Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática

Computação Natural

2018/2019

Paulo Novais, Cesar Analide, Filipe Gonçalves

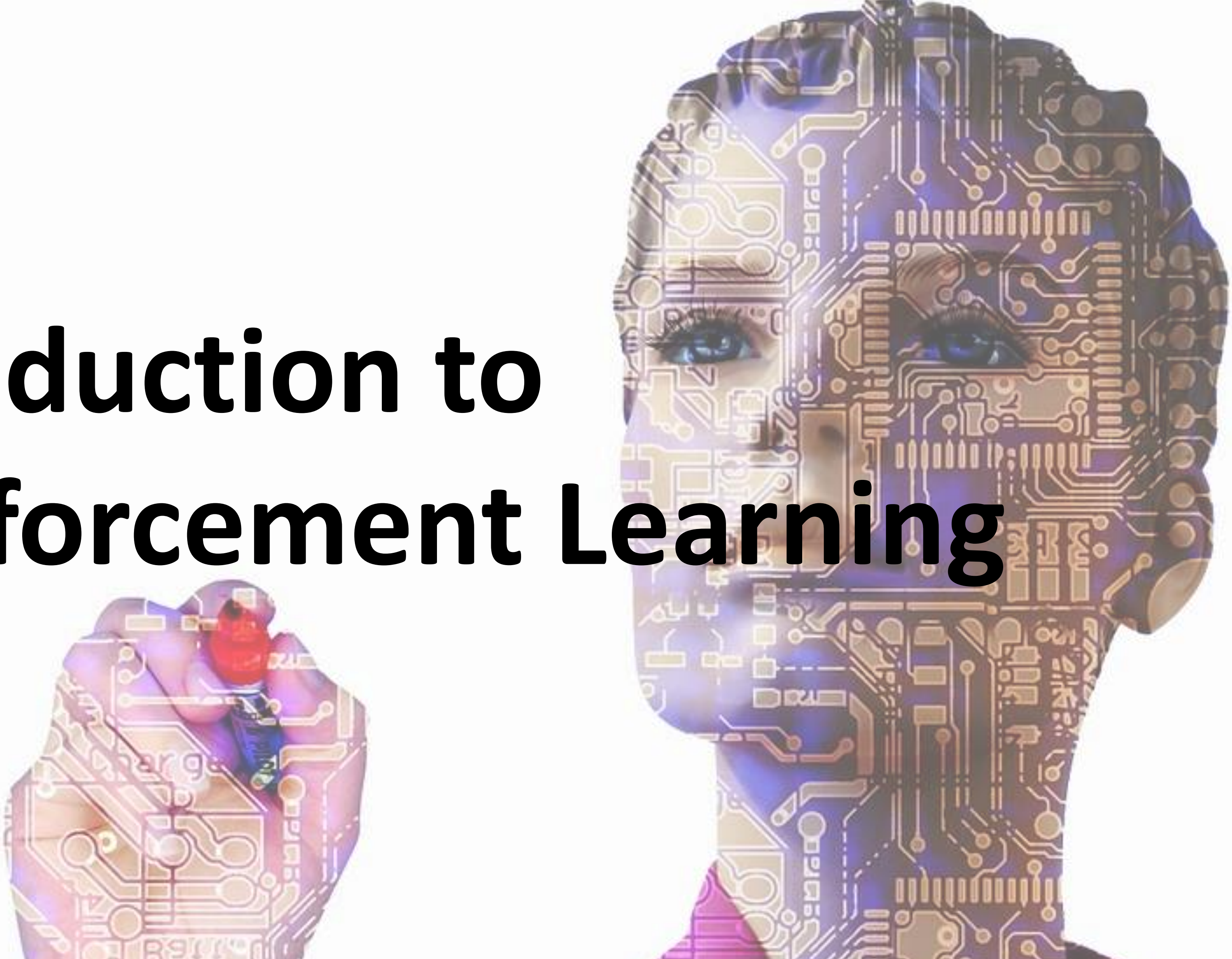
ISLab
Synthetic Intelligence Lab

Computação Natural@ 2018/2019

- Paulo Novais – pjon@di.uminho.pt
- Cesar Analide – cesar.analide@di.uminho.pt
- Filipe Gonçalves – fgoncalves@algoritmi.uminho.pt

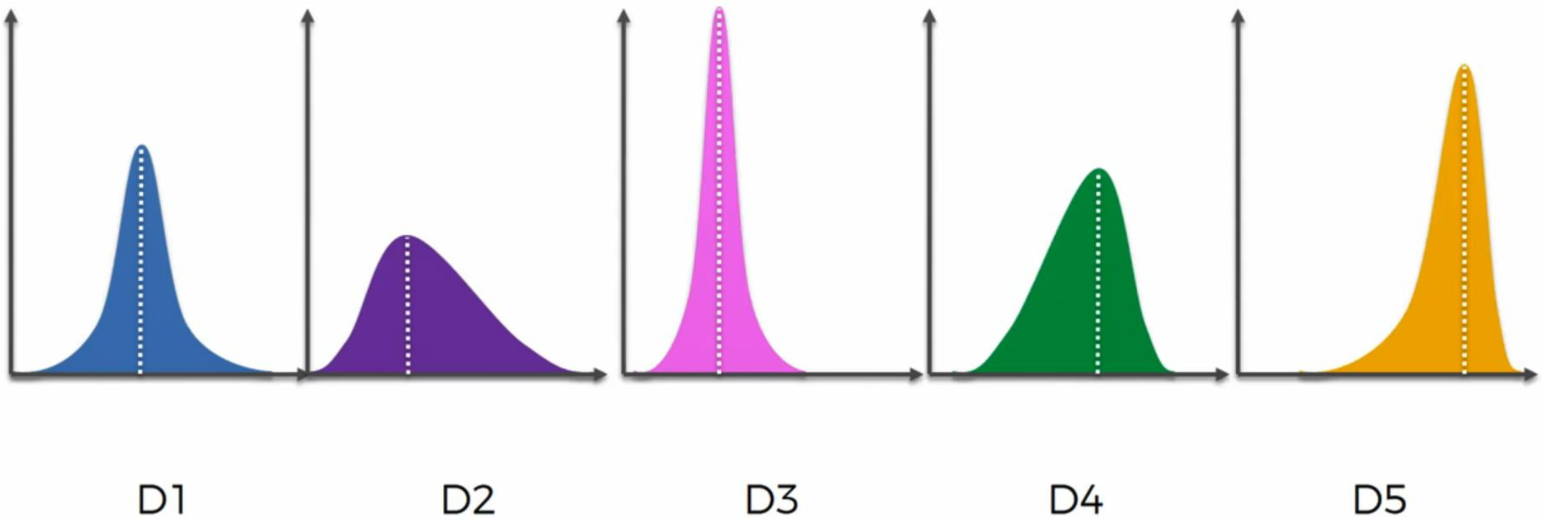
- Departamento de Informática
  Escola de Engenharia
  Universidade do Minho
- Grupo ISLab – (Synthetic Intelligence Lab)
- Centro ALGORITMI
  Universidade do Minho

Introduction to Reinforcement Learning

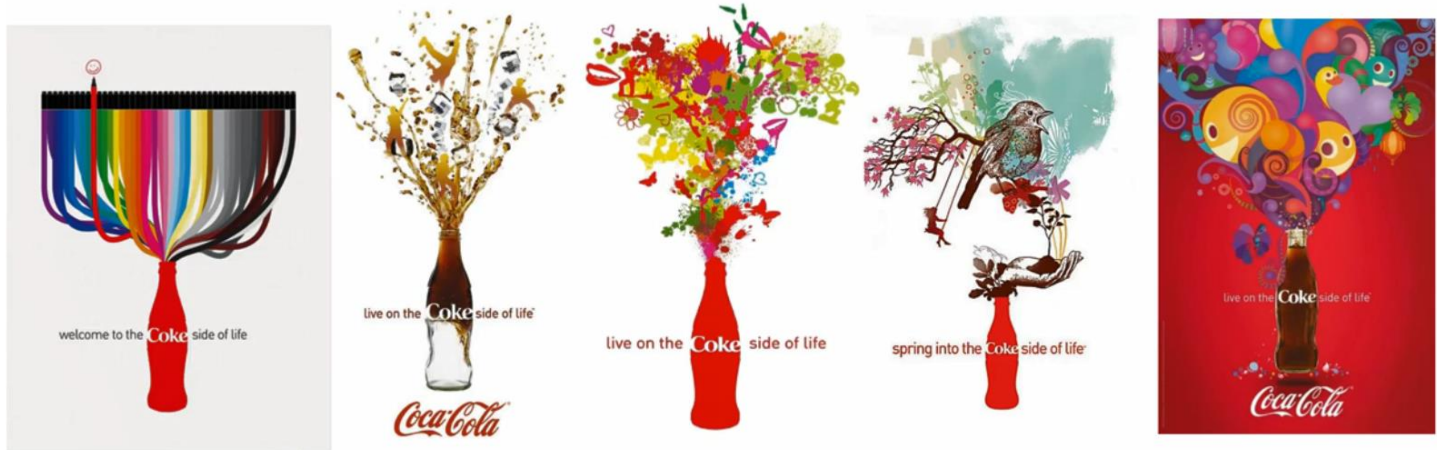


- The Multi-armed Bandit Problem (Exploration vs Exploitation Problem)

- The Multi-armed Bandit Problem (Exploration vs Exploitation Problem)



- Marketing - Ads Selection (Exploration vs Exploitation Problem)



D1 D2 D3 D4 D5

- Marketing - Ads Selection (Exploration vs Exploitation Problem)
  - We have d arms. For example, arms are ads that we display to users each time they connect to a web page
  - Each time a user connects to this web page, that makes a round
  - At each round n, we choose one ad to display to the user
  - At each round n, ad i gives reward:

    $r_i(n) \in \{0, 1\}$: $r_i(n) = 1$ if the user clicked on the ad i, 0 if the user didn't
  - Goal: maximize the total reward we get over many rounds

- Marketing - Ads Selection
  - Index: Person
  - Column: Ads
    - 1: Person clicked on Ad
    - 0: Person ignored Ad

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

- Marketing - Ads Selection
    - Random Selection

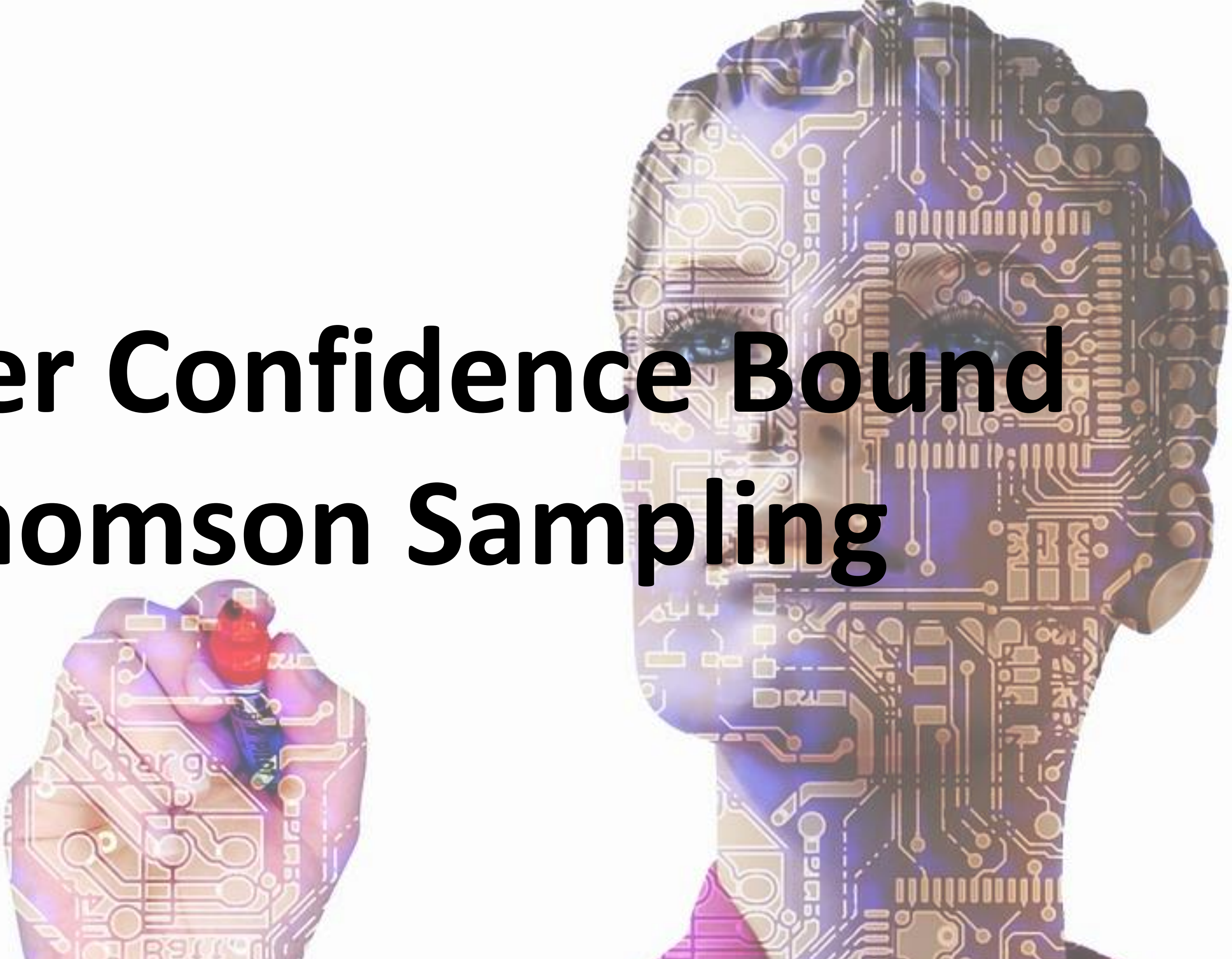
```python
# Random Selection

# 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')

# Implementing Random Selection
import random
N = 10000
d = 10
ads_selected = []
total_reward = 0
for n in range(0, N):
    ad = random.randrange(d)
    ads_selected.append(ad)
    reward = dataset.values[n, ad]
    total_reward = total_reward + reward

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

- Marketing - Ads Selection
  - Random Selection

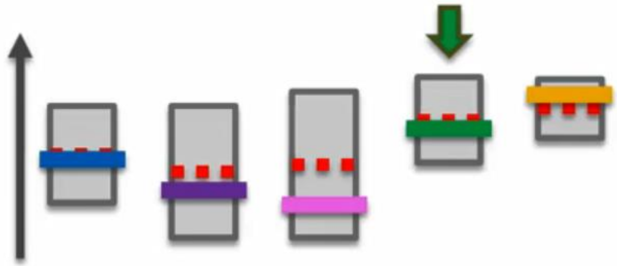
i	Type	Size	Value
0	int	1	4
1	int	1	6
2	int	1	1
3	int	1	0
4	int	1	4
5	int	1	5
6	int	1	0
7	int	1	8
8	int	1	5
9	int	1	3
10	int	1	8
11	int	1	5
12	int	1	8
13	int	1	4
14	int	1	0

total_reward	int64	1	1193

Upper Confidence Bound vs Thomson Sampling

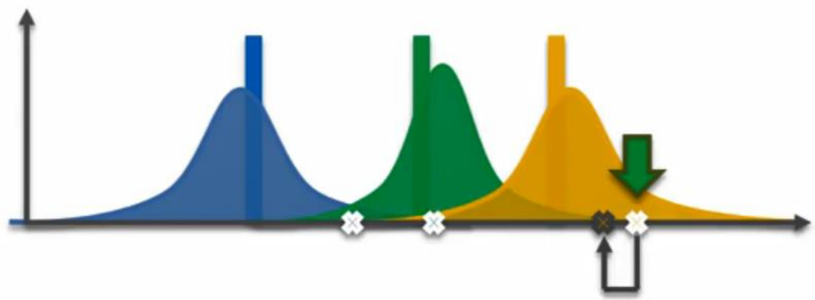


UCB



- Deterministic
- Requires update at every round

Thompson Sampling



- Probabilistic
- Can accommodate delayed feedback
- Better empirical evidence

- 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 $\bar{r}_i(n) + \Delta_i(n)$.

ISLab
Synthetic Intelligence Lab

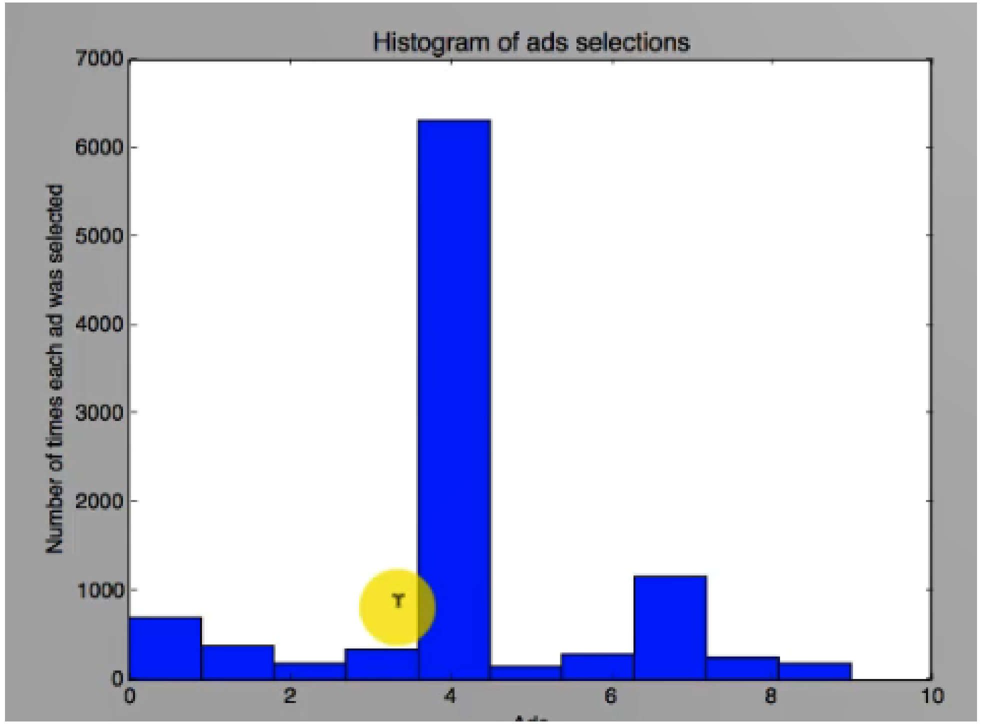
- Marketing - Ads Selection
    - Upper Confidence Bound

```python
# 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')

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

Synthetic Intelligence Lab

- Marketing - Ads Selection
    - Upper Confidence Bound

1	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



total_reward	int64	1	2178
upper_bound	float64	1	0.31017236647899182

- Thompson Sampling Algorithm

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

- $N_i^1(n)$ - the number of times the ad i got reward 1 up to round n,
- $N_i^0(n)$ - the number of times the ad i got reward 0 up to round n.

Step 2. For each ad i, we take a random draw from the distribution below:

$$\theta_i(n) = \beta(N_i^1(n) + 1, N_i^0(n) + 1)$$

Step 3. We select the ad that has the highest $\theta_i(n)$.

- Thompson Sampling Algorithm
    - Ad i gets rewards $\boldsymbol{y}$ from Bernoulli distribution $p(\boldsymbol{y}|\theta_i) \sim \mathcal{B}(\theta_i)$.
    - θ_i is unknown but we set its uncertainty by assuming it has a uniform distribution $p(\theta_i) \sim \mathcal{U}([0,1])$, which is the prior distribution.
    - Bayes Rule: we approach θ_i by the posterior distribution

$$\underbrace{p(\theta_i|\mathbf{y})}_{\text{posterior distribution}} = \frac{p(\boldsymbol{y}|\theta_i)p(\theta_i)}{\int p(\boldsymbol{y}|\theta_i)p(\theta_i)d\theta_i} \propto \underbrace{p(\boldsymbol{y}|\theta_i)}_{\text{likelihood function}} \times \underbrace{p(\theta_i)}_{\text{prior distribution}}$$

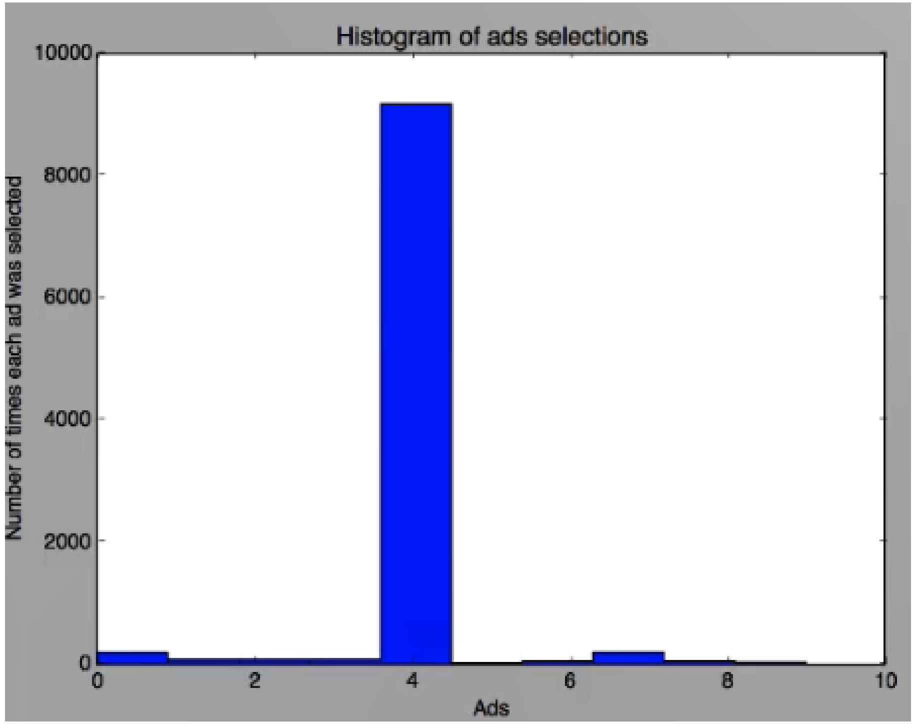
    - We get $p(\theta_i|\mathbf{y}) \sim \beta(\text{number of successes} + 1, \text{number of failures} + 1)$
    - At each round n we take a random draw $\theta_i(n)$ from this posterior distribution $p(\theta_i|\boldsymbol{y})$, for each ad i.
    - At each round n we select the ad i that has the highest $\theta_i(n)$.

- Marketing - Ads Selection
  - Thompson Sampling

```python
# Thompson Sampling

# 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')

# Implementing Thompson Sampling
import random
N = 10000
d = 10
ads_selected = []
numbers_of_rewards_1 = [0] * d
numbers_of_rewards_0 = [0] * d
total_reward = 0
for n in range(0, N):
    ad = 0
    max_random = 0
    for i in range(0, d):
        random_beta = random.betavariate(numbers_of_rewards_1[i] + 1, numbers_of_rewards_0[i] + 1)
        if random_beta > max_random:
            max_random = random_beta
            ad = i
    ads_selected.append(ad)
    reward = dataset.values[n, ad]
    if reward == 1:
        numbers_of_rewards_1[ad] = numbers_of_rewards_1[ad] + 1
    else:
        numbers_of_rewards_0[ad] = numbers_of_rewards_0[ad] + 1
    total_reward = total_reward + reward
```

- Marketing - Ads Selection
  - Thompson Sampling

i	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



total_reward	int64	1	2613

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Mestrado Integrado em Engenharia Informática
Computação Natural
2018/2019

Paulo Novais, Cesar Analide, Filipe Gonçalves