Bandit Algorithms

Learning Under Uncertainty

Antonio Coppola

ancoppola@unisa.it

April 10, 2022

The Bandit Algorithms

Posted Price Auctions

- Suppose that you have a huge amount of shoes to sell

Posted Price Auctions

- Suppose that you have a huge amount of shoes to sell
- You decide to organize a street market for selling the shoes

Posted Price Auctions

- Suppose that you have a huge amount of shoes to sell
- You decide to organize a street market for selling the shoes
- You post the price 20€ and sell a pair of shoes to any customer that are willing to pay the posted price. Each customer can buy only a pair of shoes

Posted Price Auctions

- Suppose that you have a huge amount of shoes to sell
- You decide to organize a street market for selling the shoes
- You post the price 20€ and sell a pair of shoes to any customer that are willing to pay the posted price. Each customer can buy only a pair of shoes
- Each customer that has a private valuation greater or equal than 20€ buys the shoes

Posted Price Auctions

- Suppose that you have a huge amount of shoes to sell
- You decide to organize a street market for selling the shoes
- You post the price 20€ and sell a pair of shoes to any customer that are willing to pay the posted price. Each customer can buy only a pair of shoes
- Each customer that has a private valuation greater or equal than 20€ buys the shoes

The goal is to maximize the revenue

Posted Price Auctions

- Suppose that you have a huge amount of shoes to sell
- You decide to organize a street market for selling the shoes
- You post the price 20€ and sell a pair of shoes to any customer that are willing to pay the posted price. Each customer can buy only a pair of shoes
- Each customer that has a private valuation greater or equal than 20€ buys the shoes

The goal is to maximize the revenue
You want sell the shoes at highest possible price, but a too high price might make you earn nothing

Repeated Posted Price Auctions

- Suppose now that you know an important customer that every Monday asks you the price of a pair of shoes
- Each week the customer can change its valuation of the shoes and you can change the price

Repeated Posted Price Auctions

- Suppose now that you know an important customer that every Monday asks you the price of a pair of shoes
- Each week the customer can change its valuation of the shoes and you can change the price
- Every Monday:
    1. The customer has its own valuation

Repeated Posted Price Auctions

- Suppose now that you know an important customer that every Monday asks you the price of a pair of shoes
- Each week the customer can change its valuation of the shoes and you can change the price
- Every Monday:
  1. The customer has its own valuation
  2. You post the price for a pair of shoes

Repeated Posted Price Auctions

- Suppose now that you know an important customer that every Monday asks you the price of a pair of shoes
- Each week the customer can change its valuation of the shoes and you can change the price
- Every Monday:
  1. The customer has its own valuation
  2. You post the price for a pair of shoes
  3. If the posted price is lower or equal than customer's valuation, then you sell the pair of shoes at your price

Repeated Posted Price Auctions

- Suppose now that you know an important customer that every Monday asks you the price of a pair of shoes
- Each week the customer can change its valuation of the shoes and you can change the price
- Every Monday:
  1. The customer has its own valuation
  2. You post the price for a pair of shoes
  3. If the posted price is lower or equal than customer's valuation, then you sell the pair of shoes at your price

  Every Monday you want to post a price to maximize the cumulated revenue

Repeated Posted Price Auctions: Formalization

- For simplicity we consider a two-player game between the seller and the buyer
- At each time step t the seller wants to sell the good to the buyer
- For each time step $t \in \{1, 2, \dots, T\}$:
  1. Buyer arrives with (hidden) valuation $v_t \in [0, P]$
  2. Seller sets the price $\pi_t \in [0, P]$
  3. If $\pi_t \leq v_t$, then the buyer buys the good and pays π_t

  The Seller's Goal is to maximize the **cumulated** revenue $\sum_{t=1}^{T} r_t = \sum_{t=1}^{T} \pi_t \cdot \mathbb{1}\{\pi_t \leq v_t\}$

Repeated Posted Price Auctions: A Classic Dilemma

Let's back to our example.

You are the Seller and you can post only two prices $\{10€, 20€\}$ for your pair of shoes. You are running the Repeated Posted Price Auction and you have already posted each price 5 times:

Step	1	2	3	4	5	6	7	8	9	10
Posted Price = 10	10€		10€	0		0				10€
Posted Price = 20		20€			0		0	0	0	

- Cumulated revenue for $\{\text{Price} = 10\} = 30€$
- Cumulated revenue for $\{\text{Price} = 20\} = 20€$

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

- Will you keep posting Price = 10€ , ignoring Price = 20€

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

- Will you keep posting Price = 10€ , ignoring Price = 20€
- Would you attribute the poor performance of Price = 20€ to bad luck and try it a few more times?

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

- Will you keep posting Price = 10€ , ignoring Price = 20€
- Would you attribute the poor performance of Price = 20€ to bad luck and try it a few more times? How many times?

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

- Will you keep posting Price = 10€ , ignoring Price = 20€ $\rightarrow$ Exploitation
- Would you attribute the poor performance of Price = 20€ to bad luck and try it a few more times? How many times?

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

- Will you keep posting Price $=$ 10€ , ignoring Price $=$ 20€ $\rightarrow$ Exploitation
- Would you attribute the poor performance of Price $=$ 20€ to bad luck and try it a few more times? How many times? $\rightarrow$ Exploration

Repeated Posted Price Auctions: A Classic Dilemma

You have 10 more trials altogether. What is your strategy?

- Will you keep posting Price $= 10€$, ignoring Price $= 20€ \rightarrow$ Exploitation
- Would you attribute the poor performance of Price $= 20€$ to bad luck and try it a few more times? How many times? $\rightarrow$ Exploration

Finding the right balance between exploration and exploitation is at the heart of all **Bandit** algorithms.

What are Bandits Algorithms?

What are Bandits Algorithms?

"Bandits is a simple but very powerful framework for algorithms that make decisions over time under uncertainty"

Aleksandrs Slivkins. *Introduction to Multi-Armed Bandits*. Foundations and Trends in Machine Learning, 2019

Some History

- Bandit problems were introduced by William R. Thompson in 1933
- The name comes from the 1950s, when Frederick Mosteller and Robert Bush decided to study how animals learn online
- Why Bandit? In first studies, a 'two-armed bandit' was used to model how humans learn online.

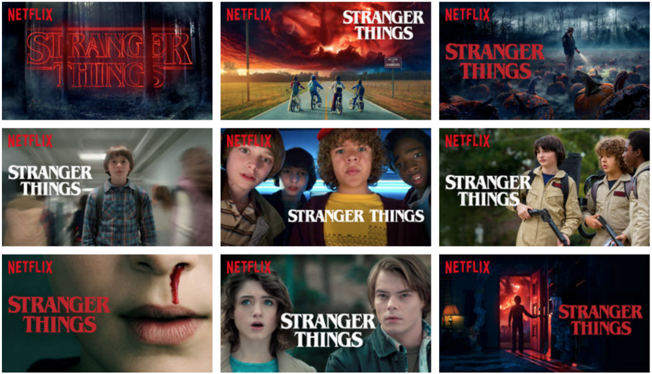
Bandit Nowadays

- Bandit Algorithms are used by major tech (e.g. Amazon[1], Zillow[2]) for configuring web interfaces:
  - News Recommendation
  - Dynamic Pricing
  - Ad Placement

[1]Daniel N. Hill et al. "An Efficient Bandit Algorithm for Realtime Multivariate Optimization". In: *CoRR* abs/1810.09558 (2018). arXiv: 1810.09558. URL: http://arxiv.org/abs/1810.09558.

[2]Luca Cazzanti. *Contextual, multi-armed bandit performance assessment*. Zilow. 2020. URL: https://www.zillow.com/tech/multi-armed-bandits/.

Netflix Artwork Personalization

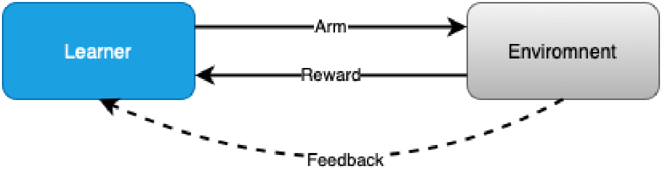


Ashok Chandrashekar et al. *Artwork Personalization at Netflix*. Netflix. 2017. URL: https://netflixtechblog.com/artwork-personalization-c589f074ad76

Bandit Nowadays

- Bandit Algorithms are used by major tech (e.g. Amazon, Zillow) for configuring web interfaces:
  - News Recommendation
  - Dynamic Pricing
  - Ad Placement
- Bandit Algorithms can be used to address emerging problems in different disciplines:
  - Auctions
  - Diffusion Mechanisms

The Language of Bandits



- The bandit problem is a sequential game between a learner (or an algorithm) and an environment
- The game is played over T rounds, T is called the horizon
- At each round the learner chooses an arm a_t collects a reward (or cost) and receives a feedback from the environment.
- With K arms we refer to the Bandit Problems as K-Armed Bandit. In general, with more than two arms we use the term Multi-Armed Bandit
- The learner's goal is to maximize its total reward over the T rounds.

The Language of Bandits

The Bandits Problem

Require: K arms, T rounds

```
for t ∈ [T] do
    Algorithm chooses some arm a_t ∈ [K]
    Environment determines reward r_t
    Algorithm receives r_t and a feedback
end for
```

Repeated Posted Price Auctions as Bandit Problems

Repeated Posted Price Auctions as Bandit Problems

- Seller

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price $\rightarrow$ Arm

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price $\rightarrow$ Arm
- Revenue

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price $\rightarrow$ Arm
- Revenue $\rightarrow$ Reward

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price $\rightarrow$ Arm
- Revenue $\rightarrow$ Reward
- Has the good been sold?

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price $\rightarrow$ Arm
- Revenue $\rightarrow$ Reward
- Has the good been sold? $\rightarrow$ Feedback

Repeated Posted Price Auctions as Bandit Problems

- Seller $\rightarrow$ Learner
- Buyer $\rightarrow$ Environment
- Posted Price $\rightarrow$ Arm
- Revenue $\rightarrow$ Reward
- Has the good been sold? $\rightarrow$ Feedback
- Cumulated Revenue

Repeated Posted Price Auctions as Bandit Problems

- Seller → Learner
- Buyer → Environment
- Posted Price → Arm
- Revenue → Reward
- Has the good been sold? → Feedback
- Cumulated Revenue → Total Reward over the T rounds

Repeated Posted Price Auctions as Bandit Problems

Bandit Posted Price Auctions

Require: K prices, T rounds

```
for t ∈ [T] do
    Buyer has (hidden) valuation v_t
    Seller posts a price a_t ∈ [K]
    Reward is r_t = a_t if v_t ≥ a_t, otherwise r_t = 0
    Seller receives r_t, and knows if the good has been sold (feedback)
end for
```

Type of Environments

Two type of environments:

- **Stochastic**: Given the arm chosen by the learner, the environment samples the reward from an (unknown) distribution
- **Deterministic Adversarial**: The environment determines the rewards deterministically, independently from the arms chosen by the learner

Posted Price with Stochastic Environment

Let's assume that the price set for the seller is: $\mathcal{P} = \{0, 0.5, 1\}$

- Suppose that, for each time step t, the buyer's valuation v_t follows the following distribution:

$$Pr[v_t = 0] = 0.25; Pr[v_t = 0.7] = 0.6; Pr[v_t = 1.5] = 0.15$$

- We can determine a reward distribution $D(a)$ for each price $a \in \mathcal{P}$. For example, if $a = 0.5$, the reward distribution is:

$$Pr[r = 1] = 0$$

$$Pr[r = 0.5] = Pr[v \geq a] = Pr[v = 0.7] + Pr[v = 1.5] = 0.75$$

$$Pr[r = 0] = Pr[v < a] = Pr[v = 0] = 0.25$$

- The expected reward for $a = 0.5$ is:

$$\mu(a) = Pr_{D(a)}[r = 0] \cdot 0 + Pr_{D(a)}[r = 0.5] \cdot 0.5 + Pr_{D(a)}[r = 1] \cdot 1 = 0.375$$

Posted Price with Deterministic Adversarial Environment

Let's assume that $\mathcal{P} = \{0, 0.5, 1\}$ and $T = 3$.

- For each time step t, the valuation v_t is chosen deterministically and does not depend on the arms chosen by the seller in the previous steps
- Adversarial setting $\rightarrow$ Buyer knows the pricing algorithm used by the seller
- We can think that the buyer determines its valuations before round 1.

Step	1	2	3
v	0.5	0	1

- For each price a the rewards are deterministic. The rewards table is chosen before the game starts

Step	1	2	3
Posted Price = 0	0	0	0
Posted Price = 0.5	0.5	0	0.5
Posted Price = 1	0	0	1

Performance Evaluation

How do we argue whether an algorithm is doing a good job?

Performance Evaluation

How do we argue whether an algorithm is doing a good job?

The problem is that some problem instances inherently allow higher rewards.

Performance Evaluation

How do we argue whether an algorithm is doing a good job?
The problem is that some problem instances inherently allow higher rewards.
Consider the stochastic setting in the slide 18 with $T = 3$.

- The Seller always chooses the price 0.5.
- Different problem instances:
  - The buyer always chooses a valuation equal to 0 $\rightarrow$ Cumulated revenue is 0;
  - The buyer always chooses a valuation equal to 0.7 $\rightarrow$ Cumulated revenue is 1.5
  - ...

Performance Evaluation

How do we argue whether an algorithm is doing a good job?

The problem is that some problem instances inherently allow higher rewards.

A standard approach:

- Compare the cumulative reward (or cost) to the cumulative reward obtained by always playing the optimal arm a^*, called the best-arm benchmark
- We analyze the regret $R(t)$ that measures the difference between the cumulative reward obtained by the bandit algorithm and the best-arm benchmark.
- Let's note $r_\tau(a^*)$ the reward obtained at step τ with the optimal arm a^*, and $r_\tau(a^*)$ the reward obtained at step τ with the arm played by the bandit algorithm at the same step a_τ

$$R(t) = \sum_{i=\tau}^{t} r_\tau(a*) - \sum_{\tau=1}^{t} r_\tau(a_\tau)$$

The Regret

- The regret at round t depends on all the arms chosen up to round t
- One of the core questions in the study of Bandits is to understand the growth rate of the regret as t grows
- Our goal: the difference between the cumulative reward obtained by the bandit algorithm and the best-arm benchmark decreases over time, and goes to zero as t increases.
- Formally, A good learner achieves sub-linear regret. This means that $lim_{t\to\infty}\frac{R(t)}{t} = 0$. Example of sub-linear regret: $R(t) = O(\sqrt{t}), R(t) = O(\log t)$

Note that the arms chosen by the algorithm are random quantities, as they may depend on randomness in rewards and/or in the algorithm. We will typically talk about expected regret $\mathbb{E}[R(T)]$.
In words: *regret measures how much the algorithm "regrets" not knowing the best arm in advance*

Stochastic Bandits

Stochastic Bandits: Problem Structure

- Bandit Feedback: The algorithm observes only the reward for the selected action, and nothing else.
- Reward Distributions: For each action a, there is an unknown distribution D_a called the reward distribution. Every time action a is chosen, the reward is sampled independently from this distribution. The expected reward for a is μ(a)
- Rewards are bounded: For simplicity, we restrict the rewards to the interval $[0, 1]$
- Goal: Maximize (Minimize) the cumulated revenue (cost) $\sum_{t=1}^{T} r_t$

Stochastic Bandits

Problem Protocol: Stochastic Bandits

Require: K arms; T rounds; reward distribution D_a for each arm a (unknown).

1: **for** $t \in [T]$ **do**
2: Algorithm chooses some arm $a_t \in [K]$
3: Reward $r_t \in [0, 1]$ is sampled independently from distribution $D_a, a = a_t$
4: Algorithm collects r_t, and observe nothing else
5: **end for**

Explore-First Algorithm

Idea: Try each arms a fixed number of times, then choose the best one.

Explore-First Algorithm

Idea: Try each arms a fixed number of times, then choose the best one.

Explore-First Algorithm

Require: K arms; T rounds; length of exploration phase for each arm N.

1. **Exploration phase**: try each arm N times
2. Select the arm $\bar{a}$ with the highest average reward (break ties arbitrarily)
3. **Exploitation phase**: play arm $\bar{a}$ in all remaining rounds

Explore-First Algorithm

Idea: Try each arms a fixed number of times, then choose the best one.

Explore-First Algorithm

Require: K arms; T rounds; length of exploration phase for each arm N.

1: **Exploration phase**: try each arm N times

2: Select the arm $\overline{a}$ with the highest average reward (break ties arbitrarily)

3: **Exploitation phase**: play arm $\overline{a}$ in all remaining rounds

Theorem

Explore-first achieves regret

$$\mathbb{E}[\mathrm{R(T)}] \leq T^{2/3} O(K \cdot \log T)^{1/3}$$

Epsilon-greedy algorithm

Problem with Explore-first: the performance in the exploration phase may be very bad if many/most of the arms have a low reward (high cost) with respect to the optimal-arm. It is usually better to spread the exploration of the arms more uniformly over the time steps.

Epsilon-greedy algorithm

Problem with Explore-first: the performance in the exploration phase may be very bad if many/most of the arms have a low reward (high cost) with respect to the optimal-arm. It is usually better to spread the exploration of the arms more uniformly over the time steps.

Epsilon-Greedy Algorithm

Require: K arms, T rounds, ϵ_t.

for each round $t = 1, 2, \ldots, T$ **do**
    Toss a coin with success probability ϵ_t
    if success **then**:
        explore: choose an arm uniformly at random
    else
        exploit: choose the arm with the highest average reward so far
    end if
end for

Epsilon-greedy algorithm: The Regret

We can derive the same regret bound as for Explore-first, but now it holds for all rounds t

Theorem

Epsilon-Greed algorithm achieves regret bound

$$\mathbb{E}[\mathrm{R(t)}] \leq t^{2/3} O(K \cdot \log t)^{1/3}$$

Non-Adaptive Exploration

- Non-Adaptive Exploration: the number of exploration rounds is fixed before the game starts
- All the algorithms showed in the previous slides are Non-Adaptive Algorithms

Claim

For any non-adaptive algorithms we can not do better than $\Omega(T^{2/3}K^{1/3})$

For achieving better performance we must use Adaptive Algorithms.

Adaptive Exploration

- Let $n_t(a)$ the number of times when action the action a was chosen up to the round t. $[n_t(a)]$ is the set of time steps when the action a was chosen up to the round t
- The average reward of a at step t is

$$\overline{\mu}_t(a) = \frac{1}{n_t(a)} \sum_{t \in [n_t(a)]} r_t$$

- The confidence radius of arm a: $rad_t(a) = \sqrt{2 log T / n_t(a)}$
- With high probability $\overline{\mu}_t(a) - rad_t(a) \leq \mu(a) \leq \overline{\mu}_t(a) + rad_t(a)$
- Upper Confidence Bound : $UCB_t(a) = \overline{\mu}_t(a) + rad_t(a)$.
- UCB is an upper bound of $\mu(a)$

Adaptive Exploration

- Let $n_t(a)$ the number of times when action the action a was chosen up to the round t. $[n_t(a)]$ is the set of time steps when the action a was chosen up to the round t
- The average reward of a at step t is

$$\overline{\mu}_t(a) = \frac{1}{n_t(a)} \sum_{t \in [n_t(a)]} r_t$$

- The confidence radius of arm a: $rad_t(a) = \sqrt{2logT/n_t(a)}$
- With high probability $\overline{\mu}_t(a) - rad_t(a) \leq \mu(a) \leq \overline{\mu}_t(a) + rad_t(a)$
- Upper Confidence Bound : $UCB_t(a) = \overline{\mu}_t(a) + rad_t(a)$.
- UCB is an upper bound of $\mu(a)$ with high probability

- Let $n_t(a)$ the number of times when action the action a was chosen up to the round t. $[n_t(a)]$ is the set of time steps when the action a was chosen up to the round t
- The average reward of a at step t is

$$\overline{\mu}_t(a) = \frac{1}{n_t(a)} \sum_{t \in [n_t(a)]} r_t$$

- The confidence radius of arm a: $rad_t(a) = \sqrt{2 log T / n_t(a)}$
- With high probability $\overline{\mu}_t(a) - rad_t(a) \leq \mu(a) \leq \overline{\mu}_t(a) + rad_t(a)$
- Upper Confidence Bound : $UCB_t(a) = \overline{\mu}_t(a) + rad_t(a)$.
- UCB is an upper bound of $\mu(a)$ with high probability
- If $n_t(a) \rightarrow \infty$:

Adaptive Exploration

- Let $n_t(a)$ the number of times when action the action a was chosen up to the round t. $[n_t(a)]$ is the set of time steps when the action a was chosen up to the round t
- The average reward of a at step t is

$$\overline{\mu}_t(a) = \frac{1}{n_t(a)} \sum_{t \in [n_t(a)]} r_t$$

- The confidence radius of arm a: $rad_t(a) = \sqrt{2logT/n_t(a)}$
- With high probability $\overline{\mu}_t(a) - rad_t(a) \leq \mu(a) \leq \overline{\mu}_t(a) + rad_t(a)$
- Upper Confidence Bound : $UCB_t(a) = \overline{\mu}_t(a) + rad_t(a)$.
- UCB is an upper bound of $\mu(a)$ with high probability
- If $n_t(a) \to \infty$: $UCB_t(a) \to \mu(a)$

UCB1

An arm a can have a large $UCB_t(a) = \overline{\mu_{\mathrm{t}}}(\mathrm{a}) + rad_t(a)$ for two reasons:

- $\overline{\mu_{\mathrm{t}}}(\mathrm{a})$ is large:
- $rad_t(a)$ is large:

UCB1

An arm a can have a large $UCB_t(a) = \overline{\mu_{\mathrm{t}}}(\mathrm{a}) + rad_t(a)$ for two reasons:

- $\overline{\mu_{\mathrm{t}}}(\mathrm{a})$ is large: the arm a is likely to have a high reward
- $rad_t(a)$ is large:

UCB1

An arm a can have a large $UCB_t(a) = \overline{\mu_{\mathrm{t}}}(\mathrm{a}) + rad_t(a)$ for two reasons:

- $\overline{\mu_{\mathrm{t}}}(\mathrm{a})$ is large: the arm a is likely to have a high reward
- $rad_t(a)$ is large: the arm a has not been explored much

The two terms in $UCB_t(a)$ represent, resp., exploitation and exploration.

UCB1

An arm a can have a large $UCB_t(a) = \overline{\mu_{\mathrm{t}}}(\mathrm{a}) + rad_t(a)$ for two reasons:

- $\overline{\mu_{\mathrm{t}}}(\mathrm{a})$ is large: the arm a is likely to have a high reward
- $rad_t(a)$ is large: the arm a has not been explored much

The two terms in $UCB_t(a)$ represent, resp., exploitation and exploration.
Idea: [*Optimism in the face of uncertainty*] At each time step t takes the arm a that maximize $UCB_t(a)$

UCB1

Require: K arms, horizon T

1: **for** each round $t = 1, \dots, T$ **do**
2: Chose a_t as the arm a which maximizes $UCB_t(a)$
3: Observe the reward r_t
4: Update upper confidence bounds:

$$\begin{cases} UCB_{t+1}(a) = \frac{1}{n_t(a)+1} \sum_{t \in [n_t(a)] \cup t} r_t + \sqrt{\frac{2 \log T}{n_t(a)+1}} & \text{if } a = a_t, \\ UCB_{t+1}(a) = UCB_t(a) & \text{otherwise} \end{cases}$$

5: **end for**

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. "Finite-Time Analysis of the Multiarmed Bandit Problem". In: *Mach. Learn.* 47.2–3 (2002), 235–256. ISSN: 0885-6125. DOI: 10.1023/A:1013689704352. URL: https://doi.org/10.1023/A:1013689704352

UCB1

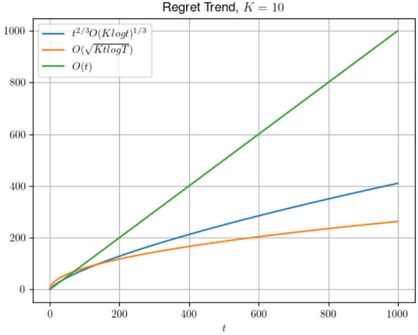
Theorem

UCB1 satisfies achieves regret bound

$$\mathbb{E}[\mathrm{R(t)}] = O(\sqrt{K \cdot t \log T}) \text{ for all } t \leq T$$

Stochastic Bandits: Summary

- We use the notion of regret for evaluating the performance of a Bandits Algorithms
- Our aim is to achieve sub-linear regret
- Non-Adaptive Algorithm: $\mathbb{E}[\mathrm{R(t)}] \leq t^{2/3} O(K \cdot \log t)^{1/3}$
- Adaptive-Algorithm: $\mathbb{E}[\mathrm{R(t)}] \leq O(\sqrt{K \cdot t \cdot \log T})$



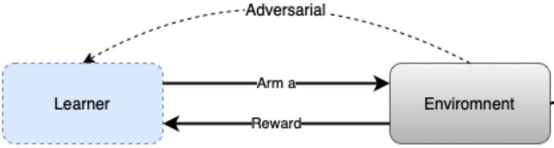
*Exercise

Exercise. Consider the Repeated Posted Price Auction described in Slide 5 with a discrete price set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Specifically, for each time step t the price π_t belongs to $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Moreover, assume that the buyer samples its valuation from a fixed distribution $D(\cdot)$.

1. Suppose that you are the seller. Design a Bandit Algorithm for the Repeated Posted Price Auction
2. Determine the Regret of the proposed algorithm
3. What happens if the price set is continuous? i.e. for each time step t the price π_t belongs to the continuous set $(0, 1]$

Adversarial Bandits

Adversarial Bandits: Problem Structure



Step	1	2	...
Arm a1	0.2	0.5	...
Arm a2	0.7	0.8	...
...	...	...	...

- Bandit Feedback: The algorithm only observes the reward for the selected action
- Deterministic Environment: The environment determines the rewards deterministically, independently from the arms chosen by the learner
- Adversarial Environment: The environment knows the learner algorithm
- Rewards are bounded: For simplicity, we restrict the rewards to the interval $[0, 1]$
- Goal: Maximize the cumulated revenue $\sum_{t=1}^{T} r_t$

The Adversarial nature of the environment is crucial

- Let's back to the Repeated Posted Price Auction
- Suppose that the Seller algorithm is deterministic:
  - The minimum price is p
  - The Seller adapts the posted price by observing the rewards $\rightarrow$ if the buyer always have the same valuation, then the posted price converges to the buyer valuation
- The Buyer knows the Seller's algorithm

Adversarial Environment

The Adversarial nature of the environment is crucial

- Let's back to the Repeated Posted Price Auction
- Suppose that the Seller algorithm is deterministic:
  - The minimum price is p
  - The Seller adapts the posted price by observing the rewards $\rightarrow$ if the buyer always have the same valuation, then the posted price converges to the buyer valuation
- The Buyer knows the Seller's algorithm
- The buyer does not act truthfully: it imposes its valuation equal to p
- The Seller has no chance to sell the shoes at a price higher than p

Adversarial Environment

The Adversarial nature of the environment is crucial

- Let's back to the Repeated Posted Price Auction
- Suppose that the Seller algorithm is deterministic:
  - The minimum price is p
  - The Seller adapts the posted price by observing the rewards $\rightarrow$ if the buyer always have the same valuation, then the posted price converges to the buyer valuation
- The Buyer knows the Seller's algorithm
- The buyer does not act truthfully: it imposes its valuation equal to p
- The Seller has no chance to sell the shoes at a price higher than p

Claim

The learner have to use a randomised policy

Adversarial Bandit Algorithm vs Mechanism Design

Mechanism Design

- Problem: Buyers can act strategically by misreporting their valuation to increase their utility

Adversarial Bandit Algorithm

- Problem: Buyers can act strategically by misreporting their valuation to increase their utility

Adversarial Bandit Algorithm vs Mechanism Design

Mechanism Design

- Problem: Buyers can act strategically by misreporting their valuation to increase their utility
- Solution: Determine Truthful Mechanism $\rightarrow$ Revel private valuation it is a dominant strategy

Adversarial Bandit Algorithm

- Problem: Buyers can act strategically by misreporting their valuation to increase their utility

Adversarial Bandit Algorithm vs Mechanism Design

Mechanism Design

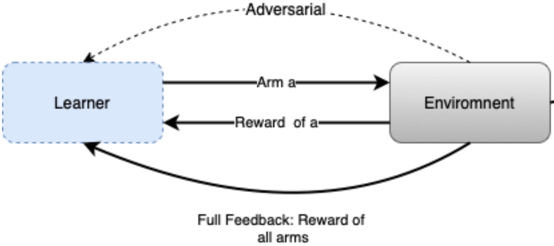
- Problem: Buyers can act strategically by misreporting their valuation to increase their utility
- Solution: Determine Truthful Mechanism $\rightarrow$ Revel private valuation it is a dominant strategy

Adversarial Bandit Algorithm

- Problem: Buyers can act strategically by misreporting their valuation to increase their utility
- Solution: Randomize the strategy

Warm Up: Adversarial Bandits with Full Feedback

Let's make the seller's life simpler...



Step	1	2	...
Arm a1	0.2	0.5	...
Arm a2	0.7	0.8	...
...	...	...	...

- Full Feedback: The algorithm observes the outcome not only for the chosen arm but for all the other arms as well

Hedge

We introduce a randomized algorithm called Hedge

- The algorithm takes into account the reward of all arms
- Hedge maintains a weight $w_t(a)$ for each arm a
- The higher is the weight for the arm the higher is the confidence about the arm
- Hedge uses a randomized algorithm to select the arm: the higher is the confidence about the arm the higher is the probability of selecting the arm
- Idea: Initially all the arms have the same confidence. If an arm receives an high reward we increase our confidence about the arm. At each time step we pick the arm with the higher confidence with high probability

Hedge

Require: K arms, T rounds, $\epsilon \in [0, 1/2]$

1: Initialize the weights $w_i(a) = 1$ for each arm a.
2: **for** $t \in [T]$ **do**
3: Let $p_t(a) = \frac{w_t(a)}{\sum_{a' \in [K]} w_t(a')}$
4: Sample an arm a_t from $p_t(\cdot)$
5: **for** $a \in [K]$ **do**
6: Observe $r_t(a)$
7: $w_{t+1}(a) \leftarrow w_t(a) \cdot (1 - \epsilon)^{(1 - r_t(a))}$
8: **end for**
9: **end for**

Nicolò Cesa-Bianchi et al. "How to Use Expert Advice". In: *J. ACM* 44.3 (1997), 427–485. ISSN: 0004-5411. DOI: 10.1145/258128.258179. URL: https://doi.org/10.1145/258128.258179

Hedge

Theorem

Assume that the reward is lower bounded, i.e. $r_t \geq c$, for some known $c > 0$ and for all t. Then Hedge achieves regret:

$$\mathbb{E}[\mathrm{R(T)}] \leq O(2 \cdot \sqrt{T \cdot \log K})$$

Reduction from Bandit Feedback to Full Feedback

- Use Hedge to solve a Bandit Adversarial Problem
- Sample an action according to the probability distribution defined by Hedge
- Problem: At each time step Hedge needs the reward of all the arms
- Solution: Define a fake reward for all the arms

Reduction from Bandit Feedback to Full Feedback

Reduction Schema

Require: K arms, T rounds, $\epsilon > 0$ for Hedge

for $t \in [T]$ **do**

  Call `Hedge`, receive the probability distribution p_t over $[K]$

  Selection rule: use p_t to pick arm $\mathrm{a_t}$

  Observe the reward $r_t(\mathrm{a_t})$ of the chosen arm (*Bandit Feedback*)

  Fake rewards: Define *fake rewards* $\widehat{\mathrm{r_t}}(\mathrm{a})$ for all arms

  Return *fake rewards* to `Hedge`

end for

Exp3

Exp3 completes the Reduction Schema:

- Selection Rule: (Explore and randomize) With probability $\gamma \in [0, 1/2)$ pick an arm uniformly at random, otherwise draw an arm from p_t
- Fake rewards:

$$\widehat{r_t}(a) = \begin{cases} 1 - \frac{1-r_t(a)}{p_t(a)}, & \text{if } a = a_t, \\ 1, & \textit{otherwise} \end{cases}$$

Peter Auer et al. "The Nonstochastic Multiarmed Bandit Problem". In: *SIAM J. Comput.* 32.1 (2003), 48–77. ISSN: 0097-5397. DOI: 10.1137/S0097539701398375. URL: https://doi.org/10.1137/S0097539701398375

Exp3

Exp3

Require: K arms, T rounds, $\epsilon \in [0, 1/2]$ for Hedge, $\gamma \in [0, 1/2)$

for $t \in [T]$ **do**

  Call `Hedge`, receive the probability distribution p_t over $[K]$

  With prob. γ pick an arm a_t uniformly at random; otherwise draw an arm a_t from p_t

  Observe the reward $r_t(\mathrm{a_t})$ of the chosen arm

  Compute the fake rewards for all arms $a \in [K]$:

$$\widehat{r_t}(a) = \begin{cases} 1 - \frac{1 - r_t(a)}{p_t(a)}, & \text{if } a = a_t, \\ 1, & \textit{otherwise} \end{cases}$$

  Return *fake rewards* to `Hedge`

end for

Exp3

Theorem

Assume that the reward is lower bounded, then Exp3 achieves regret:

$$\mathbb{E}[\mathrm{R(T)}] \leq O(2\sqrt{T \cdot K \cdot \log K})$$

Exp3 Parameters

How to impose parameters ϵ and γ? Maths can help!

- Remark $r_t \in [c, 1], \forall t$
- $\gamma \in [0, \frac{1}{2T})$
- $\epsilon = \sqrt{\frac{(1-\gamma) \cdot \log K}{3KT}}$

This choice of parameters guarantees the regret showed in the last slide!

Adversarial Bandits: Summary

- The learner has to randomise its strategy;
- We reduce our Bandit Problem to a Full-Feedback Problem
- Exp3:

$$\mathbb{E}[\mathrm{R(T)}] \leq O(2\sqrt{T \cdot K \cdot \log K})$$

*Exercise

Exercise. Consider the Repeated Posted Price Auction described in Slide 5 with a discrete price set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Specifically, for each time step t the price π_t belongs to $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. Moreover, assume that the buyer determines its valuation for all weeks before the game starts, the buyer knows the algorithm used by the seller and the buyer's valuation is at least 0.2

1. Suppose that you are the seller. Design a Bandit Algorithm for the Repeated Posted Price Auction
2. Determine the Regret of the proposed algorithm
3. What happens if the price set is continuous? i.e. for each time step t the price π_t belongs to the continuous set $(0, 1]$

References

References

Reference Books:

- Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambrdige Univeristy Press, 2019, https://tor-lattimore.com/downloads/book/book.pdf
- Aleksandrs Slivkins. *Introduction to Multi-Armed Bandits*. Foundations and Trends in Machine Learning, 2019, https://arxiv.org/abs/1904.07272