# Reinforcement Learning

## Upper Confidence Bound (UCB)

### UCB Intuition

Could you please explain in greater details what a distribution is? What are on the x-axis and y-axis?

Let us assume that we have an experiment with ad clicking 100 times, and compute what is the frequency of ad clicking. Then we repeat it again for another 100 times. And again for another 100 times. Hence we obtain many frequencies. If we repeat such frequency computations many times, for example 500 times, we can plot a histogram of these frequencies. By the Central Limit Theorem it will be bell-shaped, and its mean will be the mean of all frequencies we obtained over the experiment. Therefore in conclusion, on the x-axis we have the different possible values of these frequencies, and on the y-axis we have the number of times we obtained each frequency over the experiment.

In the first Intuition Lecture, could you please explain why D5 (in orange) is the best distribution? Why is it not D3 (in pink)?

In this situation, 0 is loss, and 1 is gain, or win. The D5 is the best because it is skewed so we will have average outcomes close to 1, meaning that there we have more wins, or gains. And actually all casino machines nowadays are carefully programmed to have distribution like D1 or D3. But it is a good concrete example.

### UCB in Python

Why does a single round can have multiple 1s for different ads?

Each round corresponds to a user connecting to a web page, and the user can only see one ad when he/she connects to the web page, the ad that is being shown to him/her. If he/she clicks on the ad, the reward is 1, otherwise it’s 0. And there can be multiple ones because there are several ads that the user would love to click on. But only a crystal ball would tell us which ads the user would click on. And the dataset is exactly that crystal ball (we are doing a simulation here).

Could you please explain in more details what we do with the rewards at lines 33 & 34?

Let’s explain clearly:

* In line 33 we are just getting the reward at the specific round n. So we get a 1 if the user clicked on the ad at this round n, and 0 if the user didn’t.
* In line 34 we are getting the sum of all the rewards up to round n. So this sum is incremented by 1 if the user clicked on the ad at this round n, and stays the same if the user didn’t.

Does the UCB strategy really come into effect during the first rounds?

Not at the beginning, first we must have some first insights of the users response to the ads. That is just the beginning of the strategy. In real world for the first ten users connecting to the webpage, you would show ad 1 to the first user, ad 2 to the second user, ad 3 to the third user,..., ad 10 to the 10th user. Then the algorithm starts.

What was the purpose of this algorithm? Why couldn’t I just count which type of ad converted the most and then used that?

Because at each round there is a cost (the cost of putting one ad on the page). So basically the purpose of UCB is not only to maximize the revenue but also to minimize the cost. And if you just count which type of ad converted the most, that would require you to experiment a lot of rounds and therefore the cost would be high and definitely not minimized.

### UCB in R

How to get this kind of dataset?

In real life, you could do this by data streaming, using Spark or Hive, meaning you would get real time data. Otherwise if you want evaluate your Reinforcement Learning models, you can simulate a dataset like the one we use in the tutorials.

What is exactly the ’number of selections’?

’number of selections’ is the number of times the ad was displayed to a user.

I don’t understand why there is no ad selection by the algorithm in the first 10 rounds. The instructor said there is no strategy at the beginning so the first round will display Ad 1, 2nd round Ad 2, 3rd round Ad 3, and so on.. So why is there no selection by UCB?

Because at this point we know nothing about ads and we cannot apply our formula because it involves division by how many times an ad was shown. We may not divide by 0.

In Python the winning ad was ad no. 4 and in R it was ad no. 5. However the same

Ads\_CTR\_Optimisation.csv file was used. Why we are getting two different results using the same logic?

Because indexes in Python start from 0 and indexes in R start from 1. Hence Python and R both get the same ad.

Could you please provide a R implementation of the UCB regret curve?

|  |
| --- |
| *# Importing the dataset*  dataset = read.csv(’Ads\_CTR\_Optimisation.csv’)  *# Implementing UCB* N = 10000 d = 10  ads\_selected = integer(0) numbers\_of\_selections = integer(d) sums\_of\_rewards = integer(d) total\_reward = 0  rewards\_at\_each\_step=integer(0)  best\_selection=(rowSums(dataset)==0) *# my addition* best\_rewards\_at\_each\_step=integer(0) *# my addition*  for (n in 1:N) {  ad = 0  max\_upper\_bound = 0  for (i in 1:d) { if (numbers\_of\_selections[i] > 0) {  average\_reward = sums\_of\_rewards[i] / numbers\_of\_selections[i] delta\_i = sqrt(3/2 \* log(n) / 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  }  } |

5

10

15

20

25

|  |
| --- |
| ads\_selected = append(ads\_selected, ad)  numbers\_of\_selections[ad] = numbers\_of\_selections[ad] + 1 reward = dataset[n, ad]  sums\_of\_rewards[ad] = sums\_of\_rewards[ad] + reward total\_reward = total\_reward + reward  rewards\_at\_each\_step=c(rewards\_at\_each\_step, total\_reward) *#my addition* best\_rewards\_at\_each\_step[n]=sum(best\_selection[1:n]) *# my addition* }  *# Regret curve*  plot(best\_rewards\_at\_each\_step, pch=’.’, col=3, main="Real and Imaginary Rewards; Regret", ylab="Reward Numbers")  points(rewards\_at\_each\_step, pch=".", col=4)  points((best\_rewards\_at\_each\_step-rewards\_at\_each\_step), pch=’.’, col="darkgoldenrod") legend(’topleft’, legend=c("Best", "Real", "Regret"), col=c(7,4, "darkgoldenrod"), horiz=F, lty=1, lwd=2) |

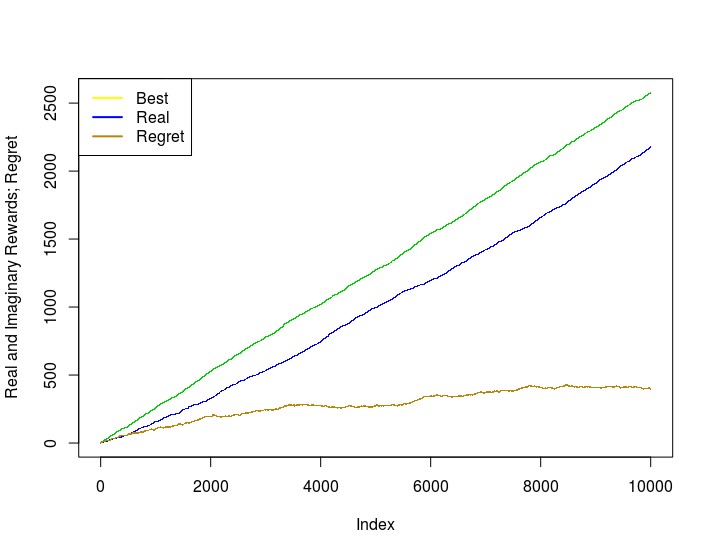
30

35

40

45

And we obtain the following plot:



## Thompson Sampling

### Thompson Sampling Intuition

Why is the yellow mark the best choice, and not the green mark?

The yellow mark is the best choice because it is the furthest from the origin on the x-axis, which therefore means that it has the highest estimated return.

How is Thompson Sampling better than UCB?

Thompson Sampling is better than UCB in terms of convergence of the regret. The regret is the difference between the optimal reward and the reward you accumulate with your algorithm. Thompson Sampling shows a better regret curve than UCB in my experience. Also, the fact that UCB is deterministic as opposed to Thompson Sampling being stochastic, helps making Thompson Sampling outperform UCB. Besides you will see in the practical sections that Thompson Sampling finds the best ad faster and with more certainty than UCB.

I don’t understand how Thompson Sampling can accept delayed feedback. Please explain. When doing Thompson Sampling, we can still perform updates in our algorithm (like making new guesses for the distributions with existing data, sampling from the guessed distribution, etc) while we are waiting for the results of an experiment in the real world. This would not hinder our algorithm from working. This is why it can accept delayed feedback.

What are further examples of Thompson Sampling applications?

Another potential application of Multi-armed bandits (MAB) can be the online testing of algorithms.

For example, let’s suppose you are running an e-commerce website and you have at your disposal several Machine Learning algorithms to provide recommendations to users (of whatever the website is selling), but you don’t know which algorithm leads to the best recommendations.

You could consider your problem as a MAB problem and define each Machine Learning algorithm as an "arm": at each round when one user requests a recommendation, one arm (i.e. one of the algorithms) will be selected to make the recommendations, and you will receive a reward. In this case, you could define your reward in various ways, a simple example is "1" if the user clicks/buys an item and "0" otherwise. Eventually your bandit algorithm will converge and end up always choosing the algorithm which is the most efficient at providing recommendations. This is a good way to find the most suitable model in an online problem. Another example coming to my mind is finding the best clinical treatment for patients: each possible treatment could be considered as an "arm", and a simple way to define the reward would be a number between 0 (the treatment has no effect at all) and 1 (the patient is cured perfectly).

In this case, the goal is to find as quickly as possible the best treatment while minimizing the cumulative regret (which is equivalent to say you want to avoid as much as possible selecting "bad" or even sub-optimal treatments during the process).

### Thompson Sampling in Python

Where can I find some great resource on the Beta distribution? The best is the following: [Beta Distribution](https://stats.stackexchange.com/questions/47771/what-is-the-intuition-behind-beta-distribution)

The histogram shows 4 as the best ad to select. The highest bar is at 4 on the x axis. Why is it being said as 5?

It’s just because indexes in Python start from 0. So the ad of index 4 it ad number 5.

I am curious as to how one would apply Thompson Sampling proactively when running this theoretical ad campaign. Would you iterate the program over each round (i.e. each time all adds were presented to the user)?

First, a data engineer makes a pipeline for reading the data from website and reacting to it in real time.

Then a web visit triggers a response to recompute our parameters and to choose an ad for the next time.

### Thompson Sampling in R

Could you please explain in your own words the difference between the Bernoulli and Beta distributions?

The Bernoulli distribution is applied to discrete probabilities because it gives the probability of success of an outcome, whereas the Beta distribution is applied to dense (continuous) probabilities, using a dense probability function. That’s the main difference. Here are some excellent resources:

[Bernoulli distribution](http://www.statisticshowto.com/bernoulli-distribution/)

[Beta distribution](http://www.statisticshowto.com/beta-distribution/)

Could you please provide a great source on Bayesian Inference? Here is a great one: [Bayesien Inference](https://brohrer.github.io/how_bayesian_inference_works.html)

How is Thomson Sampling heuristic quickly able to find that 5th advertisement is the best one in comparison to the Upper Confidence Bound heuristic?

It is hard to explain the reason theoretically, that would require to do research and write a long mathematical proof. But intuitively, it could be because UCB is based on optimistic assumptions whereas Thompson Sampling is based on relevant probabilities through the Bayesian approach.

How to plot the Thompson Sampling Regret Curve in R?

Check out the last question in the UCB in R section. Basically you simply need to add the following at the end of your code:

|  |
| --- |
| rewards\_at\_each\_step=c(rewards\_at\_each\_step, total\_reward) best\_rewards\_at\_each\_step[n]=sum(best\_selection[1:n])  }  plot(best\_rewards\_at\_each\_step, pch=’.’, col=3, main="Real and Imaginary Rewards; Regret", ylab="Reward Numbers")  points(rewards\_at\_each\_step, pch=".", col=4)  points((best\_rewards\_at\_each\_step-rewards\_at\_each\_step), pch=’.’, col="darkgoldenrod") legend(’topleft’, legend=c("Best", "Real", "Regret"), col=c(7,4, "darkgoldenrod"), horiz=F, lty=1, lwd=2) |

5

10