

The Multi-Armed Bandit Problem and Its Solutions

January 23, 2018 · 10 min · Lilian Weng

▼ Table of Contents

- Exploitation vs Exploration
- What is Multi-Armed Bandit?
 - Definition
 - Bandit Strategies
- ϵ -Greedy Algorithm
- Upper Confidence Bounds
 - Hoeffding's Inequality
 - UCB1
 - Bayesian UCB
- Thompson Sampling
- Case Study
- Summary
- References

The algorithms are implemented for Bernoulli bandit in [lilianweng/multi-armed-bandit](https://lilianweng.github.io/posts/2018-01-23-multi-armed-bandit/).

Exploitation vs Exploration

The exploration vs exploitation dilemma exists in many aspects of our life. Say, your favorite restaurant is right around the corner. If you go there every day, you would be confident of what you will get, but miss the chances of discovering an even better option. If you try new places all the time, very likely you are gonna have to eat unpleasant food from time to time. Similarly, online advisors try to balance between the known most attractive ads and the new ads that might be even more successful.

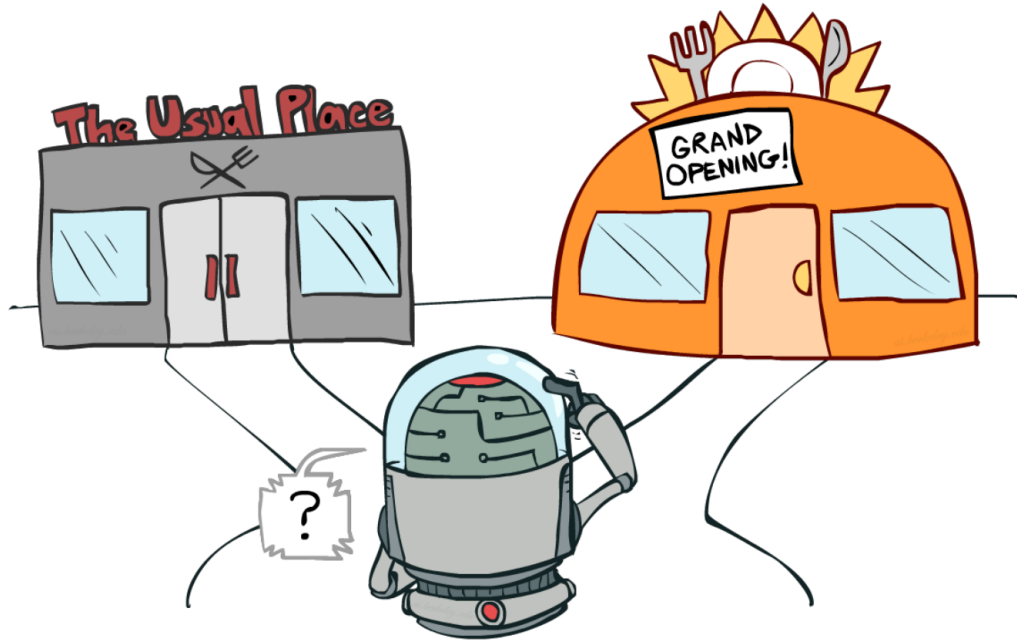


Fig. 1. A real-life example of the exploration vs exploitation dilemma: where to eat? (Image source: UC Berkeley AI course [slide](#), [lecture 11](#).)

If we have learned all the information about the environment, we are able to find the best strategy by even just simulating brute-force, let alone many other smart approaches. The dilemma comes from the *incomplete* information: we need to gather enough information to make best overall decisions while keeping the risk under control. With exploitation, we take advantage of the best option we know. With exploration, we take some risk to collect information about unknown options. The best long-term strategy may involve short-term sacrifices. For example, one exploration trial could be a total failure, but it warns us of not taking that action too often in the future.

What is Multi-Armed Bandit?

The multi-armed bandit problem is a classic problem that well demonstrates the exploration vs exploitation dilemma. Imagine you are in a casino facing multiple slot machines and each is configured with an unknown probability of how likely you can get a reward at one play. The question is: *What is the best strategy to achieve highest long-term rewards?*

In this post, we will only discuss the setting of having an infinite number of trials. The restriction on a finite number of trials introduces a new type of exploration problem. For instance, if the number of trials is smaller than the number of slot machines, we cannot even try every machine to estimate the reward probability (!) and hence we have to behave smartly w.r.t. a limited set of knowledge and resources (i.e. time).

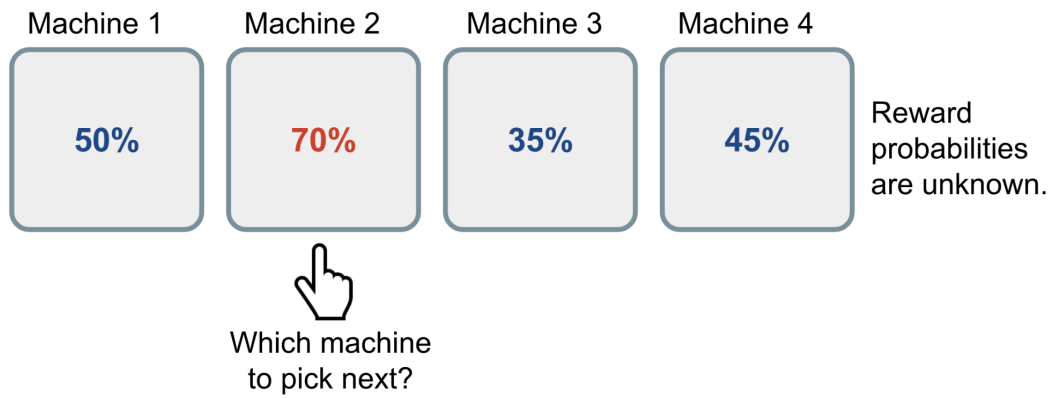


Fig. 2. An illustration of how a Bernoulli multi-armed bandit works. The reward probabilities are **unknown** to the player.

A naive approach can be that you continue to playing with one machine for many many rounds so as to eventually estimate the “true” reward probability according to the law of large numbers. However, this is quite wasteful and surely does not guarantee the best long-term reward.

Definition

Now let’s give it a scientific definition.

A Bernoulli multi-armed bandit can be described as a tuple of $\langle \mathcal{A}, \mathcal{R} \rangle$, where:

- We have K machines with reward probabilities, $\{\theta_1, \dots, \theta_K\}$.
- At each time step t , we take an action a on one slot machine and receive a reward r .
- \mathcal{A} is a set of actions, each referring to the interaction with one slot machine. The value of action a is the expected reward, $Q(a) = \mathbb{E}[r|a] = \theta$. If action a_t at the time step t is on the i -th machine, then $Q(a_t) = \theta_i$.
- \mathcal{R} is a reward function. In the case of Bernoulli bandit, we observe a reward r in a *stochastic* fashion. At the time step t , $r_t = \mathcal{R}(a_t)$ may return reward 1 with a probability $Q(a_t)$ or 0 otherwise.

It is a simplified version of Markov decision process, as there is no state \mathcal{S} .

The goal is to maximize the cumulative reward $\sum_{t=1}^T r_t$. If we know the optimal action with the best reward, then the goal is same as to minimize the potential regret or loss by not picking the optimal action.

The optimal reward probability θ^* of the optimal action a^* is:

$$\theta^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a) = \max_{1 \leq i \leq K} \theta_i$$

Our loss function is the total regret we might have by not selecting the optimal action up to the time step T :

$$\mathcal{L}_T = \mathbb{E} \left[\sum_{t=1}^T (\theta^* - Q(a_t)) \right]$$

Bandit Strategies

Based on how we do exploration, there several ways to solve the multi-armed bandit.

- No exploration: the most naive approach and a bad one.
- Exploration at random
- Exploration smartly with preference to uncertainty

ϵ -Greedy Algorithm

The ϵ -greedy algorithm takes the best action most of the time, but does random exploration occasionally. The action value is estimated according to the past experience by averaging the rewards associated with the target action a that we have observed so far (up to the current time step t):

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{\tau=1}^t r_\tau \mathbb{1}[a_\tau = a]$$

where $\mathbb{1}$ is a binary indicator function and $N_t(a)$ is how many times the action a has been selected so far, $N_t(a) = \sum_{\tau=1}^t \mathbb{1}[a_\tau = a]$.

According to the ϵ -greedy algorithm, with a small probability ϵ we take a random action, but otherwise (which should be the most of the time, probability $1-\epsilon$) we pick the best action that we have learnt so far: $\hat{a}_t^* = \arg \max_{a \in \mathcal{A}} \hat{Q}_t(a)$.

Check my toy implementation [here](#).

Upper Confidence Bounds

Random exploration gives us an opportunity to try out options that we have not known much about. However, due to the randomness, it is possible we end up exploring a bad action which we have confirmed in the past (bad luck!). To avoid such inefficient exploration, one approach is to decrease the parameter ϵ in time and the other is to be optimistic about options with *high uncertainty* and thus to prefer actions for which we haven't had a confident value estimation

yet. Or in other words, we favor exploration of actions with a strong potential to have a optimal value.

The Upper Confidence Bounds (UCB) algorithm measures this potential by an upper confidence bound of the reward value, $\hat{U}_t(a)$, so that the true value is below with bound $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability. The upper bound $\hat{U}_t(a)$ is a function of $N_t(a)$; a larger number of trials $N_t(a)$ should give us a smaller bound $\hat{U}_t(a)$.

In UCB algorithm, we always select the greediest action to maximize the upper confidence bound:

$$a_t^{UCB} = \operatorname{argmax}_{a \in \mathcal{A}} \hat{Q}_t(a) + \hat{U}_t(a)$$

Now, the question is *how to estimate the upper confidence bound*.

Hoeffding's Inequality

If we do not want to assign any prior knowledge on how the distribution looks like, we can get help from "Hoeffding's Inequality" — a theorem applicable to any bounded distribution.

Let X_1, \dots, X_t be i.i.d. (independent and identically distributed) random variables and they are all bounded by the interval $[0, 1]$. The sample mean is $\bar{X}_t = \frac{1}{t} \sum_{\tau=1}^t X_\tau$. Then for $u > 0$, we have:

$$\mathbb{P}[\mathbb{E}[X] > \bar{X}_t + u] \leq e^{-2tu^2}$$

Given one target action a , let us consider:

- $r_t(a)$ as the random variables,
- $Q(a)$ as the true mean,
- $\hat{Q}_t(a)$ as the sample mean,
- And u as the upper confidence bound, $u = U_t(a)$

Then we have,

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq e^{-2tU_t(a)^2}$$

We want to pick a bound so that with high chances the true mean is blow the sample mean + the upper confidence bound. Thus $e^{-2tU_t(a)^2}$ should be a small probability. Let's say we are ok with a tiny threshold p :

$$e^{-2tU_t(a)^2} = p \text{ Thus, } U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}$$

UCB1

One heuristic is to reduce the threshold p in time, as we want to make more confident bound estimation with more rewards observed. Set $p = t^{-4}$ we get **UCB1** algorithm:

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}} \text{ and } a_t^{UCB1} = \arg \max_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Bayesian UCB

In UCB or UCB1 algorithm, we do not assume any prior on the reward distribution and therefore we have to rely on the Hoeffding's Inequality for a very generalize estimation. If we are able to know the distribution upfront, we would be able to make better bound estimation.

For example, if we expect the mean reward of every slot machine to be Gaussian as in Fig 2, we can set the upper bound as 95% confidence interval by setting $\hat{U}_t(a)$ to be twice the standard deviation.

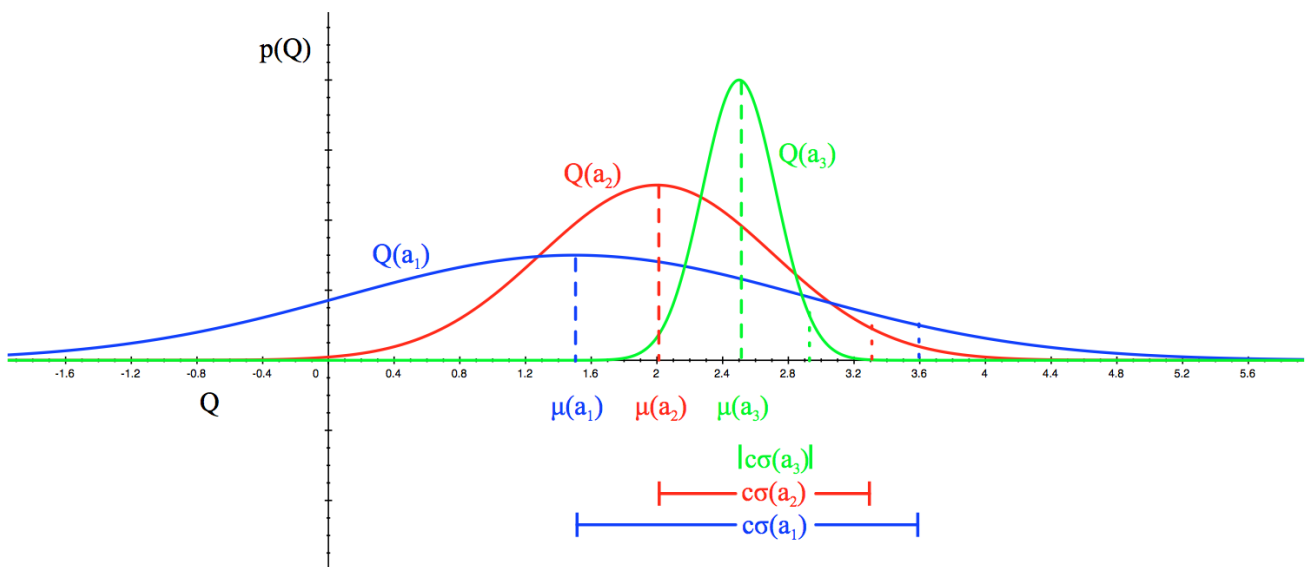


Fig. 3. When the expected reward has a Gaussian distribution. $\sigma(a_i)$ is the standard deviation and $c\sigma(a_i)$ is the upper confidence bound. The constant c is a adjustable hyperparameter. (Image source: [UCL RL course lecture 9's slides](#))

Check my toy implementation of [UCB1](#) and [Bayesian UCB](#) with Beta prior on θ .

Thompson Sampling

Thompson sampling has a simple idea but it works great for solving the multi-armed bandit problem.



Fig. 4. Oops, I guess not this Thompson? (Credit goes to Ben Taborsky; he has a full theorem of how Thompson invented while pondering over who to pass the ball. Yes I stole his joke.)

At each time step, we want to select action a according to the probability that a is **optimal**:

$$\begin{aligned}\pi(a \mid h_t) &= \mathbb{P}[Q(a) > Q(a'), \forall a' \neq a \mid h_t] \\ &= \mathbb{E}_{\mathcal{R} \mid h_t}[\mathbb{1}(a = \arg \max_{a \in \mathcal{A}} Q(a))]\end{aligned}$$

where $\pi(a; \mid; h_t)$ is the probability of taking action a given the history h_t .

For the Bernoulli bandit, it is natural to assume that $Q(a)$ follows a Beta distribution, as $Q(a)$ is essentially the success probability θ in Bernoulli distribution. The value of $\text{Beta}(\alpha, \beta)$ is within the interval $[0, 1]$; α and β correspond to the counts when we **succeeded** or **failed** to get a reward respectively.

First, let us initialize the Beta parameters α and β based on some prior knowledge or belief for every action. For example,

- $\alpha = 1$ and $\beta = 1$; we expect the reward probability to be 50% but we are not very confident.
- $\alpha = 1000$ and $\beta = 9000$; we strongly believe that the reward probability is 10%.

At each time t , we sample an expected reward, $\tilde{Q}(a)$, from the prior distribution $\text{Beta}(\alpha_i, \beta_i)$ for every action. The best action is selected among samples: $a_t^{TS} = \arg \max_{a \in \mathcal{A}} \tilde{Q}(a)$. After the true reward is observed, we can update the Beta distribution accordingly, which is essentially doing Bayesian inference to compute the posterior with the known prior and the likelihood of getting the sampled data.

$$\begin{aligned}\alpha_i &\leftarrow \alpha_i + r_t \mathbb{1}[a_t^{TS} = a_i] \\ \beta_i &\leftarrow \beta_i + (1 - r_t) \mathbb{1}[a_t^{TS} = a_i]\end{aligned}$$

Thompson sampling implements the idea of probability matching. Because its reward estimations \tilde{Q} are sampled from posterior distributions, each of these probabilities is equivalent to the probability that the corresponding action is optimal, conditioned on observed history.

However, for many practical and complex problems, it can be computationally intractable to estimate the posterior distributions with observed true rewards using Bayesian inference. Thompson sampling still can work out if we are able to approximate the posterior distributions using methods like Gibbs sampling, Laplace approximate, and the bootstraps. This tutorial presents a comprehensive review; strongly recommend it if you want to learn more about Thompson sampling.

Case Study

I implemented the above algorithms in lilianweng/multi-armed-bandit. A BernoulliBandit object can be constructed with a list of random or predefined reward probabilities. The bandit algorithms are implemented as subclasses of Solver, taking a Bandit object as the target problem. The cumulative regrets are tracked in time.

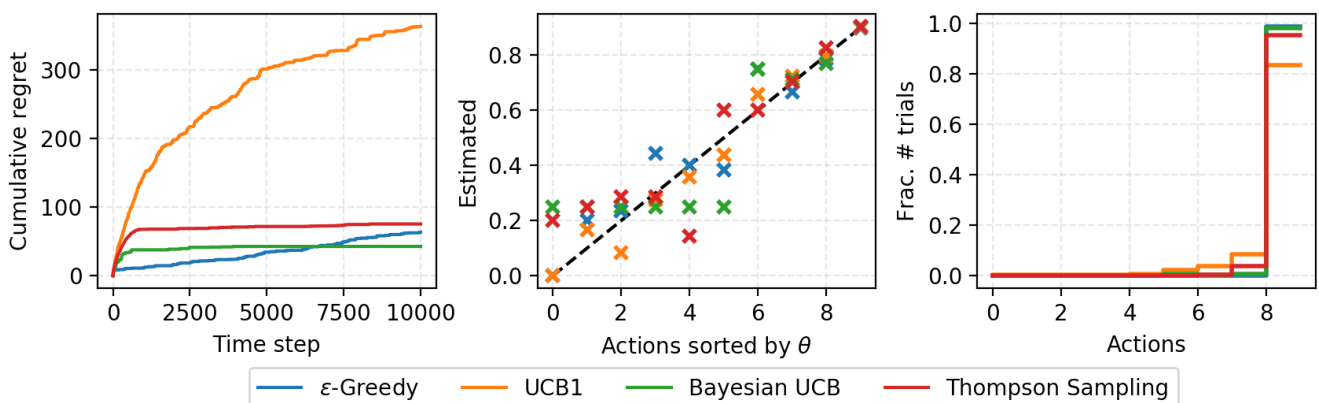


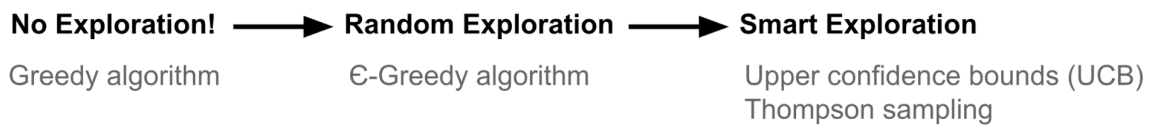
Fig. 4. The result of a small experiment on solving a Bernoulli bandit with $K = 10$ slot machines with reward probabilities, $\{0.0, 0.1, 0.2, \dots, 0.9\}$. Each solver runs 10000 steps.

(Left) The plot of time step vs the cumulative regrets. (Middle) The plot of true reward probability vs estimated probability. (Right) The fraction of each action is picked during the 10000-step run.*

Summary

We need exploration because information is valuable. In terms of the exploration strategies, we can do no exploration at all, focusing on the short-term returns. Or we occasionally explore

at random. Or even further, we explore and we are picky about which options to explore — actions with higher uncertainty are favored because they can provide higher information gain.



Cited as:

```

@article{weng2018bandit,
  title = "The Multi-Armed Bandit Problem and Its Solutions",
  author = "Weng, Lilian",
  journal = "lilianweng.github.io",
  year = "2018",
  url = "https://lilianweng.github.io/posts/2018-01-23-multi-armed-bandit/"
}
  
```

References

- [1] CS229 Supplemental Lecture notes: [Hoeffding's inequality](#).
- [2] RL Course by David Silver - Lecture 9: [Exploration and Exploitation](#)
- [3] Olivier Chapelle and Lihong Li. ["An empirical evaluation of thompson sampling."](#) NIPS. 2011.
- [4] Russo, Daniel, et al. ["A Tutorial on Thompson Sampling."](#) arXiv:1707.02038 (2017).

reinforcement-learning

exploration

math-heavy

«

»

A (Long) Peek into Reinforcement Learning

Object Detection for Dummies Part 3: R-CNN Family



