Game Playing with Monte-Carlo Tree Search

Proof of Concept: Bandit Problem

Alexandra Danciu

What are Bandit Problems?

Bandit Problems are statistical problems that involve trying to gain as many points over a limited time as possible. There will be multiple places to visit that include an unknown number of points each. The goal is to find a balance between exploring and exploiting.

Exploring means going out to new places and figuring out how many points they can give you, most of the time the points will vary at different times so the average number of points would be best.

Exploiting is going to the place where you know the average number of points given is high and you are likely to gather the most points here.

One example would be going to a restaurant to eat. Say you had 300 days, and you wanted to go out to eat each night at 3 different restaurants. Each restaurant will give you a certain amount of happiness to go eat there each night. One restaurant could give you an average of 10 happiness with a deviation of 2.5 each way, a second restaurant gives you 8 happiness on average with a 2 deviation each way, and the third restaurant offers an average 5 happiness with 12.5 deviation.

You want to have a balance between going to a restaurant and trying its menu and seeing how much happiness it gives you and going to your favourite restaurant that you are more confident will make you happy.

Restaurant 1

Restaurant 2

Restaurant 3

Naïve Strategies

There are a number of different algorithms you can use to solve the bandit problem, however the efficiency of some of them compared to other, better, algorithms are lacking.

Explore Only

One of these less efficient algorithms that still gets the job done is Explore Only. With this algorithm the entire 300 days is spent exploring and picking random restaurants to dine at. By the end you have spent a considerable amount of time at the wrong restaurants and have lost some points on the way. Each restaurant will end up having an equal 100 days spent at them.

Exploit Only

Another inefficient, naïve algorithm would be the Exploit Only, that is similar to Explore Only where it does only one half of the explore/exploit in the problem.

With this algorithm you will use the first three days to visit each restaurant once and which restaurant gave you the highest happiness level you will visit for the rest of the 297 days.

While this may seem like an ok algorithm, the happiness points given from each restaurant have a deviation. On day one you may visit the best restaurant, number 1, and get a low deviation that gives you 7 instead of the average 10, and then you go to restaurant 2 on day 2 and get an 8. You would then continue to go to restaurant 2 for the remainder of the days and would only be visiting the second-best restaurant, lowering your chances for higher total happiness points.

E-Greedy

This is one of the better algorithms that could be used to generate decent results.

With this algorithm you start by denoting an E value, say E = 10%.

What this means is that, each day, there is now a 10% chance that you will visit a random restaurant and a 90% chance that you will visit the restaurant that has historically given you the most happiness points. These exploit days, the 10% chance that you visit a random restaurant, give you more evidence into which restaurant is the best. It will either ensuring that you switch restaurants to the best one in the case that the first time you went to one a deviation happened, or it will make you more confident that the restaurant you are currently going to is the best one.

The issue with this algorithm is that the performance of it depends on the chosen epsilon value and the value of the possible deviations.

Upper Confidence Bound (UCB)

Works off E-Greedy method.

The strategy is that:

At each time T, pick point R, such that

mean of R + √ 2 \* current time T / number of times visited R  
A math equation with numbers and symbols

Description automatically generated

Code

…

Comparison

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Round | Parameters | 1. Explore  Only | 2. Exploit Only | 3. E-Greedy | 4. UBC | Total Points |
| 1 | Total Rounds: 100  Arms: 4  Arm1: 10, 4  Arm2: 5, 5  Arm3: 7, 3  Arm4: 8, 2  Base,  Normal parameters | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
|  |  |  |  |  |  |  |
| 2 | Total Rounds: 500  Arms: 4  Arm1:10, 4  Arm2: 5, 5  Arm3: 7, 3  Arm4: 8, 2  A lot more rounds run | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
|  |  |  |  |  |  |  |
| 3 | Total Rounds: 100  Arms: 8  Arm1:10, 4  Arm2: 5, 5  Arm3: 7, 3  Arm4: 8, 2  Arm5: 6, 4  Arm6: 3, 3  Arm7: 4, 2  Arm8: 1, 5  Double the number of arms | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | Arm1:  Arm2:  Arm3:  Arm4:  Arm5:  Arm6:  Arm7:  Arm8: | 1:  2:  3:  4: |
| 4 | Total Rounds: 100  Arms: 4  Arm1: 10, 4  Arm2: 20, 5  Arm3: 30, 3  Arm4: 40, 2  Greatly increase the distance between points given | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
|  |  |  |  |  |  |  |
| 5 | Total Rounds: 100  Arms: 4  Arm1: 10, 20  Arm2: 5, 15  Arm3: 7, 18  Arm4: 8, 21  Greatly increase the deviation | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |
| Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | Arm1:  Arm2:  Arm3:  Arm4: | 1:  2:  3:  4: |

Bibliography

[ritvikmath](https://www.youtube.com/@ritvikmath)(2020) Multi-Armed Bandit: Data Science Concepts. Available at: <https://www.youtube.com/watch?v=e3L4VocZnnQ>

[ritvikmath](https://www.youtube.com/@ritvikmath)(2020) Best Multi-Armed Bandit Strategy? (feat: UCB Method). Available at: <https://www.youtube.com/watch?v=FgmMK6RPU1c>

[Nikolay Manchev](https://domino.ai/blog/author/nikolay-manchev)(2023) Reinforcement learning: The K-armed bandit problem. Available at: <https://domino.ai/blog/k-armed-bandit-problem>

[Nikolay Manchev](https://domino.ai/blog/author/nikolay-manchev)(2022) Semi-uniform strategies for solving K-armed bandits. Available at: <https://domino.ai/blog/semi-uniform-strategies-for-solving-k-armed-bandits>

samishawl(2020) Upper Confidence Bound Algorithm in Reinforcement Learning. Available at:<https://www.geeksforgeeks.org/upper-confidence-bound-algorithm-in-reinforcement-learning/>

[Ankit](https://www.analyticsvidhya.com/blog/author/ankit2106/)(2023) Reinforcement Learning Guide: Solving the Multi-Armed Bandit Problem from Scratch in Python. Available at:<https://www.analyticsvidhya.com/blog/2018/09/reinforcement-multi-armed-bandit-scratch-python/>