21-12-2021 Tuesday

Week #17 Lecture # 23

Deep Reinforcement Learning

we won't have y, we only have x & z.

first we have

$$y = f(x)$$

used x to predict y.

but now we use

x, z to predict y

z is Reward

$$y = f(x)$$

$$z \rightarrow \text{Reward}$$

x = State

on action, if it is good, we give +ve reward z.

or on action, if it is not good, we give -ve reward z.

Unsupervised learning or unsupervised learning.

For example,

in bricks out game

using Previous Deep Neural Net techniques we have to play a lot of games & collect a large dataset & give it as an input & learn steps each move right, left or stay.

This is hectic bez it is not possible to play all games & gather dataset.

Now:

State:

basically input for image, full screen image

in above scenario, screen, or ball position, direction, bricks available, bricks position

Action:

paddle movement right, left, or stay

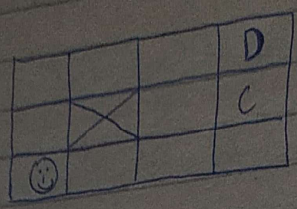
Environment:

This entire game

Reward:

usually score of game & we try to maximize it.

Example:



D = Destination
C = Concert
X = not reachable

Reward ∈ could be +ve or -ve. All together 12 states.

Agent has to reach Destination

1 episode: Agent starts from its initial position to a terminal state

D has reward 100
C has reward -100

Rewards are also given when moving from one state to another state called Immediate Reward

we first Compute utility value of each state

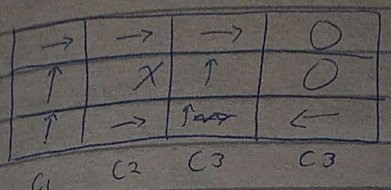
which is reward accumulated till terminal state.
So utilities values for each state is

88 →	92 →	96 →	100
84 ↑	X	92	-100
80 ↑	84	88	84

every step reward was -4

Now agent could go up or right

Policy matrix



	C1	C2	C3	C3
r3	→	→	→	0
r2	↑	X	↑	0
r1	↑	→	↑	←

There could be multiple terminal states.

This was a Deterministic environment meaning our action taken gives Deterministic state that we intended to go.

In an undeterministic environment, we have 2 Reinforcement learning.

① Model Based Learning.

Transition Model = a matrix

Starting cell	Destination	Probability
1,1	→ 2,1	0.8
1,1	→ 1,2	0.1
1,1	→ 1,1	0.1

This matrix gives probabilities that to reach a Destination from current cell to adjacent cells.

Compute Scores.

For Deterministic environment

$S_t = \gamma + U(S_{t+1})$

γ = reward immediate reward

S_{t+1} next State Reward

2 loops one for all the States

One loop to infinity until matrix values don't change

Value Iteration Algorithm:

0	0	0	100
0	0	0	-100
0	0	0	0

↓

-4	-4	96	100
-4	-4	-104	-100
-4	-4	-4	-104

↓

-4	92	96	100
-4	-4	92	-100
-4	-4	-4	-104

but in unDeterministic Environment Value Iteration becomes.

92	a	100	
-4	b	-100	
-4	-4	-104	

100

a $S_{t} = 0.8 * S_{next} + 0.1 * a + 0.1 * b + r$

r: Immediate Reward

Bellman Equation.

Utility or Value $= V_t = R_t + \gamma \sum_{s'} T(s, a, s') * V(s')$

gamma called Discount factor

γ values is b/w 0, 1

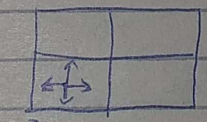
one iteration is Bellman update

Q learning:

Enhanced version of Value iteration,

Before we were computing one utility value for one state.

Now, for each action, we check utility value



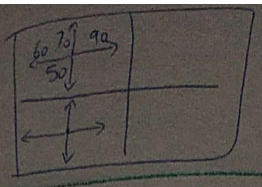
max value in each direction it is called Q value

In Q learning,

we don't know in which state we can go to from this current state

No transition Model, Model Free Reinforcement learning

States	Actions: Up	Down	Left	Right
S_1				
S_2				
S_3				
S_4				
S_5				
S_6				



Now, we have

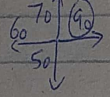
$$Q(S,a) = Q(S,a) + \alpha\left[r + \max(Q(S',a)) - Q(S,a)\right]$$

α = learning rate, start from a larger value & then reduce it as learning has been conducted.

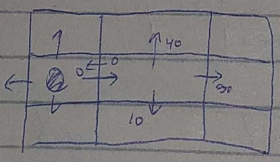
S = current state

a = action

$\max(Q(S',a))$ → 90 will be returned so Q value of next state is taken & we add immediate reward & we substract it from current state Q value



Now, example.



$$Q(S_{21}, \text{right}) = 0 + \alpha[-4 + \max(40, 50, 10, 0) - 0]$$

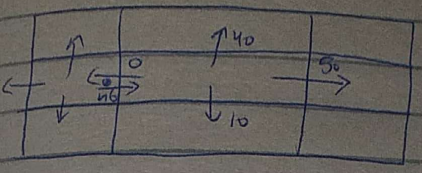
$$= 0 + \alpha[-4 + 50 - 0]$$

$$= 0 + \alpha[46 - 0]$$

$$= 0 + \alpha(46)$$

$$Q(S_{21}, \text{right}) = \alpha * 46$$

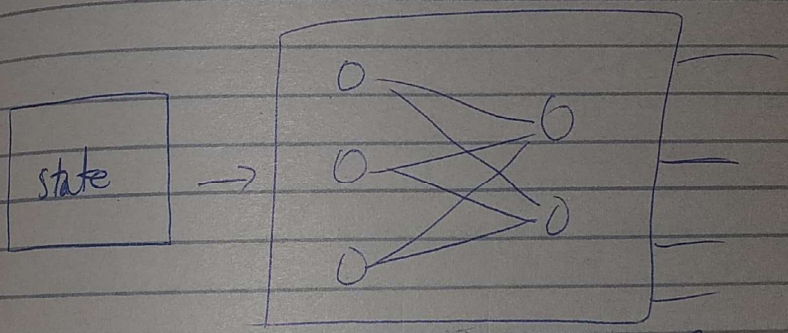
Now we have



The Problem:-

As No. of states or No. of Actions increase, this algorithm fails. So we go back to Neural Net.

Solution:



Act as Q table

Predicts Every Action Q Value

we learn our Q Table

Then whatever is max for one action we assign that Q value to that Action.

This Eq. Equation is used for loss.