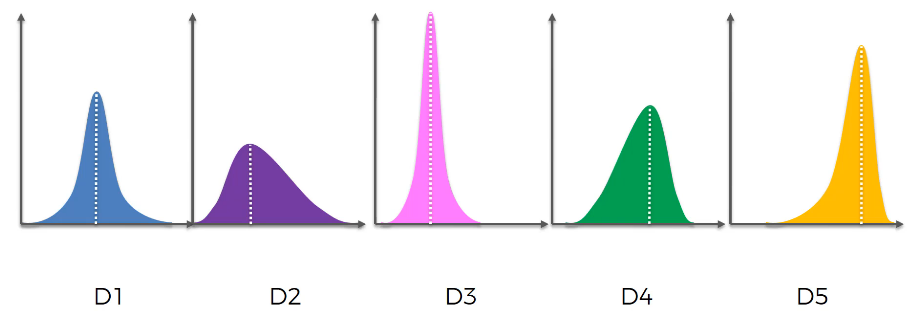
1. Upper Confidence Bound (UCB)

1.1 The Multi-Armed Bandit Problem

The MABP is a problem in which fixed limited resources must be allocated between competing choices in a way that maximizes their expected gain, when each choice’s properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating resources to the choice. This is a classical reinforcement learning problem that exemplifies the exploration-exploitation tradeoff dilemma. The name comes from imagining a gambler at a row of slot machines who has to decide which machines to play, how many times to play each machine and in which order to play them, and whether to continue with the current machine or different machine.

An assumption to be considered is that each machine has a certain distribution and it picks its outcome from this distribution only. Let’s say we have 5 machines. The distribution for each machine is given below:



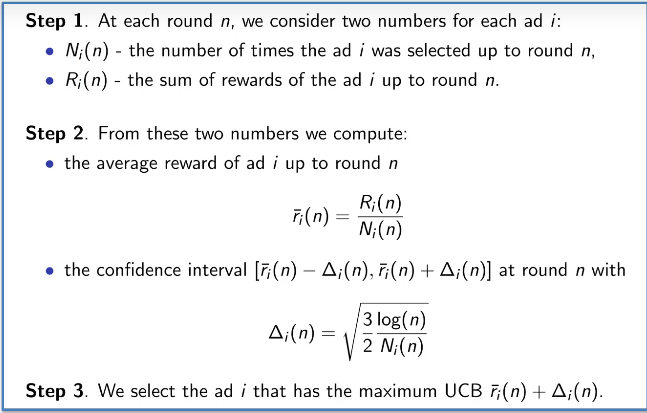
From the above distribution, we would pick the 5th machine since it has higher mean, median and mode. But we don’t know these distributions. The goal is to find the best machine without spending much money and time.

Another real-life example of this problem could be an ad campaign. Out of some selected ads, we want to finalize an ad which would bring the highest return of all.

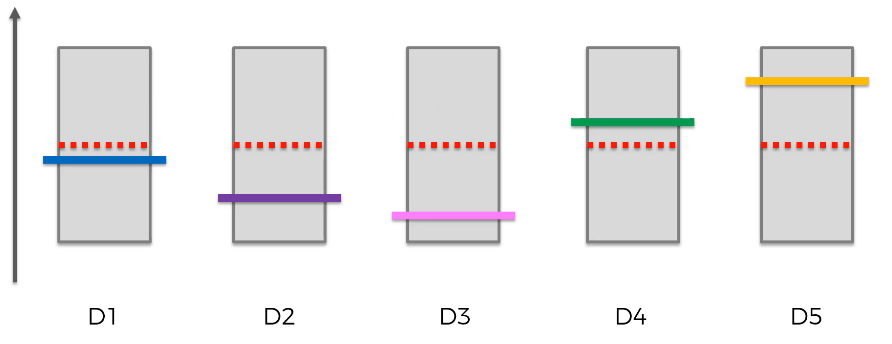
In mathematical terms, the MAB problem can be written as follows:

* 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 *ri(n)* ∈ {0 , 1}: *ri(n)* = 1 if the user clicked on the ad i, 0 if the user didn’t
* Our goal is to maximize the total reward we get over many rounds

1.2 Upper Confidence Bound Algorithm



For each arm, the algorithm initially assumes a starting point which is same for all arms since we don’t know which arm has higher distribution. Then, the formulas in the algorithm create a confidence band and it is designed in such a way that we have a very high level of certainty that the confidence bands will include the actual return.



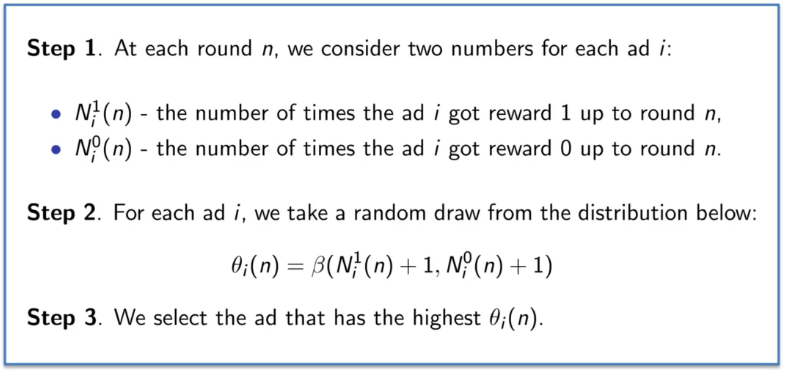
The next step is to choose the arm with the highest confidence bound. For this, we randomly choose an arm (or ad in our example) and see if a person rolled that arm (or clicked on that ad) or not. If a person does not click on that add, the starting point shifts down and the confidence interval shrinks. This process is repeated again for different arms (or ads). When a person clicks on an ad, the starting point goes up and the confidence bound shrinks.

2. Thompson Sampling

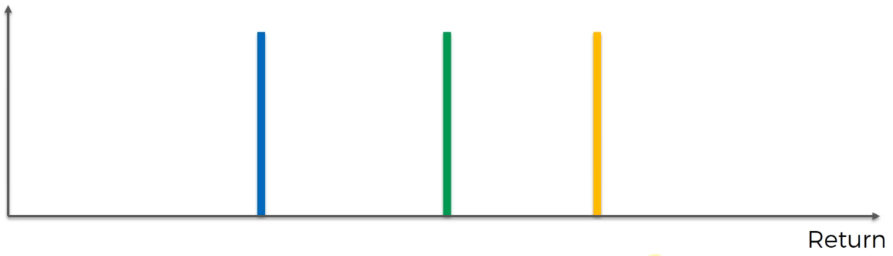
Consider the example of the Multi-arm bandit problem. In mathematical terms, the MAB problem can be written as follows:

* 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 *ri(n)* ∈ {0 , 1}: *ri(n)* = 1 if the user clicked on the ad i, 0 if the user didn’t
* Our goal is to maximize the total reward we get over many rounds

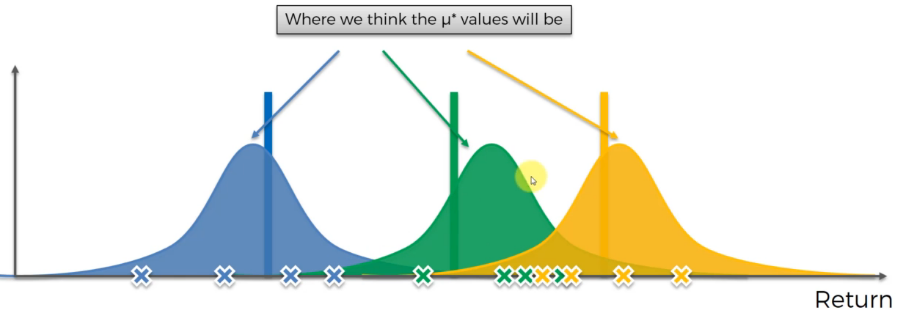
2.1 Thompson Sampling Algorithm



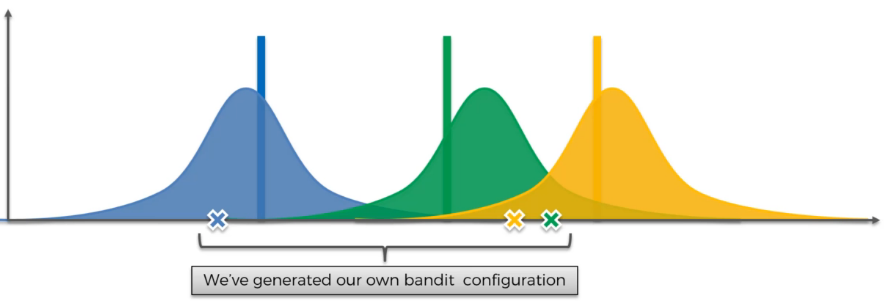
Consider an example of 3 bandits. Following graph shows the distribution of these bandits with their expected return. Assume that the vertical lines are the centers of the distribution of the respective machines.



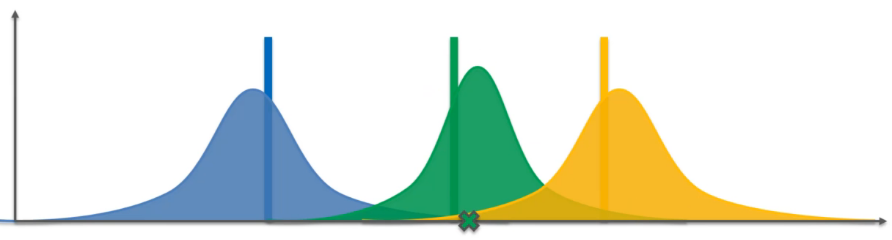
After some trial runs, the Thompson sampling algorithm will start constructing a distribution from the data it received from the trial runs.



The algorithm then pulls out a value out of each distribution and it considers these values as the actual expected value of the machines.



From the above 3 values, it shows that the green machine has the highest return. Therefore, we pull the lever on the green machine and we get a certain return from the green machine which is going to be very close to the actual distribution (green vertical line). Since there is some difference between the actual and simulated return of the green machine, the algorithm will adjust this difference by shifting the simulated output towards the actual output. Due to this, the algorithm’s confidence has increased and hence the curve becomes narrower.



Now, the entire algorithm repeats until we get to a point where the distribution are refines substantially.