Final Project: Bandit Learning

Ziyu Shao

School of Information Science and Technology
ShanghaiTech University

May 16, 2023

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

Example of Online Problem

- A senior undergraduate student of SIST in ShanghaiTech University have applied many top graduate programs around the world.
- He will receive several offers in a sequential order. After looking at an offer, he must decide whether to accept it (and terminate the process) or to reject it. Once rejected, an offer is lost.
- Suppose that the only information he has at any time is the relative rank of the present offer compared with previously ones.
- His objective is to maximize the probability of selecting the best offer.
- Please help help him to find the optimal policy and compute the corresponding probability.

Offline Problems vs. Online Problems

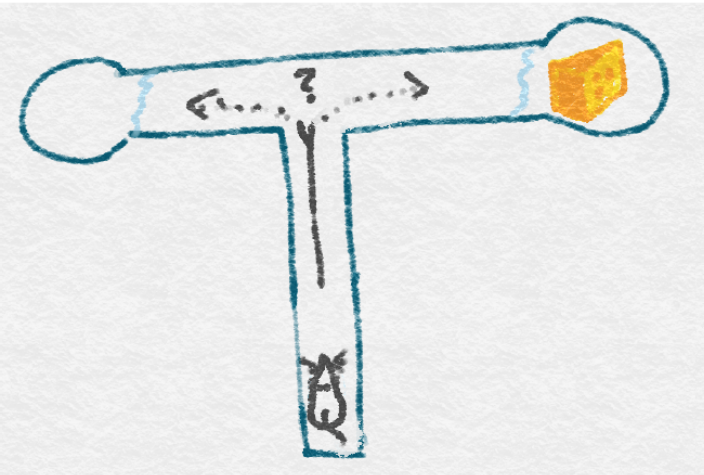
- Offline Problems
  - Assumed complete information: for example, know the whole input in advance
  - Example: the shortest path problem (topology & costs between nodes are given)
- Online Problems: decision, optimization and learning
  - Having no or incomplete knowledge of the future
  - Multiple decisions are made sequentially based on a piece-by-piece input
  - The sequential decisions are irrevocable
  - Example: playing the Chess/Go
- Bandit model & algorithms for online problems

One-Armed Bandit

Multi-Armed Bandit

History

First bandit algorithm proposed by Thompson (1933)



Bush and Mosteller (1953) were interested in how mice behaved in a T-maze

Widely Applications

- Clinical trials/dose discovery
- Recommendation systems (movies/news/etc)
- Advertisement placement
- A/B testing
- Monte Carlo tree search algorithm (game playing)
- Resource allocation
- Network routing
- Dynamic pricing (e.g.,for Amazon products)
- Ranking (e.g.,for search)

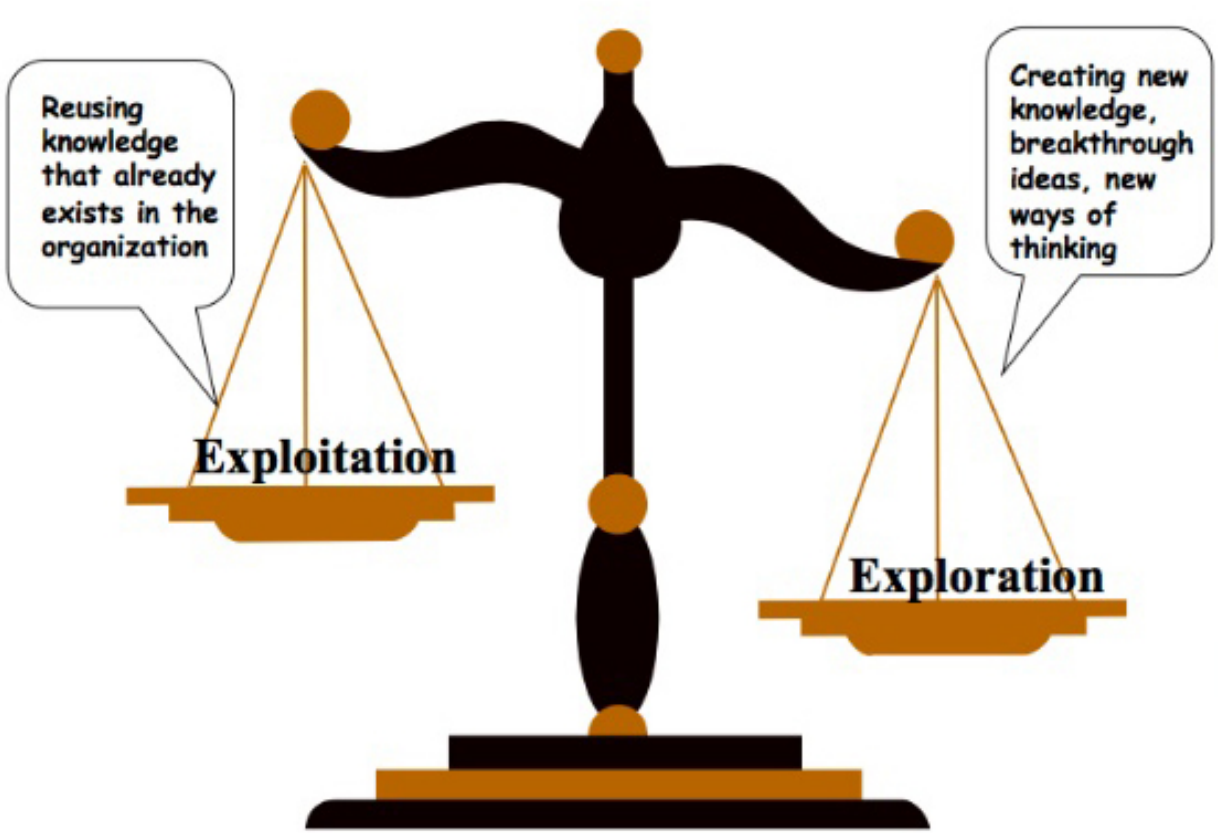
Why Bandit?

- A perfect model for online decision making under uncertainty
- Isolate an important component of reinforcement learning: exploration-vs-exploitation
- Rich and beautiful mathematically

Exploration vs. Exploitation Dilemma

- Online decision-making involves a fundamental choice
  - **Exploitation**: Make the best decision given current information
  - **Exploration**: Gather more information
- The best long-term strategy may involve short-term sacrifices
- Gather enough information to make the best overall decisions

Exploration vs. Exploitation Dilemma



Examples in Real Life

- Restaurant Selection
  - **Exploitation**: Go to your favorite restaurant
  - **Exploration**: Try a new restaurant

Example of A Two-Armed Bandit

Time	1	2	3	4	5	6	7	8	9	10	11	12
Left arm	$1	$0			$1	$1	$0					
Right arm			$1	$0								

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

Basic Setting

- We consider a time-slotted bandit system ($t = 0, 1, 2, \dots$) with three arms.
- We denote the arm set as $\{1, 2, 3\}$. Pulling each arm j ($j \in \{1, 2, 3\}$) will obtain a reward r_j, which satisfies a Bernoulli distribution with mean θ_j (Bern(θ_j)), *i.e.*,

$$r_j = \begin{cases} 1, & w.p.\ \theta_j, \\ 0, & w.p.\ 1 - \theta_j, \end{cases}$$

- θ_j are parameters within $(0, 1)$ for $j \in \{1, 2, 3\}$.

Basic Setting

- Now we run this bandit system for N ($N \gg 3$) time slots.
- At each time slot t, we choose one and only one arm from these three arms, which we denote as $I(t) \in \{1, 2, 3\}$. Then we pull the arm $I(t)$ and obtain a reward $r_{I(t)}$.
- Our objective is to find an optimal policy to choose an arm $I(t)$ at each time slot t such that the expectation of the aggregated reward is maximized, *i.e.*,

$$\max_{I(t), t=1,\ldots,N} \quad \mathbb{E}\left[\sum_{t=1}^{N} r_{I(t)}\right].$$

Basic Setting

- If we know the values of $\theta_j, j \in \{1, 2, 3\}$, this problem is trivial.
- Since $r_{I(t)} \sim \text{Bern}(\theta_{I(t)})$,

$$\mathbb{E}\left[\sum_{t=1}^{N} r_{I(t)}\right] = \sum_{t=1}^{N} \mathbb{E}[r_{I(t)}] = \sum_{t=1}^{N} \theta_{I(t)}.$$

- Let $I(t) = I^* = \underset{j \in \{1,2,3\}}{\arg\max}\ \theta_j$ for $t = 1, 2, \ldots, N$, then

$$\max_{I(t), t=1,\ldots,N} \mathbb{E}\left[\sum_{t=1}^{N} r_{I(t)}\right] = N \cdot \theta_{I^*}.$$

- However, in reality, we do not know the values of $\theta_j, j \in \{1, 2, 3\}$. We need to estimate the values θ_j via empirical samples, and then make the decisions at each time slot.

Classical Bandit Algorithms

- ϵ-Greedy
- Upper Confidence Bound (UCB)
- Thompson Sampling

ϵ-Greedy Algorithm

Algorithm 1 $\epsilon-$greedy Algorithm

Initialize $\hat{\theta}(j) = 0, \text{count}(j) = 0, j \in \{1, 2, 3\}$

1: **for** $t = 1, 2 \ldots, N$ **do**

2:

$$I(t) \leftarrow \begin{cases} \underset{j \in \{1,2,3\}}{\arg\max}\ \hat{\theta}(j) & w.p.\ 1-\epsilon \\ \text{randomly chosen from} \{1,2,3\} & w.p.\ \epsilon \end{cases}$$

3: $\quad \text{count}(I(t)) \leftarrow \text{count}(I(t)) + 1$

4: $\quad \hat{\theta}(I(t)) \leftarrow \hat{\theta}(I(t)) + \frac{1}{\text{count}(I(t))} \left[r_{I(t)} - \hat{\theta}(I(t)) \right]$

5: **end for**

UCB Algorithm

Algorithm 2 UCB Algorithm

for $t = 1, 2, 3$ **do**
    $I(t) \leftarrow t$
    $\text{count}(I(t)) \leftarrow 1$
    $\hat{\theta}(I(t)) \leftarrow r_{I(t)}$
end for
for $t = 4, \ldots, N$ **do**

$$I(t) \leftarrow \underset{j \in \{1,2,3\}}{\arg\max} \left(\hat{\theta}(j) + c \cdot \sqrt{\frac{2 \log t}{\text{count}(j)}} \right)$$

    $\text{count}(I(t)) \leftarrow \text{count}(I(t)) + 1$
    $\hat{\theta}(I(t)) \leftarrow \hat{\theta}(I(t)) + \frac{1}{\text{count}(I(t))} \left[r_{I(t)} - \hat{\theta}(I(t)) \right]$
end for

Thompson Sampling Algorithm

Algorithm 3 Thompson sampling Algorithm

Initialize Beta parameter $(\alpha_j, \beta_j), j \in \{1, 2, 3\}$

1: **for** $t = 1, 2 \ldots, N$ **do**
2: **for** $j \in \{1, 2, 3\}$ **do**
3: Sample $\hat{\theta}(j) \sim \text{Beta}(\alpha_j, \beta_j)$
4: **end for**

$$I(t) \leftarrow \arg\max_{j \in \{1,2,3\}} \hat{\theta}(j)$$

$$\alpha_{I(t)} \leftarrow \alpha_{I(t)} + r_{I(t)}$$
$$\beta_{I(t)} \leftarrow \beta_{I(t)} + 1 - r_{I(t)}$$

5: **end for**

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

Setting

- Suppose we obtain the Bernoulli distribution parameters from an oracle, which are shown in the following table below.

Arm j	1	2	3
θ_j	0.7	0.5	0.4

- Choose $N = 5000$ and compute the theoretically maximized expectation of aggregate rewards over N time slots. We call it the oracle value. Note that these parameters $\theta_j, j = 1, 2, 3$ and oracle values are unknown to the three bandit algorithms.

Setting

Implement three bandit algorithms with following settings:

- ϵ-greedy with $\epsilon = 0.1, 0.5, 0.9$.
- UCB with $c = 1, 5, 10$.
- Thompson Sampling with
  $\{(\alpha_1, \beta_1) = (1, 1), (\alpha_2, \beta_2) = (1, 1), (\alpha_3, \beta_3) = (1, 1)\}$ and
  $\{(\alpha_1, \beta_1) = (601, 401), (\alpha_2, \beta_2) = (401, 601), (\alpha_3, \beta_3) = (2, 3)\}$

Tasks

- Each experiment lasts for N $= 5000$ time slots, and we run each experiment 200 trials. Results are averaged over these 200 independent trials.
- Compute the gaps between the algorithm outputs (aggregated rewards over N time slots) and the oracle value. Compare the numerical results of ϵ-greedy, UCB, and Thompson Sampling. Which one is the best? Then discuss the impacts of ϵ, c, and α_j, β_j respectively.
- Give your understanding of the exploration-exploitation trade-off in bandit algorithms.
- We implicitly assume the reward distribution of three arms are independent. How about the dependent case? Can you design an algorithm to exploit such information to obtain a better result?

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

Bayesian Bandits

- There are two arms which may be pulled repeatedly in any order.
- Each pull may result in either a success or a failure.
- The sequence of successes and failures which results from pulling arm i ($i \in \{1, 2\}$) forms a Bernoulli process with unknown success probability θ_i.
- A success at the t^{th} pull yields a reward γ^{t-1} ($0 < \gamma < 1$), while an unsuccessful pull yields a zero reward.
- At time zero, each θ_i has a Beta prior distribution with two parameters α_i, β_i and these distributions are independent for different arms.

Bayesian Bandits

- These prior distributions are updated to posterior distributions as arms are pulled.
- Since the class of Beta distributions is closed under Bernoulli sampling, posterior distributions are all Beta distributions.
- How should the arm to pull next in each time slot be chosen to maximize the total expected reward from an infinite sequence of pulls?

Tasks

- One intuitive policy suggests that in each time slot we should pull the arm for which the current expected value of θ_i is the largest. This policy behaves very good in most cases. Please design simulations to check the behavior of this policy.
- However, such intuitive policy is unfortunately not optimal. Please provide an example to show why such policy is not optimal.

Tasks

- For the expected total reward under an optimal policy, show that the following recurrence equation holds:

$$
\begin{aligned}
R_1(\alpha_1, \beta_1) =& \frac{\alpha_1}{\alpha_1 + \beta_1}[1 + \gamma R(\alpha_1 + 1, \beta_1, \alpha_2, \beta_2)] \\
&+ \frac{\beta_1}{\alpha_1 + \beta_1}[\gamma R(\alpha_1, \beta_1 + 1, \alpha_2, \beta_2)]; \\
R_2(\alpha_2, \beta_2) =& \frac{\alpha_2}{\alpha_2 + \beta_2}[1 + \gamma R(\alpha_1, \beta_1, \alpha_2 + 1, \beta_2)] \\
&+ \frac{\beta_2}{\alpha_2 + \beta_2}[\gamma R(\alpha_1, \beta_1, \alpha_2, \beta_2 + 1)]; \\
R(\alpha_1, \beta_1, \alpha_2, \beta_2) =& \max\{R_1(\alpha_1, \beta_1), R_2(\alpha_2, \beta_2)\}.
\end{aligned}
$$

- For the above equations, how to solve it exactly or approximately?
- Find the optimal policy.

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

Core of AlphaGo & AlphaZero

- 2016-2018: Google DeepMind team developed AlphaGo & AlphaZero System, where the core algorithm is MCTS(Monte Carlo Tree Search) + Neural Network
- Core of MCTS: UCT (multi-stage UCB)

After humanity spent thousands of years improving our tactics, computers tell us that humans are completely wrong... I would go as far as to say not a single human has touched the edge of the truth of Go.

Ke Jei,
9 dan Go player

robots will never understand the beauty of the game the same way that we humans do

Lee Sedol,
9 dan Go player

Outline

1. Bandit Background
2. Multi-Armed Bandit Algorithms
3. Part I: Basic Bandits (Required)
4. Part II: Bayesian Bandits (Optional)
5. Part III: Monte Carlo Tree Search (Optional)
6. References

References

- Reinforcement Learning: An Introduction (second edition), R. Sutton & A. Barto, 2018.
- Bandit Algorithms, T. Lattimore & C. Szepesvari, 2020.

Remark