What do you call a man with a rubber toe?  *A Roberto*

The credit assignment problem: basically cartpole problem, if we have a sum of rewardsa as the only guidance.

We want to reward and tell the program after the pole fell which one of its steps were good and which were to blame, basically the NN will give a probability of moving right, therefore moving left is 1-p(moving right) at each interval of training, however, when we finish an episode maning that the pole has fallen how do we reward the correct probabilities that it was having?

Such as a dog, if you reward it hours after it has done a good job will it know which one of the behaviors it did was good? It won’t mabe it’ll think it was the last behavior or action, but this is incorrect, beacue the last action cannot be a summerizaton of all actons taken during.

Which of the 100 actions were the *good ones*  which ones were bad

Logsigmoid squesh the output between [0,1]

Credit assignment for each step, we make a action, and there will be many times we don’t get a reward, the ONE teim we get a non-zero reward is if the round is won, now forexample if we hav 1,000 moves and the moves and get a reward of +1, how the fuck do we know if 90 and 500 were the moves we need that made the win possible? And all the others were bad?, this is called credit assignment problem, maybe it had to do with frame 10 and frame 70,

Policy Gradients

Q-Learning

100 games each game is made of 200 frames(theorthetically)

to do the math, that means 100 games and each game produces 200 frames, which in turn means each frame returns an action 100X200 = 20,000 actions that this traning session ha produced (training protocol) 20,000 for going up or down

now we know we can get a gradient for the actions ,

suppoused we won 12 games and lost 88 games,

the number of actions that won were 12 \* 200(frames) each game = 2400

the num of actions that lost were 88 \* (200) = 17600 actions(decisions)

reward are used to tell which state-action apirs are good

*discount reward:* to give later rewards less weight, kinda of like giving your reward values a depreciation amount, “the amount that you get in the future is not as important as the amount you’re getting now”, 100$ now is not the same or could not be the sam as 100$ tomorrow in the market

each time the ball goes past a player then we either get 1 or -1 of 0 rewards

0 means nothing happended(continue playing)

if we get -1, we need to count this into our session

if we get +1, we need to count this int our session

so between the zeros (during game, have a running session)

play 10 games,

record the reward states as current

enqueuer all the reqards and their network associated,

now after 10 games for each of the 10 networks,

lets find the cost function(or the fitness function)

sort the networks by reward and get the top 5% performers(5 networks)

makesure they are not breeding themselves (if par1 != par2) create then fill remaining spots

kill off 7%(7)

refill( with parameters such as, hidden layers, weights, bias)

mutate the remaining older networks, changing randomizing their weights or bias, or layer amount, introduces variability or competition from outside the population, which will help seek for optimizsing the parameters outside (global minia)