Reinforcement learning

Damoun Ayman
Github repository

November 25, 2020

1 Bandits algorithms

Bandit models first appeared in the 1930s to model gambling. The term bandit comes from gambling where slot machine can be thought as one-armed bandits. In the so-called one-armed bandit problem, a user faces $K \geq 2$ slot machines. Each giving an average reward that the user does not know a priori. For each of these actions, he will select a machine that will maximize his winnings.
Problem: which slot machine should we play at each turn when their payoffs are not necessarily the same and initially unknown?
This kind of problem is far from being unique to gambling. It is present in many situations where we have to choose how to allocate resources sequentially among several options without knowing the outcomes of those options. Below is a list of the most frequent applications:

- **Design of experiments**:this is particularly the case for *clinical tests*, where the effectiveness of treatments is not known and the only way to find out is to test them on real patients.
- **Website Optimization**: online ad placement, web page personalization [6].
- **Cognitive radio**: is a radio that can be programmed and configured dynamically to use the best wireless channels [4]
- **Network Routing**: Routing is the mechanism by which paths are selected in a network to route data from a sender to one or more recipients. Allocation of channels to the right users, such that the overall throughput is maximised, can be formulated as a Multi-Armed Bandit Problem (MABP).
- **RecommenderSystems**

1.1 Stochastic Bandits

The focus here is on stochastic problems.

> **Definition 1.1.** *Formal definition*
>
> - *single state S={s}*
> - *A: set of actions (known as **arms**)*
> - *space of rewards (often re-scaled to be [0,1])*

No trasition fonction to be learned since there is a single state. We need to learn the *stochastic reward fonction*.
Let us consider K arms (actions) defined by distributions $(\nu_k)_{1 \leq k \leq K}$ with values in [0, 1], of unknown law. At each moment, the agent chooses an arm $A_t \in 1, \ldots$ K and observes a reward $r_k(t)$, independent of past rewards , generated according to the law of the arm A_t. Note $\mu_k = E[A_k(t)]$ the expectation of the k-arm. The best arm is the one for which the expectation is the highest. The best arm is noted with a star:

$$k^\star = \arg\max_{1 \leq k \leq K} \mu_k$$
$$\mu^\star = \mu_{k^\star} = \max_{1 \leq k \leq K} \mu_k$$

The most popular performance measure for bandit algorithms is the total regret, defined at n as:

$$R_n = n\mu^\star - \sum_{t=1}^{n} r_t$$

which represents the difference in cumulative rewards between what the agent got and what he could have gotten on average if he had played optimally from the beginning. The focus is on defining strategies that have a small average cumulative regret.

$$\mathbb{E}[R_n] = n\mu^\star - \mathbb{E}[\sum_{t=1}^{n} \mu_{I_t}] = \mathbb{E}\sum_{k=1}^{K} T_k(n)\,(\mu^\star - \mu_k) = \mathbb{E}\sum_{k=1}^{K} T_k(n)\Delta_k$$

where $\Delta_k = \mu^\star - \mu_k$ is the gap between the optimal arm and the k-arm and $T_k(n) = \sum_{t=1}^{n} 1\{I_t = k\}$ is the number of times the arm k has been pulled until the instant n. Therefore, a good bandit algorithm should pull the sub-optimal arms rarely.

1.2 Lai and Robbins

Theorem 1.1. *For any uniformly good algorithm* π :

$$\liminf_n \frac{R^\pi(n)}{\log(n)} \geq \sum_{a \neq a^\star} \frac{\mu_{a^\star} - \mu_a}{KL\,(\mu_,\mu_{a^\star})}$$

where $KL(a,b) = a\log\left(\frac{a}{b}\right) + (1-a)\log\left(\frac{1-a}{1-b}\right)$ (KL *divergence*)

An algorithm satisfying the hypothesis of the Lai and Robbins theorem is uniformly efficient.

2 Presentation of bandit algorithms

2.1 epsilon-Greedy Algorithm

is one of the simplest possible algorithms for trading off exploration and exploitation. In computer science, a greedy algorithm is an algorithm that always takes whatever action seems best at the present moment, even when that decision might lead to bad long term consequences. The ϵ-Greedy algorithm is almost a greedy algorithm because it generally exploits the best available option, but every once in a while the ϵ-Greedy algorithm explores the other available options.

Note: The value of ϵ determines the fraction of the time when the algorithm explores available arms, and exploits the ones that have performed the best historically the rest of the time.

Algorithm 1: `ϵ-greedy pseudo code`

```
Input: Actions and epsilon(ϵ)
p = random()
if p < ϵ then
    pull random action;
else
    pull current best action;
end
```

2.2 UCB Strategy

The Upper Confidence Bound (UCB) algorithm [1] is constructed from the Chernoff/Hoeffding inequality, which quantifies the probability with which the empirical mean of independent realizations of a random

variable deviates from its expectation. UCB pays attention to not only the arms values but also to arms confidence. At each step, UCB algorithm follows the "optimism in face of uncertainty" principle to select thearm with the highest upper confidence bound, defined by :

$$U_{k,t} = \mu_i + \sqrt{\frac{\alpha \log t}{N_i}}$$

where μ_i is the mean of arm i, N_i is the number of times that arm i is played up-to time t, and $\alpha > 0$.
It can be seen that UCB of an arm is composed by the estimated mean reward and the confidence level of the arm.

Algorithm 2: `UCB pseudo code`

```
for t ← 0 to K do
    Play arm A_t = t
end
for t ← N + 1 to T do
    Play arm A_t = arg max_k (U_{k,t})
end
```

2.3 KL UCB "Kullback-Leibler UCB"

Theorem 2.1. *the binary Kullback-Leibler divergence of x and y is defined as the KL of two Bernoulli laws of means x and y, which is defined as :*

$$\mathrm{kl}(x, y) := x \log\left(\frac{x}{y}\right) + (1 - x) \log\left(\frac{1 - x}{1 - y}\right).$$

Instead of using an upper bound of a confidence interval around this mean, the Kullback-Leibler pseudo-distance is used to obtain the optimal confidence interval [3]:

$$U_{k,t} = \sup_{q \in [0,1]} \left\{ q : \mathrm{kl}(\hat{\mu}_k(t), q) \leq \frac{f(t)}{N_k(t)} \right\}.$$

Algorithm 3: `KL-UCB pseudo code`

Require: Time horizon T, non-decreasing function $f : \mathbb{N} \to \mathbb{R}$
for $t \leftarrow 1$ **to** K **do**
    Select $I_t = t$
end
for $t \leftarrow K + 1$ **to** T **do**
    Select

$$I_t \in \underset{i \in \{1,\ldots,K\}}{\operatorname{argmax}} \{U_i(t)\}$$

    with

$$U_i(t) \overset{\text{def}}{=} \sup \left\{ \mu \in \bar{I} : d\left(\widehat{\mu}_i(t), \mu\right) \leqslant \frac{f(t)}{N_{i,t}} \right\}$$

end

2.4 Thompson sampling

It is a randomized algorithm [3] based on Bayesian inference. The reward is modeled using an a priori distribution associated with each arm. At each time step t, a draw is made in each distribution and the arm with the highest draw is played. The reward obtained is then used to update the distribution of the played arm.During the first rounds of play, the distributions will be close to the uniform before concentrating around the empirical average of each arm.

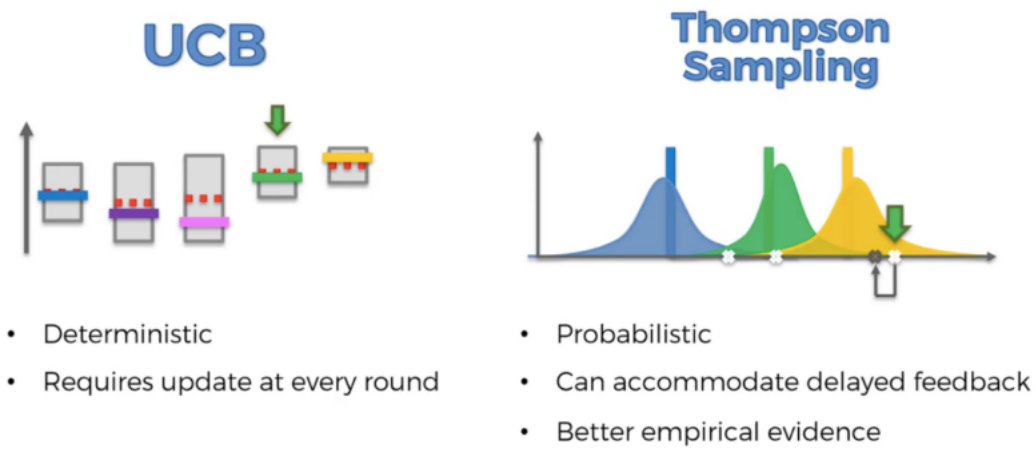

Figure 1: UCB compared to Thompson Sampling

Algorithm 4: `Thompson Sampling pseudo code`

Input: $X, K = |X|$
$t \leftarrow 1$
$\forall x \in X$: Success $_x \leftarrow 0$
$\forall x \in X$: Fail $s_x \leftarrow 0$
while $t \leq T$ **do**
  $\forall x \in X : \Theta_{x,t} \sim$Beta(Success$_x + 1$,Fail $s_x + 1$)
  $x_t \leftarrow \text{argmax}\left(\Theta_{x,t}\right)$ play arm x_t and acquire u_t
  Success$_{x_t} \leftarrow$Success$_{x_t} + u_t$
  Fails$_{x_t} \leftarrow$Fails$_{x_t} + (1 - u_t)$
end

2.5 Best Empirical Sampled Average (BESA)

The BESA algorithm [2] is based on sub-sampling. In order to obtain a fair comparison between the different arms, the observations of each arm are sub-sampled so that the averages are calculated with the same number of observations. Contrary to Thompson Sampling, BESA is non-parametric and therefore more robust.

Algorithm 5: `BESA pseudo code for two arms`

Require: Two arms a, b, current time t
1: Sample $I_t^a \sim \text{Wr}\left(N_t(a); N_t(b)\right)$ and $I_t^b \sim \text{Wr}\left(N_t(b); N_t(a)\right)$
2: Define $\tilde{\mu}_{t,a} = \widehat{\mu}\left(X^a_{1:N_t(a)}\left(I_t^a\right)\right)$ and $\tilde{\mu}_{t,b} = \widehat{\mu}\left(X^b_{1:N_t(b)}\left(I_t^b\right)\right)$
3: Choose (break ties by choosing the arm with the smaller N_t)

$$a_t = \underset{a' \in \{a,b\}}{\text{argmax}} \tilde{\mu}_{t,a'}$$

Algorithm 6: `BESA pseudo code for a multi-armed bandit`

Require: Set of arms $\mathcal{A}$ of size A, current time t
if $\mathcal{A} = \{a\}$ **then**
  Choose $a_t = a$
else
  Choose $a_t = \text{BESA}_t\left(\text{BESA}_t\left(\{a_i\}_{1 \leqslant i < \lceil A/2 \rceil}\right), \text{BESA}_t\left(\{a_i\}_{\lfloor A/2 \rfloor < i \leqslant A}\right)\right)$
end

2.6 Softmax algorithm

The Softmax algorithm [5] picks each arm with a probability that is proportional to its average reward. the Actions with greater average rewards are picked with higher probability.

Algorithm 7: `Boltzmann exploration(Softmax) pseudo code`

Given initial empirical means $\mathrm{u}_1(0), \ldots, \mathrm{u}_\mathrm{K}(0)$

$$p_i(t+1) = \frac{e^{u_i(t)/\tau}}{\sum_{j=1}^{K} e^{u_j(t)/\tau}}, i = 1, \ldots, K$$

White τ controls the choice, $\tau \rightarrow$infinity the algorithm will pick arms uniformly at random.

3 strengths and weaknesses

	pros	cons
Epsilon-greedy bandit	Easy to understand and to implement. Will not get stuck in some local optimal state.	More unlikely to miss the best arm. How do you pick the value epsilon? This is a tricky problem to solve and the wrong epsilon value could lead to either 0 exploration or too much exploration. Does not consider confidence interval
UCB1	Once there's enough accumulated data, the algorithm exploits almost all of the time. that UCB obtains greater reward on average than Epsilon-greedy.	Requires update at evrey round.
KL UCB	Chernoff's bound is used to define the upper confidence. KL-UCB is never worse than that of UCB1	The Implementation Difficulty.
Thompson sampling	robustness in performance regardless of arms with close reward averages and arms with big difference in reward averages.	
BESA	Invariant under rescaling of the rewards	The Implementation Difficulty.
Softmax algorithm	finds the best-performing variation quickly	Need to choose τ carefully

References

[1] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning* 47 (May 2002), pp. 235–256. DOI: 10.1023/A:1013689704352.

[2] Akram Baransi, Odalric-Ambrym Maillard, and Shie Mannor. "Sub-sampling for Multi-armed Bandits". In: *Proceedings of the European Conference on Machine Learning* (Sept. 2014), p. 13. URL: https://hal.archives-ouvertes.fr/hal-01025651.

[3] Nicolas Galichet. “Contributions to Multi-Armed Bandits : Risk-Awareness and Sub-Sampling for Linear Contextual Bandits”. Theses. Université Paris Sud - Paris XI, Sept. 2015. URL: https://tel.archives-ouvertes.fr/tel-01277170.

[4] Wassim Jouini et al. “Multi-armed bandit based policies for cognitive radio’s decision making issues”. In: *3rd international conference on Signals, Circuits and Systems (SCS)*. Djerba, Tunisia, Nov. 2009, 6 pages. URL: https://hal-supelec.archives-ouvertes.fr/hal-00421252.

[5] Volodymyr Kuleshov and Doina Precup. “Algorithms for multi-armed bandit problems”. In: *CoRR* abs/1402.6028 (2014). arXiv: 1402.6028. URL: http://arxiv.org/abs/1402.6028.

[6] J.M. White. *Bandit Algorithms for Website Optimization: Developing, Deploying, and Debugging*. O’Reilly Media, 2012. ISBN: 9781449341589. URL: https://books.google.fr/books?id=xnAZLjqGybwC.