Strategy for Two Players Multi-Armed Bandit

Background and Motivation

As one of the team leaders, you know that nothing keeps your fellow elves more productive and motivated than a steady supply of candy canes! But all seven levels of the Candy Cane Forest are closed for revegetation, so the only ones available are stuck in the break room vending machines. And even though you receive free snacks on the job, the vending machines are always broken and don't always give you what you want.

Due to social distancing, only two elves can be in the break room at once. You and another team leader will take turns trying to get candy canes out of the 100 possible vending machines in the room, but each machine is unpredictable in how likely it is to work. You do know, however, that the more often you try to use a machine, the less likely it will give you a candy cane. Plus, you only have time to try 2000 times on the vending machines until you need to get back to the workshop!

If you can collect more candy canes than the other team leaders, you'll surely be able to help your team win Santa's contest! Try your hand at this multi-armed candy cane challenge!

Problem Definition

Your objective is to find a strategy to beat your opponent as much as possible.

Both participants will work with the same set of N vending machines (bandits). Each bandit provides a random reward based on a probability distribution specific to that machine. Every round each player selects ("pulls") a bandit, the likelihood of a reward decreases by 3%.

Each agent can see the move of the other agent, but will not see whether a reward was gained in their respective bandit pull.

Methodlogy

1. Pure Greedy base on Average Profit in history

   In this strategy, we play each machine one time in the beginning. After that, each time we play, we calculate the average profit of each machine in history. In other words, we choose to play the machine that makes the following formula have the highest value:

   $Avg_i = \frac{w_i}{n_i}$

   w_i: Rewards we have gotten on machine i in history.
   n_i: Times we have played on machine ${i} in history.

Note that the strategy is straightforward and doesn't consider the effect of the decaying rate.

2. Epsilon-Delta Strategy

   The problem is an "exploitation" and "exploration" tradeoff dilemma. Epsilon-Delta is one of the simple and famous strategies. We choose a probability p for exploration. Each time when we play, we use p of chance randomly choose a machine to play (exploration), and another $1-p$ of the chance to play the machine that has highest average reward in history just like we mentioned in pure greedy strategy (exploitation).

$$chosen\ machine = \begin{cases} random, & chance\ of\ p \\ argmax\ Avg_i, & chance\ of\ 1-p \end{cases}$$

3. UCB Strategy

   UCB formula is one of the famous mathematical methods for balancing the "exploitation" and "exploration" strategy. Each time when we play, we choose a machine i which makes the following formula has the highest value:

$$UCB = \frac{w_i}{n_i} + c * \sqrt{\frac{lnN}{n_i}}$$

   w_i: Rewards we have gotten on machine i in history.
   n_i: Times we have played on machine i in history.
   c: Exploration parameter, equals to $\sqrt{2}$ in theory.
   N: Times we have played on all machines in history.

4. Advance Greedy Method base on Maximum Likelihood Estimation

   We would like to maximize our expected profit each time we play based on the observation of historical results. To know the expected profit of each machine, we have to consider the effect of the decaying rate. For a specific machine i, we have history information I, representing in a series $(player_j, w_j)$. If we have a given probability p for machine i beginning expected profit, we could calculate $P(I|p)$ easily:

$$P(I|p) = \prod_{j=0} \begin{cases} (0.97)^j * p, & player_j = me \wedge w_j = 1 \\ (1 - p * (0.97)^j), & player_j = me \wedge w_j = 0 \\ 1, & otherwise \end{cases}$$

   However, what we want to know is the value of p. So we could try to find the value $p = p*$ which makes $P(p|I)$ have the highest value. We use the likelihood function $L(p|I) = P(I|p)$ to find the maximum likelihood estimation of p.
   Each time we play, we find a probability p_i for machine i which has maximum likelihood. Then we choose the machine which has the highest expected value. The expected value for machine i:

$EV_i = p_i * (0.97)^{n_i}$

p_i: maximum likelihood estimation of p on machine i.
n_i: times we and out opponent have played on mahine i.

5. Advance UCB Method by Combining Advance Greedy and Pure UCB

   We could deem the advanced greedy method as a strategy to predict the expected reward of each machine more precisely. Hence we could take advantage of it to adjust the formula of UCB:

   $newUCB = EV_i + c\sqrt{\frac{\ln N}{n_i}}$

   EV_i: Ecpected value of machine i by calculating maximum likelihood.
   n_i: Times we play on machine i in history.
   c: Exploration parameter, equals to $\sqrt{2}$ in theory.
   N: Times we play on all machines in history.

6. Maching Learning Predicting Strategy

   To predict the expected reward of each machine, we could apply a machine learning model to solve it. We use the times we and our opponent on machine i and the average reward of machine i as input features. Then we train a model by generating random self-play data. Each time we play, we choose the machine which has the highest model predict output to play. We will try following ML models to predict each machine's profit.:

   - Linear Regressor
   - Decision Tree Regressor
   - K-Neighbors Regressor
   - Epsilon-Support Vector Regressor
   - Random Forest Regressor

7. Combining Advance UCB with a ML/DL model:

   We could also let the output of a machine learning model affect the decision to advance UCB strategy. Hoping the model's output will increase the performance of the advanced UCB strategy. The new UCB formula is described as follows:

   $newUCB = EV_i + c\sqrt{\frac{\ln N}{n_i}} + \beta y_i$

   EV_i: Ecpected value of machine i by calculating maximum likelihood.
   n_i: Times we have played on machine i in history.
   c: Exploration parameter, equals to $\sqrt{2}$ in theory.

N: Times we have played on all machines in history.
β: An hyper parameter. It will affect the importance of the ML/DL model's predicted result.
y_i: Output of the ML/DL model on machine i.

8. Analysis method by competing with a random agent

   To compare and analysis whether each method is good or not, we run agent builded up by each strategy against a random agent to see the scoring status.

9. ELO rating system analysis

   The Elo rating system is a method for calculating the relative skill levels of players in zero-sum games such as chess. It is named after its creator Arpad Elo, a Hungarian-American physics professor. We compete with all the agents with each other. If agent A has a rating of R_A and agent B rating of R_B, the exact formula (using the logistic curve with base 10) for the expected score of player A is

   $$E_A = \frac{1}{1+10^{(R_B-R_A)/400}}$$

   Similarly the expected score for agent B is

   $$E_B = \frac{1}{1+10^{(R_A-R_B)/400}}$$

   Suppose player A (again with rating R_A) was expected to score E_A points but actually scored S_A points $(win = 1, tie = 0.5, lose = 0)$. The formula for updating that player's rating is

   $$R_A' = R_A + K(S_A - E_A)$$
   $$R_B' = R_B + K(S_B - E_B)$$

10. Analysis by Mean Square Error

    Some methods will calculate the original probability to find the Maximum Likelihood Estimation, so we use MSE to calculate the loss between the probability predicted by the agent and the actual probability. And use it as one of the indicators for judging the quality of the agent. The following is the formula for calculating mean square error:

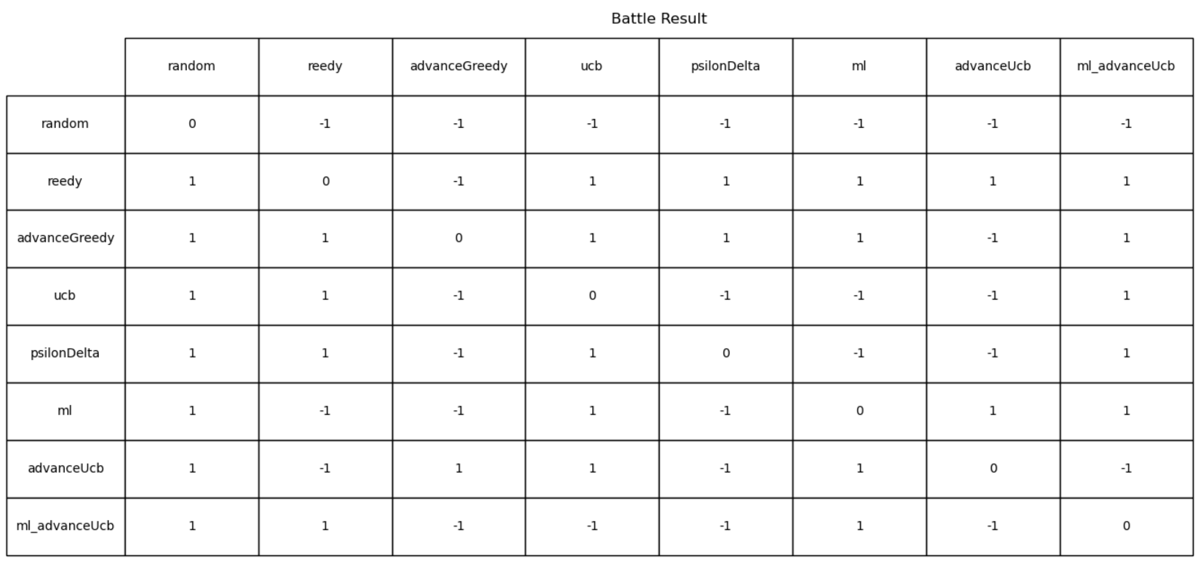
    $$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2$$

    N: Number of data we have.
    y_i: predict value of ith data.
    t_i: exact value of ith data.

Data Collection and Analysis Result

Analysis

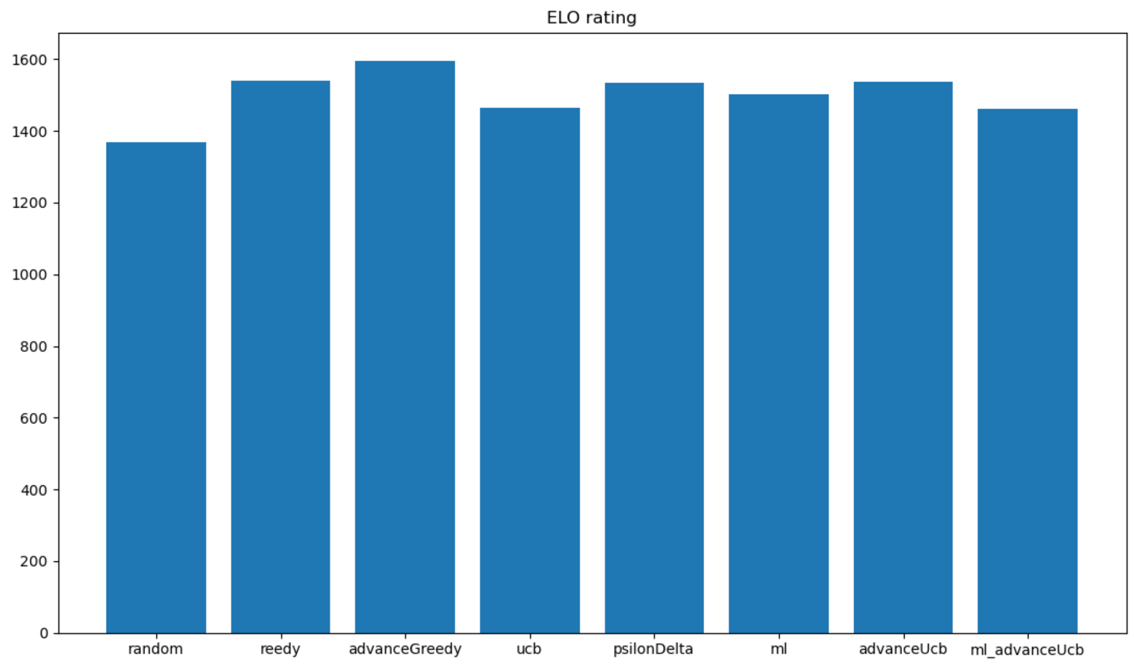
Battle Result



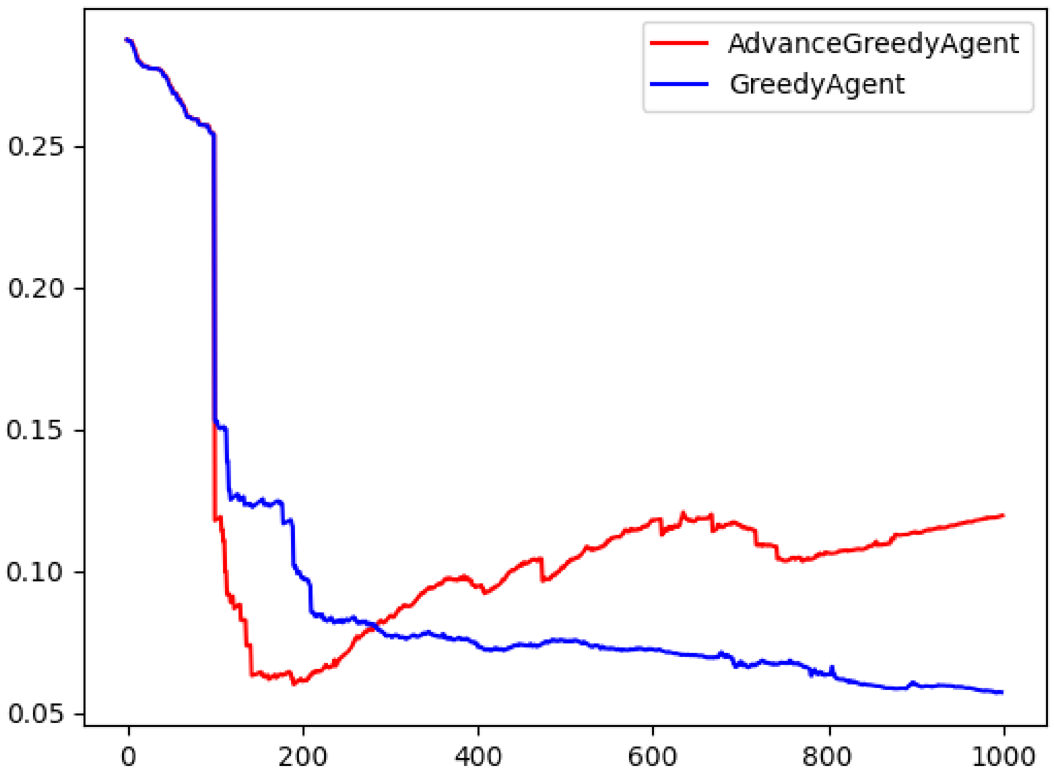
Battle Result

	random	reedy	advanceGreedy	ucb	psilonDelta	ml	advanceUcb	ml_advanceUcb
random	0	-1	-1	-1	-1	-1	-1	-1
reedy	1	0	-1	1	1	1	1	1
advanceGreedy	1	1	0	1	1	1	-1	1
ucb	1	1	-1	0	-1	-1	-1	1
psilonDelta	1	1	-1	1	0	-1	-1	1
ml	1	-1	-1	1	-1	0	1	1
advanceUcb	1	-1	1	1	-1	1	0	-1
ml_advanceUcb	1	1	-1	-1	-1	1	-1	0

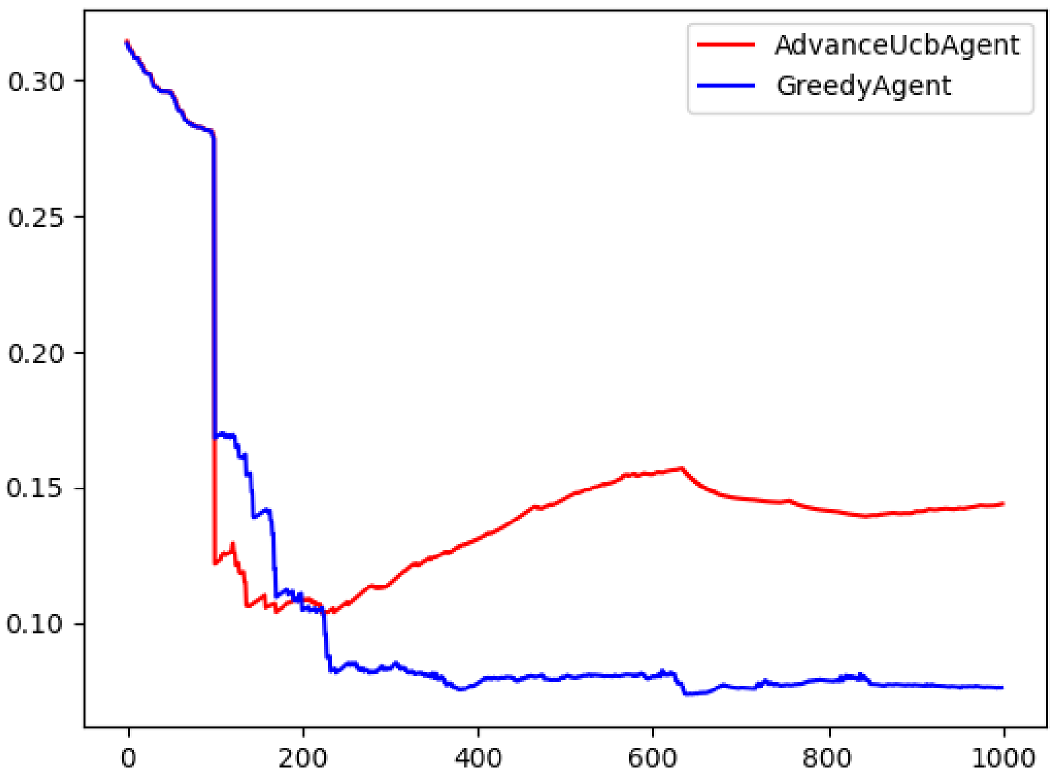
ELO Rating



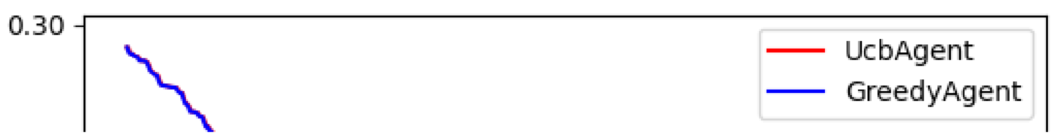
Loss of Initial Probility Prediction



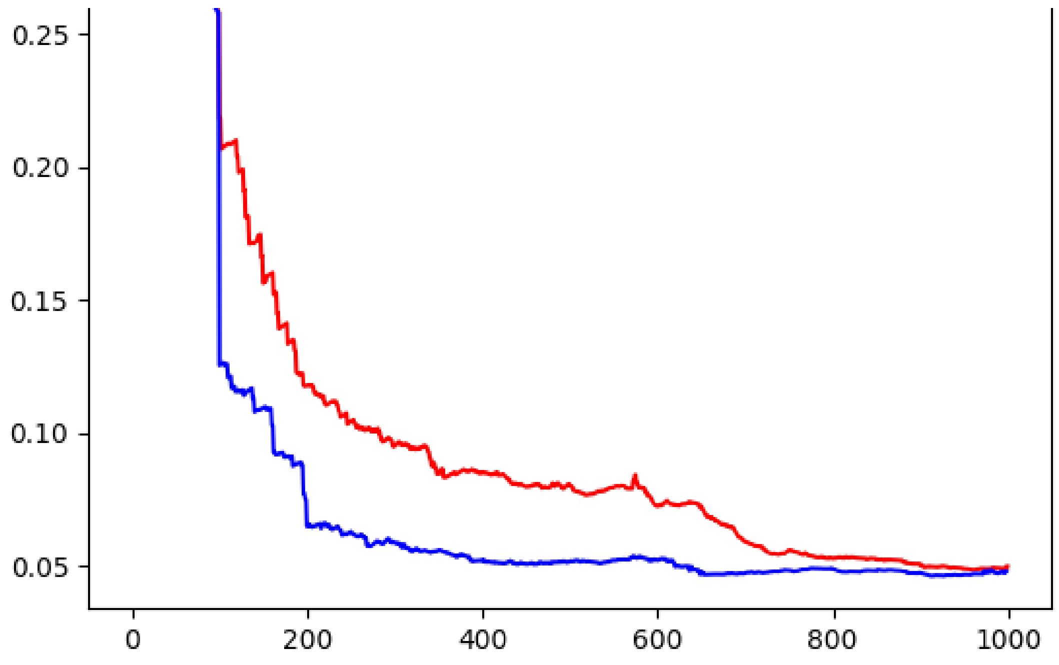
AdvanceGreedyAgent
GreedyAgent
0.25
0.20
0.15
0.10
0.05
0
200
400
600
800
1000

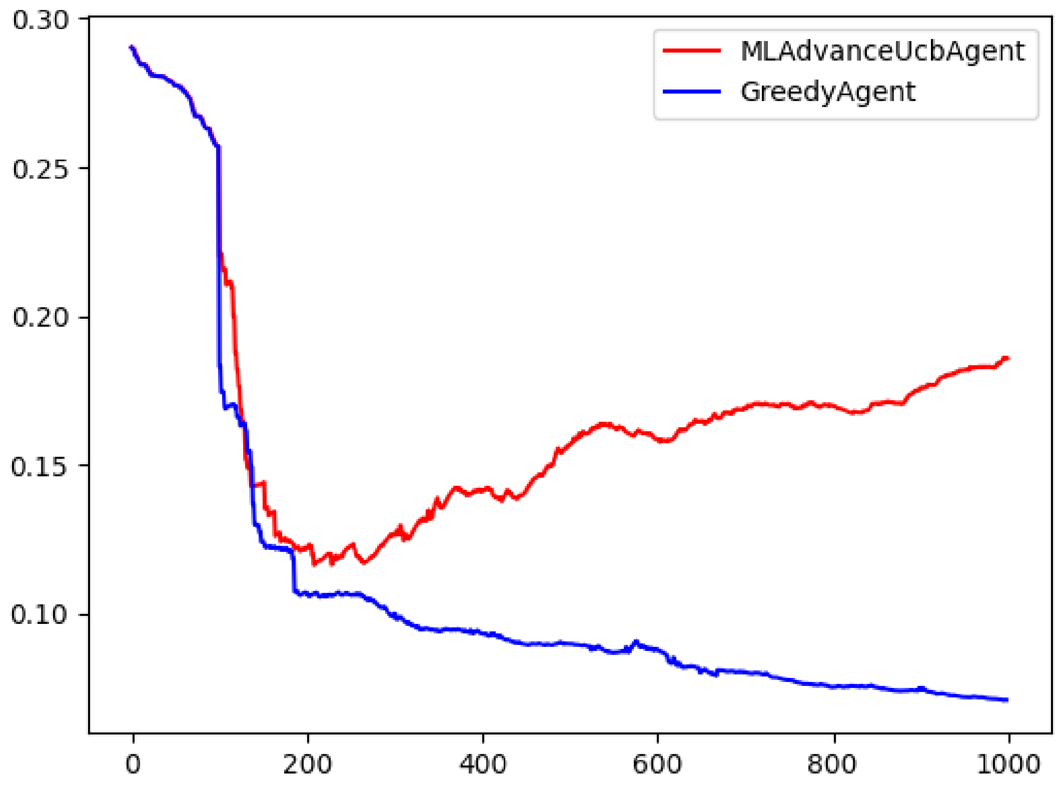


AdvanceUcbAgent
GreedyAgent
0.30
0.25
0.20
0.15
0.10
0
200
400
600
800
1000



UcbAgent
GreedyAgent
0.30





Result

1. Advance Greedy Agent has the best ELO rating and the highest winning rate.
2. Greedy Agent has a surprisingly strong performance. Its ELO rating is only lower than Advance Greedy Agent. It shows that even though we have a decaying rate, average history reward is still a great likelihood function.
3. While using Greedy or Advance Greedy Agent. If we, unfortunately, don't get a reward on the first try of machine i, then the machine is dead, which means the strategy will no longer

play on that machine. We guess it leads to a considerable loss, and it is probably the reason we get a slight improvement using advanced UCB with a very small c.

4. Even though Advance Greedy Agent has a higher ELO rating and performs better than pure greedy, Advance Greedy Agent generally guesses the initial probability of each machine worse than Greedy Agent in terms of mean square error.

Conclusion

1. Advance Greedy Agent has the strongest performance, so we recommend playing the game with greedy strategy base on maximum likelihood estimation.
2. In the UCB formula, we have a hyperparameter - “c” which is theoretically equal to $\sqrt{2}$. However, we found that The smaller the “c”, the better the performance of the model. So we thought “exploitation” is much more important than “exploration” in this game.
3. Using the UCB strategy combing greedy strategy with maximum likelihood estimation and choosing a relatively very small exploration parameter c will result in a slight improvement in the performance.

Future Work

1. Try applying deep learning models such as LSTM, transformer, or LGBM.
2. Analysis of each machine learning model and finding better hyper parameters for each model.
3. Try combining the machine learning model and rule base algorithm for a better strategy.
4. Research on optimizing each hyperparameter.

Reference

- Aleksandrs Slivkins (2019), “Introduction to Multi-Armed Bandits”, Foundations and Trends® in Machine Learning: Vol. 12: No. 1-2, pp 1-286. http://dx.doi.org/10.1561/2200000068 (http://dx.doi.org/10.1561/2200000068)
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 3146–3154.