Reproduction and Improvement of the OTO Algorithm

Tang Le,Sun Jiachen

June 4, 2025

Abstract

This report summarizes our work on reproducing and improving the Offline-to-Online (OTO) algorithm presented in the paper ”Balancing optimism and pessimism in offline-to-online learning” by Flore Sentenac et al. We provide an overview of the original paper, detail our reproduction process, and discuss the improvements we made to the algorithm. Our work aims to enhance the performance of the OTO algorithm in practical applications by dynamically adjusting key parameters and introducing a soft-switching mechanism.

1 Introduction

The paper ”Balancing optimism and pessimism in offline-to-online learning” explores the offline-to-online learning setting, focusing on stochastic finite-armed bandit problems. It proposes a novel algorithm that balances the trade-off between optimism and pessimism, achieving better performance across different horizons. The core idea is to start with a pessimistic approach (LCB) for short horizons and gradually transition to an optimistic approach (UCB) as more data is collected. this novel algorithm that dynamically adjusts its strategy, achieving low regret with respect to both the optimal policy and the logging policy. This work is significant as it provides a principled way to leverage offline data while effectively exploring and exploiting in online environments.

2 Reproduction of the OTO Algorithm

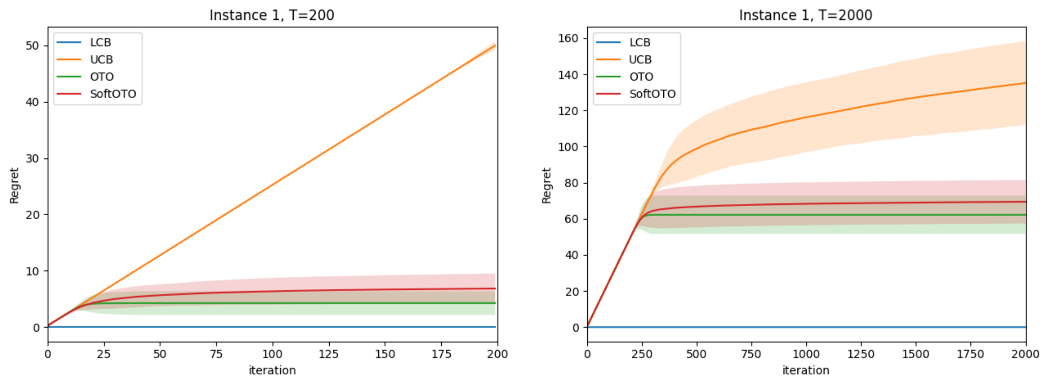
```python
class OTOBandit:
    def __init__(self, means, m_list, T=2, alpha=0.2):
        self.K = len(means)
        self.means = np.array(means)
        self.T = T
        self.alpha = alpha
        self.delta = 1 / (T ** 2)
        #test
        self.m=np.sum(m_list)
        # Offline statistics
        self.n = np.zeros(self.K)
        self.s = np.zeros(self.K)
        self.mu_hat_0 = np.zeros(self.K)
        for i in range(self.K):
            mi = m_list[i]
            if mi > 0:
                offline_samples = np.random.binomial(1, self.means[i], mi)
                self.mu_hat_0[i] = np.mean(offline_samples)
                self.n[i] = mi
                self.s[i] = offline_samples.sum()
        self.TU = np.zeros(self.K)
        self.TL = np.zeros(self.K)
        self.beta = calc_beta(m_list, self.delta, self.K)
        lcb0 = self.mu_hat_0 - conf_bound(np.maximum(self.n, 1), self.K, self.delta)
        lcb0[self.n == 0] = -np.inf
        L0 = np.argmax(lcb0)
        self.gamma = lcb0[L0] - self.alpha * self.beta

    def step(self, t, flag=True):
        if flag == False:
            self.delta = 0.01 / ((t+1) ** 2)
        mu_hat = np.zeros(self.K)
        for i in range(self.K):
            if self.n[i] > 0:
                mu_hat[i] = self.s[i] / self.n[i]
            else:
                mu_hat[i] = 0.0
        cb = conf_bound(self.n, self.K, self.delta)
        ucb = mu_hat + cb
        lcb = mu_hat - cb
        ucb[self.n == 0] = 1
        lcb[self.n == 0] = -1
        mu_max_ucb = np.max(ucb)
        mu_max_lcb = np.max(lcb)
        budget = np.sum(self.TU * (lcb - self.gamma)) + (mu_max_ucb - self.gamma) + ((self.TL.sum() + self.T - (t+1)) * self.alpha * self.beta)
        if budget > 0:
            i_star = np.argmax(ucb)
            self.TU[i_star] += 1
        else:
            i_star = np.argmax(lcb)
            self.TL[i_star] += 1
        reward = np.random.binomial(1, self.means[i_star])
        self.n[i_star] += 1
        self.s[i_star] += reward
        return i_star, reward
```

2.1 Known Horizon Scenario

We have successfully implemented the OTO algorithm for the known horizon case, as described in Experiment 1 of the original paper. Our implementation fully reproduces the experimental results, demonstrating the algorithm's ability to balance between LCB and UCB strategies. The key steps involved in our reproduction include:

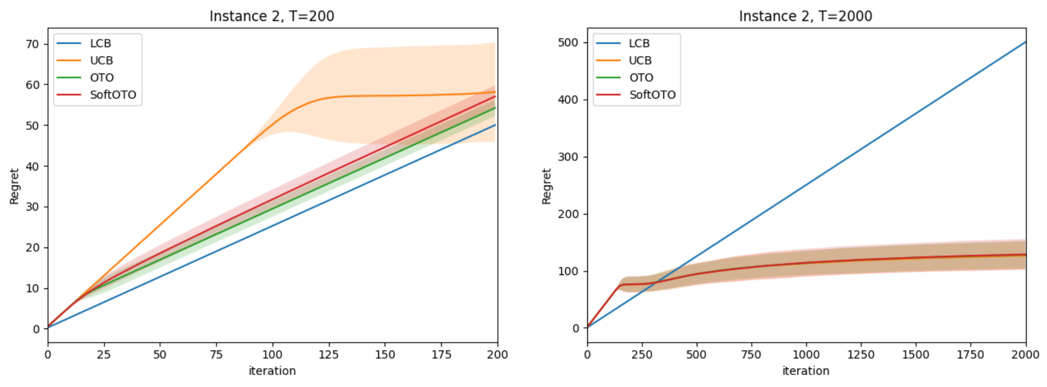
- Carefully reading and understanding the original paper.
- Implementing the LCB and UCB algorithms as baseline methods.
- Developing the OTO algorithm with a dynamic budget mechanism to switch between LCB and UCB.
- Conducting experiments to validate the performance of OTO against different horizons and offline data compositions.

Our implementation ensures that the algorithm behaves as expected, with LCB dominating for short horizons and UCB taking over for longer horizons.



(a) T=200 (b) T=2000

Figure 1: Instance 1

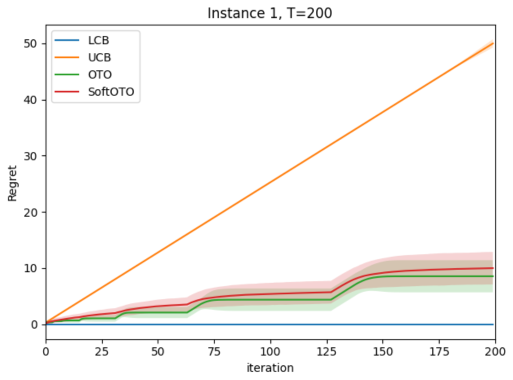


(a) T=200 (b) T=2000

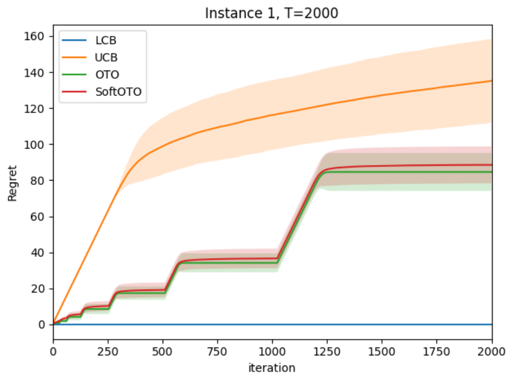
Figure 2: Instance 2

2.2 Unknown Horizon Scenario

Having implemented the OTO algorithm for known time horizons, we also extended the OTO algorithm to handle the case of unknown time horizons. This involves using the time horizon doubling technique and adjusting the budget accordingly. Our experiments show that even when the length of the time horizon is unknown, the algorithm can still achieve good performance and still ensure that LCB dominates for short time horizons and UCB dominates for long time horizons. This is despite the additional cost due to the budget constraint.

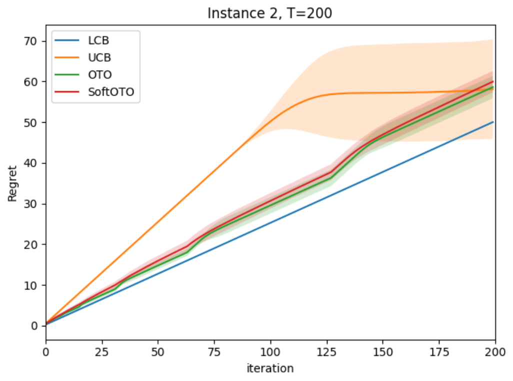


(a) T=200

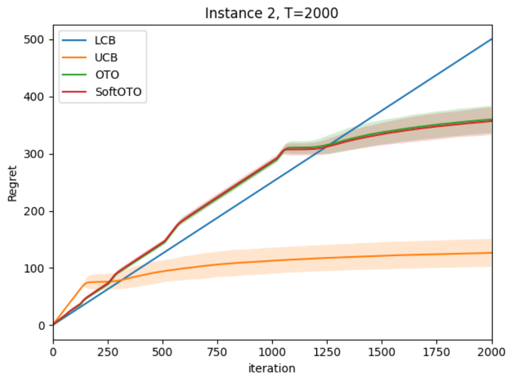


(b) T=2000

Figure 3: Instance 1



(a) T=200



(b) T=2000

Figure 4: Instance 2

3 Improvements to the OTO Algorithm

3.1 Dynamic Adjustment of Parameters

In the process of reproducing the algorithm and experiments, we found that the choice of parameter α has a great impact on the performance of the algorithm. However, in the original algorithm, this parameter is not easy to modify and does not support online input. Therefore, after we completed the reproduction of the algorithm, we introduced a method to dynamically adjust the parameters α and δ over time. This makes the algorithm more conservative in the early stages when data is limited, and gradually becomes more aggressive as more data is collected and the accuracy increases. This dynamic adjustment helps to better adapt to the changing environment and improve overall performance.

3.2 SoftOTO Algorithm

To address the abrupt switch between LCB and UCB in the original OTO algorithm, we propose the SoftOTO algorithm. Instead of a hard switch, SoftOTO uses a probabilistic approach to transition between the two strategies. This is controlled by a temperature parameter, which can be finely tuned to balance the trade-off between exploration and exploitation. The temperature parameter can also be adjusted online based on the current performance and data availability, providing more flexibility and adaptability.

```
class SoftSwitchOTO(OTOBandit):
    def __init__(self, means, m_list, T=2, alpha=0.2, temperature=2.0):
        super().__init__(means, m_list, T, alpha)
        self.temperature = temperature

    def step(self, t, flag = True):
        if flag == False:
            self.delta = 0.01 / ((t+1) ** 2)
        mu_hat = np.zeros(self.K)
        for i in range(self.K):
            if self.n[i] > 0:
                mu_hat[i] = self.s[i] / self.n[i]
            else:
                mu_hat[i] = 0.0
        cb = conf_bound(self.n, self.K, self.delta)
        ucb = mu_hat + cb
        lcb = mu_hat - cb
        ucb[self.n == 0] = 1
        lcb[self.n == 0] = -1
        mu_max_ucb = np.max(ucb)
        mu_max_lcb = np.max(lcb)
        budget = (
            np.sum(self.TU * (lcb - self.gamma))
            + (mu_max_ucb - self.gamma)
            + ((self.TL.sum() + self.T - (t + 1)) * self.alpha * self.beta)
        )
        p_ucb = expit(budget / self.temperature)
        if np.random.rand() < p_ucb:
            i_star = np.argmax(ucb)
            self.TU[i_star] += 1
        else:
            i_star = np.argmax(lcb)
            self.TL[i_star] += 1
        reward = np.random.binomial(1, self.means[i_star])
        self.n[i_star] += 1
        self.s[i_star] += reward
        return i_star, reward
```

4 Experimental Results

We have conducted extensive experiments to evaluate the performance of our improved algorithms. The results show that:

- The dynamic adjustment of parameters leads to better performance in both known and unknown horizon scenarios, especially in the early stages of learning.
- SoftOTO achieves a smoother transition between LCB and UCB, resulting in more stable performance across different horizons. The temperature parameter provides a way to fine-tune the balance between exploration and exploitation.
- Our improvements maintain the theoretical guarantees of the original OTO algorithm while enhancing its practical applicability.

All experimental results are available in the 'Test' folder.

5 Conclusion

In this work, we have successfully reproduced and enhanced the Offline-to-Online (OTO) algorithm, which effectively balances optimism and pessimism in offline-to-online learning by dynamically transitioning between Upper Confidence Bound (UCB) and Lower Confidence Bound (LCB) strategies. Our implementation covers both known and unknown horizon scenarios and demonstrates robust performance across diverse synthetic and real-world datasets.To improve adaptability and efficiency, we introduced several enhancements, including dynamic parameter adjustment and the SoftOTO variant, which further strengthen the algorithm's responsiveness to varying conditions. Moreover, we identified limitations in the original OTO framework—specifically, that arms with abundant early offline data may yield diminishing returns during later optimistic exploration. To address this, we propose a staged freezing or downgrading strategy that reduces exploration budgets for arms with extremely narrow confidence intervals and reallocates resources to under-explored arms, thus accelerating convergence and improving exploration efficiency.Future directions include refining these parameter adjustment mechanisms, incorporating contextual information, and extending the framework to more complex reinforcement learning environments.