Reinforcement Learning

Upper confidence Bound Algorithm to solve N-arm Bandit Problem

Reinforcement Learning

- Branch of Machine Learning - also called Online Learning.
- It is used when training machines to perform tasks such as walking (AI).
- It is used to solve interacting problems where the data observed up to time t is considered to decide which action to take at time $t + 1$.

Reinforcement Learning

- Desired outcomes provide the AI with reward, undesired with punishment.

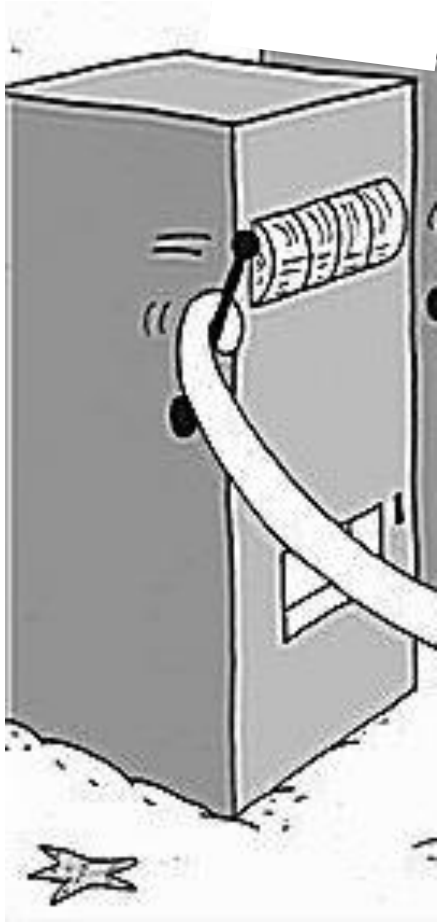
- Machines learn through trial and error.

- In this part, we will implement the Upper Confidence Bound (UCB) Reinforcement Learning models to solve N-armed Bandit Problem.

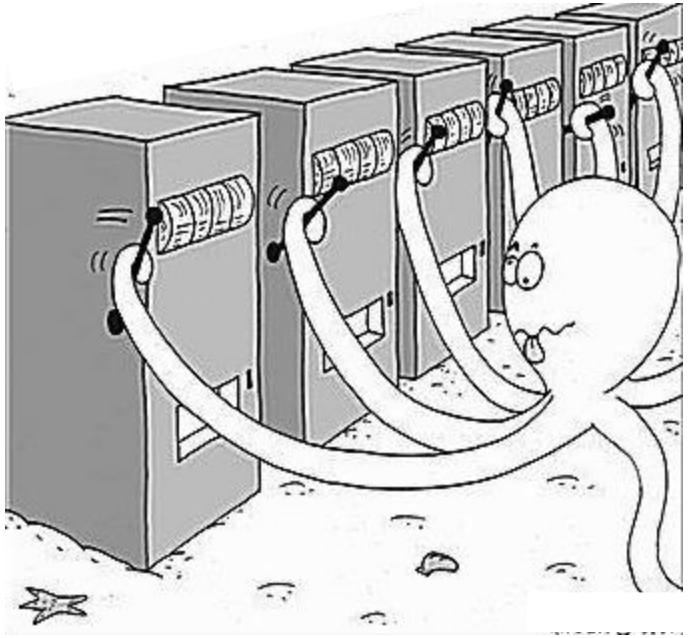
Multi-arm Bandit Problem

- Bandit - someone who steals your money

- One-armed bandit is a simple slot machine wherein you insert a coin into the machine, pull a lever, and get an immediate reward.



Multi-Armed Bandit Problem (MABP)



A multi-armed bandit is a complicated slot machine wherein instead of 1, there are several levers which a gambler can pull, with each lever giving a different return.

The probability distribution for the reward corresponding to each lever is different and is unknown to the gambler.

Upper Confidence Bound (UCB)

- Example Data Set: Click-Through-Rate (CTR) data set
  - Social network advertisement
  - 10000 rows with 10 columns
  - 10 columns – 10 different advertisements of a new SUV car
  - Marketing Dept. of the SUV car company
    - Decide which one is best advt and that can be put-up in the social network
    - Machine learning

Social_Network_Ad

	User ID	Gender	Age	EstimatedSalary	Purchased
1	15624510	Male	19	19000	0
2	15810944	Male	35	20000	0
3	15668575	Female	26	43000	0
4	15603246	Female	27	57000	0
5	15804002	Male	19	76000	0
6	15728773	Male	27	58000	0
7	15598044	Female	27	84000	0
8	15694829	Female	32	150000	1
9	15600575	Male	25	33000	0
10	15727311	Female	35	65000	0
11	15570769	Female	26	80000	0
12	15606274	Female	26	52000	0

Showing 1 to 12 of 400 entries

Index	Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
0	1	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	1	1	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	1	0	0	1	0	0
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	1	0	0
19	0	0	0	0	0	0	0	0	1	0
20	0	1	0	0	0	0	0	1	0	0
21	0	0	0	0	1	0	0	0	0	1

Index	Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
9978	0	0	0	0	1	0	0	0	0	0
9979	0	0	1	0	0	0	1	0	0	0
9980	1	1	0	1	0	0	0	0	0	0
9981	0	0	0	0	0	0	0	0	0	0
9982	0	1	0	0	0	0	0	0	0	0
9983	0	0	0	0	1	0	0	1	1	0
9984	0	0	0	0	1	0	0	0	0	0
9985	0	0	0	0	0	0	0	1	0	0
9986	0	0	0	0	1	0	0	0	0	0
9987	0	0	0	0	1	0	0	0	0	0
9988	1	0	0	0	1	0	0	0	0	0
9989	0	0	0	0	0	0	0	0	0	0
9990	0	0	0	1	0	0	0	0	0	0
9991	0	1	0	1	1	0	1	0	0	0
9992	0	0	0	1	0	0	1	0	0	0
9993	0	0	0	0	1	0	0	0	1	0
9994	0	0	1	0	0	0	0	0	1	0
9995	0	0	1	0	0	0	0	1	0	0
9996	0	0	0	0	0	0	0	0	0	0
9997	0	0	0	0	0	0	0	0	0	0
9998	1	0	0	0	0	0	0	1	0	0
9999	0	1	0	0	0	0	0	0	0	0

Upper Confidence Bound Algorithm

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i(n)$ - the number of times the ad i was selected up to round n,
- $R_i(n)$ - the sum of rewards of the ad i up to round n.

Step 2. From these two numbers we compute:

- the average reward of ad i up to round n

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

- the confidence interval $[\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)]$ at round n with

$$\Delta_i(n) = \sqrt{\frac{3}{2}\frac{\log(n)}{N_i(n)}}$$

Step 3. We select the ad i that has the maximum UCB $\bar{r}_i(n) + \Delta_i(n)$.

```
# Upper Confidence Bound

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Ads_CTR_Optimisation.csv')
```

Index	Ad 1	Ad 2	Ad 3	Ad 4	Ad 5	Ad 6	Ad 7	Ad 8	Ad 9	Ad 10
0	1	0	0	0	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0

```
# Implementing UCB
import math
N = 10000
d = 10
ads_selected = []
numbers_of_selections = [0] * d
sums_of_rewards = [0] * d
total_reward = 0
```

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i(n)$ - the number of times the ad i was selected up to round n,
- $R_i(n)$ - the sum of rewards of the ad i up to round n.

Step 2. From these two numbers we compute:

- the average reward of ad i up to round n

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

- the confidence interval $[\bar{r}_i(n) - \Delta_i(n), \bar{r}_i(n) + \Delta_i(n)]$ at round n with

$$\Delta_i(n) = \sqrt{\frac{3}{2}\frac{\log(n)}{N_i(n)}}$$

```
for n in range(0, N):
    ad = 0
    max_upper_bound = 0
    for i in range(0, d):
        if (numbers_of_selections[i] > 0):
            average_reward = sums_of_rewards[i] / numbers_of_selections[i]
            delta_i = math.sqrt(3/2 * math.log(n + 1) / numbers_of_selections[i])
            upper_bound = average_reward + delta_i
        else:
            upper_bound = 1e400
        if upper_bound > max_upper_bound:
            max_upper_bound = upper_bound
            ad = i
```

Step 3. We select the ad i that has the maximum UCB $\bar{r}_i(n) + \Delta_i(n)$.

```
ads_selected.append(ad)
numbers_of_selections[ad] = numbers_of_selections[ad] + 1
reward = dataset.values[n, ad]
sums_of_rewards[ad] = sums_of_rewards[ad] + reward
total_reward = total_reward + reward
```

```
# Visualising the results
plt.hist(ads_selected)
plt.title('Histogram of ads selections')
plt.xlabel('Ads')
plt.ylabel('Number of times each ad was selected')
plt.show()
```

Selected Ads

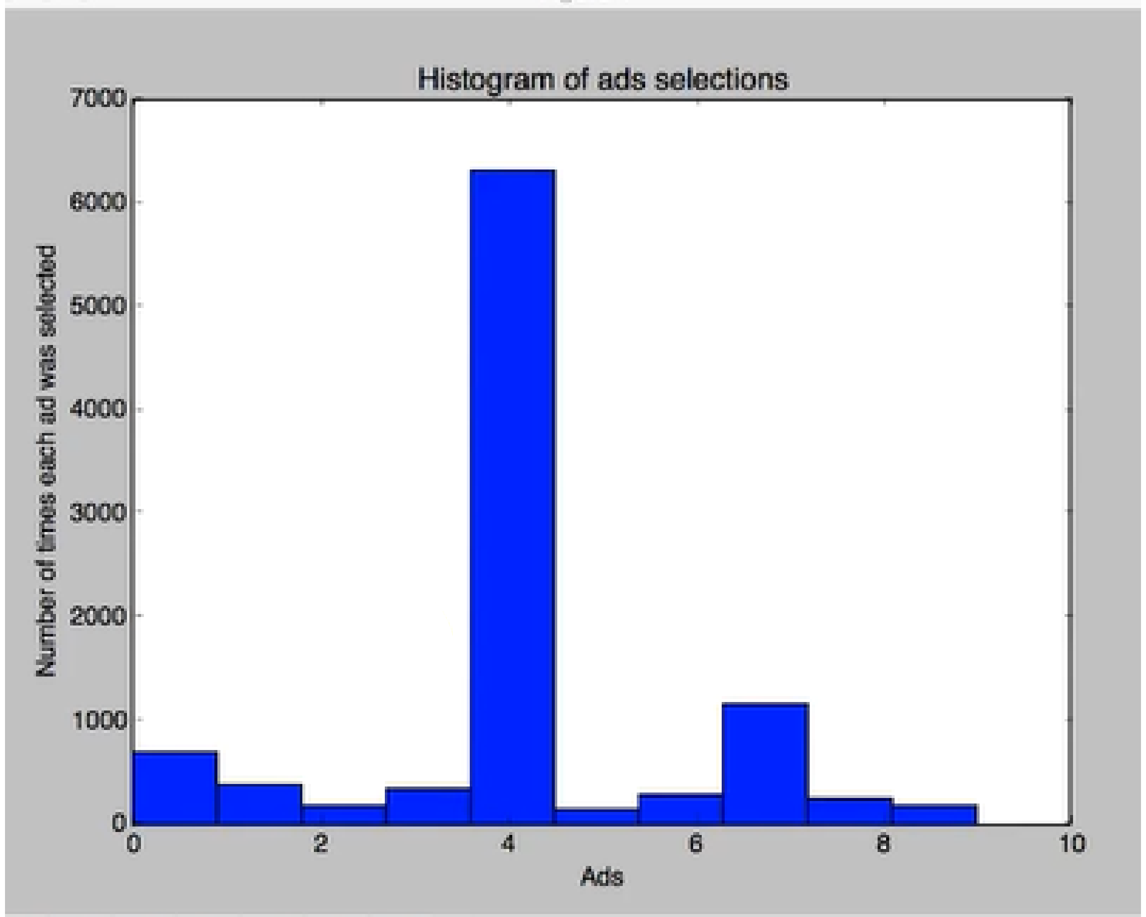
↕	Type	Size	Value
11	int	1	1
12	int	1	2
13	int	1	3
14	int	1	4
15	int	1	5
16	int	1	6
17	int	1	7
18	int	1	8
19	int	1	9
20	int	1	0
21	int	1	0
22	int	1	1
23	int	1	2
24	int	1	3
25	int	1	4

↕	Type	Size	Value
5367	int	1	4
5368	int	1	4
5369	int	1	4
5370	int	1	4
5371	int	1	4
5372	int	1	4
5373	int	1	4
5374	int	1	4
5375	int	1	4
5376	int	1	4
5377	int	1	4
5378	int	1	4
5379	int	1	4
5380	int	1	4
5381	int	1	4

#	Type	Size	Value
8552	int	1	4
8553	int	1	4
8554	int	1	4
8555	int	1	4
8556	int	1	4
8557	int	1	4
8558	int	1	4
8559	int	1	4
8560	int	1	4
8561	int	1	4
8562	int	1	4
8563	int	1	4
8564	int	1	4
8565	int	1	4
8566	int	1	4

#	Type	Size	Value
9985	int	1	4
9986	int	1	4
9987	int	1	4
9988	int	1	4
9989	int	1	4
9990	int	1	4
9991	int	1	4
9992	int	1	4
9993	int	1	4
9994	int	1	4
9995	int	1	4
9996	int	1	4
9997	int	1	4
9998	int	1	4
9999	int	1	4

Name	Type	Size	Value
N	int	1	10000
ad	int	1	4
ads_selected	list	10000	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ...]
average_reward	float64	1	0.037634408602150539
d	int	1	10
dataset	DataFrame	(10000, 10)	Column names: Ad 1, Ad 2, Ad 3, Ad 4, Ad 5, Ad 6, Ad 7, Ad 8, Ad 9, Ad 10
delta_i	float	1	0.27253795787684126
i	int	1	9
max_upper_bound	float64	1	0.31169506812765957
n	int	1	9999
numbers_of_selections	list	10	[705, 387, 186, 345, 6323, 150, 292, 1170, 256, 186]
reward	int64	1	0
sums_of_rewards	list	10	[120, 47, 7, 38, 1675, 1, 27, 236, 20, 7]
total_reward	int64	1	2178
upper_bound	float64	1	0.31017236647899182


Histogram of ads selections
Number of times each ad was selected
Ads
0
1000
2000
3000
4000
5000
6000
7000
0
2
4
6
8
10

Thank you