Artificial Intelligence Lab

Even Semester 2022

Advertisement Presentation Using AI

(Reinforcement Learning)

Team Members

Manas 9919103037 (F2)
Tushar 9919103083 (F3)
Tathagat 9919103084 (F3)

Submitted To

Dr. Raju Pal

Assistant Professor (Senior Grade)

Problem Statement

Advertisements are a crucial part of PR teams for any organization that seeks to establish better market control for their product. Advertising on large scales however, is expensive and there goes a lot of research in how and what to advertise that targets the audience better.

Therefore we aim to use an AI solution that uses Reinforcement Learning and uses Multi Arm Bandit problem to solve this problem.

The adds will be shown accordingly to user clicks and the most profitable add thus will be shown subsequently.

Motivation

Motivation of the project is to use reinforcement learning techniques and apply them in real world scenario to save time and cost of the organization.

Tools and Technology Used

Python Libraries: Numpy, Pandas, Matplotlib, Gym Environment
IDEs: Spyder, Jupyter Notebook

Solutions Used

Multi Arm Bandit Problem:

The multi-armed bandit problem is a classic reinforcement learning example where we are given a slot machine with *n* arms (bandits) with each arm having its own rigged probability distribution of success. Pulling any one of the arms gives you a stochastic reward of either R=+1 for success, or R=0 for failure. Our objective is to pull the arms one-by-one in sequence such that we maximize our total reward collected in the long run.

Greedy Approach:

Every time we plug into a socket we get a reward, in the form of an amount of charge, and every reward we get lets us calculate a more accurate estimate of a socket's true output. If we then just choose the socket with the highest estimate hopefully this will be the best available socket.

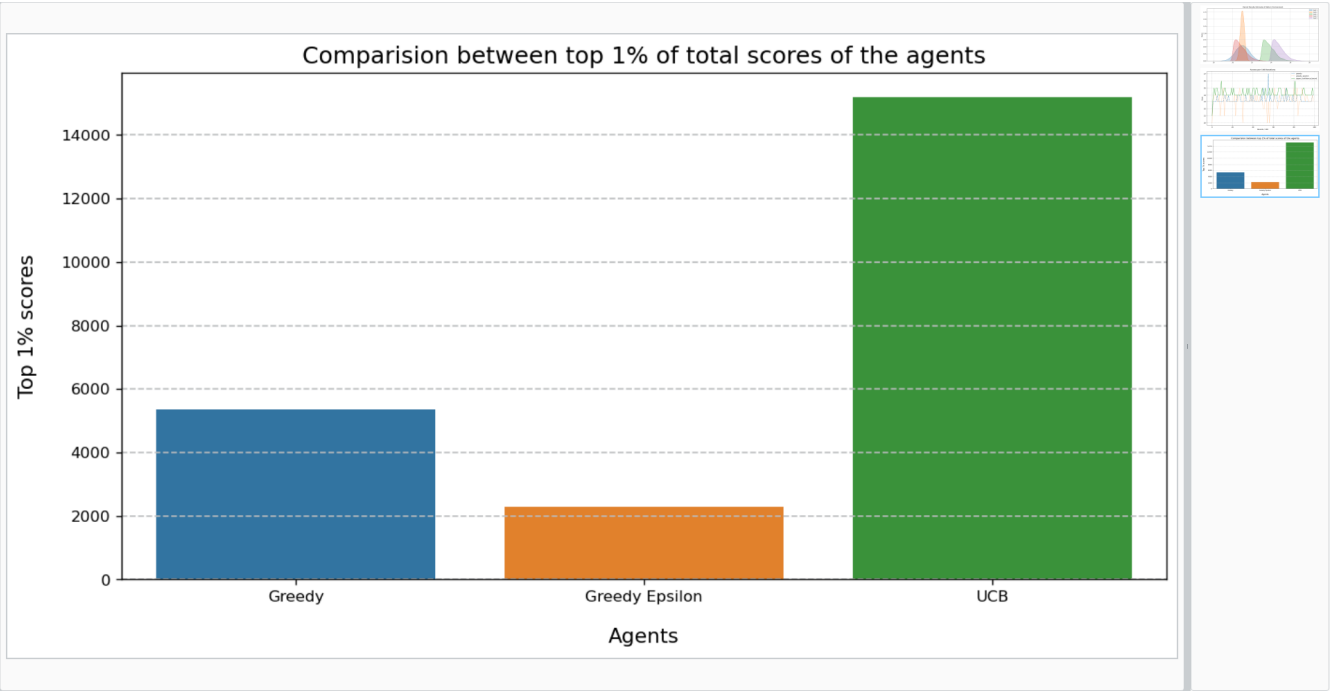
When selecting the action with the highest value, the action chosen at time step '*t*', can be expressed by the formula:

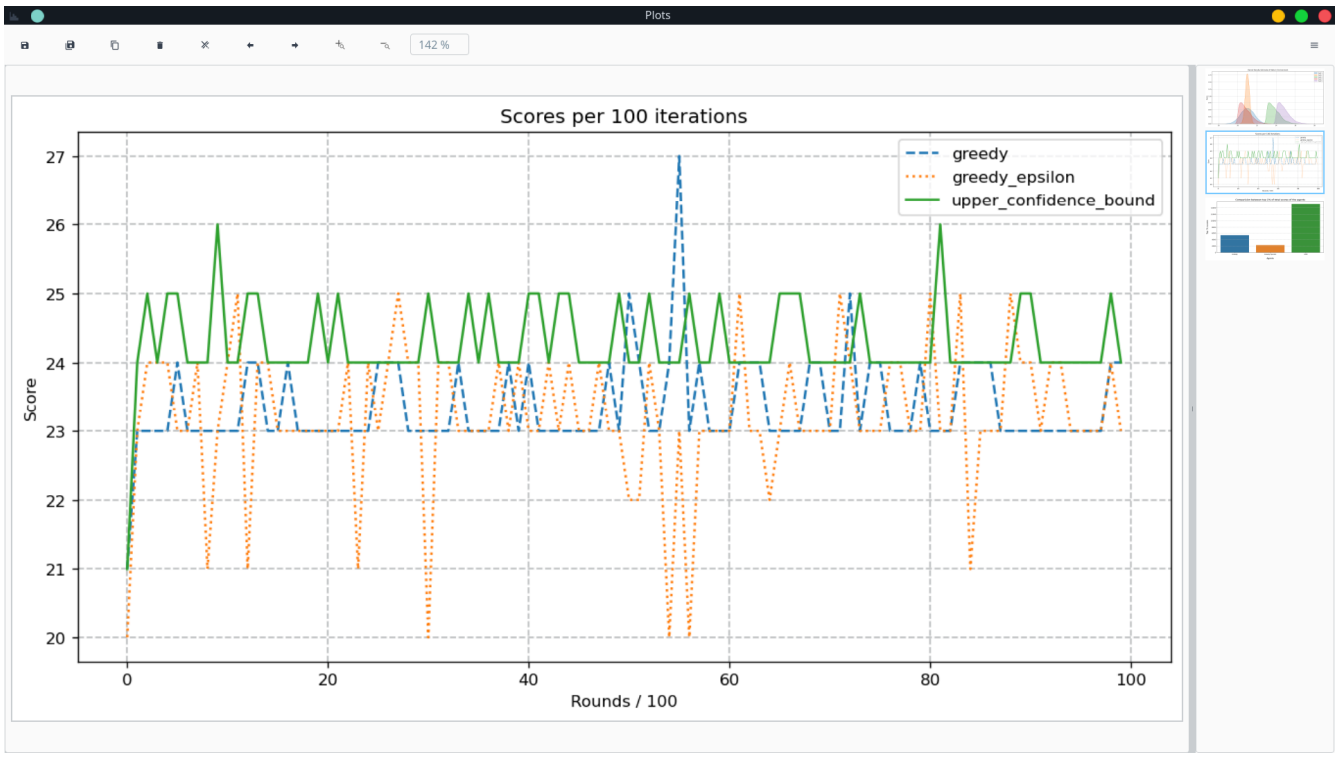
here "*argmax*" specifies choosing the action '*a*' for which *Q (a)* is maximised . Remember that '*Q (a)*' is the estimated value of action '*a*' at time step *'t',* so we're choosing the action with the highest currently estimated value.

UCB:

Rather than performing exploration by simply selecting an arbitrary action, chosen with a probability that remains constant, the UCB algorithm changes its exploration-exploitation balance as it gathers more knowledge of the environment. It moves from being primarily focused on exploration, when actions that have been tried the least are preferred, to instead concentrate on exploitation, selecting the action with the highest estimated reward.

Code Output




Plots
142 %
Scores per 100 iterations
greedy
greedy_epsilon
upper_confidence_bound
Score
Rounds / 100
27
26
25
24
23
22
21
20
0
20
40
60
80
100


Plots
118 %
Kernel Density Estimate of Adds in Environment
Add 1
Add 2
Add 3
Add 4
Add 5
Density
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
18
20
22
24
26
28