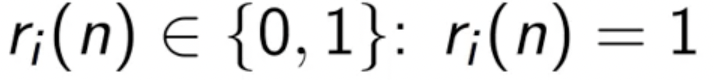
**Upper Confidence Bound**

**(UCB)**

* The Multi-Armed Bandit Problem:
* When we read the words “Multi-Armed Bandit”, the first thing that comes to mind is a robber, or a bad guy with multiple arms. But it is referring to a slot machine as it has a handle/lever to pull.
* They are called bandit because these machines were the easiest way to lose money, it was a 50% chance you would lose money back in the day, you would lose money more than you would make, and casino owners put some sort of a bug in it so people would lose even more money. Thus, a bandit.
* The multi-armed bandit is a challenge that a person is faced with when they come up to a whole set of these machines. Basically, you are faced with a challenge, you have got a set of these machines (let’s say 5 or 6), so how would you play them to maximize your return from the number of games you can play.
* Do describe the problem in more detail, we have got to mention that the assumption here is that each of these machines has a distribution behind it.
* So, there is a distribution of numbers/outcomes out of which the machine picks the results – you pull the lever, and it picks out randomly a result/outcome from its distribution.
* To keep it simple, it decided whether you win or lose based on the distribution that’s built into the machine.
* But the problem is we don’t know the distribution used in this machine, and they are assumed to be different for these machines. Sometimes, they could be similar, but by default they are different.
* And the goal is to figure out which distribution­ is the best for us.

Chart, histogram

Description automatically generated

* Above we have got 5 distributions for 5 machines, and just by looking we can say that the one on the right (D5) is the best distribution because it’s got the most favorable outcome – the highest mean, median, and mode.
* And if we knew these distributions, we would always bet on the machine with distribution with D5.
* But we don’t know that in advance, and our goal is to figure it out.
* To add to that, we are already spending money to find that, and the longer we take to figure out, the more money we will spend on the wrong ones.
* So, there are these two factors in play – exploration and exploitation. You need to explore the machines to find out which one is the best one, and at the same time you need to, as soon as you can, start exploiting these machines to make maximum return.
* There is a mathematical concept behind this called ‘Regret’.
* The goal here is to find the best machine and spend the least amount exploring.
* And it is important to understand that there is a machine with the best/most optimal return value.
* Even though these machines they have something like jackpot, we are assuming that there are distributions that are finite, and out of them there is a best one that we are looking for.
* The most common modern application of this we can think of, and the one we will explore is advertising.
* Let’s say a company is running campaign for their product where they have multiple ads for the campaign, and we want to find out which is the best ad and works best.
* Right now, we don’t know which ad works the best. There is a distribution behind each ad, but we don’t know that until 1000s of people look at them.
* One way to approach the problem is just run an A-B test – take all the ads you have, run a huge A-B test, wait until you have a large enough sample to conclude which ad is the best. But the problem with that approach is that a lot of resources would be spent to run those tests. It would be pure exploration and not much of exploitation.
* The goal is to find out which is the best one in the actual launch campaign, find the best one in the quickest way possible and start exploiting it along the way.
* **Upper Confidence Bound** **–**
* We have *d* arms. For example, arms are ads 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  if the user clicked on the ad *I*, 0 if the user didn’t.
* The goal is to maximize total rewards received over many rounds.
* And this is the algorithm for Upper Confidence Bound:

Table

Description automatically generated with low confidence

* Instead of understanding the algorithm, we will try to understand what is happening in the background.
* We have *n* one-armed bandits, each with its own probability distribution, and we want to find the machine which has the highest return. And in our case, the machines are the ads, and we want to find which ads have the most user interaction rate. We can do a simple A|B test, but we don’t have the time and the money to do the exploration. Thus, we want to maximize the return value right from the start before the campaign starts.

Chart, histogram

Description automatically generated

* So, we will take the expected values of the distribution and plot them on a vertical axis.

Chart

Description automatically generated

* Those are the expected values for each of the distributions. But again, we don’t know that.
* How this algorithm works is it assumes some starting point for every distribution. Because we can’t distinguish between these machines/ads, the starting point is assumed the same for all the machines to be at the same level.

A picture containing chart

Description automatically generated

* Now what the algorithm does is that those formula that are behind the algorithm, they create a confidence band, and it is designed in such a way that with a very high level of certainty those confidence band will include the actual return or the actual expected return.
* So basically, the first couple of rounds are going to be trial runs. We are going to intentionally try out each of the machines at least one time each to place the value on the plot and come up with a confidence band, which is going to be very large at the start, but it is designed specifically in a way that the expected value with a very high level of confidence/certainty falls in the confidence bound which is built around the red empirical values chosen at the very start.

Chart, box and whisker chart

Description automatically generated

* Then, out of all of them, we pick the machine with the highest confidence bound. Right now, it can be any of those machines, they have the same upper confidence bound. But in this case, they all have the same confidence bound – we don’t know the solid lines yet.
* So, let’s say we pick machine with distribution D3.
* Next, we pull the lever of that machine, or play that ad next, and see whether the webpage visitor clicked on it or not. And in this case, the person didn’t click on it.

Chart, box and whisker chart

Description automatically generated

* Thus, the red value (it is the value set as a start value, also the observed average) was decreased. And now we have another observation for this machine that was added to the sample of observations of this machine. And according to the law of large number, the observed average in the long run is going to converge with the expected return/average for that distribution.
* And it is very likely that the value is going to go down. Because we have an extra observation, the second thing that happens is the confidence bound interval becomes smaller.
* Then we repeat the step for every other machines/ad, and as expected the starting value slowly starts to converge with the expected return/average of the respective distribution. Because it is a random distribution, there is a chance that the observed average can go up as well.
* We keep adding observations, and the more we add observations the more the observed average will converge towards to the expected value, and the more the confidence bounds will decrease in size.

Chart, box and whisker chart

Description automatically generated

* And after certain iterations, even though we found out the optimal distribution and are exploiting it to get the best result, the algorithm will still run the suboptimal machines/ads to decrease the confidence bounds.
* This is in essence the whole concept behind the Upper Confidence Bound (UCB) algorithm, and this is how it solves the Multi-Armed Bandit Problem.