Reinforcement Learning

Reward

feedback may be delayed

time matters - ?

Markov Decision Process

bunch of states and actions

State transitions

Probabalistic → 1st order MDP only current state determins

→ 2nd order

= current and prev (P)

Stochastic shortest path problem.

(uncertainty)

Have to make

a policy lookup table ~~x~~ when this state do this.

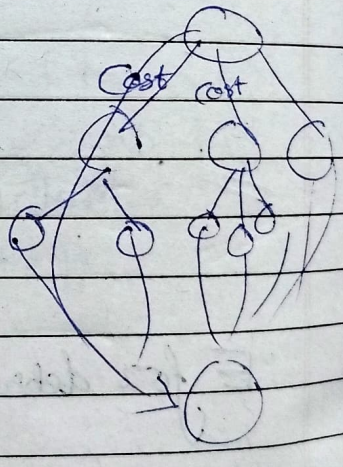
Now goal it to minimis cost

for a policy and

get to know Expected

avg

cost



How? Bellman Eq

determine how good a choice is.

Why not use

A* / Dyjkstra? when deterministic graph

No probability

Page No.
Date: / /

RL Algos

Value based — (determining value function that gets reward) based on policy for max rewards

Policy based.

Value function —determine→ policy —that→ Rewards.

↳ ① State VF $V(s)$

$S \rightarrow V(s) \rightarrow R$ — tells how good to be in S

② State-Action VF $Q(s,a)$ — (optimal decision by agent in environment)

$s, a \rightarrow Q(s,a) \rightarrow R$ (q value) — how good to be in s and to take action a

BE answers what longterm reward to expect if i take best action in a state } Value of state

BE for deterministic env — No probability

$$V(s) = \max_a \left(R(s,a) + \gamma V(s') \right)$$

$V(s)$: Value of state; $\max_a$: pick action that maxs; $R(s,a)$: Reward; γ: discount factor; $V(s')$: when you take a, goto next state

$S \xrightarrow{a} S'$ recursive

Page No:
Date: / /

R.L. Algos

Value based → Policy based.

Value based: you determine value function that give Reward } basis for policy } to max rewards

Value function —determine→ policy —max→ Rewards.

↳ (1) State VF $V(s)$: $s \rightarrow V(s) \rightarrow R$ (total reward agent gets to be in s)

(2) State-Action VF $Q(s,a)$ → R (q value)
- s, a
- action: latent depiction by agent in environment
- R: how good is it in s and to take action a

BE answers: what longterm reward to expect if i take best action in a state } Value of state

BE for deterministic env^n — No probability

$$V(s) = \max_a \left(R(s,a) + \gamma V(s') \right)$$

- $V(s)$: Value of state
- $\max_a$: pick action that maxes
- $R(s,a)$: Reward.
- γ: discount factor
- $V(s')$: when you take a & go to next state

$(s \xrightarrow{a} s')$ accuracice

Discount factor?

How to know which action at each S?

RL

Agent gets state S_t at instant t from environment on basis of S_t and policy applies (a_t) to environment.

Environment responds by S_{t+1} and also gives time after action gets R_{t+1}

For a_{t+1}

Current policy that maps S to A

$\pi(A_t | S_t)$

Return

$G_t = R_{t+1} + R_{t+2} \cdots + R_T$

when $T = \infty$

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

γ is discount

Want to max (G)

Now as R, A, S all is stochastic

We do not have defined (G_t)

Can find expectation value of G_t.

Get sequence of control actions that maximise $E(G_t)$

Page No.:
Date: / /

Dynamics of MDP (conditional)

is the probability of that reaching state S' and getting (r) at t if already in S at $t-1$ and a done.

$$p(s', r | s, a)$$

~~= Prob~~

So State transition prob.

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

reaching S' given in S in $t-1$ and action a

(*)

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

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

Policy $\pi(a|s)$ Prob of selecting a when at s.

$\pi(\,)$ is a function

a_1 is or a_2 ...

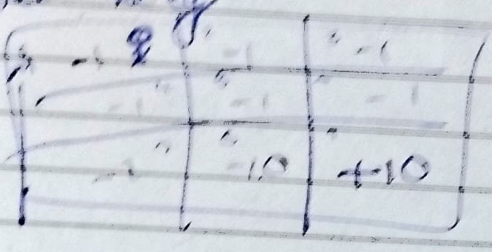
gives us probability

goal is to get a good $\pi(\,)$

Temporal Difference Learning.

is a way value based RL algo use to Iteratively Learn SVF/SAVF
(to generate Q-values)

Let take eg.



Let init Value function of all states with 0

S_1	0
S_2	0
S_3	0
⋮	
S_n	

here $V(S)$ is to be 0

Let agent take action a to S_2 then

$V(S_1) = R(S_2) + V(S_2)$ ← Bellman (deterministic)
observed

$= -1 + 0$

$= -1$ but $V(S_1)$ expected $= 0$

Then Temporal Difference Error $= -1$
(over two diff time steps)

$= V(S_1)_{observed} - V(S_1)_{expected}$

Obs − Exp

Now new better learned.

$V(S_1)_{new} = V(S_1) + \alpha\,(TDE)$
↳ Learning Rate

after 1 time step.

Repeat

After learned, agent will take path that will result in Right at $\max \Sigma Q$ value. get the Target Policy

Page No.:
Date: / /

Behaviour Policy is what is used to learn.

Q Learning

OFF policy

SAVF

State → Q() → Q value
action

-1	-1	-1
-1	-1	-1
-1	-10	+10

Our Q values is a table ∴ find such 'a' as optimum policy will

Random Values from now

Q	Left	Right	down	up
S_1	-0.5	0.7	1.3	
S_2	0.3	0.75	2	-0.5
S_3				

Let S_1 then randomly S_2

a = Right

BE

$$Q(S_1, \text{Right})_{\text{observed}} = R(S_2) + \gamma \max_a Q(S_2, a)$$

max Q then (a)

$= -1 + 0.1 * 2 = 0.8$

i.e. down

Check for which a in state S_2 we have max $Q(S_2, a)$

$TDE = Q(S_1, \text{Right})_{obs} - Q(S_1, \text{Right})_{Expec}$

$= -0.8 - 0.7 = -1.5$

Updated Value $Q(S_1, \text{Right})_{new} = Q(S_1, R) + \alpha\, TDE$

Page No.:
Date: / /

Chk Code

Q. Learning project

For shortest path. go

We use negative rewards even in the normal traversal blocks because

if +ve, then to maximise value, our agent would keep on walking on blocks.

So lil -ve for path, and huge -ve for forbidden area.

(3) Deep Q learning NN + Q learning

(4) CNN

ANN for input data with spacial structure

means img of a cat → Img, Vid, Text

is

Cat because all the pixels are arranged in that specific way

CNN captures spacial relationships

Working

(1) Generating set of feature maps data for CNN

(2) Simplify FM by pooling

(3) Flatten the pFM

7-34!

Page No.:
Date : / /

(4) Connect fpFM to NN

(★) FM ... 1st step ⇒ Apply (filters) / feature detector (kernels)

determine if a particular part of image contains a specific feature

Result is a FM

Collection of FM ≡ Convolutional layer.

(★) Eg Apply 3×3 filter to every 3×3 section of img

Multiply each element of filter to corresponding cell in image. Add (divide by total cells in filter) get degree of overlap

→ Why No ANN for img.?

Too many inp, weights (Computation)

Treats local pix same as far apart pix

Sensitive to location of obj

→ Max Pooling ≡ reduce size of each feature map

Take the max value from the section (current) (2×2 for eg)

→ Window jumps by its size rather 1px

→ Flattening = 2D pFM to 1D vector,

all pFMs in one vector

Feature layer → Input layer to NN

nn.Module

Page No.:
Date: / /

Class ConvNet (nn.Module)

(init) Conv(1)
pool
Conv2
11 3
fc1
fc1

(forward)

Exploration vs Exploitation tradeoff.

ε-greedy strategy / Implementation code

Tip to make it better: Start from 1 then decay to 0.01

DqL new best friend

Friendship with value iteration ended.

Image inp from eg game

few consec frame as one inp fed as a state to NN get q values of corresponding to actions in that state.

Experience Replay (Training) & Replay Memory

Experience of agent at (t)

$e_t = (s_t, a_t, r_{t+1}, s_{t+1})$

Replay memory set contains these e_t from ∞ episodes

Usually stores last N e_t.

Why Exp Replay?

We randomly sample from the replay memory set to train network

Called Exp Replay

Why? not normally using sequential experience

To break correlation between consec samples

Otherwise, samples highly correlate → inefficient learning

db weights initialization how?

Page No.:
Date: / /

→ init replay mem capacity N
→ init weights
→ for episode
→ start state
→ action
→ execute
→ reward & next state
→ Store experience

Training Startsss !

→ Take any rand experience and feed forward
→ Calc Loss (Net fn q and Bellman fn q)
optimal

Again Loss
⇒ $q_*(s,a) - q(s,a)$
⇒ $E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a') \right]$

isse liye feed s' to NN and get max q from 2nd pass required for optimal q value.

$= E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right]$

⇒ Then gradial descentss

Page No.:
Date: / /

A new contender appears!!

(Moving target)

Why? Issues with older approach

What? THE 2nd PASS!

PROBLEM

As for generating Optimal q with bellman we run the NN again for S_{t+1} but it uses the same nudged weights that in the first place he were trying improve

When update happen? target q value (2nd pass value) the update :o

q value → approach Q^* but Q^* → also moves away.

Solution? ~~2nd pass~~ (not on same NN)

Obtain (target q) from Separate Target NN

(it's a clone) and its weight only update after a custom timestep

2nd pass now in Target NN!

New hyper parameter

Page No.:
Date: / /

Now, coming to 2048 !!

Keep biggest in corner / (lower) More rewards higher bottom full

2048 is stochastic! so more intensive

Approach ① Monte Carlo Tree Search.

② heuristic driven search algo.

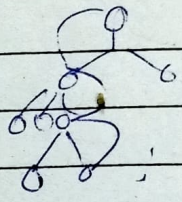
{Search by itself}

Tree search + q learning

→ explore all ($\epsilon = 1$)

→ get best (suboptimal)

→ keep looking, updates q values



Look similar to Minimax not used as not efficient

→ MCTS only searches couple layer deep

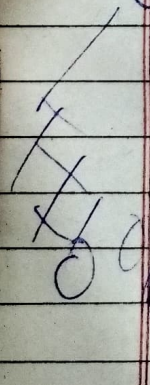
Steps ① Tree traversal.

② Node expansion

③ Rollout

④ Backprop

Root node



Leaf nodes and positions