COMP4600/8460 - Advanced Algorithms **S2 2022**

Assignment 4

[Xueer Zhangsong] ([u7079320]) Date: October 16, 2022

Note: *Plagiarism is strictly prohibited. Even though you may discuss with classmates, you should write your homework assignment by yourself.*

1 Questions

Q1) (10 Marks) Prove that if we set $\hat{p}_i = M\left(1 - (1-\frac{1}{\hat{s}})(1+\frac{1}{k\hat{s}})^{i-1}\right)$, then $\mathcal{A}_{\mathsf{kmin}}$ achieves the competitive ratio $\hat{s}$, where $\hat{s}$ is the solution to $\frac{1-\frac{m}{M}}{1-\frac{1}{\hat{s}}} = (1+\frac{1}{k\hat{s}})^k$

Q2) (10 Marks) Consider a variant of dating problem – each candidate has a score sampled form a uniform distribution in $[0, 1]$. The candidates are ranked according to their scores. Provide an online strategy that maximizes the chance of picking the candidate of the highest score

Q3) (10 Marks) Simulate WMA and WMU, considering the following experts:

1. Expert who makes a uniformly random prediction all the time
2. Expert who always predicts the same as the outcome of the previous day
3. Expert who predicts the future the outcome of the next day with accuracy of $\frac{1}{2} + \delta$, where δ is a small constant
4. Create your own expert and explain your model

Generate the outcome of each day based on a uniform distribution

Vary the number of days T and ϵ to test the accuracy of WMA and WMU

2 Answers

2.1 Q1

(1) k min search

Definition 1 (K-Min Search) *Given a sequence of prices* $(p_1, p_2, \ldots, p_T)$ *over time* $(1, \ldots, T)$. *Need to buy* k *items, and only one item bought at each time. The price range is* $m < p_t < M$ *for all* $t = 1, \ldots, T$. *If* i *items of* k *are not bought before time* $T - i + 1$, *then force to buy all* i *items in* $(p_{T-i}, \ldots, p_T)$. *The offline optimal for the K-Min Search is to pick the* k *lowest prices.*

Let $\hat{p}_i$ be the threshold buying price of the i-th price. There are k of them $\hat{p}_1 \geq \hat{p}_2 \geq \ldots \geq \hat{p}_k$. The decrease of the thresholds indicate that buying the first item is easier than the second items.

According to the definition, adversary has $k+1$ options:

Case 1: $\mathcal{A}_{kmin}$ sell at $\hat{p}_1$ and $(p_{T-k+1}, \ldots, p_T)$, there are $k-1$ item unsold sold before $T-k$.

Case 2: $\mathcal{A}_{kmin}$ sell at $(\hat{p}_1, \hat{p}_2)$ and $(p_{T-k+1}, \ldots, p_T)$, there are $k-2$ item unsold sold before $T-k+1$.

...

Case k: $\mathcal{A}_{kmin}$ sell at $(\hat{p}_1, \mathrm{p}_2, \ldots, \hat{p}_k)$, there are 0 item unsold sold before T.

Case k+1: $\mathcal{A}_{kmin}$ sell at $(p_{T-k}, \ldots, p_T)$, there are K item unsold sold before $T-k-1$.

Because $\mathcal{A}_{kmin}$ attend to pick the lowest price and the price range is between m and M, thus we define $\hat{p}_{k+1} = m$.

K-Min search is a minimization problems, the Competitive ratios is $max\{\frac{\mathcal{A}_{kmin}}{OPT}\}$. The Competitive ratio $\hat{s}$ for each case of K-Min search are as follow:

Case 1:

$$\text{Price Sequence: } (\ldots, \hat{p}_1, \ldots, \underbrace{\hat{p}_2 + \epsilon, \ldots, \hat{p}_2 + \epsilon}_{k}, \ldots, \underbrace{M, \ldots, M}_{k-1})$$

$$\text{Competitive ratio: } \frac{\hat{p}_1 + (k-1)m}{k\hat{p}_2}$$

Case 2:

$$\text{Price Sequence: } (\ldots, \hat{p}_1, \ldots, \hat{p}_2, \ldots, \underbrace{\hat{p}_3 + \epsilon, \ldots, \hat{p}_3 + \epsilon}_{k}, \ldots, \underbrace{M, \ldots, M}_{k-2})$$

$$\text{Competitive ratio: } \frac{\hat{p}_1 + \hat{p}_2 + (k-2)m}{k\hat{p}_3}$$

...

Case k:

$$\text{Price Sequence: } (\ldots, \hat{p}_1, \ldots, \hat{p}_k, \ldots, \underbrace{M, \ldots, M}_{k})$$

$$\text{Competitive ratio: } \frac{\sum_{i=1}^{k} \hat{p}_i}{k\hat{p}_{k+1}} = \frac{\sum_{i=1}^{k} \hat{p}_i}{km}$$

Case $k+1$:

$$\text{Price Sequence: } (\ldots, \underbrace{\hat{p}_1+\epsilon, \ldots, \hat{p}_1+\epsilon}_{k}, \ldots, \underbrace{M, \ldots, M}_{k})$$

$$\text{Competitive ratio: } \frac{kM}{k\hat{p}_1}$$

Equating all the competitive ratios to solve $\hat{s}$.

$$\frac{kM}{k\hat{p}_1} = \frac{\hat{p}_1 + (k-1)m}{k\hat{p}_2} = \frac{\hat{p}_1 + \hat{p}_2 + (k-2)m}{k\hat{p}_3} = \frac{\sum_{i=1}^{k} \hat{p}_i}{km} = \hat{s}$$

(2) solve $\hat{s}$ Given $\hat{p}_i = M\Big(1 - (1-\frac{1}{\hat{s}})(1+\frac{1}{k\hat{s}})^{i-1}\Big)$, apply it to calculate $\sum_{i=1}^{k} \hat{p}_i$:

$$\begin{aligned}
\sum_{i=1}^{k} \hat{p}_i &= \sum_{i=1}^{k} M\Big(1 - (1-\frac{1}{\hat{s}})(1+\frac{1}{k\hat{s}})^{i-1}\Big) \\
&= M\sum_{i=1}^{k}\Big(1 - (1-\frac{1}{\hat{s}})(1+\frac{1}{k\hat{s}})^{i-1}\Big) \\
&= M\Big(k - \sum_{i=1}^{k}(1-\frac{1}{\hat{s}})(1+\frac{1}{k\hat{s}})^{i-1}\Big) \\
&= M\Big(k - (1-\frac{1}{\hat{s}})\sum_{i=1}^{k}(1+\frac{1}{k\hat{s}})^{i-1}\Big) \\
&\text{According to geometric series:} \\
&= M\Big(k - (1-\frac{1}{\hat{s}})\frac{(1+\frac{1}{k\hat{s}})^{k}-1}{(1+\frac{1}{k\hat{s}})-1}\Big) \\
&= M\Big(k - k\hat{s}(1-\frac{1}{\hat{s}})((1+\frac{1}{k\hat{s}})^{k}-1)\Big)
\end{aligned}$$

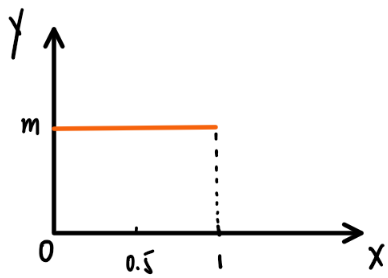
Apply this result to the equation above:

$$\begin{aligned}
\frac{\sum_{i=1}^{k} \hat{p}_i}{km} &= \hat{s} \\
M\Big(k - k\hat{s}(1-\frac{1}{\hat{s}})((1+\frac{1}{k\hat{s}})^{k}-1)\Big) &= km\hat{s} \\
k - k\hat{s}(1-\frac{1}{\hat{s}})((1+\frac{1}{k\hat{s}})^{k}-1) &= \frac{km\hat{s}}{M} \\
k - \frac{km\hat{s}}{M} &= k\hat{s}(1-\frac{1}{\hat{s}})((1+\frac{1}{k\hat{s}})^{k}-1) \\
1 - \frac{m\hat{s}}{M} &= \hat{s}(1-\frac{1}{\hat{s}})((1+\frac{1}{k\hat{s}})^{k}-1) \\
\frac{1-\frac{m\hat{s}}{M}}{\hat{s}(1-\frac{1}{\hat{s}})} + 1 &= (1+\frac{1}{k\hat{s}})^{k} \\
\frac{\frac{1}{\hat{s}} - \frac{m}{M} + 1 - \frac{1}{\hat{s}}}{(1-\frac{1}{\hat{s}})} &= (1+\frac{1}{k\hat{s}})^{k} \\
\frac{1-\frac{m}{M}}{1-\frac{1}{\hat{s}}} &= (1+\frac{1}{k\hat{s}})^{k}
\end{aligned}$$

Thus proof that $\hat{s}$ is solution to $\frac{1-\frac{m}{M}}{1-\frac{1}{\hat{s}}} = (1+\frac{1}{k\hat{s}})^k$.

2.2 Q2

According to the question, the score of each candidate are sampled form a uniform distribution in [0,1].



This means that if there are m candidates whose score is 0.5, then there will also be m candidates whose score is 0.6. The order of all candidates is shuffled. Assuming that the candidates have a total of k scores, then the total number of candidates will be $n = km$.

Because, if the best candidate is in the past, we can no longer select it anymore, thus the strategy should prevent that from happening. In other word, the algorithm will always choose the best candidate from either the current candidate or a candidate in the future.

Consider the current candidate is the i-th candidate.

Then, there are only two situations left, they leads to statements:

$$\mathbf{P}(\text{i-th candidate is the the best}) + \mathbf{P}(\text{the best candidate is in the future}) = 1$$
$$\mathbf{P}(\text{the best candidate is in the future}) = 1 - \mathbf{P}(\text{i-th candidate is the the best})$$

Because the scores are uniform distribution in [0,1], if the i-th candidate's score is s_i, then the possibility of other candidate's score is lower than s_i is s_i.

$$\mathbf{P}(\text{i+1-th candidate's score} \leq s_i) = s_i$$
$$\mathbf{P}(\text{all n - i candidates' score} \leq s_i) = \prod_{i+1}^{n} s_i = (s_i)^{n-i}$$
$$\mathbf{P}(\text{the best candidate is in the future}) = 1 - (s_i)^{n-i}$$

Decided to choose the i-th candidate if:

$$\mathbf{P}(\text{i-th candidate is the the best}) \geq \mathbf{P}(\text{the best candidate is in the future})$$

When $\mathbf{P}$(i-th candidate is the the best) $=$ $\mathbf{P}$(the best candidate is in the future), both of the probability is equal to $\frac{1}{2}$, thus:

$$\mathbf{P}(\text{the best candidate is in the future}) \leq \frac{1}{2}$$
$$1 - (s_i)^{n-i} \leq \frac{1}{2}$$
$$s_i \geq \frac{1}{2}^{\frac{1}{n-i}}$$

Let $\frac{1}{2}^{\frac{1}{n-i}}$ be the threshold, if s_i is greater then the threshold, then pick the i-th candidate and stop the algorithm. Otherwise, keep checking the next candidate.

I calculate the probability of selecting the best candidate by display a testing picture:

```python
import numpy as np
import matplotlib.pyplot as plt

num = []
pro = []
for m in range(2,150):
    p = 0
    n = 0
    while n < 100:
        select = 0
        random_array = np.random.uniform(0.0, 1.0, m)
        M = max(random_array)
        for i in range(m):
            th = (0.5)**(1/(m-i))
            if random_array[i] >= th:
                select = random_array[i]
                break
        if M == select:
            p += 1
        n += 1

    num.append(m)
    pro.append(p/100)
```

Figure 1. python code of the test

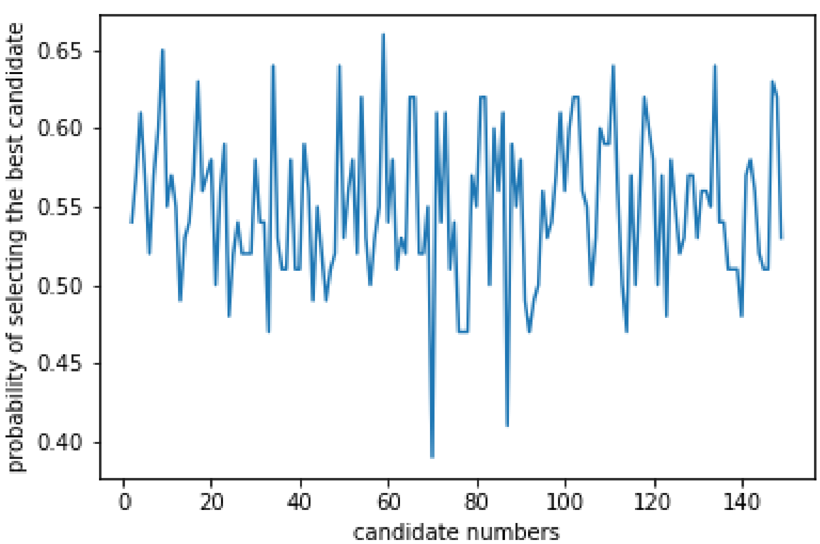


Figure 2. test result

Compared with probability of choosing the best candidate by Sample-and-Choose Algorithm(37%)d, this strategy gets a higher chance of picking the best candidate.

2.3 Q3

(0) Expert-Four

1. According to the question, I generate the fourth expert with the model: always predicts the next outcome as the opposite of the previous one.

And example of expert four prediction will be:

outcome: [True, False, True, True, False, True, True, True, False, False]

prediction: [True, False, True, False, True, False, True, False, True, False]

2. Another kind of expert taking ideas from the one bit branch prediction of CPU.

The example will be:

outcome: [True, False, True, True, False, True, True, True, False, False]

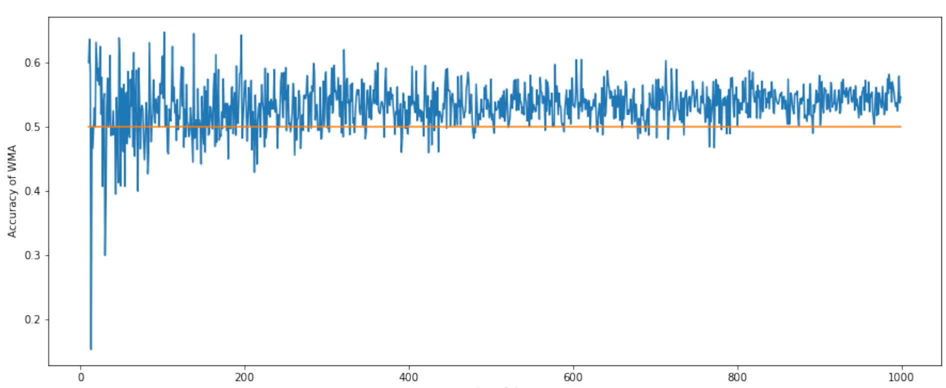
prediction: [True, True, False, True, True, False, True, True, True, False]

I think the quality of the second Expert-Four is higher than the first, because the second Expert-Four will make the next judgment based on the actual output

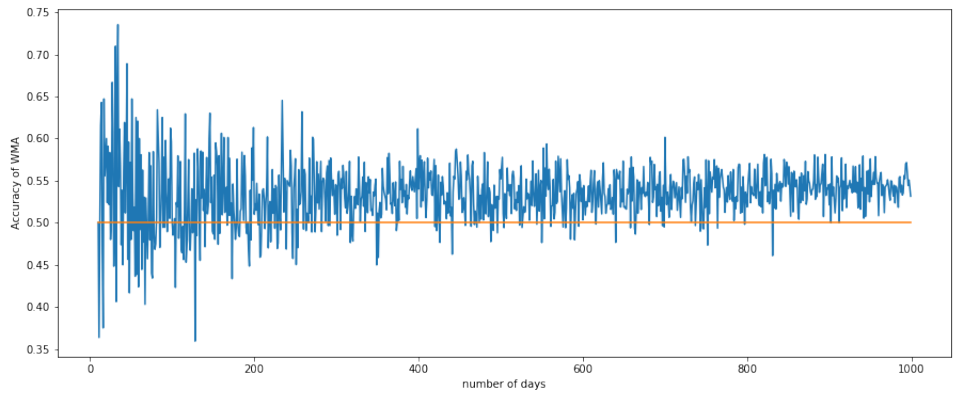
(1) WMA

I set the range of the testing day from 10 to 1000.

First test with the first Expert-Four. The line chart of the accuracy of WMA changes as the number of days increase is as follows:



Then test with the second Expert-Four:

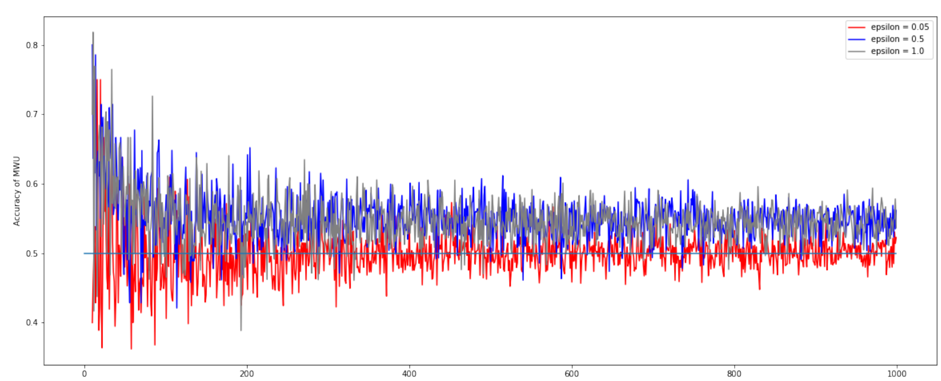


It can be seen from the analysis in Figure 1 and Figure 2 that for WMA, as the number of days increases, the accuracy gradually tends to be stable, oscillating between 0.5 and 0.6.

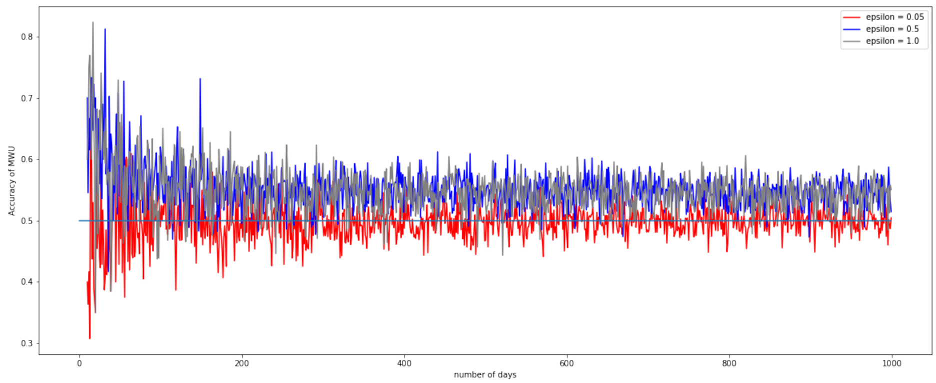
Using different expert-four will not make a difference when the number of days is large. However, when the number of days is small, the lower bound of WMA with the second type of Expert-Four is higher than the WMA with the first type of Expert-Four. That is to say, increasing the expert's quality, can only improve the accuracy of WMA when the number of days is small.

(2) MWU

Same as before, I set the range of the testing day from 10 to 1000. First test with the first Expert-Four. The line chart of the accuracy of MWU changes as the number of days increase is as follows:



Then test with the second Expert-Four:



As the number of days increases, the accuracy gradually tends to be stable, oscillating between 0.5 and 0.6, when $\epsilon \geq \frac{1}{2}$. When $\epsilon \leq \frac{1}{2}$ accuracy tends to oscillating between 0.45 and 0.55. The lower bound and the upper lower of the latter are both smaller then former. When $\epsilon \geq \frac{1}{2}$, the increase of ϵ can't improve the accuracy anymore.

The analysis results for experts of different quality are the same as those of WMA, that is, only when the number of days is small, the MWU with high quality experts have a higher lower bound than those with low quality experts.

(3) MWU vs WMA

It can be seen from the previous analysis that as the number of days increases, the accuracy of using WMA and using MWU($\epsilon \geq \frac{1}{2}$) is not much different. When the number of days is small, the accuracy of using MWU($\epsilon \geq \frac{1}{2}$) is relatively high compare with using WMA.

...

References

[BE] Borodin & El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press.