Final Projects

Ziyu Shao

School of Information Science and Technology
ShanghaiTech University

December 12, 2023

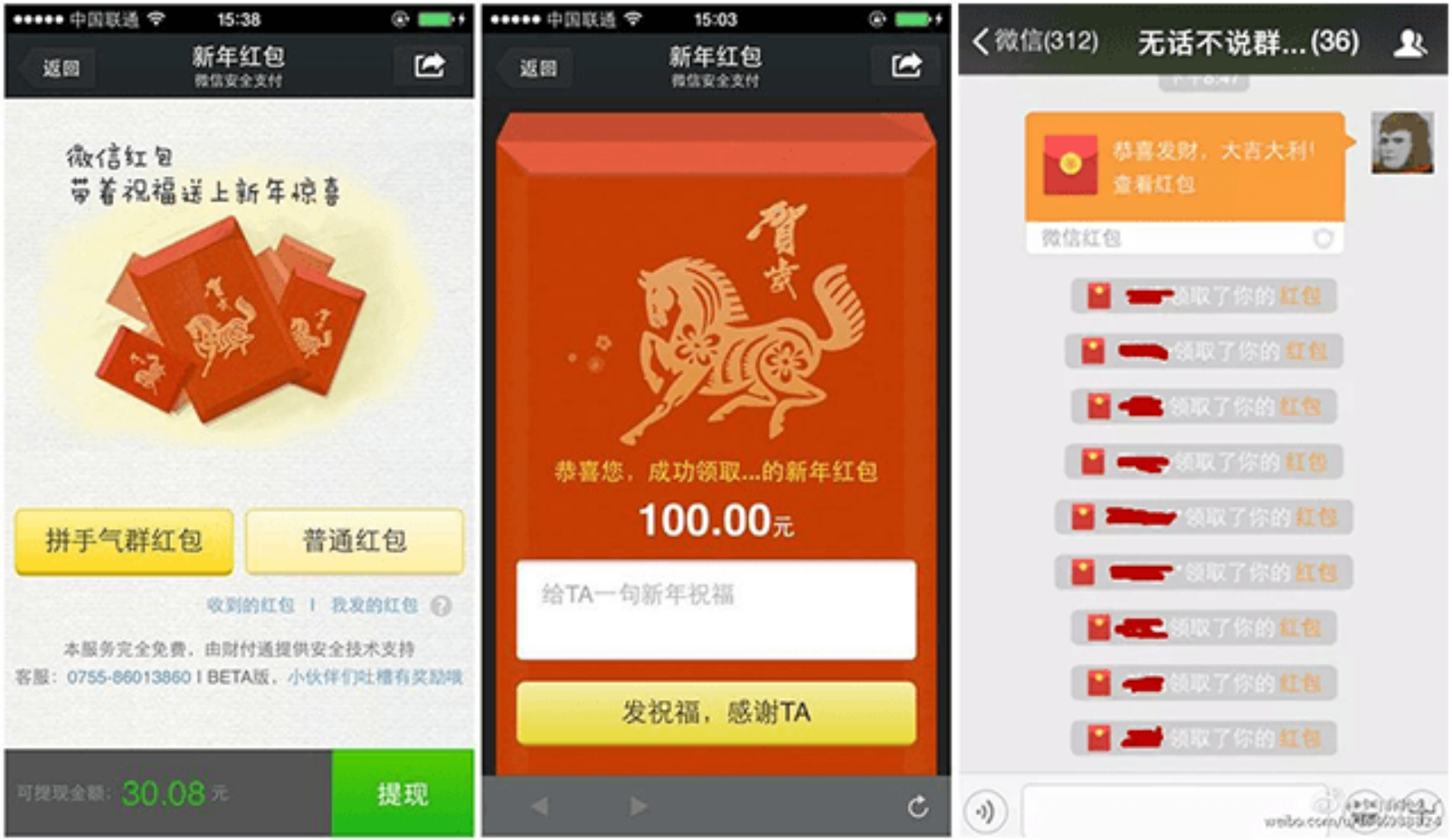
Project Requirement

- Choose one and only one option
- The project can be done by a team with no more than three students.
- Your team is required to use Python for the programming part. Your team needs to submit all things in Jupyter Notebook format, including Python codes, simulation results, analysis, discussions, tables, figures, *etc*.
- Always keep the **academic integrity** in mind and remember to give credit to any source that inspires you.
- Due date: 2024/1/21 10:59pm

Outline

Outline

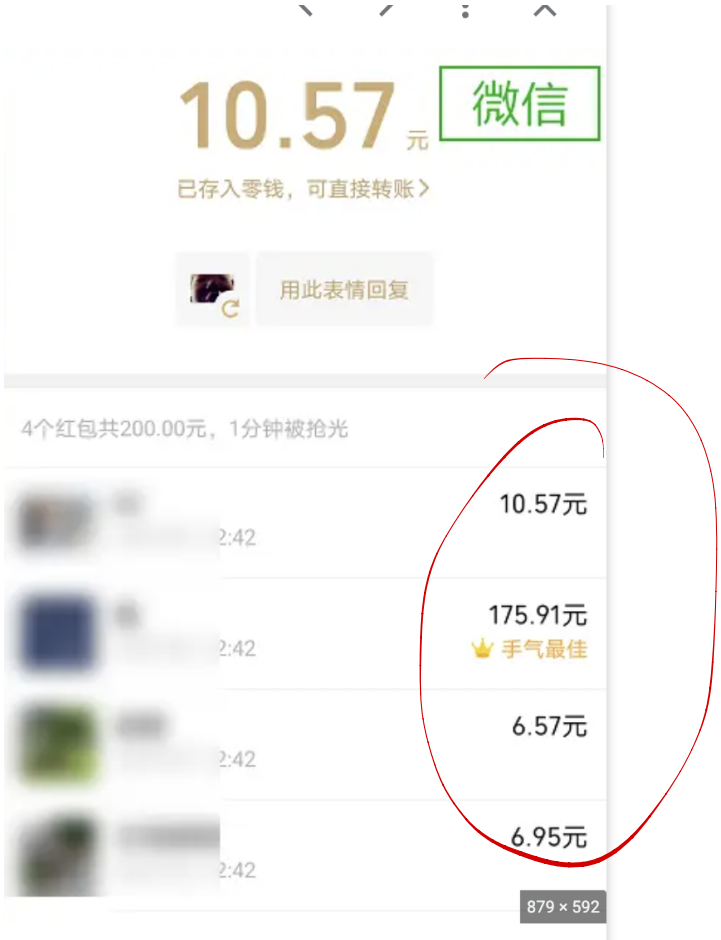
Wechat Red Envelope



Reverse Engineering the Mechanism

- Obtain enough data sets first.
- Based on such data sets, try to figure out the underlying random allocation mechanism for money distribution.
- How such mechanism relate to many factors including the amount of total money, the number of users in a group, individual user's profile(the activity history of user account, the country of user account, et al.), and even operating systems (iOS and Android)?
- Any suggestions for better mechanism?

The Lucky One



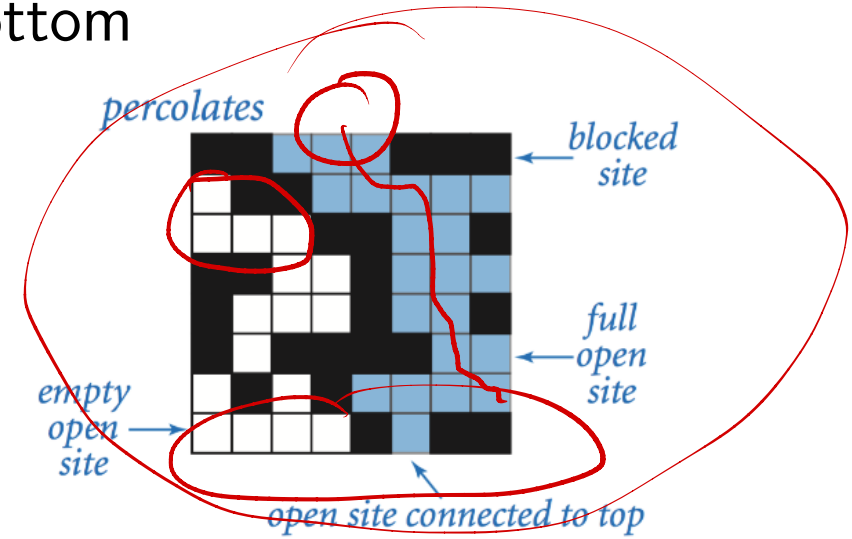
Decision-Making Policy

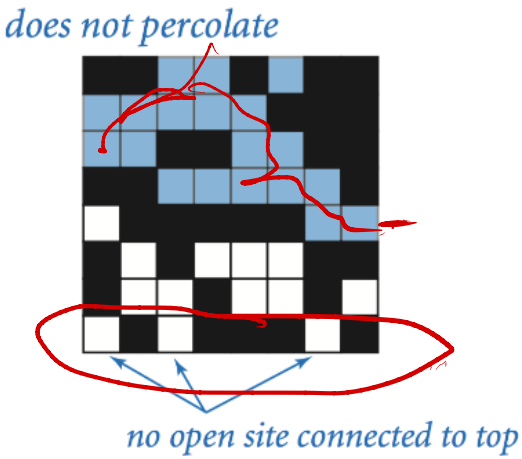
- Now you want to be the luck one, please design a smart strategy to grab the money as much as possible.
- Use your data sets to justify the advantage of your strategy.

Outline

Part I: Percolation over Finite Grid (Required)

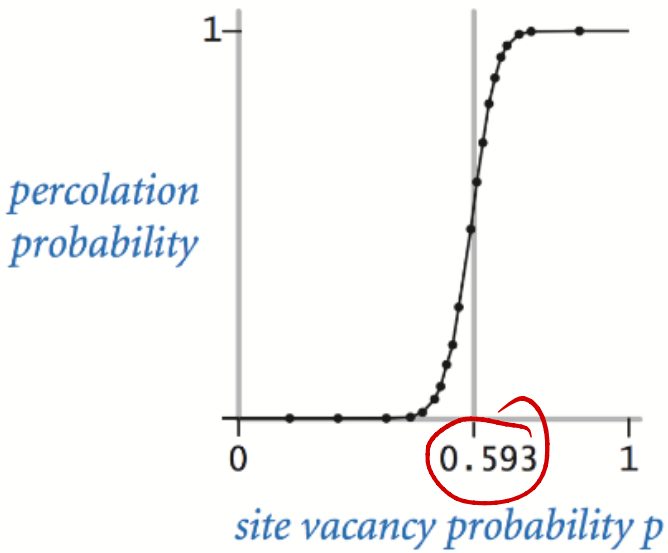
- Example: A fluid flow through a porous medium
- Blocked Site: fluid can not flow through it
- Open Site: fluid can flow through it
- Full Open Site: there exist a path (site chain) between it and the top open sites
- Empty Open Site: Such path does not exist
- Percolation: There exists at least one full open site at the bottom

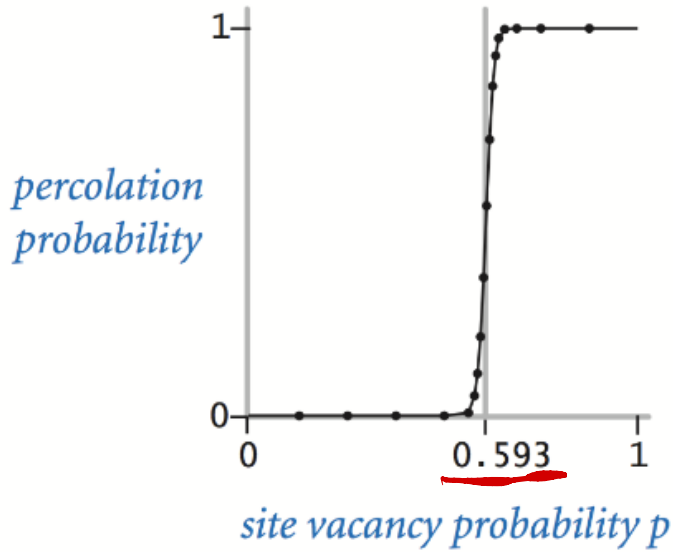




Phase Transition

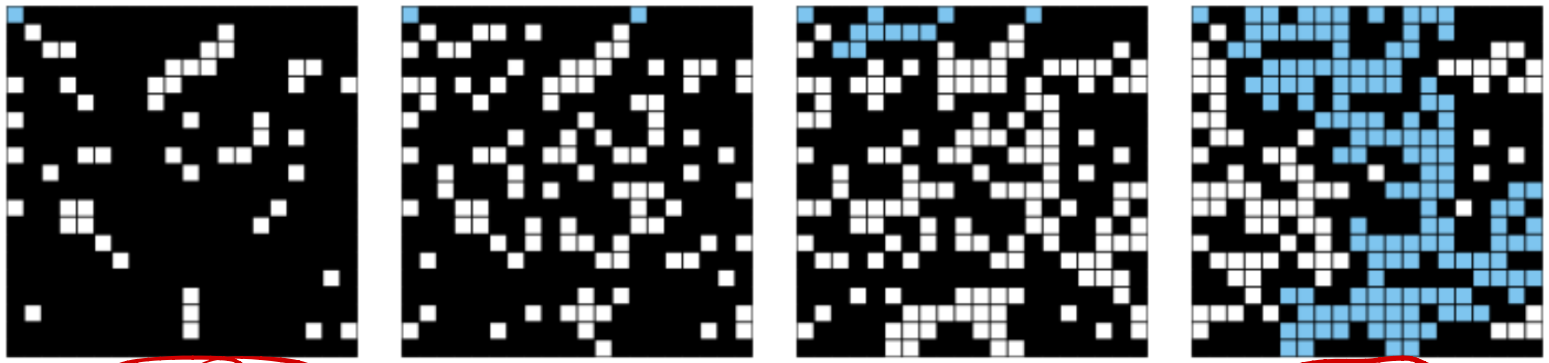
- $n \times n$ grid: each site is independently being open site with probability p (site vacancy probability)
- Percolation threshold value of p: below it without percolation; above it with percolation
- Left:20×20 grid; Right: 100×100 grid.





Estimation of Percolation Threshold

- Initialize all sites to be blocked and black.
- Repeat the following until the system percolates:
  - Choose a site uniformly at random among all blocked sites.
  - Open the site.
- The fraction of sites that are opened when the system percolates provides an estimate of the percolation threshold.
- A 20-by-20 lattice with estimate of the percolation threshold is $204/400 = 0.51$ because the system percolates when the 204th site is opened.



50 open sites *100 open sites* *150 open sites* *204 open sites*

Estimation of Percolation Threshold

- Repeating this computation experiment T times and averaging the results, we obtain a more accurate estimate of the percolation threshold. Let x_t be the fraction of open sites in computational experiment t. The sample mean $\bar{x}$ provides an estimate of the percolation threshold:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_T}{T}.$$

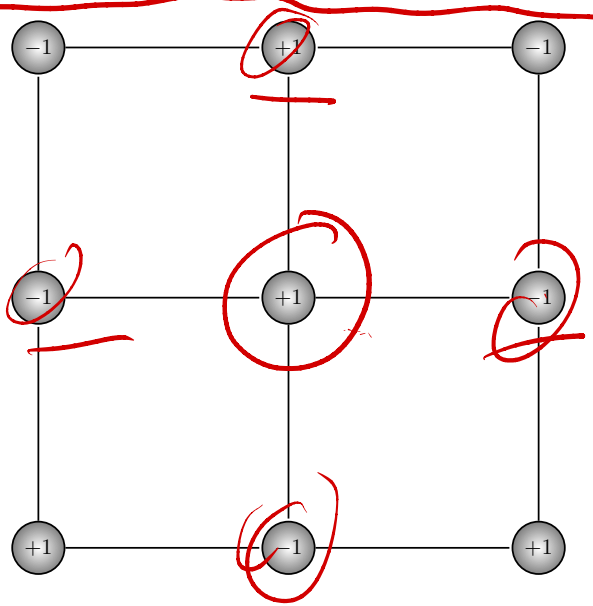
- Task: Estimate the percolation threshold for $20 - by - 20$ grid, $50 - by - 50$ grid, and $100 - by - 100$ grid via Monte Carlo simulation.

Part II: Ising Model (Required)

- In certain alloys, particularly those containing Fe, Co or Ni, the electrons have a tendency to align their spins in a common direction. This phenomenon is called **ferromagnetism**
- The spin ordering diminishes as the temperature of the sample increases and, above a certain temperature T_c, the magnetization vanishes entirely.
- This is a **phase transition** and T_c is called the **critical temperature** or the **Curie temperature**.

Ising Model

- Invented by the physicist Wilhelm Lenz in 1920, who gave it as a problem to his student Ernst Ising.
- The one-dimensional Ising model was solved by Ising in 1924
- The two-dimensional Ising model was solved by Lars Onsager in 1944.
- The three-dimensional Ising model is still open



Ising Model

- Spin configuration as $\boldsymbol{\sigma} = \{\sigma_v, v \in V\}$, where $\sigma_v \in \{-1, +1\}$.
- The corresponding energy is

$$H(\boldsymbol{\sigma}) = -\sum_{(v,w)\in E} \sigma_v \sigma_w.$$

- Define the set of all possible spin configurations as Ω, and define the Gibbs distribution(or called Boltzmann Distribution) over Ω as follows:

$$\pi_{\boldsymbol{\sigma}} = \frac{e^{-\beta H(\boldsymbol{\sigma})}}{Z}, \forall \boldsymbol{\sigma} \in \Omega.$$

- β is the constant representing the inverse of the temperature, and Z is the normalization constant (also called the partition function)

Ising Model

- **Task A**: Show that given all other vertexes' spin value, the conditional distribution of vertex k 's spin is:

$$\begin{cases} P(\sigma_k = +1 | \boldsymbol{\sigma}_{-k}) = \dfrac{1}{1 + e^{-2\beta\left(\sum_{v \sim k} \sigma_v\right)}} \\ P(\sigma_k = -1 | \boldsymbol{\sigma}_{-k}) = \dfrac{1}{1 + e^{2\beta\left(\sum_{v \sim k} \sigma_v\right)}}. \end{cases}$$

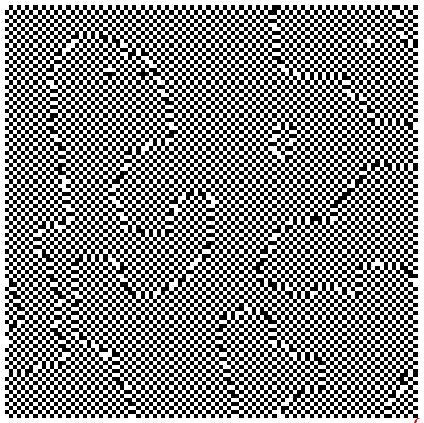
  where

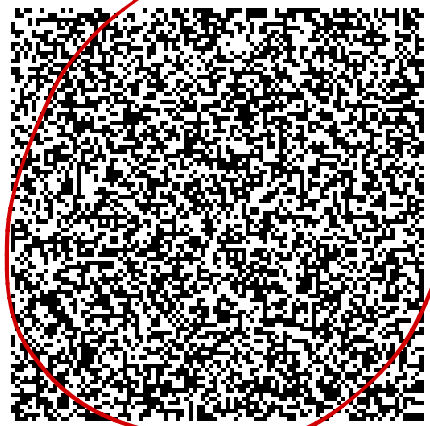
$$\boldsymbol{\sigma}_{-k} = (\sigma_1, \ldots, \sigma_{k-1}, \sigma_{k+1}, \ldots, \sigma_{|V|}).$$

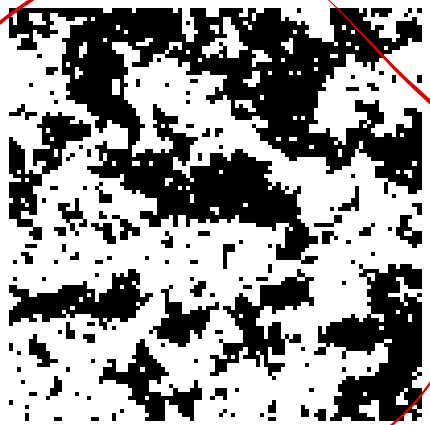
- **Task B**: Simulate the two-dimensional Ising model with parameters as follows: $n = 100$, $\beta = -1, 0, 0.441, 0.8$.
- **Task C**: Simulate the two-dimensional Ising model with parameters as follows: $n = 300$, $\beta = -5, 0.2, 0.441, 0.6$.

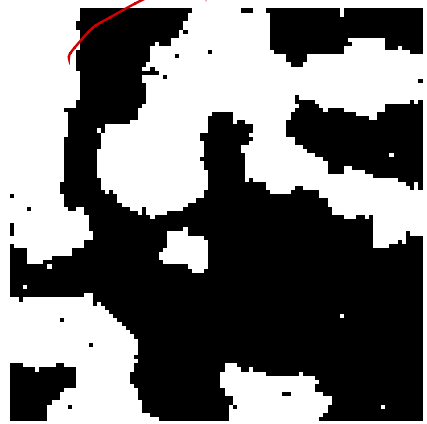
Possible Simulation Results

- The two-dimensional Ising model with parameters as follows: $n = 100$, $\beta = -1.0, 0.441, 0.8$.

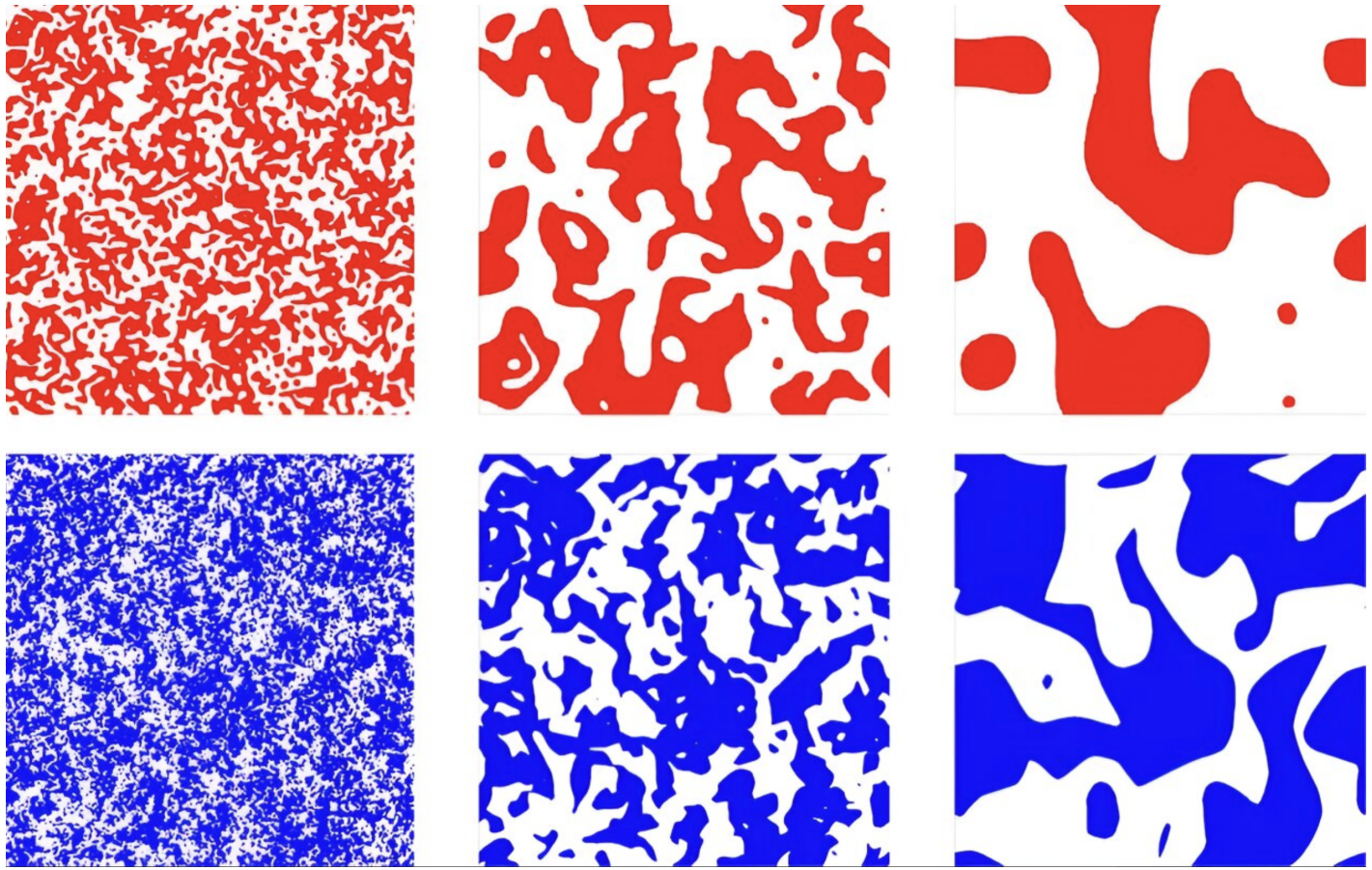








Colorful Simulation Results



Outline

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 him to find the optimal policy and compute the corresponding probability.

Offline Problems vs. Online Problems

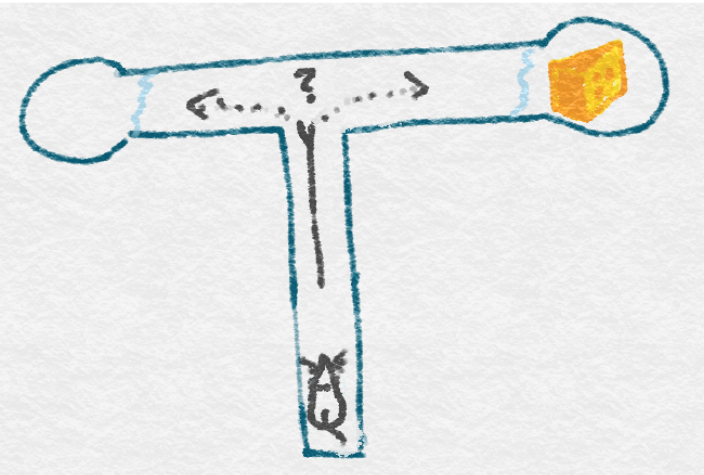
- Offline Problems
  - Assumed complete information: for example, know the whole input in advance
  - Example: the shortest path problem (topology & costs between nodes are given)
- Online Problems: decision, optimization and learning
  - Having no or incomplete knowledge of the future
  - Multiple decisions 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 (AI)
- Resource allocation
- Network routing
- Dynamic pricing (e.g.,for Amazon products)
- Ranking (e.g.,for search)

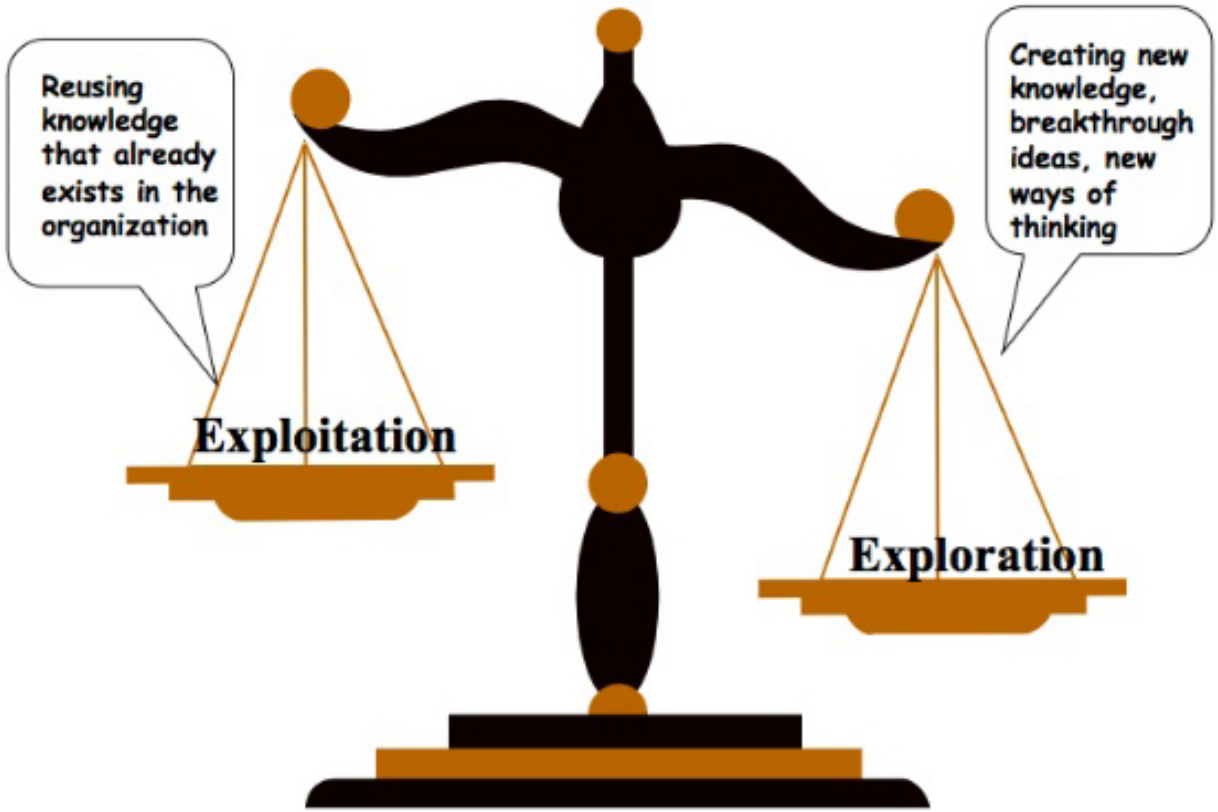
Why Bandit?

- 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								

$\frac{3}{5} = 0.6$

$\frac{1}{2} = 0.5$

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.*,

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

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.

Part I: Classical Bandit Algorithms (Required)

- 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:

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

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

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

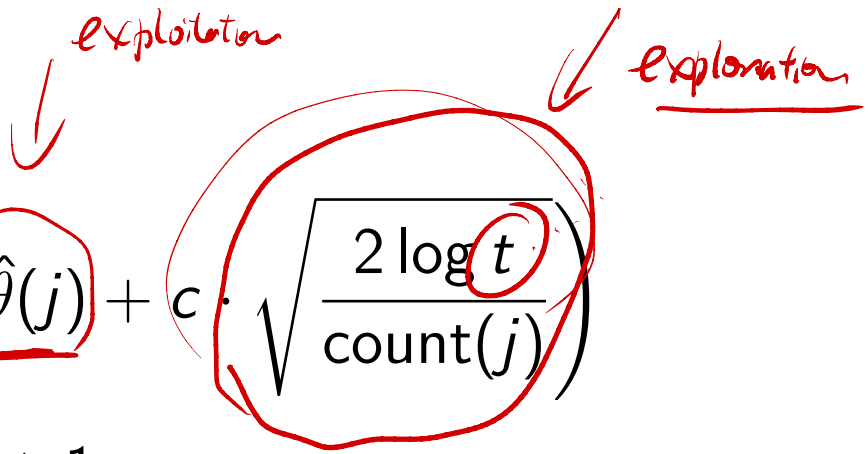
5: **end for**

UCB Algorithm

Algorithm 2 UCB Algorithm

1: **for** $t = 1, 2, 3$ **do**
2: $I(t) \leftarrow t$
3: $\text{count}(I(t)) \leftarrow 1$
4: $\hat{\theta}(I(t)) \leftarrow r_{I(t)}$
5: **end for**
6: **for** $t = 4, \ldots, N$ **do**
7:

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



8: $\text{count}(I(t)) \leftarrow \text{count}(I(t)) + 1$
9: $\hat{\theta}(I(t)) \leftarrow \hat{\theta}(I(t)) + \frac{1}{\text{count}(I(t))} \left[r_{I(t)} - \hat{\theta}(I(t)) \right]$
10: **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**

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?

Part II: Bayesian Bandits (Optional)

- 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.

References

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

Remark