	x_3	x_4	
fval/Σfval	1				5	→ fval
	2	1	3	2	10	
	.	2	1	9		
	;				0	
		1	0	2		
	8					

$x_1\ x_2\ x_3\ x_4$ → candidate soln. | Chromosome

Suppose $f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2$ → soln $= 0,0,0,0$

Each sample gives a soln

Fitness score may be any other fn of the fval.

lowest score is the soln.

(3) selection

(3) ~~selection~~

Encoding in (2):

min $f(x_1, \ldots x_4)$.

(a) bit encoding — good for small int, higher int higher ~~bits~~ numbers of bits

candidate soln: 1 3 5 2

$\downarrow$ 0001 $\downarrow$ 0011 $\downarrow$ 0101 $\downarrow$ 0010

random cs

cs: | 1011 | 1000 | 0000 | 1110 | we have fitness score for each cs.

Roulette wheel selection /

Tournament "

Roulette wheel: Fitness score in each sectors

$\xrightarrow{\text{Arrange}}$

0.01
0.02
0.03
0.04
0.05
0.15
$\boxed{0.5}$
0.8

say Random no. = 0.6.

Choose the cs where the cumsum is crossed (0.5) the Random no. $\longrightarrow$ repeat 8 times $\downarrow$ pop size is fixed.

$\boxed{\text{Mate in pool}}$ $\leftarrow$ all parents are kept here $\leftarrow$

This is biased towards higher numbers.

Tournament ~~Alt~~ selection: for min for max 70-80%.

Select randomly hyperparameter 40-50% of the pop.^{n} & make smallest ~~FS~~ FS go to mating pool.

$\downarrow$ do this 8 times

Hence largest one doesn't get selected

Repetition in mating pool ~~can~~ is there

If we do 70-80%, then lowest value will be repeated many no. of times $\because$ it will be there in most of the 70-80% selections

(4) Crossover : Generate offspring from mating pool

Choose any two diff parents from mating pool. to generate new popⁿ.
constrained.

★ 1 point / 2 point crossover / N point / Uniform Crossover

1 point :

Total 16 length

1110 + 0.01 takes length randomly → say 7

c 1011 1.00;0 0000 1110 + 0.01

c 0111 100;1 1100 1111 → 0.2

Offs : 1011 1001 1001111 → 0.3 fs → FS may increase be careful

Any half of parents may join

2 point : Choose 2 random length

N " : " N "

Stochasticity involved ↓ goes to global opt.

Uniform Crossover :

Generate random mask

1101	0011	1001	1000

: { If 1 → from 1st parent; 0 → " 2nd " }

→ offs 1011 1000 0100 1111

Helps in generating offspring w/ FS < parents FS

(5) How to know if children doing better than parents ?

Do mutation → avoids trapping in local minima.

Randomly change children string

Offs → 1011 1000 1000 1110

Mask : 1000 0000 0001 0100 (with less prob of 1 : $\frac{2}{10}$). 20%

0001 1000 1001 1010 of entire length is changed

(8)

FS for Offs after crossover & after mutation

(8)

we have 24 CS → 8 Curr pop + 8 crossover + 8 Mutⁿ. } 24 CS

usually "mutation prob" ≤ 20%.

choose lowest 8 FS as New popⁿ

(6) Replacement : New genⁿ for next iti

Since all the f^n eval are independent tasks, hence high no. of f^n eval is not expensive as it can be parallelized.

Repeat 3 – 6 until converged.

Convergence criteria :–

- No. of itr
- No. of itr w/o improvement
- Tolerance on obj.fn (if we know the min f^n value).
  - ↓ or tolerance of obj fn from 1 step to another.

Population should be diversified at each itr.
If New Gen^n has FS b/w 0.01 and 0.02, it is not diversified and no improvement of soln may occur.
So we may choose percent of popu^n from top, middle & bottom FS

No guarantee that crossover will produce offspring which performs better than both parents.
So we have a strategy : Elitist Strategy

Elite Ratio ~ 5% : preserve the best candidate from current pop^n & rest from crossover + mutation

(for new Gen^n)

Crossover Probability : How often a chromosome is being changed.

$\left\{\begin{array}{l}1\ 0\ 0\ 0 \\ 2\ 1\ 0\ 1\end{array}\right.$: crossover prob = 50% as 2 chromosomes are changed
: " " = 25% " 1 " " "

= 2 1 0 0

we define this probability for mating, $P_c = 0.8 = 80\%$ (say)

					0.3	0.55
Dom 1001	1100	1010	1111	:	0	0
0110	0000	1101	1001	:		

Per Parent : (taking each as dominant parent once)
For each bit, generate a random no. } So we get 2 diff children ~~from 2~~ ~~diff~~ parent
If random no. $> P_c$: don't change the bit (take from dom parent)
$< P_c$: ~~don't~~ take bit from other parent (non dominant)

We had fixed in this strategy.
But we see in case of ② 1000 → Less change / P_c
① 2101 → More change / P_c

So we change P_c adaptively.
say we put $P_{c1} = 0.9$, $P_{c2} = 0.1$. $P_{c1} + P_{c2} \neq 1$.

FS for ① : f_1 f_{avg} = Avg FS of curr. Pop^n $f_{min} = \min\{$FS of curr pop$\}$
FS " ② : f_2

$$P_c = \begin{cases} \dfrac{(P_{c1} - P_{c2})(f_{min} - \cancel{f_{avg}})}{f_{min} - f_{avg}} & , f < f_{avg} \\ P_{c_1} = \text{highest prob.} & , \text{otherwise} \end{cases}$$

⇒ any f^n can be defined.

then go for prev. strat with this

P_{c2} = How much we need to change the candidate soln

$P_c = 1$ → No change, $P_c = 0$ → Full change.

If the 2 solns from crossover are close to the parents then the f^n is not changing much in each itr^n & it gets ~~stu~~ stuck in the local minima. So we go for mutation (as less as possible mut^n prob). This will also create diversified pop^n.

In Derivative based method, converge to the soln from initial pt.
" Genetic Algo " , we jump discretely anywhere.
Convergence is hypothesized → micro schema Theorem, "simulated annealing", macro mutation hypothesis

Rate of convergence : empirically
→ based on population being generated.
→ should span entire search space : diversified.

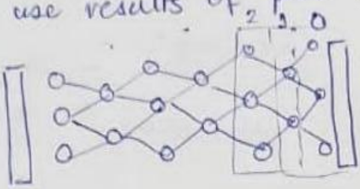
(*) If only 1 child is desired, take only 1 dominant parent (one w/ less FS)
If 2 child are desired, take both as dominant by turn.

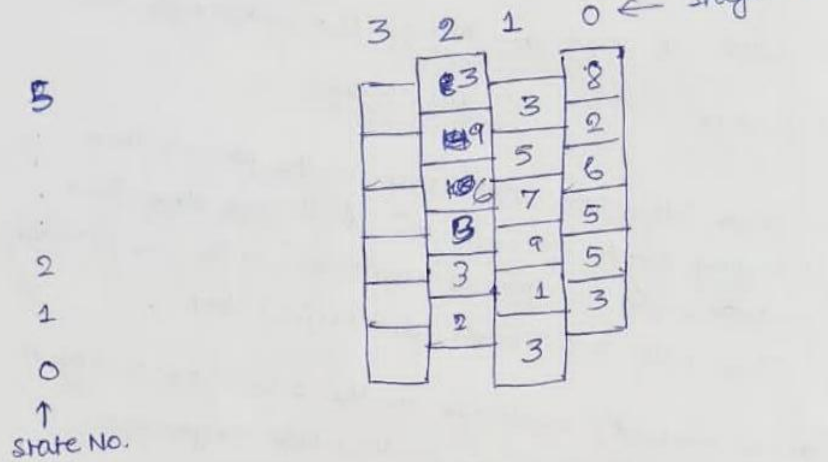
Dynamic Programming

Road Network:

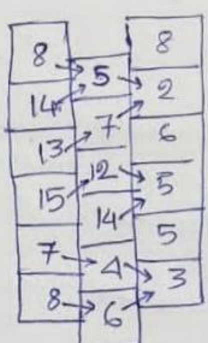
Brute force method – infeasible for large no. of layers & nodes.

So recursive way is used to solve layer wise & use results of prev. layers / subproblems.

Start  Destination   Nodes: waiting time



Stage :



Stage 0: All states have 1 soln.

Stage 1: Solve for minm taking lower sum for stage (1+0)

$v_n(S_n) = \text{Min}\{ t_n(S_n) + v_{n-1}(S_{n-1}) \}$
(cost to go)

n = stage No.

S_n = state No. of stage n

State transition

$$S_{n-1} = \begin{cases} S_n + 1, & \text{choose up } \& \text{ } n \text{ is even} \\ S_n - 1, & \text{" down } \& \text{ } n \text{ is odd} \\ S_n, & \text{O.W} \end{cases}$$

t_n = 1step time duration at stage n } for each state.

$V_n = \Sigma$ time till stage n

<u>Decision Making</u> : ~~maximise~~ min Σ cost OR maximise Σ rewards at each stage. and minimize the risk / uncertainty at each stage.

Markov Prop: Future depends on present, not past.

MDP: <Markov Decision Process>

$<S, A, P, R>$ $\quad$ S = Finite set of states

(x, u, f, g, J) $\quad$ A = " " " actions

$P^a(s, s') = P[S_{t+1} = s' \mid S_t = s, A_t = a]$. — state trans prob.

$S = \{1, 2 \ldots 9\}$ $\quad$ R = Reward fn

$A = \{u, l, d, r\}$ $\quad$ $= R^a(s) = E[R_{t+1} \mid S_t = s, A_t = a]$ (scalar)

$R(s, a, s')$ $\quad$ $R(s,a) = E[R(s,a,s')]$

Deterministic

$P(2|5, u) = 1$

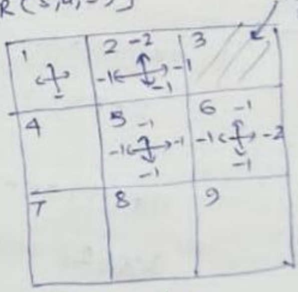
$P(1|5, u) = 0$

Stochastic

$P(2|5, u) = 0.5$

$P(3|5, u) = 0.5$

supposing wind blows towards rt up 0.5



9 S, 4 A

$$R(s,a) = \left.\begin{aligned} R(2,l) &= -1 \\ R(2,d) &= -1 \\ R(2,u) &= -2 \end{aligned}\right\}$$

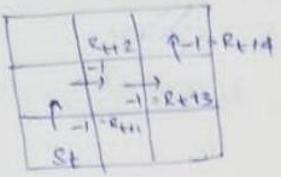
G_t (Return) = Sum of future rewards $R_{t+1} + R_{t+2} + R_{t+3} + \ldots R_T$ — T = Terminal state

$G_t = R_{t+1} + G_{t+1}$

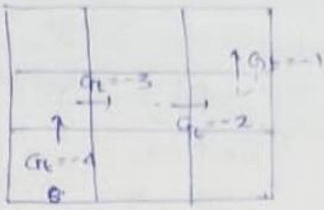
Policy, π: $\pi(a|s) = P[A_t = a \mid S_t = s]$ — fn of state you're in, prob dist of actions at each state $S: \to (A)$ (stochastic)

$\pi(s) = a$

In Policy grid given, all actions are given
: Deterministic policy



Imm rewards



Returns

Value functions :

State-value fn, $v_\pi(s) = E[G_t \mid S_t = s]$

Action " fn, $q_\pi(s,a) = E[G_t \mid S_t = s, A_t = a]$

-4 ↓	-3 ↓	0
-3 →	-2 →	-1 ↑
↑ -4	↑ -3	↑ -2

v_π

$q_\pi(G_t \mid 7, r) = -4$ followed rt & then follows policy

$q_\pi(G_t \mid 7, u) = -4$ followed

$v_\pi(s) = \max_a q_\pi(s,a)$

For deterministic,

$v_\pi(s) = G_t$ at each grid

$q_\pi(7, l) = -6$ Left wall : (stay & get -2)

$q_\pi(7, r) = -4$

Bellman Expectation Eqn

In deterministic env,

$v_\pi(s) = R^a(s) + v_\pi(s')$

$v_\pi(4) = (-1) + (-4)$
$= -5$

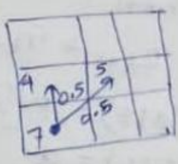
$v_\pi(1) = (-1) + v_\pi(4)$
$v_\pi(2) = (-1) + v_\pi(5)$
$v_\pi(4) = -1 + v_\pi(5)$
} 9 eqn, 9 variables

In stoch env.

$v_\pi(s) = R^a(s) + \sum_{s' \in S} p^a(s,s')\, v_\pi(s')$

$v_\pi(7) = (-1) + [v_\pi(4) \times 0.5 + v_\pi(5) \times 0.5]$

9 eqn, 9 var



$$v_\pi^{k+1}(s) = R^a(s) + \sum_{s' \in S} P^a(s,s')\, v_\pi^k(s') \quad (*)$$

start with $v_\pi^{(0)}(s')$ as say $\{0, 0, \ldots 0\}$

→ Solve w/ iterative scheme

→ Policy Evaluation

→ There always exists a soln which is unique given some mild conditions.

Finding the Optimal Policy:

Control Problem

Given an MDP $\langle S, A, P, R \rangle$, find opt. pol. π^*

π^* s.t.,

$$v_{\pi^*}(s) \geq v_\pi(s) \;\forall\, s \in S \;\forall\, \pi \text{ where } \pi \neq \pi^*$$

In given policies, $v_{\pi_2}(s) \geq v_{\pi_1}(s) \;\forall\, s$

$\pi_1 \rightarrow$ Sub optimal

$\pi_2, \pi_3 \rightarrow$ optimal

* Multiple optimal policies (maybe),
But optimal fn value is same for all

Opti Problem :— {best / optimal policy}

$\max_\pi v_\pi(s)$ / $\left\{ \begin{array}{l} v_{\pi^*}(s) = \max v_\pi(s) \\ v^*(s) = \max v_\pi(s) \end{array} \right\}$

rewards
if cost, then min.

Bellman Optimality Eqn:

$$v^*(s) = \max_\pi E\left[R(s,a) + v^*(s')\right]$$

$$V_n(S_n) = \text{Min}\,\{ r_n(S_n) + v_{n-1}(S_{n-1}) \}$$

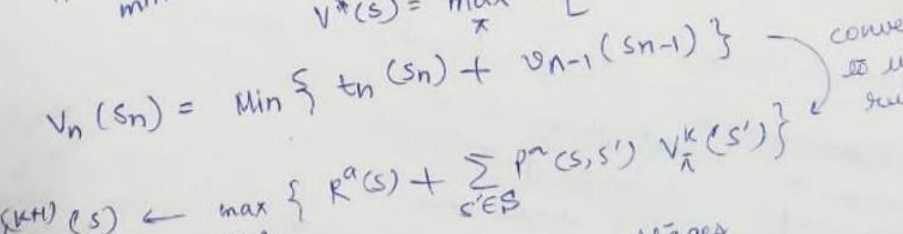
converted to update rule

$$(*)\; v^{(k+1)}(s) \leftarrow \max_{a \in A} \left\{ R^a(s) + \sum_{s' \in S} P^a(s,s')\, V_\pi^k(s') \right\}$$

Value Iteration Algo, $k \rightarrow$ stages

spcl case of GPI in RL

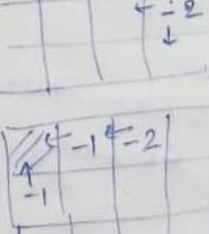
Initialize



V_1

$$V^*(s) = \max\{-1+0, -1+0 \ldots\}$$
$$= -1$$

$R(a,s) = -1$, $v(s') = 0$ ← state I land up in



In each stage, one diag entries get converged.

Based on domain knowledge, we can prune out some nodes which don't get updated further.

DP methods are model based;

Model of world has to be known a priori

$p^a(s,s')$ has to be known a priori

This is offline planning.

What if model is not known?

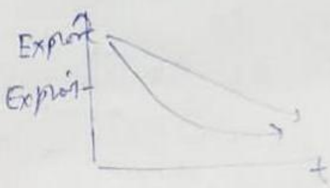
→ Reinforcement Learning : learning to decide through interactions w/ world.

Learning from experience

Regret: not knowing a priori & hence missing out on exploiting

Tradeoff : Explore & Exploit.

initially do this lot (so that we get a good model of the world)



w/o Exploration, we may learn the suboptimal policy.

More " , more better / optimal solution.

Prediction & control Problem : (Model is not known)

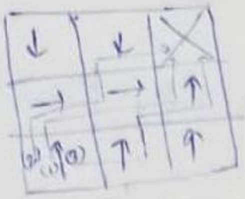
Policy Evaluation : Find value fn given policy

Find optimal policy * Gen policy itr.

Prediction : $q_\pi(s,a) = \underset{\text{sample mean}}{E}\left[G_t \mid S_t = s, A_t = a\right]$

$$v_\pi^{(k+1)}(s) = R^a(s) + \underbrace{\sum p^a(s,s')}_{\text{not known}} v_\pi^k(s)$$

Find: Action value fn by Estimating Expectation



$G^{(1)} = -4$

$G^{(2)} = -4$

$G^{(3)} = 4$

$q_\pi(\text{start}, \uparrow)$

$= \frac{(-4)+(-4)+(-4)}{3}$

$= (-4)$

Take sample mean of return thru episodes.

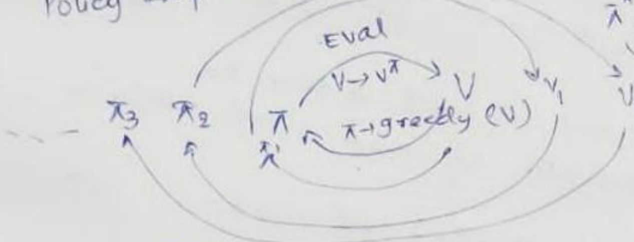
we know value fn of all states we visited

i.e, of $\{2, 3, 4, 5, 6, 7, 8\}$

<u>Control</u>: Gen Pol. Itr (GPI) (start w/ random policy)

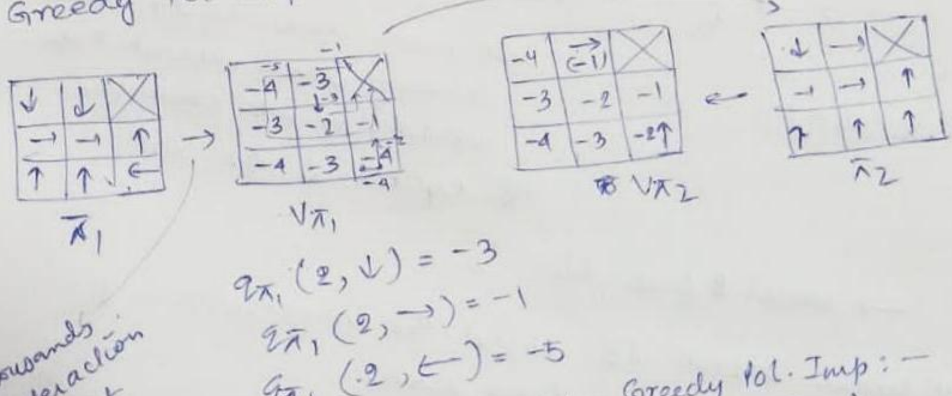
Eval: $v_\pi(s)$ for π

Policy Improvement: Generate improved π' from $v_\pi(s)$



$\pi^* \ldots \pi_3 > \pi_2 > \pi_1$ converged to optimal policy. (guaranteed to converge)

Greedy Pol. Imp: $\pi'(s) = \arg\max_a q_\pi(s,a) \; \forall s$



π_1 $\quad V_{\pi_1}$ $\quad V_{\pi_2}$ $\quad \pi_2$

$q_{\pi_1}(2, \downarrow) = -3$

$q_{\pi_1}(2, \rightarrow) = -1$

$q_{\pi_1}(2, \leftarrow) = -5$

thousands of interaction w/ π_1 to get the expectation of all actions to get V_{π_1}

Theoretically $\to \infty$ times

Practically $\to$ large no. of times

Epsilon Greedy Pol. Imp: —

$\varepsilon \in (0,1)$

$\pi'(s) = \arg\max_a q_\pi(s,a)$ wp $1-\epsilon$

wp ϵ any other actions

say $\pi'(a|a) = \begin{cases} \uparrow, & 1-\epsilon+\epsilon/4 \\ \downarrow, \leftarrow, \rightarrow, & \frac{\epsilon}{4} \end{cases}$

Monte Carlo control

$$\mu_n = \frac{x_1 + \dots + x_n}{n} \Rightarrow \mu_n = \mu_{n-1} + \frac{1}{n}(x_n - \mu_{n-1})$$

$$q_n = \frac{G_1 + \dots + G_n}{n} \Rightarrow \underbrace{q_n}_{\text{Action value fn.}} = q_{n-1} + \frac{1}{n}[\underbrace{G_n}_{\text{return}} - q_{n-1}]$$

$q_\pi(s,a) \leftarrow$ same

$= E[G_t \mid S_t, A_t]$

⇓ for particular (s,a)

Policy Eval.

$\alpha_t = \frac{1}{n}$: for same state, we are updating the q_π

$\epsilon \leftarrow 1/k$, $\pi \leftarrow \epsilon$ - greedy(q) Policy Improve

$N(S_t, A_t)$ = No. of times we have done the interaction w/ same state & action

$\epsilon \rightarrow$ অতএব explore করতে হবে wp ϵ

$\epsilon \rightarrow \frac{1}{k}$ Exploration ↑ k↑ $\frac{1}{k}$↓ ϵ↓

$\epsilon \propto (s,a)$

bcz more knowledge we already have about the world, so we chance of explore ↓ as k↑

→ Model free RL

* Model based RL: build the model within & then plan.

upper time limit set

only $\bar{\pi}_i$'s are shown, no v_i's shown

$q \rightarrow$ DNN

Monte Carlo → Q learning → DQL Deep Q learning

apply NN

Two min paper.

movmt of the bar in the policy