Multi-path Bandit Models

Kartik Nair Variar

Indian Institute of Technology, Bombay

18d070051@iitb.ac.in

December 3, 2021

Overview

What is a Bandit?

- A bandit is an old-fashioned lever-operated slot machine. The lever is called an arm, which we pull, to receive a reward according to the number displayed on screen. Of course, we need to pay money to play.
- The lesser the reward, the more we “regret” playing.
- In general, we extend this idea to bandits with k arms.
- We would like to pick the arms that minimize our regret over time.

Figure: A bandit

Why Learn About Bandits?

Decision making in uncertainty is a challenge that we face all the time and bandits provide a simple model of this dilemma. Bandit problems were introduced by William R. Thompson while he was looking into 'cruelty of running a clinical trial blindly'. Other practical applications include:

- **Recommendation Systems:** Out of all videos available, Which video should YouTube recommend to its users?
- **Dynamic Pricing:** What is the optimum price of a certain product so as to maximize profit?
- **Advert Placement:** Where does a website place its adverts so that more viewers click on them without reducing its traffic?

to name a few...

The Language of Bandits

Bandit Problem

A bandit problem is a sequential game between a **learner** and an **environment**. The game is played over **n** rounds called **horizon**.

- For round $t \in [n]$, the learner chooses **action** $A_t \in \mathcal{A}$ (**action set**) and the environment returns a **reward** $X_t \in \mathbb{R}$, correspnding to A_t.
- if, in each round, the number of possible actions is **k**, then our bandit is called a **k-armed bandit**.
- Intuitively, A_t should depend on the **history** which is the sequence $H_{t-1} = (A_1, X_1, ..., A_{t-1}, X_{t-1})$.
- A learner adopts a **policy** to interact with the environment, which maps the histories to the actions.

The Language of Bandits

- An environment maps the history sequences ending in actions, to rewards.
- A common objective of the learner is to maximize the **cumulative reward**, $\sum_{t=1}^{n} X_t$.
- The environment is generally unknown to the learner but in most cases, the environment lies in some set $\mathcal{E}$, called the **environment class**. (We consider the environment to be 1-subgaussian during our discussion on the algorithms.)
- The **regret** of a learner relative to a policy π is the difference between the total expected reward using π and the expected reward collected by the learner over n rounds.
- The **worst-case regret** is the maximum regret over all environments in $\mathcal{E}$.

Stochastic Bandits

Stochastic Bandit

A stochastic bandit is a collection of distributions $\nu = (P_a : a \in \mathcal{A})$.

The learner and the environment interact sequentially over n rounds such that, for $t \in [n]$, the learner chooses action $A_t \in \mathcal{A}$ and the environment **samples** a reward $X_t \sim P_{A_t}$. From this we observe,

a) $P_{A_t|A_1,X_1,\ldots,A_{t-1},X_{t-1}} = P_{A_t}$
b) The learner cannot use future observations while making current decisions.

In general, the learning objective is to maximize $\mathbf{S_n} = \sum_{t=1}^{n} X_t$

Challenges

- The issue with stochastic bandit problems is that the learner does not know the distributions that govern each arm.
- It also need not know the horizon before hand.
- As S_n is a random variable, we cannot treat a stochastic bandit problem as a simple optimization problem, we need to assign a utility on the distribution of S_n.

To solve the problem of utility, we shall choose the policy with the highest **expected** value of S_n, though it may vary from application to application.

Even if we know n and our utility, we still don't know anything about the environment, $\nu = (P_a : a \in \mathcal{A})$. The learner usually has partial information about the environment, i.e, $v \in \mathcal{E}$ or the environment belongs to the environment class.

Unstructured and Structured Bandits

Unstructured Bandit

If $\mathcal{A}$ is finite and $\exists$ a set of distributions $\mathcal{M}_a$ for all actions $a \in \mathcal{A}$ such that,

$$\mathcal{E} = \{\nu = (P_a : a \in \mathcal{A}) | P_a \in \mathcal{M}_a, \forall a \in \mathcal{A}\}$$

In short $\mathcal{E} = \times_{a \in \mathcal{A}} \mathcal{M}_a$. A bandit problem is usually called a "distribution" bandit based on the set of distributions $\mathcal{M}_a$ for all $a \in \mathcal{A}$.
Examples: Bernoulli, Gaussian, subgaussian etc...

Structured Bandit

If the $\mathcal{M}_a$'s are the same for all $a \in \mathcal{A}$, then the bandit is called structured.

Regret and Regret Decomposition

Let $\nu = (P_a : a \in \mathcal{A})$ be a stochastic bandit, we define

$$\mu_a(\nu) = \int_{-\infty}^{\infty} x P_a(x) dx$$
$$\mu_a^*(\nu) = \max_{a \in \mathcal{A}} \mu_a(\nu)$$

Regret

Assuming that $\mu_a(\nu)$ exists and is finite for all $a \in \mathcal{A}$. The **regret** of a policy π on bandit ν is

$$R_n(\pi, \nu) = n\mu_a^*(\nu) - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right]$$

where $\mathbb{E}\left[\sum_{t=1}^{n} X_t\right]$ is with respect to the probability measure induced by π acting on ν.

Regret and Regret Decomposition

We notice that on minimizing regret, we maximize S_n.

Lemma

Let ν be a stochastic bandit environment, then.

a) $R_n(\pi, \nu) \geq 0, \forall \pi$

b) *The policy π of choosing $A_t \in argmax_a \mu_a, \forall t$ satisfies $R_n(\pi, \nu) = 0$*

c) *If $R_n(\pi, \nu) = 0$, then $\mathbb{P}(\mu_{A_t} = \mu_a^*) = 1, \forall t \in [n]$.*

Proof: Intuitive.

Revised Objective:

1. $\forall \nu \in \mathcal{E}$, $\lim_{n \to \infty} \frac{R_n(\pi,\nu)}{n} = 0$ (Sub-linearity).
   Even stronger objectives are,
2. $\forall \nu \in \mathcal{E}$, $R_n(\pi, \nu) \leq Cn^p$ for some $C > 0, p < 1$
3. $\forall \nu \in \mathcal{E}$, $R_n(\pi, \nu) \leq C(\nu) f(n)$ for some $C : \mathcal{E} \to [0, \infty)$ and $f : \mathbb{N} \to [0, \infty)$ such that C and f are as small as possible.

Decomposing the Regret

Let $\nu = (P_a : a \in \mathcal{A})$ be a stochastic bandit, we define $\Delta_a(\nu) = \mu^*(\nu) - \mu_a(\nu)$ called the **sub-optimality gap** or the **action gap** or **immediate regret** of action a. $T_a(t) = \sum_{s=a}^{t} \mathbb{I}\{A_s = a\}$ is the number of times action a was chosen after t rounds.

Lemma (Regret Decomposition)

For any policy π and stochastic bandit environment ν with finite $\mathcal{A}$ and horizon $n \in \mathbb{N}$,

$$R_n = \sum_{a \in \mathcal{A}} \Delta_a \mathbb{E}\left[T_a(n)\right]$$

Decomposing the Regret (Contd.)

Proof:

$$\begin{aligned}
R_n = n\mu^* - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right] &= \sum_{t=1}^{n} \mathbb{E}[\mu^* - X_t] \\
&= \sum_{a\in\mathcal{A}} \sum_{t=1}^{n} \mathbb{E}[\mathbb{I}\{A_t = a\}(\mu^* - X_t)] \\
&= \sum_{a\in\mathcal{A}} \sum_{t=1}^{n} \mathbb{E}[\mathbb{I}\{A_t = a\}]\mathbb{E}[(\mu^* - X_t)|\mathbb{I}\{A_t = a\}] \\
&= \sum_{a\in\mathcal{A}} \Delta_a \mathbb{E}\left[T_a(n)\right]
\end{aligned}$$

Hence Proved.

Canonical Bandit Model[1] (Finite Horizon)

A policy on interaction with the environment produces a tuple of random variables $H_n = (A_1, X_1, ..., A_n, X_n)$. We want to construct a probability space that carries these random variables. For each $t \in [n]$, let $\Omega_t = ([k] \times \mathbb{R})^t$ and $\mathcal{F}_t = \mathcal{B}(\Omega_t)$, then

Define: A policy $(\pi_t)_{t=1}^n$ is such that, π_t is a probability kernel of $(\Omega_{t-1}, \mathcal{F}_{t-1}) \to ([k], 2^{[k]})$ such that,

a) $A_t \sim \pi_t(.|A_1, X_1, ..., A_{t-1}, X_{t-1})$, almost surely.

b) $X_t|A_1, X_1, ..., A_{t-1}, X_{t-1}, A_t \sim P_{A_t}$, almost surely.

Canonical Bandit Model

Let $p_{\nu\pi}$ be our probability measure with respect to distribution, on $(\Omega_n, \mathcal{F}_n)$, corresponding to our canonical bandit, then we can define it as

$$p_{\nu\pi}(a_1, x_1, ..., a_n, x_n) = \prod_{t=1}^{n} \pi_t(a_t|a_1, x_1, ..., a_{t-1}, x_{t-1}) p_{a_t}(x_t)$$

[1] Not understood completely.

Stochastic Bandit Algorithms

Until now, we have looked at the basics of Bandits, their language and the definitions of some important attributes of bandit models and policies. Now we look at **bandit algorithms** that implement certain basic policies and analyze the regret of these algorithms. In the following slides, we assume the environment class to be **1-subgaussian** ($\mathcal{E}_{SG}^{k}(1)$). A detailed analysis of subgaussian random variable is given in the **References**.

We shall discuss the following algorithms in detail,

- **The Explore-Then-Commit Algorithm**
- **Upper Confidence Bounds** (Specifically $UCB(\delta)$)

These algorithms can be extended to other environment classes as well, but the approach to regret analysis would change.

The Explore-Then-Commit Algorithm

The **ETC** algorithm, as the name suggests, first **explores** all the arms for a certain number of rounds and then **commits** to the arm whose mean reward is the highest. The question is for how many rounds should our algorithm explore? Thus, the ETC algorithm is characterized by $m \in \mathbb{N}$, which is the number of times it explores each arm. As there are k actions, the algorithm explores for mk rounds. We denote the **average reward** received for arm i after t rounds as $\hat{\mu}_i(t)$, given by

$$\hat{\mu}_i(t) = \frac{1}{T_i(t)} \sum_{s=1}^{t} \mathbb{I}\{A_s = i\} X_s$$

The algorithm is given in the next slide.

The Explore-Then-Commit Algorithm

Algorithm 1: `ETCAlgorithm`

```
Input: m
for $t \in [n]$ do
    if $t \leq mk$ then
        $A_t = (t \mod k) + 1$ ;
        $X_t$ sampled by the environment from $P_{A_t}$ ;
        $\hat{\mu}_{A_t} = T_{A_t}\hat{\mu}_{A_t}$ ;
        $T_{A_t} = T_{A_t} + 1$ ;
        $\hat{\mu}_{A_t} = \frac{\hat{\mu}_{A_t} + X_t}{T_{A_t}}$ ;
    end
    $A_t = argmax_{a \in \mathcal{A}}(\hat{\mu_a})$ ;
    $X_t$ sampled by the environment from $P_{A_t}$ ;
end
return $H_n$ (History over n rounds) ;
```

The Explore-Then-Commit Algorithm (Regret Analysis)

Theorem (ETC regret analysis)

When ETC interacts with a 1-subgaussian bandit and $1 \leq m \leq n/k$*,*

$$R_n \leq m \sum_{i=1}^{k} \Delta_i + (n - mk) \sum_{i=1}^{k} \Delta_i \exp\left(-\frac{m\Delta_i^2}{4}\right)$$

Proof: WLOG, we assume that the first arm is optimal, i.e., $\mu_1 = \mu^* = \max_i \mu_i$. We know that $R_n = \sum_{i=1}^{k} \Delta_i \mathbb{E}[T_i(n)]$. In the first mk rounds, each arm is chosen m times, therefore,

$$\begin{aligned} \mathbb{E}[T_i(n)] &= m + (n - mk)\mathbb{P}(A_{mk+1} = i) \\ &\leq m + (n - mk)\mathbb{P}(\hat{\mu}_i(mk) \geq max_{a \in \mathcal{A}} \hat{\mu}_a(mk)) \end{aligned}$$

The Explore-Then-Commit Algorithm (Regret Analysis)

Observing that,

$$\begin{aligned}\mathbb{P}(\hat{\mu}_i(mk) \geq max_{a\in\mathcal{A}}\hat{\mu}_a(mk)) &\leq \mathbb{P}(\hat{\mu}_i(mk) \geq \hat{\mu}_1(mk)) \\ &\geq \mathbb{P}(\hat{\mu}_i(mk) - \mu_i - (\hat{\mu}_1(mk) - \mu_1) \geq \Delta_i)\end{aligned}$$

For 1-subgaussian random variables, we know that $\hat{\mu}_i(mk) - \mu_i - (\hat{\mu}_1(mk) - \mu_1) \sim \text{subG}(\sqrt{\frac{2}{m}})$, from which we can conclude that

$$\mathbb{P}(\hat{\mu}_i(mk) - \mu_i - (\hat{\mu}_1(mk) - \mu_1) \geq \Delta_i) \leq \exp\left(-\frac{m\Delta_i^2}{4}\right)$$

Substituting back into the inequality of $\mathbb{E}[T_i(n)]$ and then back into R_n, we get,

$$R_n \leq m\sum_{i=1}^{k}\Delta_i + (n - mk)\sum_{i=1}^{k}\Delta_i \exp\left(-\frac{m\Delta_i^2}{4}\right)$$

Hence Proved.

The Explore-Then-Commit Algorithm (Experiment)

Assuming $k = 2$, and $\Delta_1 = 0$, $\Delta_2 = \Delta$, the bound on R_n for large n simplifies to,

$$R_n \leq m\Delta + (n - 2m)\Delta \exp\left(\frac{-m\Delta^2}{4}\right) \leq m\Delta + n\Delta \exp\left(\frac{-m\Delta^2}{4}\right)$$

Finding the **optimal m** by minimizing the upper bound,

$$m = \max\left\{1, \lceil \frac{4}{\Delta^2} \log\left(\frac{n\Delta^2}{4}\right) \rceil \right\}$$

and so, the **worst-case regret** for the optimal m is given by,

$$R_n \leq \min\left\{n\Delta, \Delta + \frac{4}{\Delta}\left(1 + \max\left\{0, \log\left(\frac{n\Delta^2}{4}\right)\right\}\right)\right\}$$

We shall see the results of our experiment in the next slide.

The Explore-Then-Commit Algorithm (Experiment)

The following plot shows the result of varying Δ from 0 to 1, keeping the horizon n as 1000 rounds, for optimal m. We see that the regret is almost always less than the theoretical upper bound or the worst-case regret. Thus, it can be shown that $R_n = \mathcal{O}(\sqrt{n})$, which is sub-linear, when m is optimally chosen.

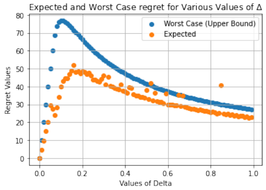

Figure: Regret and worst-case Regret for the ETC algorithm

The problem with ETC is the requirement of prior knowledge of the horizon n and Δ to find the optimal m. We shall see that $UBC(\delta)$ is independent of the knowledge of Δ.

Upper Confidence Bound Algorithms ($UCB(\delta)$)

- We saw earlier, that one of the issues of the ETC algorithm was that it depended on the prior knowledge of the sub-optimality gaps. The UCB algorithm does not depend on the prior knowledge of sub-optimality gaps.
- The UCB algorithm is based on the principle of **optimism in the face of uncertainty**.
- The question now is just how optimistic should we be?

In the case of bandits, we assign to each arm a value, called the **upper confidence bound**, which is an overestimate of the unknown mean with high probability.

Using Chernoff bound on a sequence of 1-subgaussin random variables given by $(X_t)_{t=1}^n$ with mean μ and empirical mean $\hat{\mu}$, we can show that

$$\mathbb{P}\left(\mu \geq \hat{\mu} + \sqrt{\frac{2\log(1/\delta)}{n}}\right) \leq \delta, \forall \delta \in (0,1)$$

Upper Confidence Bound Algorithms ($UCB(\delta)$)

Using the above, we can define the **UCB** for arm i at time $t-1$ for a certain **confidence level** δ as follows,

$$UCB_i(t-1,\delta) = \infty \text{ if } T_i(t-1) = 0$$
$$UCB_i(t-1,\delta) = \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}} \text{ otherwise.}$$

The intuition is that δ is an upper bound on the probability of the event that the above UCB is an underestimate of the true mean.

Now, we give an intuition for $UCB(\delta)$ algorithm. An algorithm should explore an arm if and only if,

a) The arm is promising.

b) The arm hasn't been explored many times.

We shall see from the following pseudo-code that the $UCB(\delta)$ algorithm satisfies the above two.

Upper Confidence Bound Algorithms ($UCB(\delta)$)

Algorithm 2: `UCBAlgorithm`

```
Input: k and $\delta$

for $t \in [n]$ do
    $A_t = argmax_i UCB_i$ ;
    $X_t$ is sampled from the environment ;
    $\hat{\mu}_{A_t} = T_{A_t}\hat{\mu}_{A_t}$ ;
    $T_{A_t} = T_{A_t} + 1$ ;
    $\hat{\mu}_{A_t} = \frac{\hat{\mu}_{A_t}+X_t}{T_{A_t}}$ ;
    $UCB_{A_t} = \hat{\mu}_{A_t} + \sqrt{\frac{2\log(1/\delta)}{T_{A_t}}}$ ;
end
return $H_n$ (History over n rounds) ;
```

Here UCB_i in the argmax is called the **index** of arm i.The addition term to the empirical mean is called the **confidence width** or the **exploration bonus**.

Upper Confidence Bound Algorithms (Regret Analysis)

Choosing δ is a difficult problem, we shall take $\delta = 1/n^2$ while analysing our regret.

Theorem ($UCB(\delta)$ regret analysis)

When UCB(δ) interacts with a k-armed 1-subgaussian bandit such that for any horizon n, $\delta = 1/n^2$, then

$$R_n \leq 3\sum_{i=1}^{k} \Delta_i + \sum_{i:\Delta_i>0} \frac{16\log(n)}{\Delta_i}$$

Proof: The proof is complicated. A pdf containing a detailed proof is provided in the references.
The above bound can be proved to be sub-linear. (Proof in the pdf.)

$$R_n \leq 8\sqrt{nk\log(n)} + 3\sum_{i=1}^{k} \Delta_i$$

Upper Confidence Bound Algorithms (Experiment)

The parameters for the experiment are the same as those in ETC experiment except that we compare the expected regret of the ETC algorithm with various values of m vs that of the UCB(δ) algorithm.

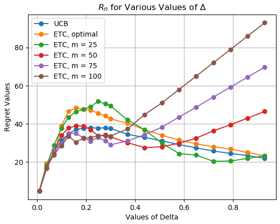

Figure: Performance of UCB vs ETC for various values of Δ

We observe that UCB performs better in terms of minimizing regret as compared to ETC in general. [2]

[2] In the book, the performance of optimal ETC turned out to be better than UCB but in my implementation, UCB was better. This might be due to a more optimal choice of m. I followed the equation in slide 20.

What Next?

- There are more efficient implementations of stochastic bandit algorithms, in fact more efficient UCB algorithms.
- We have looked at a special case of subgaussian bandits but we can look at similar ideas to the above presented for other types of environment classes, like Bernoulli bandits.
- Next, we can abandon all assumptions on the environment class. This leads us to adverserial bandits.
- Reinforcement learning...

References

The GitHub link given below contains all the relevant references.
Link to Bandits GitHub

The proof for UCB regret analysis can be found here.
Structure of repository:

- ETC algorithm implementation.
- References.
- UBC vs ETC implementation
- Package containing bandit classes, policy classes and algorithm classes.

The End