Bandit Convex Optimization with Unbounded Memory

Jiho Cha
jdc394@cornell.edu
Cornell University
Ithaca, New York, USA

ABSTRACT

Online Convex Optimization (OCO) is a popular framework to model online sequential decision making by a learner, making it an invaluable tool for machine learning and data science. In the traditional framework, a learner chooses a decision from a decision set in each round and suffers a loss. That loss is solely based on the decision made in the current round, and is chosen in an adversarial fashion. Afterwards, the learner learns the cost associated with each decision they could have made in that round for future reference. However, in many applications of OCO, the learner's loss may also be contingent on past decisions, and they may not get the luxury of learning what other decisions would have costed them in hindsight.

In this paper, we consider a previously established generalization for OCO that captures long-term dependence on past decisions and expand upon it by also limiting the feedback received by the learner from full to bandit. Under this framework, we present and analyze a general method for taking any Bandit Convex Optimization (BCO) with Memory algorithm and converting it into a BCO with Unbounded Memory algorithm with asymptotically negligible overhead. Next, we present an algorithm for the BCO with Memory problem, which may be of independent interest. We prove an $O(T^{2/3})$ regret bound for this algorithm, which is, to the best of our knowledge, the lowest presented regret bound for general convex functions. Together, our results imply the existence of agents that can make decisions optimally, even in difficult and complex systems that traditional OCO may not realistically model.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Theory of computation** → **Online algorithms**; • **Mathematics of computing** → **Convex optimization**.

KEYWORDS

Machine Learning, Optimization, Online Algorithms

ACM Reference Format:
Jiho Cha. 2024. Bandit Convex Optimization with Unbounded Memory. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

KDD'24, August 2024, Barcelona, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

There exist many scenarios within data science that can be described as a learner making multiple sequential decisions within an environment. Such examples include predictions from expert advice [16], recommendation systems [5, 13], and routing [4]. Furthermore, reinforcement learning as a whole hinges upon this principle. Online Convex Optimization (OCO) [12] has become a popular framework for modelling these sequential decision-making problems. At a high level, the OCO framework can be described by the following game, played over T rounds: in each round, an adversary chooses a convex loss function f_t over a convex set, and a learner chooses a decision in the function's domain, without prior knowledge of f_t. The learner then suffers a loss, which is the value of f_t evaluated at their decision. Subsequently, the learner is given knowledge of f_t, which can be used to aid future decisions. The traditional benchmark to evaluate an algorithm's performance is the notion of regret, i.e. the difference between the total cost incurred by the algorithm and the cost that would be incurred from playing the best fixed decision in every round. Regret is typically measured with respect to the number of rounds the above game is played - e.g. an algorithm with regret $O(\sqrt{T})$ would achieve a regret on the order of 100 after 10000 rounds of play. In general, we strive to design algorithms with a sublinear regret bound, as this implies that our algorithm makes choices that attain less regret as the game goes on. In previous iterations of KDD, OCO has been used to model portfolio selection [7], detect anomalies [18], and learn with fairness [21].

This paper explores two enhancements to the traditional OCO framework that allow it to capture applications lying beyond the original scope of the model. First, in many applications of OCO, the cost of a learner in a given round depends on both their current and past decisions. In the case of online linear control [1], even the learner's first decision could affect every subsequent decision. The loss functions in conventional OCO only depend on the latest decision, making it difficult, if not impossible, to capture long term dependence. Second, the learner may receive bandit feedback, namely, when they only learn the loss value that they received after each round, rather than the full loss function.

Recommendation systems are a good example of an application that encounters both issues. A host such as Netflix or Spotify, after choosing a piece of media to recommend to a user, will not know what would happen if they instead chose another piece to recommend at that time. Furthermore, former recommendations may have an impact on the user's current response (e.g. recommending the same song over and over may incline a user to click on it). Therefore, it is important to study these two generalizations of OCO separately, as well as to understand how they interact with one another, in order to properly model how learners can learn efficiently even in this complex, difficult, yet often more realistic

setting. Despite there being work in this effort for a constant memory length with bandit feedback [9, 10], they are unable to consider the entire history of decisions, which is important in many novel applications of OCO.

In this paper, we derive regret bounds for the Online Convex Optimization with Unbounded Memory framework introduced by Kumar et al. [14] but with bandit feedback, rather than full feedback. We show that reductions from the unbounded memory case to the constant memory case incur no additional regret asymptotically. This suggests that arbitrarily increasing the memory length to capture the entire history of decisions will not experience any additional loss. Additionally, we utilize a technique inspired by [3] to modify any $O(\sqrt{T})$ regret memoryless (i.e. memory length of 1) BCO algorithm (e.g. [6, 20]) to an $O(T^{2/3})$ regret BCO-M (BCO with finite memory) algorithm, which might be of independent interest. This is, to the best of our knowledge, the first proven result to achieve an $O(T^{2/3})$ upper regret bound for BCO-M with general convex functions. Together, these results show an $O(T^{2/3})$ upper regret bound for BCO with Unbounded Memory (BCO-UM). This discovery lets us create agents that can quickly learn and make optimal decisions in settings that are more realistic than what OCO and its existing generalizations could model previously.

1.1 Related work

Our paper mostly builds off of the work established by Kumar et al. They develop a generalization of the traditional OCO framework that allows the learner's loss to depend on the entire history of decisions. However, they leave open the case where the learner observes bandit feedback as future work. Other papers on the topic of OCO with memory [2] and its generalizations [17, 22], including BCO-M [10], do not go beyond a constant memory length. Our work aims to bridge the gap between these two frameworks by finding an algorithm that attains sublinear regret with both bandit feedback and with an unbounded memory length.

Recently, Suggala et al. [20] details an algorithm that achieves $O(T^{2/3})$ regret for BCO-M. However, they make additional quadratic and smoothness assumptions on the loss functions, whereas our algorithm can apply to general convex functions. This algorithm is heavily inspired by [3], which also details a reduction to memoryless algorithms.

2 PRELIMINARIES

Due to space constraints, we refer the reader to [12], [19], and [15] for a comprehensive introduction to OCO and bandits. In this paper, we consider an instance of the OCO with Unbounded Memory framework developed by Kumar et al., except we modify the feedback from full to bandit. An instance is specified by the tuple $(\mathcal{X}, \mathcal{H}, A, B)$, where $\mathcal{X}$ is our decision space (i.e. the set of all possible decisions our learner can choose from), $\mathcal{H}$ is our history set, and $A : \mathcal{H} \to \mathcal{H}$ and $B : \mathcal{X} \to \mathcal{H}$ are linear operators that update the history to incorporate our most recent decision using an update equation that we present in the following paragraph. Note that $\mathcal{X}$ and $\mathcal{H}$ are subsets of vector spaces, and that their dimensions need not be finite.

In each round $t \in [T]$, the adversary picks a convex function $f_t : \mathcal{H} \to \mathbb{R}$. The learner then chooses $x_t \in \mathcal{X}$, the history is updated via $h_t = Ah_{t-1} + Bx_t$, and the learner suffers loss $f_t(h_t)$. This value is the only information that is revealed to the learner in each round. The learner's objective is to minimize the expected loss that they incur.

We also adopt the following notation from [14]:

Definition 2.1. Given $f_t : \mathcal{H} \to \mathbb{R}$, the function $\tilde{f}_t : \mathcal{X} \to \mathbb{R}$ is defined by $\tilde{f}_t(x) := f_t(\sum_{i=0}^{t-1} A^i Bx)$.

Note that an algorithm that only plays x in every round will result in a history vector $h_t = \sum_{i=0}^{t-1} A^i Bx$. The previous definition captures the loss such an algorithm will suffer at round t. We now formally introduce the definition of regret, which is the metric used to measure an algorithm's performance:

Definition 2.2. The regret of an algorithm $\mathcal{A}$ is defined to be

$$\text{Regret}_{\mathcal{A}} = \sum_{t=0}^{T} f_t(h_t) - \min_{x \in \mathcal{X}} \sum_{t=0}^{T} \tilde{f}_t(x).$$

In layman's terms, the regret of the learner is the difference between an algorithm's total accumulated loss and the minimum accrued cost over all algorithms whose strategy is to play the same decision in every round.

We also make the following assumptions about the model, the loss functions, and the environment:

1. The learner has prior knowledge of the operators A and B during the course of the algorithm.
2. The operator A takes the form $A(y_0, y_1, y_2, ...) = (0, A_0 y_0, A_1 y_1, ...)$, where $A_i : \mathcal{X} \to \mathcal{X}$ are linear operators and have operator norm strictly less than 1.
3. B satisfies $B(x) = (x, 0, 0, ...)$.
4. The functions f_t are convex.
5. The functions f_t are L-Lipschitz continuous.
6. The diameter of the convex decision space is at most D, i.e. $\forall x, x' \in \mathcal{X}, ||x - x'|| \leq D$.

Assumptions 1, 4, 5, and 6 are common in the previous literature on OCO. Furthermore, Assumptions 2 and 3 capture the notion that in many motivating examples, the history at the end of a round is a sequence of linear transformations of past decisions. Furthermore, the additional assumption that the A_i's have operator norm strictly less than 1 captures the idea that recent decisions tend to have a higher impact on the current loss compared to earlier decisions.

3 ALGORITHM AND REGRET ANALYSIS

We begin by considering the special case where our system follows the dynamics of BCO with ρ-discounted Infinite Memory.

Definition 3.1. (BCO with ρ-discounted Infinite Memory)
A problem is said to follow the dynamics of BCO with ρ-discounted Infinite Memory if A and B are defined to be the following:

$$A(x_t, x_{t-1}, x_{t-2}, ...) = (0, \rho x_t, \rho x_{t-1}, ...)$$
$$B(x) = (x, 0, 0, ...),$$

where $\rho \in (0, 1)$ is a discounting factor that exponentially reduces the effect of past decisions.

Under this definition, an algorithm that picks decisions $x_0, x_1, ..., x_t$ during the first t rounds will have a resulting history of $(x_t, \rho x_{t-1}, \rho^2 x_{t-2}, ...\rho^t x_0)$. Observe that these definitions satisfy Assumptions 2 and 3. Later in this section, we show how our results generalize for any linear operator A and B that satisfy these assumptions.

We now present our strategy for picking the decisions $x_{0:t}$ that attains sublinear regret for the BCO-UM problem, where $x_{0:t}$ is shorthand for $(x_t, x_{t-1}, ..., x_1, x_0)$. We achieve this by performing a reduction to any algorithm that achieves sublinear regret for the BCO-M problem. To illustrate our reduction, we use the algorithm detailed in Section 3 of Gradu et al. [10], which attains a regret bound of $O(T^{3/4})$. However, we note that any algorithm can be used in its place, which we justify later in the paper. In each round, the learner uses a BCO-M algorithm $\mathcal{B}$ to pick a decision $x \in \mathcal{X}$, receives a cost value $f_t(h_t)$, and returns this value back to $\mathcal{B}$ for use in subsequent rounds. This process is made explicit through the following algorithm:

Algorithm 1 BCO with Unbounded Memory

1: **Input:** $\mathcal{X} = \mathbb{R}^d$, T H, δ, $\{\eta_t\}$
2: Initialize $x_1 = ... = x_H \in \mathcal{X}$ arbitrarily.
3: Sample $u_1, ..., u_H$ uniformly from the unit sphere of dimension d.
4: Set $y_i = x_i + \delta u_i \ \forall i \in [H]$.
5: Set $g_i = 0 \ \forall i \in [H]$.
6: Play $y_i \ \forall i \in [H-1]$.
7: **for** $t = H, ..., T$ **do**
8: Play y_t and incur cost $f_t(h_t)$
9: Set $g_t = \frac{d}{\delta} f_t(h_t) \sum_{i=0}^{H-1} u_{t-i}$
10: Set $x_{t+1} = \Pi_{\mathcal{X}}[x_t - \eta_t g_{t-(H-1)}]$
11: Sample u_{t+1} uniformly from the unit sphere of dimension d.
12: Set $y_{t+1} = x_{t+1} + \delta u_{t+1}$.
13: **end for**

In the pseudocode, lines 1-5 emulate how Gradu's algorithm pick the first H iterates, where H is a hyper parameter that controls the length of the memory for the BCO-M algorithm reduction. Line 8 is where we play the decision that Gradu's algorithm would make for all future iterates. Finally, lines 8-11 emulate how Gradu's algorithm determines the next iterate, given the cost value of the current round. In the algorithm, δ and $\{\eta_t\}$ are hyper parameters on the order of $T^{-1/4}$ and $t^{-3/4}$ respectively which dictate the algorithm's learning rate, and $\Pi_{\mathcal{X}}[x]$ is simply the projection of x back onto $\mathcal{X}$ in case the gradient descent step causes x to fall out of the set.

3.1 Regret Analysis

We now derive a regret upper bound for a learner that adheres to the algorithm above. As we are performing a reduction to BCO-M, we wish to consider functions that only depend on a constant number of past decisions to aid our analysis. This motivates the following definition:

Definition 3.2. Let $x_{0:t}$ be our sequence of decisions. This makes our history at time t be $h_t = (x_t, \rho x_{t-1}, ..., \rho^t x_0)$. Now, define

$$F_{t,H}(h_t) = f_t(x_t, \rho x_{t-1}, ..., \rho^{H-1} x_{t-(H-1)}, 0, ..., 0),$$

where f_t is the cost function at time t.

In other words, $F_{t,H}$ is a partial evaluation of f_t on the first H elements of its history. This artificially creates a function with a constant number of arguments by "zero-ing out" every argument that precedes the most recent H decisions. We define $\tilde{F}_{t,H}(x)$ similarly. We are now ready to prove our main theorem:

Theorem 3.3. *Consider the bandit convex optimization with unbounded memory problem. Algorithm 1 satisfies*

$$Regret_{\mathcal{A}} = \sum_{t=0}^{T} f_t(h_t) - \min_{x \in \mathcal{X}} \sum_{t=0}^{T} \tilde{f}_t(x) \leq O(T^{3/4} H^{3/2} d D^{4/3} L^{2/3}).$$

Proof. Consider the following 3 inequalities:

$$\sum_{t=1}^{T} f_t(h_t) - \sum_{t=1}^{T} F_{t,H}(h_t) \leq LD\rho^H \sqrt{H} T$$

$$\sum_{t=1}^{T} F_{t,H}(h_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} \tilde{F}_{t,H}(x) \leq O(T^{3/4} H^{3/2} d D^{4/3} L^{2/3})$$

$$\min_{x \in \mathcal{X}} \sum_{t=1}^{T} \tilde{F}_{t,H}(x) - \min_{y \in \mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(y) \leq LD\rho^H \sqrt{H} T.$$

Note that by triangle inequality, these inequalities imply that

$$\sum_{t=1}^{T} f_t(h_t) - \min_{x \in \mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x) \leq 2LD\rho^H \sqrt{H} T + O(T^{3/4} H^{3/2} d D^{4/3} L^{2/3}).$$

Setting $H = \frac{\log T}{1-\rho}$ makes $2LD\rho^H \sqrt{H} T$ on the order of $LD(1-\rho)^{-1} \frac{\log T}{T}$, which is absorbed by the term $O(T^{3/4} H^{3/2} d D^{4/3} L^{2/3})$. Thus, proving the above 3 inequalities will imply Theorem 3.3.

The second inequality comes immediately from Theorem 3.1 proven by Gradu et al. [10], as the functions present in the left hand side of the inequality are defined to have a constant memory length of H. Thus, this inequality is simply an upper bound on the regret attained by their algorithm.

We bound the first inequality by considering a particular f_t and respective $F_{t,H}$:

$$\begin{aligned}
f_t(h_t) - F_{t,H}(h_t) &= f_t(x_t, \rho x_{t-1}, \rho^2 x_{t-2}, ..., \rho^{H-1} x_{t-(H-1)}, \rho^H x_{t-H}, ...) \\
&\quad - f_t(x_t, \rho x_{t-1}, \rho^2 x_{t-2}, ..., \rho^{H-1} x_{t-(H-1)}, 0, ...) \\
&\leq L||(0, 0, ..., 0, \rho^H x_{t-H}, \rho^{H+1} x_{t-(H+1)}, ...)|| \\
&\leq L \sqrt{\sum_{i=H}^{t} (\rho^i x_{t-i})^2} \\
&\leq L\rho^H \sqrt{\sum_{i=H}^{t} (\rho^{i-H} x_{t-i})^2} \\
&\leq LD\rho^H \sqrt{\sum_{i=H}^{t} \rho^{i-H}} \\
&\leq LD\rho^H \sqrt{H}.
\end{aligned}$$

The first inequality holds from L-Lipschitzness, the second inequality comes from the definition of the norm in the history space

$\mathcal{H}$, and the second to last inequality holds from the fact that the magnitude of decisions are bounded by the diameter of the decision set. Lastly, because $H = \frac{\log T}{1-\rho}$, we have that $H \geq \log T \sum_{i=H}^{t} \rho^i$. Summing this over T rounds gives us the claimed bound.

Next, we bound the third inequality. Let x' be the decision in $\mathcal{X}$ that minimizes the value of $\sum_{t=1}^{T} \tilde{f}_t(x')$. Now, we note that

$$\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{F}_{t,H}(x) - \sum_{t=1}^{T} \tilde{f}_t(x') \leq \sum_{t=1}^{T} \tilde{F}_{t,H}(x') - \sum_{t=1}^{T} \tilde{f}_t(x').$$

We can now follow a similar process as we used for the first inequality to derive the same bound for the right hand side. This finishes the proof of our main theorem. □

We would now like to generalize our findings by relaxing the assumptions we made during our analysis (beyond those made in Section 2). Namely, we argue that any BCO-M algorithm can be used in place of the algorithm in [10] as part of our reduction, and that our analysis will hold for any A and B that follow Assumptions 2 and 3. We begin by addressing the second concern.

Note that we can let ρ equal the maximum operator norm value out of all of the A_is to achieve the same regret bounds. Our analysis only depends on the magnitude of the values in our history vector. Because we assume that the operator norm of each A_i is strictly less than 1, we can find a $\rho' \in (0, 1)$ that is at least the supremum of the set of operator norms. The regret of our algorithm under these more general dynamics must thus be at most the regret of our algorithm under BCO with ρ'-discounted Infinite Memory.

As for the first concern, the crux of our analysis demonstrated that the 3 inequalities in Theorem 3.3 were true. Observe that the specific choice of BCO-M algorithm only affects the middle inequality. The first and third inequality merely account for the discrepancy one encounters when moving from the finite framework to the unbounded framework. The maximum regret that can be incurred from the reduction is on the order of $\frac{\log T}{T}$, which is negligible due to the trivial $O(\sqrt{T})$ lower bound on any OCO algorithm (Chapter 2 of [12]). A corollary of this observation is that our BCO-UM algorithm's regret is dictated by the regret of whatever BCO-M algorithm is being reduced to. As such, to improve regret bounds, we are motivated to find a BCO-M algorithm with regret lower than $O(T^{3/4})$.

3.2 Beyond $T^{3/4}$

In this section, we describe and analyze an algorithm for BCO-M that achieves an $O(T^{2/3})$ regret bound on general convex functions, which to the best of our knowledge, is the first of its kind. Recall that in the BCO-M setting, our history is simply the H most recent decisions, where H is our memory length.

Our algorithm performs another reduction to an $O(\sqrt{T})$ algorithm for the traditional BCO (the memoryless setting) [6, 20]. This reduction process is heavily inspired by Theorem 2 in [3], which also reduce to memoryless algorithms to create learners against adversaries with memory. For the purposes of this paper, we highlight Suggala's algorithm in [20]. At a high level, our algorithm updates its decisions in accordance to Suggala's algorithm once on the order of $T^{1/3}$ rounds. This effectively slows down the rate our decisions move, which is important in attaining a sublinear bound.

Algorithm 2 BCO with Finite Memory Reduction

1: **Input:** $\mathcal{X}, T, H, \eta$
2: Initialize $x_1 = y_1 \in \mathcal{X}$ and play y_1
3: Initialize $A = I$ (the identity matrix)
4: **for** $t = 2, ..., T$ **do**
5: flip coin with success probability $\frac{1}{\sqrt[3]{T}}$
6: **if** coin flip success **then**
7: Draw $v_{t,1}, v_{t,2}$ uniformly from the unit sphere and let $y_t = x_{t-1} + \frac{1}{2}A^{-\frac{1}{2}}(v_{t,1} + v_{t,2})$.
8: Play y_t, update the history, and observe $f_t(y_{t-(H-1):t})$.
9: Set $\nabla_t = 2d f_t(y_{t-(H-1):t})A^{\frac{1}{2}}v_{t,1}$.
10: Set $H_t = 2d^2 f_t(y_{t-(H-1):t})A^{\frac{1}{2}}(v_{t,1}v_{t,2}^\top + v_{t,2}v_{t_1}^\top)A^{\frac{1}{2}}$.
11: Set $A = A + \eta H_t$
12: Set $x_t = x_{t-1} - \eta A^{-1}\nabla_t$
13: **else**
14: Play y_{t-1}
15: Set $x_t = x_{t-1}$
16: **end if**
17: **end for**

In the algorithm, lines 7-11 emulate how Suggala's algorithm picks its decisions. Line 8 is where we actually play the decision, and we use the observed cost alongside Suggala's algorithm to determine the next decision to be picked. We also highlight that randomization is necessary. An algorithm that deterministically switches decisions every $T^{1/3}$ rounds is vulnerable to an adversarial attack.

Theorem 3.4. *Algorithm 2 satisfies the following regret bound:*

$$\mathbb{E}[Regret_{\mathcal{A}}] = \sum_{t=1}^{T} f_t(y_{t-(H-1):t}) - \min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{f}_t(x) \leq O(T^{2/3}).$$

Proof. Suppose we have shown that the following two inequalities hold:

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(y_{t-(H-1):t}) - \tilde{f}_t(y_t)\right] \leq DL\sqrt{H^3}T^{2/3}$$

$$\mathbb{E}\left[\sum_{t=1}^{T} \tilde{f}_t(y_t) - \tilde{f}_t(x^*)\right] \leq O(D^{5/3}L^{2/3}T^{2/3}),$$

where x^* is the fixed decision that minimizes $\sum_{t=1}^{T} \tilde{f}_t(x)$. Then by triangle inequality, we have the desired regret bound. Thus, it is sufficient to prove the two inequalities.

For the first inequality, note that by linearity of expectation, we have that

$$\mathbb{E}\left[\sum_{t=1}^{T} f_t(y_{t-(H-1):t}) - \tilde{f}_t(y_t)\right] = \sum_{t=1}^{T} \mathbb{E}[f_t(y_{t-(H-1):t}) - \tilde{f}_t(y_t)].$$

Now, observe that most of the time, the value $f_t(y_{t-(H-1):t}) - \tilde{f}_t(y_t)$ is 0. The only way this difference is nonzero is if $y_{t-(H-1):t} \neq (y_t, y_t, ..., y_t)$. In expectation, there will be at most $HT^{2/3}$ nonzero summands, since the only time when $y_t \neq y_{t+1}$ is when success is

met using a coin with probability $1/T^{1/3}$. Every time a coin achieves success, the next H rounds will incur a nonzero loss, as it takes that much time for the "old" decision to fall out of memory. Thus, in expectation, there are at most $HT^{2/3}$ rounds that have nonzero regret. We complete the proof of the first inequality by showing that each of these rounds can incur at most $LD\sqrt{H}$ regret through the following lemma. □

Lemma 3.5. *For any 2 arbitrary histories $y_{t-(H-1):t}$, $y'_{t-(H-1):t}$ we have that $f_t(y_{t-(H-1):t}) - f_t(y'_{t-(H-1):t}) \leq LD\sqrt{H}$.*

Proof.

$$\begin{aligned} f_t(y_{t-(H-1):t}) - f_t(y'_{t-(H-1):t}) &\leq L||y_{t-(H-1):t} - y'_{t-(H-1):t}|| \\ &\leq L\sqrt{\sum_{i=0}^{H-1}(y_{t-i} - y'_{t-i})^2} \\ &\leq LD\sqrt{\sum_{i=0}^{H-1} 1} \\ &\leq LD\sqrt{H} \end{aligned}$$

□

To finish the proof of Theorem 3.4, we end by proving the second inequality.

Lemma 3.6. *We have that*

$$\mathbb{E}[\sum_{t=1}^{T} \tilde{f}_t(y_t) - \tilde{f}_t(x^*)] \leq O(D^{5/3}L^{2/3}T^{2/3}).$$

Proof. Our algorithm can be thought of as Suggala's algorithm in [20] with a time horizon of $T^{2/3}$, since we only update our decision $T^{2/3}$ times in expectation. Suggala's algorithm has an upper regret bound of $O(D^{5/3}L^{2/3}T^{1/3})$ over a time horizon of $T^{2/3}$. However, our algorithm, in expectation, will play the same decision $T^{1/3}$ times instead of just once. This increases the regret attained by a factor of $T^{1/3}$, giving us an $O(T^{2/3})$ regret bound in total. □

4 EXPERIMENTAL RESULTS

In this section, we present the results of some simulations that test our theoretical findings. We refer the reader to Appendix A for details on our setup and implementation.

4.0.1 BCO with ρ-discounted Infinite Memory. First, it behooves us to check that our algorithm does in fact attain the claimed regret bound of $O(T^{2/3})$. We run our algorithm under the dynamics of BCO with ρ-discounted Infinite Memory against a simple yet robust adversary with several values of ρ, namely 0.1, 0.3, 0.5, and 0.7, over a time horizon of $T = 5000$. For each of these values, we run our algorithm 10 times and we plot the average regret over time.

Figure 1 shows the regret that the algorithm attains over time for each rho value. Note the concavity in the plots for $\rho = 0.1, 0.3$, and 0.5, supporting the fact that our algorithm is learning over time, accruing less regret as it learns more about the environment. Additionally, we expect our regret after 5000 rounds to be in the low thousands after considering the additional parameters that are hidden by the Big-O notation. We see that this is indeed the case.

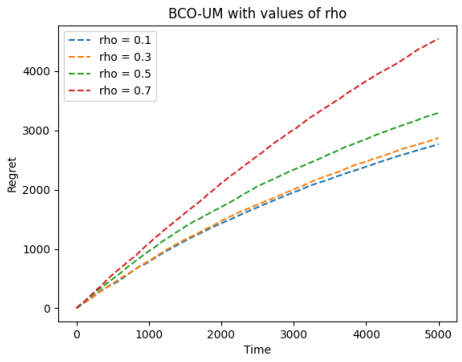

Figure 1: A plot of the accumulated regret of Algorithm 2 against an adversary that chooses the loss functions as described in Section 4.1 with a variety of ρ values.

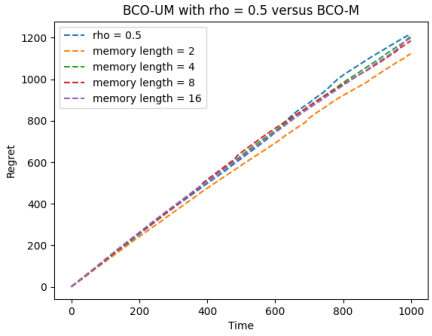

Figure 2: A plot comparing the effect of different memory lengths on our algorithm's regret.

Furthermore, we observe that a lower value of ρ seemingly results in smaller regret, which makes sense, as our algorithm is penalized less for costly decisions that were made in the past.

Larger values of ρ make it difficult for the algorithm to learn from its prior decisions and thus take longer to make progress towards the global minimum. This is why our plot for $\rho = 0.7$ does not seem to follow the same trend as the others. Indeed, running the experiment again with $\rho = 0.7$, but with a time horizon of $T = 10000$ produces a graph that exhibits a similar shape to the graphs shown above (see Figure 4, in Appendix A due to space constraints).

4.0.2 BCO-UM versus BCO-M. Next, we empirically show that our reduction does not incur much regret asymptotically, as we have proved theoretically above. For this test, we first run our algorithm against an instance of the BCO with ρ-discounted infinite memory problem, with $\rho = 0.5$ to establish a baseline. Afterwards, we run our algorithm against the BCO-M problem with memory lengths 2, 4, 8, and 16. We run each experiment 100 times and we plot the average regret over those 100 iterations over time in Figure 2.

As one can see, an environment with a longer memory length tends to incur more regret. In particular, there is a noticeable difference in regret between our BCO-UM instance and the remaining

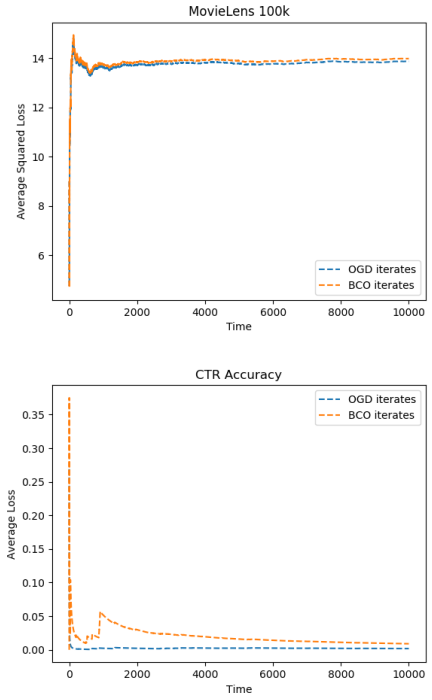

Figure 3: The average squared loss of our BCO-UM algorithm versus the prototypical OGD algorithm over time, when used for matrix completion (top) and click-through rate prediction (bottom).

BCO-M instances. However, this difference is small, offering evidence that the cost of performing our reduction is asymptotically negligible.

4.0.3 Application to Real World Systems. Finally, although the findings in this paper are mostly theoretical in nature, we demonstrate the applicability of our framework by experimenting with two real world applications, namely collaborative filtering for recommendation systems and click-through rate prediction for ad placement. Full implementation details are provided in the appendix. We hope that these results illustrate the capability of our work to capture applications that lie out of the current scope of OCO and its existing generalizations, without an unreasonable inflation of regret.

In both experiments, we compare the average loss attained by Online Gradient Descent (OGD), the prototypical algorithm used to minimize regret for the traditional OCO framework, to the average loss attained by our algorithm in the BCO-UM framework. We run both algorithms for 10000 iterations, and share the results in Figure 3.

In both experiments, we note that the loss values that our BCO-UM algorithm attains are higher than the loss values attained by the OGD algorithm. However, our BCO-UM setting better models and handles the complexity of the systems inherent in these problems, compared to the traditional (and more naive) OCO model. Therefore, the results attained by OGD may be unrealistic to compare to, and the difference in algorithmic performance is likely caused by the more difficult setting. We also see that the difference in performance is overall greater for the CTR dataset. One potential explanation behind this is that the gradient estimate that our BCO-UM algorithm uses happens to be more similar to the gradient of our loss function for matrix completion compared to click-through rate prediction. In both experiments, our gradient estimate for the BCO-UM setting is a linear function of our decision. However, our loss function for CTR involves the sigmoid function, whereas the loss function for matrix completion is a simple quadratic function of our decision (whose gradient is therefore another linear function of our decision). Thus, the behavior of our BCO-UM algorithm is similar to the behavior of OGD on the MovieLens dataset, resulting in two similar plots.

5 CONCLUSION

We have presented and analyzed a general method for taking a BCO-M algorithm and converting it into a BCO-UM algorithm with asymptotically negligible overhead. This implies that increasing the memory length of a BCO-M problem to incorporate the entire history of decisions will not increase regret asymptotically. Additionally, we presented an algorithm that achieves $O(T^{2/3})$ regret for general convex functions. In conjunction, our two findings show an $O(T^{2/3})$ regret bound for the BCO-UM problem.

Our findings allow the techniques developed for regret minimization for the BCO-M problem to immediately apply to the BCO-UM problem. Consequently, any BCO-M algorithm that improves on the current literature automatically implies a new novel algorithm for the BCO-UM problem. Our results enable the popular OCO framework to more realistically model applications whose loss at the current round depends on the entire history of past decisions. This is important for developing agents that can make decisions optimally even in complex and dynamic systems.

5.1 Future Work

One direction for future work is to relax the assumptions we make. In particular, Assumptions 2 and 3 are required for our reduction from the BCO-UM to BCO-M problem. We want our history to be in a form that makes it easy to compare with the finite memory framework. Without the structure we assume A and B have, it would be hard to directly compare the two otherwise. A potential solution may be an algorithm for the BCO-UM problem that does not rely on reductions.

Lastly, a natural question to ask is whether or not the $O(T^{2/3})$ regret bound for BCO-M can be lowered even further. There is evidence to suggest that the regret lower bound for this problem is $\Omega(T^{2/3})$, since in [20], it is shown that against an adaptive adversary, an $\Omega(T^{2/3})$ lower bound exists. However, against an oblivious adversary, an analogous $\Omega(T^{2/3})$ lower bound is yet to be proven. If this is the lower bound, our earlier algorithm would achieve a tight bound for both the BCO-M and BCO-UM problem.

6 ACKNOWLEDGEMENTS

We thank Cornell Professor Robert Kleinberg and his student, Raunak Kumar, for their guidance throughout this research project. This includes, but is not limited to, sending us existing literature on this subject for inspiration, contributing ideas for algorithms and their respective proofs, and verifying the validity of our work.

REFERENCES

[1] Naman Agarwal, Brian Bullins, Elad Hazan, Sham M. Kakade, and Karan Singh. 2019. Online Control with Adversarial Disturbances. arXiv:1902.08721 [cs.LG]

[2] Oren Anava, Elad Hazan, and Shie Mannor. 2014. Online Convex Optimization Against Adversaries with Memory and Application to Statistical Arbitrage. arXiv:1302.6937 [cs.LG]

[3] Raman Arora, Ofer Dekel, and Ambuj Tewari. 2012. Online Bandit Learning against an Adaptive Adversary: from Regret to Policy Regret. arXiv:1206.6400 [cs.LG]

[4] Baruch Awerbuch and Robert Kleinberg. 2008. Online linear optimization and adaptive routing. *J. Comput. System Sci.* 74, 1 (2008), 97–114. https://doi.org/10.1016/j.jcss.2007.04.016 Learning Theory 2004.

[5] Gábor Braun, Alejandro Carderera, Cyrille W. Combettes, Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Sebastian Pokutta. 2023. Conditional Gradient Methods. arXiv:2211.14103 [math.OC]

[6] Sébastien Bubeck, Ronen Eldan, and Yin Tat Lee. 2016. Kernel-based methods for bandit convex optimization. arXiv:1607.03084 [cs.LG]

[7] Puja Das and Arindam Banerjee. 2011. Meta optimization and its application to portfolio selection. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Diego, California, USA) *(KDD '11)*. Association for Computing Machinery, New York, NY, USA, 1163–1171. https://doi.org/10.1145/2020408.2020588

[8] Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. 2004. Online convex optimization in the bandit setting: gradient descent without a gradient. *CoRR* cs.LG/0408007 (2004). http://arxiv.org/abs/cs.LG/0408007

[9] Udaya Ghai, Arushi Gupta, Wenhan Xia, Karan Singh, and Elad Hazan. 2023. Online Nonstochastic Model-Free Reinforcement Learning. arXiv:2305.17552 [cs.LG]

[10] Paula Gradu, John Hallman, and Elad Hazan. 2020. Non-Stochastic Control with Bandit Feedback. arXiv:2008.05523 [cs.LG]

[11] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (dec 2015), 19 pages. https://doi.org/10.1145/2827872

[12] Elad Hazan. 2023. Introduction to Online Convex Optimization. arXiv:1909.05207 [cs.LG]

[13] Elad Hazan and Satyen Kale. 2012. Projection-free Online Learning. *CoRR* abs/1206.4657 (2012). arXiv:1206.4657 http://arxiv.org/abs/1206.4657

[14] Raunak Kumar, Sarah Dean, and Robert Kleinberg. 2024. Online Convex Optimization with Unbounded Memory. arXiv:2210.09903 [cs.LG]

[15] Tor Lattimore and Csaba Szepesvari. 2017. Bandit Algorithms. (2017). https://tor-lattimore.com/downloads/book/book.pdf

[16] N. Littlestone and M. Warmuth. 1989. The weighted majority algorithm. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Los Alamitos, CA, USA, 256–261. https://doi.org/10.1109/SFCS.1989.63487

[17] Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. 2021. Online Optimization with Memory and Competitive Control. arXiv:2002.05318 [cs.LG]

[18] Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, Ryan Wright, Alec Theriault, and David W. Archer. 2018. Feedback-Guided Anomaly Discovery via Online Optimization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2200–2209. https://doi.org/10.1145/3219819.3220083

[19] Aleksandrs Slivkins. 2024. Introduction to Multi-Armed Bandits. arXiv:1904.07272 [cs.LG]

[20] Arun Suggala, Y. Jennifer Sun, Praneeth Netrapalli, and Elad Hazan. 2024. Second Order Methods for Bandit Optimization and Control. arXiv:2402.08929 [cs.LG]

[21] Chen Zhao, Feng Chen, and Bhavani Thuraisingham. 2021. Fairness-Aware Online Meta-learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 2294–2304. https://doi.org/10.1145/3447548.3467389

[22] Peng Zhao, Yu-Hu Yan, Yu-Xiang Wang, and Zhi-Hua Zhou. 2023. Non-stationary Online Learning with Memory and Non-stochastic Control. arXiv:2102.03758 [cs.LG]

A REPRODUCIBILITY

In this section, we dive into the setup and implementation behind our experiments in Section 4. Our code can be found at https://github.com/jihoc1748/bco-with-memory-code.

A.1 Simulation 4.0.1 and 4.0.2 Setup

We consider an instance of the BCO with ρ-discounted Infinite Memory problem, with $\mathcal{X} = \mathbb{R}^2$. Let h'_x be the history obtained after picking the same decision x for the first t rounds. We define the cost functions to be $f_t(h_t) = ||h_t - h'_x||_{\mathcal{H}}, \ \forall t \in [T]$. If we are in the finite memory setting, we "zero-out" the history terms beyond the memory length, otherwise we use the full history as is. The adversary picks x from the decision set uniformly at random. Note that the best strategy in hindsight would be to pick x in every round, resulting in a cumulative cost of 0. Thus, the regret of an algorithm is the same as its cumulative cost. We define the norm of a history vector as such:

$$||h||_{\mathcal{H}} = ||(y_0, y_1, ...)||_{\mathcal{H}} = \sqrt{\sum_{i=0}^{T} (y_i)^2}$$

Finally, we restrict the domain of the decision set from $\mathbb{R}^2$ to $[-1, 1]^2$. This simply amounts to a re-scaling of the problem and this condition could easily be relaxed.

A.2 Simulation 4.0.3 Setup

For our first real-world experiment, we consider the problem highlighted in [13], outlined here for completion. However, we refer the reader to the original paper and to Chapter 7 in [12] for a comprehensive overview of the underlying topics. In the collaborative filtering model, we have m users, and we would like to know their opinion on n items. This can be represented by an m by n matrix M, where the entry at row i and column j indicates user i's rating of item j. However, in many cases, there are too many items for every user to reasonably rate, which leaves many unknown entries in the matrix. Naturally, we wish to estimate these entries to provide the best recommendations. While doing so, we would also like to limit the rank of the matrix, which can be interpreted as limiting the number of factors that determine a user's rating of an item. This problem is often referred to as the Matrix Completion problem.

We formulate Matrix Completion as an OCO problem via the following. In each round, the learner produces an m by n matrix X with rank less than τ. This can be thought of as providing each user $i \in [m]$ a rating for each item $j \in [n]$. The adversary then chooses a particular entry (i, j) and reveals the true rating y for that user-item interaction. The loss that the learner suffers is determined by the function $f_t(X) = (X_{ij} - y)^2$, i.e. squared loss. In the BCO-UM setting, we modify the above scenario by only providing bandit feedback to the learner and by modifying the loss function to $f_t(h_t) = \frac{1}{2} * \sum_{k=0}^{T} \rho^k (X_{ij}^{t-k} - y)^2$, where X^i is the decision matrix that the learner plays at round i. Intuitively, we are taking a weighted sum of how well our previous decision matrices would predict the rating that user i gives item j in the current round.

We use the old MovieLens 100K Dataset provided by GroupLens [11] (at https://grouplens.org/datasets/movielens/100k/) to simulate the above problem. This dataset provides 100000 ratings (on an integer scale from 1 through 5) by 943 users on 1682 movies. For simplicity, we use the original sequence of the dataset to choose the entries (i, j) that the adversary uses.

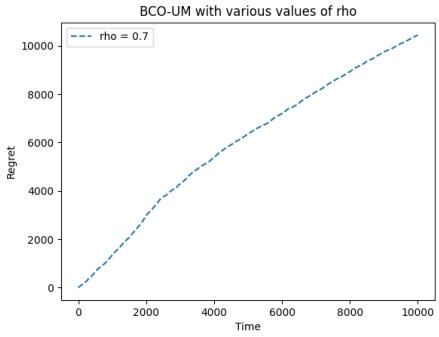

Figure 4: For a larger value of ρ, it takes our algorithm a longer time to converge to an optimal solution. For $\rho = 0.7$, it takes roughly 2000 time steps before we make substantial progress.

For our second experiment, we consider the problem of click-through rate prediction using the Criteo Click Logs dataset (at https://ailab.criteo.com/download-criteo-1tb-click-logs-dataset/). In this problem, we are given a feature vector a of dimension d representing an ad and we'd like our model to output a 1 if we believe the user will click on it, and a 0 otherwise (although our model can output any value in this interval depending on its confidence). We formulate this problem as an OCO problem via the following. In each round, the learner produces a vector $x \in \mathbb{R}^d$, which can be thought of as a filter used to differentiate favorable and unfavorable ads. Our learner classifies our vector by taking the sigmoid of the inner product between the feature vector and its filter, i.e. $y = \frac{1}{1+e^{(-x^\top a)}}$. If y' is the true label, then the loss our learner attains is $f_t(x) = \frac{1}{2}(y - y')^2$. For the BCO-UM setting, we again only provide bandit feedback to the learner and modify the loss function to $f_t(h_t) = \frac{1}{2} * \sum_{k=0}^{T} \rho^k (y_{t-k} - y')^2$, where y_{t-k} is the inner product of a and our decision at round $t - k$. Again, we can intuitively think of this function as taking a weighted sum of how well our previous filters would have classified the ad categorized by the feature vector received in the current round.

We run both experiments 2 times. For the first run, we consider the original OCO setting and use Online Gradient Descent as the learner's algorithm. For the second run, we consider the BCO-UM setting and use the classic FKM algorithm [8] as the basis for our learner. We opt for this algorithm over the ones we implemented previously because our decision set for the first experiment is no longer a subset of $\mathbb{R}^d$, which is a requirement for the other BCO algorithms discussed in this paper.

A.3 Implementation

We implement all learning algorithms with Python 3.9, alongside the numpy package to assist with matrix manipulations. All algorithms were implemented as straightforwardly as possible, following directly from the respective pseudocode. The matplotlib library was used to plot our regret graphs.