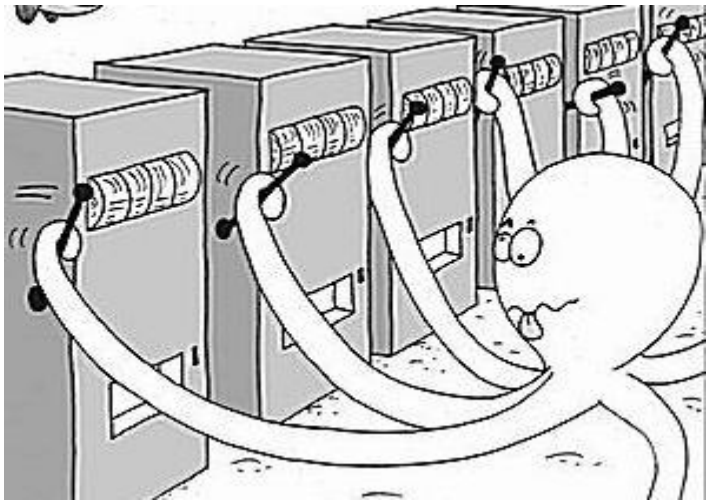
Multi-Armed Bandit

@Author https://github.com/ReactiveCJ

Definition

In probability theory , the multi-armed bandit problem is a problem in which a gambler at a row of slot machines(sometimes known as “on-armed bandit”) has to decide which machines to play, how many times to play each machine and in which order to play them. When played, each machine provides a random reward from a distribution specific to that machine. The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls.



From the another point of view, we can assume there exists one slot machine with multi arms and each time one arm should be selected before the reward is provided.

The multi-armed bandit machine can been seen as a multi Bernoulli distribution. The probability of the reward provided by each arm can be generated by one Bernoulli distribution.

This problem is also called Exploitation/Exploration problem.

Exploitation solves the Multi-armed Bandit Problem by exploiting the arm with the highest estimated value with respect to success times and rewards of previous plays.

Exploration solves the Multi-armed Bandit Problem by exploring any arm that does not have the highest estimated value based on previous plays.

The trade-off between exploitation and exploration is also faced in reinforcement learning.

Application

The bandit problem is formally equivalent to a one-state Markov decision process, so the strategy for bandit problem can be applied to many practical problems which faced to make choice at each time. We will cited two practical application here.

In Real Time Bidding of online advertisement, for any PV, we have some Adgroups of two types, one has the best estimated value maybe evaluated by CTR from the historical data, the other one has not been display on this type of PV with the enough times and maybe not has the best estimated value now. So the trade-off is to exploit the best adgroup to bid at present or to explore the other adgroup which is the best actually.

In the website/app optimization problem, we have a new logo and an old one to test which is the better one. Habitually someone would run the A/B test: randomly show the different log to the visitors. After a time, we can analysis the residence time of the groups of the visitors. But

there are a few problems. One problem is splitting the visitors into two groups too roughly. We may loss the profits of our website/app if one logo is not welcomed, even worse, we may loss our users. Another problem is even the best logo learned in A/B testing is not always the best option.

For example, users may preferred red color logo in the New Year which doesn't mean the red color logo is the best choice all of the time.

Thankfully, we can regard the website/app optimization as Multi-armed Bandit Problem. Every strategy for optimizing the website/app can been seen as the arm, so we can acquire the performance of different strategies without loss too much profits

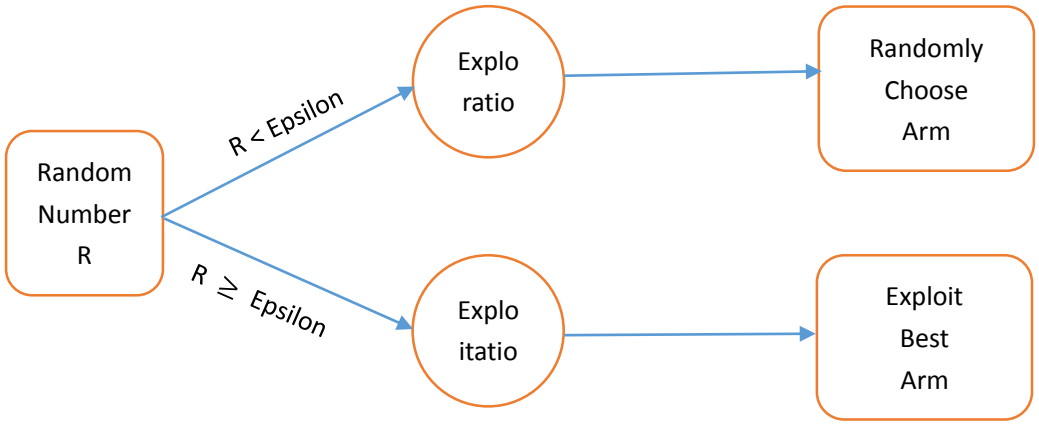
Algorithm

Many strategies exist which provide approximate solution for multi-armed problem. And I will explain and implement four algorithms.

Epsilon-Greedy：

The epsilon-greedy algorithm is a naive algorithm based on greedy strategy.

Procedure：



Parameter: Epsilon

If the epsilon is set to 1.0, the algorithm will always choose among the different arms completely at random.

If the epsilon is set to 0.0, the algorithm will never waste time on bad options.

Weakness:

- ✓ Hard to set the Epsilon, you will be over-exploring or over-exploiting
- ✓ The epsilon parameter should be set by how much the information we need to know about, so it should not be fixed.

Annealing Soft-Max：

Based on the past experiences, every arm have had two different rate of success.

For example, there are two arms named A and B. rA is the rate of success of arm A, so is rB.

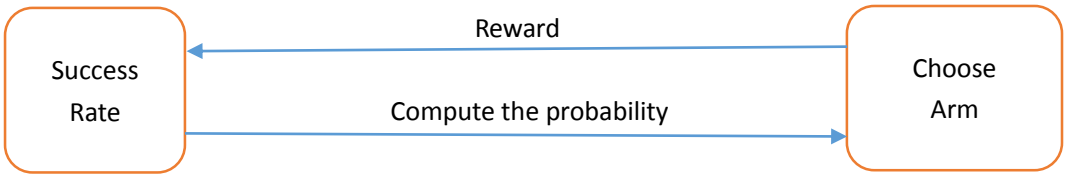
At time T, select one of the two arms with probabilities computed as follows:

— exp(rA / tau) / (exp(rA / tau) + exp(rB / tau))

— exp(rB / tau) / (exp(rA / tau) + exp(rB / tau))

Tau is temperature parameter.

Procedure：



Weakness:

The algorithm just pay too much attention on how much reward or success rate they've gotten from the arms. This means it under-explore the options whose initial experience is not rewarding but it does not have enough data to be confident about these arms. As time goes on, the data on these arms is harder to collect.

Upper Confidence Bound：

This algorithm pays attention to not only what it knows but also how much it knows.

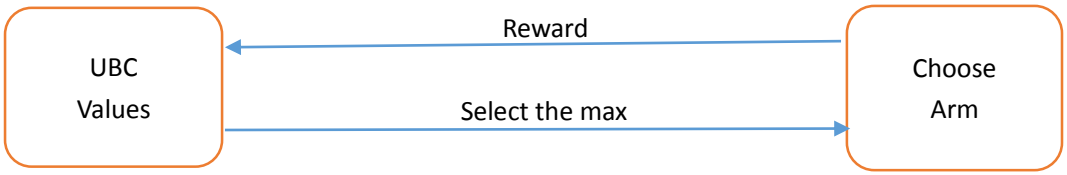
The UCB makes decision to explore based on two factors: the estimated value and the confidence of the estimated value.

Value: total reward to be gotten of one arm

Confidence: $\sqrt[2]{2 * \log(count_{total})} / count_{arm}$

UCB Value: Value + Confidence

Procedure：



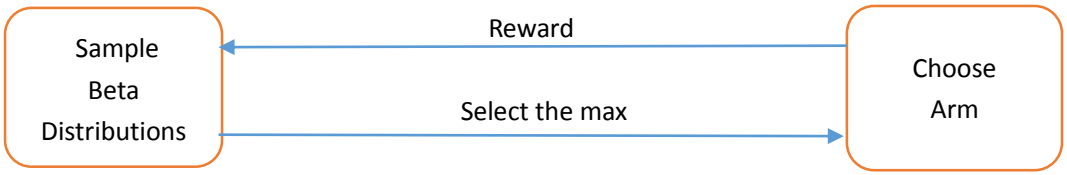
Weakness:

This is a frequentist approach because we compute the confidence just by the counts. From Bayesian view, we need a distribution and prior distribution, so we can adjust the confidence easily.

Thompson Sampling：

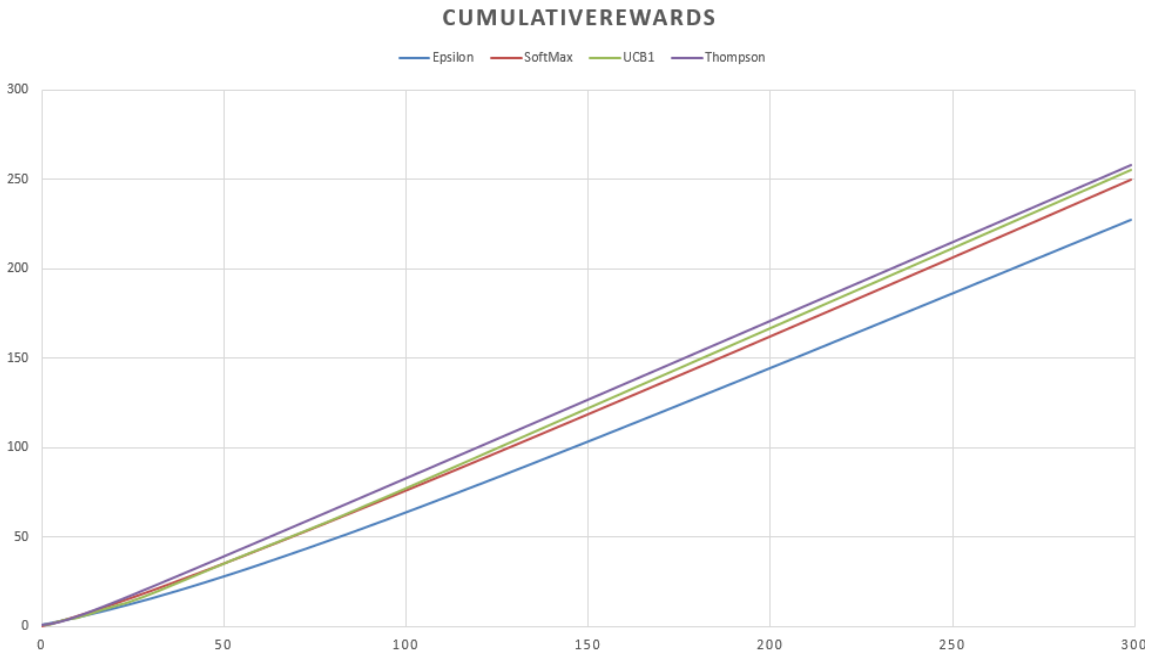
For each arm $i = 1, \ldots, N$ set $S_i = 0, F_i = 0$.
foreach $t = 1, 2, \ldots,$ **do**
  For each arm $i = 1, \ldots, N$, sample $\theta_i(t)$ from the Beta$(S_i + 1, F_i + 1)$ distribution.
  Play arm $i(t) := \arg\max_i \theta_i(t)$ and observe reward r_t.
  If $r = 1$, then $S_{i(t)} = S_{i(t)} + 1$, else $F_{i(t)} = F_{i(t)} + 1$.
end

Procedure：



Experiment

MCMC can be used to simulate the process of above algorithms. We will compare these algorithm by the accumulated rewards.(Tested on the Bernoulli Bandit)



Implementation

Build it yourself

This project can be built with sbt 0.13. And the breeze package is used for implementing beta distribution.

For **SBT**, Add these lines to your SBT project definition:

```
libraryDependencies  ++= Seq(
  "org.scalanlp" %% "breeze" % "0.11.2"
)
```

Reference

Bandit Algorithms for Website Optimization by John Myles White

Analysis of Thompson Sampling for the Multi-armed Bandit Problem by Shipra and Navin

An Information-Theoretic Analysis of Thompson Sampling by Daniel And Benjamin