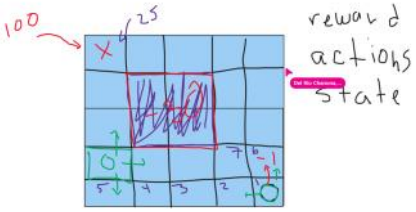
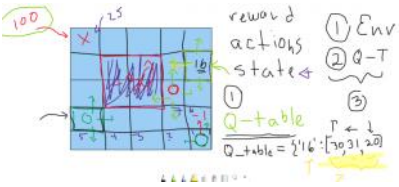
RL summer school problem 1

2025年6月23日 11:47

Problem Description:

Robot in bottom right corner, goal in top left corner, hole in the middle
Reward:
Every step -1 reward, hole area has -30 reward
If robot gets to the goal, gets 100 reward
Actions: up, right, down, left
States: every square assigned a number





First task: program the little environment so that the robot can do different actions in different states(diff states has access to diff actions); every state is assigned a value

A robot keeps track of the Q-table, which keeps track of the value of every state, and every action
The robot does not know all the correct values of states at the beginning

You would want a dictionary called a Q-table, the key will be the number of the state(1-25), the list has three values, up, left, down

When the robot gets to state 16, it will look up the list, it will choose the one to maximise the reward(so it will go left)

So second thing to programme is the Q-table
The third thing: how your agent will learn the list values over time?

Fourth thing: you would want a graph of iteration against time

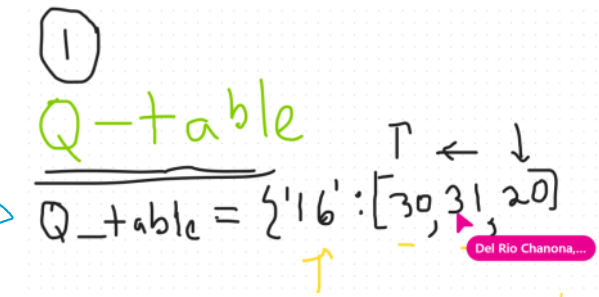
You don't care how you got there, but it cares about how much reward it will incur

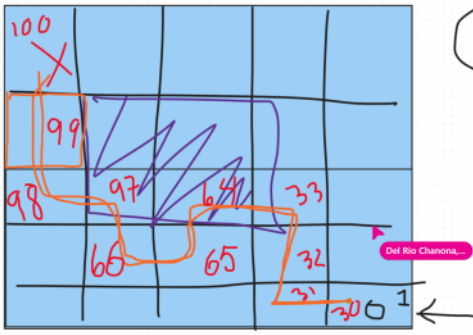
If the trip lasts a certain time/value, give it a negative reward and terminate it

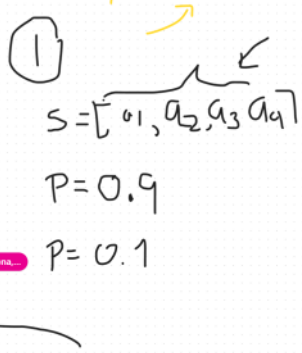
The agent has a map, but it doesn't know how much each square is worth

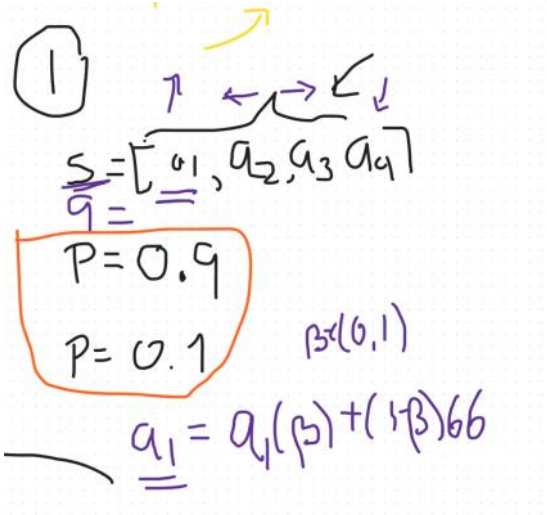
Always start at state 1

You want an agent that randomly chooses an action(initially very random
Once you have a trajectory, you back calculate the rewards of each action along the trajectory









Initially try 1000 paths

Q:What is the parameter beta in the update formula?
Beta--the learning rate in machine learning
Exact value--1-10^5
The higher beta, the slower you would learn
0.9 is fine for now

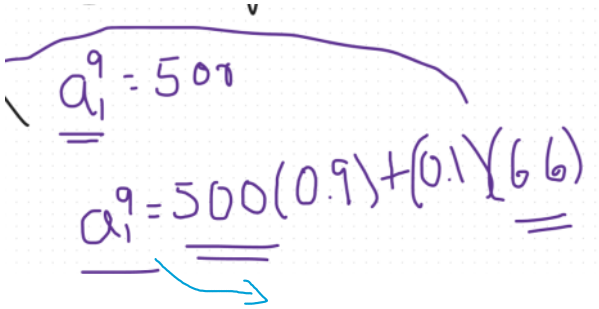
90%of time you choose the best action
10%of time you do a random action

You might also want to start with a high probability of random action, then gradually decrease the probability of doing a random action

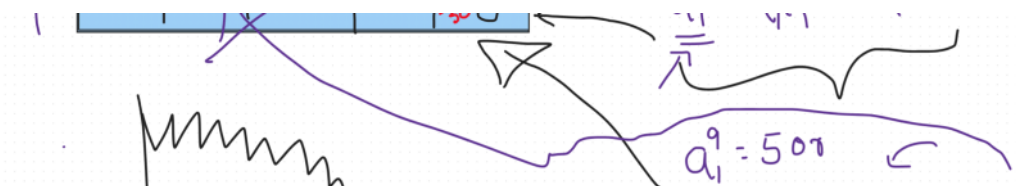
You are back-calculating the rewards because you care about the next action-the future, not the past

Q: How to distribute the reward values in the first run?
Uniform distribution or random normal distribution for initial reward values for next actions(a1 etc)
If the initial generated values are small, the agent would not explore
If you start with larger values, the agent will explore more

When the space is small, you can try high values.



Updated value of a1



actions(a1 etc)
If the initial generated values are small, the agent would not explore
If you start with larger values, the agent will explore more

When the space is small, you can try high values, so the agent explores everywhere in the small space
Vice versa for a larger space

You can experiment with larger environments as well

Harder challenge you can add:
90% probability it will land precisely where it goes next
10% probability the robot lands to the tile next to the one it wants to go