ADAPTIVE ONLINE LEARNING AND ITS APPLICATIONS IN INVENTORY MANAGEMENT

A Dual Degree Project Report

submitted by

RISHHANTH MAANAV V

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
&
MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2021

THESIS CERTIFICATE

This is to certify that the thesis titled **Adaptive Online Learning and Its Applications In Inventory Management**, submitted by **Rishhanth Maanav V (EE16B036)**, to the Indian Institute of Technology, Madras, in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Abhishek Sinha
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 15th June 2021

ACKNOWLEDGEMENTS

Firstly, I would like to thank my guide, Prof Abhishek Sinha who permitted me to explore my area of interest. The freedom he gave allowed me to experiment with my ideas. He was a constant source of support even at dark times when the work was not going as planned. In addition to this, the institute has provided me with the best of opportunities to help me to enhance my knowledge and also to grow personally. Needless to say, my parents have been the ultimate source of security and happiness. Dad's guidance and Mom's love have shaped me to be the person I am today.

ABSTRACT

KEYWORDS: Online Learning; Online Convex Optimisation; Inventory Management; Online Gradient Descent; Regret;

Online learning is a subset of machine learning, where the learner receives data and suffers losses sequentially. Such a structure is seen in many real-life problems such as in finance, capital management etc. Inventory management is an important problem which falls under this structure. Online learning algorithms can thus be applied to the inventory management problem.

One caveat is that the classical online learning framework is memoryless, in that, the current decisions do not affect the future costs. However, the inventory management problem has an element of memory in the form of future storage or delay costs associated with the current decision. Initially, we begin by considering a memoryless version of the inventory management.

There have been some works in the area of online learning with memory which propose algorithms for this setting. The standard inventory problem with memory is thus a candidate for this setting. In this work, we verify the same. The adversary however can be dynamic, hence, we finally consider the notion of dynamic policy regret for problems with memory. We propose an algorithm which is an adaptation of an existing algorithm, establish its regret guarantees, and apply it to the inventory problem.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
OCO	Online Convex Optimisation
OGD	Online Gradient Descent
OMD	Online Mirror Descent
FTRL	Follow the Regularised Leader
OGDM	Online Gradient Descent for Memory OCO
ADER	Adaptive learning for Dynamic Environment
ADERM	Adaptive learning for Dynamic Environment with Memory

NOTATION

$\mathbf{x}_t$	Inventory level at step t
$\mathbf{y}_t$	Purchase decision at step t
$\mathbf{u}_t$	Demand at step t
S_t	Base-stock decision at step t
$\mathcal{Y}$	Decision set
$\mathcal{U}$	Demand set
$\mathcal{S}$	Base-stock decisions set
η	learning rate of OGD algorithm
β	temperature parameter of Hedge and Exponential Weighting algorithm

CHAPTER 1

INTRODUCTION

1.1 Preamble

Online Learning frameworks have been efficiently used to solve a lot of complex problems in real life. The main advantage of using online learning algorithms are due to the strong theoretical guarantees which come embedded with them. There have been several works particularly in recent times to explore and expand the context of these algorithms to serve the expanding complexity of problems which could potentially be solved through the application of techniques from online learning. Although a lot of these complex problems which are solved under the online learning framework could be solved using classical probability theory, for example using the theory of Markov Decision Processes, such classical methods require some probabilistic assumptions on the underlying data-generating process. Under the online learning framework, however, no such assumption are made on the process in which the underlying data-generation occurs. Almost all problems which have been solved using classical theory, are well-suited candidates for the application of online learning algorithms.

One such problem studied quite extensively under the Markov Decision Process setting is the inventory problem. The optimal algorithm, albeit in expectation, has been well known and presented extensively. However, there are quite a few shortcomings of the same. First of all, the only guarantee of the optimality of the algorithm is in expectation and moreover such an algorithm requires the knowledge of the underlying probability distribution. In other words, this means that only if infinitely many instances of the data sequence underlying the problem is generated by a probability distribution, which is assumed to be known to us, the average performance of the algorithm is optimal. However, there is no guarantee how good or bad its performance is for a single sequence of data. Attempting to solve the same problem under the online learning framework and if the corresponding algorithm used has a regret upper bound, there is a strong guarantee what the worst performance of the algorithm for any data sequence could be.

1.2 Online Learning Framework

In many scenarios, data received by learning algorithms is sequential and not always instantaneous whereas decisions must be made instantaneously. Online learning algorithms were developed to account for this fact and to learn from data sequentially. There are several examples such cases of sequential data flow and instantaneous decisions like stock investing decisions and obviously inventory management. In a typical online learning problem, a sequence of data is made available to a learner but the framework is restrictive in the sense that the learner has to make a decision before receiving further data. In making the decision, the learner suffers some loss or obtains some reward chosen by the environment. From this point, the environment which generates both the reward and the loss is referred to as the adversary for the reason that such a nomenclature is followed while defining the regret. The online learning framework is very vast and covers a wide range of problems. However a general problem posed under this framework follows,

for $t = 1, ..., T$ **do**

1: the adversary chooses a context $\mathbf{x}_t \in \mathcal{X}$ and reveals it to the learner; the adversary chooses the true $\mathbf{y}_t \in \mathcal{Y}$;

2: the learner chooses a decision $\hat{\mathbf{y}}_t \in \mathcal{Y}$;

3: the adversary reveals the true $\mathbf{y}_t \in \mathcal{Y}$;

4: the adversary reveals the loss $f_t : \mathcal{Y} \mapsto \mathbb{R}$;

5: the learner suffers a loss of $f_t(\hat{\mathbf{y}}_t)$;

end for

Note that it isn't necessary for the adversary to reveal the context and/or the true label, all that is required is the loss/reward to be received by the learner. In fact a true value need not even exist as is the case with the inventory problem. This means that steps 1 and 3 needn't necessarily occur. Also, we will consider only losses and not rewards as any problem with rewards can be posed as a loss problem by simply negating the reward. Such a framework generally covers most online learning problems. For example, the 'simple' problem of bit prediction can be brought into this framework where there is no context $\mathbf{x}_t$ and the true labels $\mathbf{y}_t$ are bits i.e 0,1 revealed after a prediction . The learner is supposed to predict the next incoming bit $\mathbf{y}_t$ based on the history of received and predicted bits. The adversary generated loss function is simply the 0-1 loss

$f_t(\mathbf{y}_t) = |\hat{\mathbf{y}}_t - \mathbf{y}_t|$. The above framework is only a vanilla backbone of the online learning framework. There can be several modifications to this above discussed framework as is the case with learning in the presence of experts.

1.3 Prediction with Experts and Regret

In many scenarios, the learner will have access to the advice of several experts to make his decision. Formally, the prediction with experts problem is posed as,

decision space $\mathcal{D}$, outcome space $\mathcal{Y}$, set of experts $\mathcal{E}$

for $t = 1, ..., T$ **do**

1: the adversary chooses the next outcome $\mathbf{y}_t \in \mathcal{Y}$

2: the expert advice $\left\{ f_{E,t} \in \mathcal{D} : E \in \mathcal{E} \right\}$ is chosen and revealed to the forecaster

3: the learner chooses the prediction $\hat{\mathbf{y}}_t \in \mathcal{D}$;

4: the adversary reveals the true $\mathbf{y}_t \in \mathcal{Y}$;

5: the adversary reveals the loss $f_t : \mathcal{Y} \mapsto \mathbb{R}$;

6: the learner suffers a loss of $f_t(\hat{\mathbf{y}}_t)$; and each expert E incurs a loss of $f_t(f_{E,t})$;

end for

Note that as before, a true outcome need not exist and it is only the loss which is required by the learner. In such a setting comparing the performance of the learner with that of the performance of the expert which incurs the least loss cumulatively in the time horizon. Mathematically, we can define a quantity which does the above comparison as,

$$\begin{aligned} R_T &= \max_{E \in \mathcal{E}} \sum_{t=1}^{T} \left(f_t(\hat{\mathbf{y}}_\mathbf{t}) - f_t(f_{E,t}) \right) \\ &= \sum_{t=1}^{T} f_t(\hat{\mathbf{y}}_\mathbf{t}) - \min_{E \in \mathcal{E}} \sum_{t=1}^{T} f_t(f_{E,t}) \\ &= \hat{L}_T - \min_{E \in \mathcal{E}} L_{E,T} \end{aligned}$$

where $\hat{L}_T = \sum_{t=1}^{T} f_t(\hat{\mathbf{y}}_\mathbf{t})$ and $L_{E,T} = \sum_{t=1}^{T} f_t(f_{E,t})$. The quantity which we have defined is the static **regret** for the prediction with experts problem. Most literature in online learning focuses on regrets, although the regrets defined could be different as per

the goals of the works. However even though different regrets are defined, the main goal of regret analysis is to compare the performance of an algorithm or learner with that of a comparator.

1.4 Online Convex Optimisation

Online convex optimisation (OCO) problems involve a slightly more restrictive study involving convexity constraints on the sets of decisions and the loss functions. It is a slightly more generic setting than prediction with experts. A learner under the online convex optimisation setting makes decisions iteratively without knowing the outcome which a loss function chosen by an adversary. The loss function is revealed to the learner once he makes the decision. Formally, the online convex optimisation is a process described by the following steps,

for $t = 1, ..., T$ **do**

1: the learner chooses $\mathbf{x}_t \in \mathcal{K}$

2: the adversary chooses the loss $f_t : \mathcal{K} \mapsto \mathbb{R}$;

3: the learner suffers a loss of $f_t(\mathbf{x}_t)$;

end for

Additionally there are certain constraints on the decision set and the loss function. The decision set $\mathcal{K} \subseteq \mathbb{R}^n$ is constrained to be a bounded convex set under the OCO setting. The loss functions $f_t : \mathcal{K} \mapsto \mathbb{R}$ too are assumed to be bounded and convex. The goal of the algorithms is to sequentially minimise the loss suffered. However, how good or bad an algorithm performs should be measured in an absolute sense because the loss function can keep varying with iteration. This leads us to considering a measure which compares the performance of a learner/algorithm to a comparator. As in the previous case of prediction with expert advice, we define the regret. The static regret for the OCO problem is defined as,

$$R_T = \sum_{t=1}^{T} f_t(\mathbf{x_t}) - \min_{\mathbf{x} \in \mathcal{K}} \sum_{t=1}^{T} f_t(\mathbf{x})$$

The comparator in this case is static in the sense that it doesn't vary with iteration and

thus the regret defined above is termed as static regret.

1.5 Inventory Management Process

So far, the frameworks of online convex optimisation and prediction with experts have been described. We can now outline the inventory management process. The inventory process is a standard one in which, iteratively, goods of a single type are purchased by an inventory manager/seller to replenish the inventory and sold to consumers/buyers. Additional goods are stored in the inventory for demands in the future. Of course the purchasing and storage incur costs to the inventory manager. There are two ways to view the profits associated with goods sold to buyers/customers. The first trivial way is the total revenue earned by selling the goods and the profits thus earned. Another non-trivial way is to consider the loss of sales, i.e. revenue lost because of inability to sell the demanded amount of goods to customers due to insufficient levels of inventory, to incur a delay cost. While the first model is a profit based model, the second model is a delay cost model. We consider both these models for further basic analysis and then choose one for application of online learning algorithms and obtain theoretical guarantees on the regret.

1.5.1 Profit based model

We mathematically describe the inventory process in which the reward is based on the profit model. Formally, the process can be described by the following series of steps,

initial inventory level $\mathbf{x}_1 = 0$

for $t = 1, ..., T$ **do**

1: the learner chooses the amount of goods to be purchased $\mathbf{y}_t \in \mathcal{Y}$ to replenish the inventory $\mathbf{x}_t$

2: the intermediate inventory level $\mathbf{x'}_t = \mathbf{x}_t + \mathbf{y}_t$

3: the adversary chooses the demand for the goods $\mathbf{u}_t$;

4: the amount of goods sold is $\min(\mathbf{x'}_t, \mathbf{u}_t)$

5: the carried over inventory to the next step is $\mathbf{x}_{t+1} = \max(0, \mathbf{x'}_t - \mathbf{u}_t)$

6: the learner suffers a cost of $l_t(\mathbf{y}_t) = c\mathbf{y}_t + h\mathbf{x}_{t+1}$;

7: the learner earns a reward of $r_t(\mathbf{y}_t) = p \min(\mathbf{x'}_t, \mathbf{u}_t)$

8: the total loss suffered by the learner is $f_t(\mathbf{y}_t) = l_t(\mathbf{y}_t) - r_t(\mathbf{y}_t)$

end for

The intermediate level of inventory $\mathbf{x'}_t$ represents the amount of goods in the inventory at step t after replenishing it but before goods being sold to customers. The amount of goods sold is trivially the minimum of the amount of goods in inventory and that demanded which is embedded in step 4 above. Obviously, the amount of goods remaining in inventory which is carried over to the next step $t+1$ is the amount of intermediate inventory minus the amount of amount of goods sold which is represented in step 5. The carrying over of inventory incurs a holding cost h per unit and the purchasing of goods incurs a cost of c per unit both of which are embedded in the cost l_t while the revenue earned is p per unit sold which is the sole component of r_t. The total loss incurred is the total cost minus total revenue. For practical purposes we assume $p > c$. Additionally, the inventory is always non-negative in this model which is of course what we would expect.

1.5.2 Delay Cost based model

The delay cost model considers the lost sales rather than the revenue. However, the inventory evolution differs from the previous case and in fact is a linear process. In fact the ease of representing this process is one of the reasons why we chose this model for our survey of online learning algorithms. Formally, the delay cost inventory model can be described by the following steps,

initial inventory level $\mathbf{x}_1 = 0$

for $t = 1, ..., T$ **do**

1: the learner chooses the amount of goods to be purchased $\mathbf{y}_t \in \mathcal{Y}$ to replenish the inventory $\mathbf{x}_t$

2: the adversary chooses the demand for the goods $\mathbf{u}_t$;

3: the carried over inventory to the next step is $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t$

4: the learner suffers a cost of $f_t(\mathbf{y}_t) = c\mathbf{y}_t + \max(h\mathbf{x}_{t+1}, -d\mathbf{x}_{t+1})$;

end for

The first point to note here is that the loss function simply includes only a cost and

no revenue which was of course the goal of this model i.e, to eliminate the use of revenue in the loss function. Additionally, the inventory can be negative where a negative inventory indicates that the inventory levels were not sufficient to satisfy incoming demand. Essentially, for a delay of a unit item for every step incurs a cost of d. As in the previous model, the purchase of goods to replenish inventory incurs a cost of c per unit item and a positive inventory carried over to the next step incurs a holding cost of h per step per unit item. Thus, if the inventory level is positive, we incur holding cost and if the inventory level is negative, we incur delay cost both these costs are embedded in the expression, $\max(h\mathbf{x}_t, -d\mathbf{x}_t)$. This model is the main model which we shall consider for future survey on online algorithms and their theoretical regret guarantees, although we analyse the optimal stationary policy in hindsight for the profit based model.

Although both models of the inventory management problem seem to fall under the online convex optimisation and prediction with experts framework, there are minor modifications required so that they are wholly compatible with the model. To be more specific, the upper bound on the loss functions, without any modifications to the problem setup, is not necessarily independent of the time horizon which we shall cover in more detail while analysing the online algorithms. These changes are either related to the way decisions are made by the algorithm or the way in which previous purchases and demands are included in the current inventory level.

Another way to view the inventory management problem is through the use of states. States of system wholly describe the current properties of a system which would not change unless external factors act on it. In this case, the state of the inventory level is simply the inventory level which wouldn't change unless purchases are made or goods are sold. The state of the inventory system basically captures the previous amounts of purchases made and the amounts of goods sold. Thus, the inventory problem is not memoryless. Whereas memoryless problems are ideal candidates for OCO algorithms, we consider the state variable as an input variable to the loss function chosen by the adversary. We also consider modifications of the vanilla OCO algorithms to deal with the case of memory.

CHAPTER 2

RELATED WORK AND LITERATURE SURVEY

2.1 Caching

A concept with is different from inventory management albeit related to it is caching. In fact an inventory is very similar to a cache in the sense that both represent the levels of items available to be sold to customers, goods in the case of inventory and files in the case of cache. The difference lies in the fact that while goods sold from an inventory need to be replenished, files stored in a cache need no replenishing even if they are hit by user requests. It is therefore worth exploring works on caching to gain an understanding on how online learning algorithms might be great tools for the inventory management problem. Bhattacharjee *et al.* (2020) establish the fundamental limits of the caching problem under the online learning setting. In fact they devise an Follow the regularised leader (FTRL) like algorithm called follow the perturbed leader (FTPL) to achieve a sub-linear upper regret bound under the single cache setting. The bound they achieve is $\mathcal{O}(\sqrt{T})$. Additionally, hey devise aregret lower bound on the caching problem agnostic of the algorithm. This lower bound is very interesting because it is also $\mathcal{O}(\sqrt{T})$ and essentially, the upper bound on the regret of FTPL differs only by a constant factor thus proving its asymptotic optimality. They consider a reward based learning setup although it can be converted to a loss based setup by simply negating the reward. In addition to this, they consider a network of caches and devise a modification of the FTPL algorithm which achieves a similar $\mathcal{O}(\sqrt{T})$ regret upper bound. They also describe an online gradient ascent (OGA) algorithm, which is essentially the reward setting equivalent of the online gradient descent (OGD) algorithm which also achieves a similar regret upper bound. By establishing, a fundamental regret lower bound of $\mathcal{O}(T)$ for the network caching problem, they establish the asymptotic optimality of the FTPL and OGA algorithm. In fact, under the future works section, they mention the problem of inventory management as a variant of the caching problem and that was a great basis for our current work.

2.2 Inventory Management Markov Decision Process

A classical approach to the inventory management problem is by formulating it as a Markov Decision Process (MDP). Bertsekas (2000) formulates the inventory management problem as an MDP and proceeds to devise an optimal algorithm based on Dynamic Programming. The delay-cost model of inventory process is presented in this book. Dynamic programming is a standard approach to solve MDPs optimally. The usage of an MDP formulation and the look-ahead approach in dynamic programming to derive an optimal algorithm requires knowledge of all the underlying probability distributions. Moreover, the optimality referred to , in this case, is the optimality of the total loss or reward, in expectation with respect to the transition probabilities of the environment. It essentially means that on several repetitions of the process, the average total rewards/losses obtained are optimal. There are no derived theoretical guarantees or bounds on the worst-case performance of the algorithm. However, the dynamic programming algorithm devised has certain interesting properties. In fact, we incorporate some characteristics of this algorithm in our initial online learning algorithm.

The interesting aspect of this algorithm is that even through it controls the amount of goods purchased, it does so in an indirect way by monitoring the inventory levels. Specifically, the inventory is reset every time it falls beyond a particular value called the base-stock. Drawing inspiration from this algorithm, we formulate and analyse online algorithms which attempt to control the purchases to inventory via the base-stock. Additionally, the structure of the problem under the base-stock assumption makes is possible to prove bounds on the loss function which is an essential requirement for the underlying regret guarantees of some algorithms based on experts. Additionally, the work considers the MDP under both the finite-horizon and the discounted infinite-horizon setting. Although structurally different, it is indeed remarkable that under both these settings the optimal policy in expectation is a similar base-stock policy.

The delay-cost based inventory process, in addition to that discussed before, in the MDP formulation involves the specification of the probability distribution of the incoming demand $\mathbf{u}_t$. Precisely, the inventory level, $\mathbf{x}_t$, the purchase amount $\mathbf{y}_t$, and the incoming demand, $\mathbf{u}_t$ at step t evolves as,

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t$$

Thus, the conditional probability of the next state $\mathbb{P}(\mathbf{x}_{t+1}|\mathbf{y}_t, \mathbf{x}_t)$ which is the state transition probability depends only on the probability distribution of $\mathbf{u}_t$. The per-step total cost function is $f_t(\mathbf{y}) = c\mathbf{y}_t + \max(h\mathbf{x}_{t+1}, -d\mathbf{x}_{t+1})$ which is exactly the same which we consider under the delay-cost inventory process with the parameters c, h, d representing the purchase, holding and delay costs per unit item respectively.

2.3 Works in Online Convex Optimisation

Research in online learning is an active area with several works dealing with increasingly complex scenarios. Algorithms like Hedge (exponentially weighted) in the case of prediction with experts and online gradient descent in the case of online convex optimisation have well-established and well-known regret upper bounds. However, these bounds are conditional on the loss functions and/or the decision set satisfying certain assumptions. Naturally, as the complexity of the framework increases, the assumptions become stronger in nature. Although, many of these assumptions are either easily satisfied or could be modelled into the problem on which these algorithms are applied. Shalev-Shwartz (2012) and Cesa-Bianchi and Lugosi (2006) establish the theoretical regret guarantees of several of the algorithms which are considered basic. These algorithms like Online Mirror Descent, Online Gradient Descent, Hedge, and Follow the Perturbed Leader (FTRL) are considered basic in the sense that several works which consider complex frameworks have proven that adaptations of these algorithms have strong regret upper bound guarantees. It is indeed a requirement that regret upper bounds are sublinear in nature. We consider the two basic algorithms OGD and Hedge in our initial analysis under different settings.

The inventory management problem is slightly unique in the sense that there is an involvement of memory. Initially, we consider the memory involved as an input of the adversary, thus isolating the a current step of the problem from the previous steps, making the problem memoryless. However, such an assumption has several drawbacks which we shall consider. Additionally, it is worth noting that in these cases, the regret is with respect to the instantaneous decisions where the cummulative effect of decisions taken together is not considered. Thus, there is a need to consider the memory involved too. Essentially, this memory is represented by the inventory state and can be

decomposed as a result of the previous decisions made. This memory reflects itself in the loss function. We consider, a bounded memory case, where only the most recent m decisions are a part of the memory of the inventory process. Merhav *et al.* (2002) considers the experts with memory framework and presents an algorithm with $\mathcal{O}(T^{2/3})$ regret upper bound. György and Neu (2011) establishes an algorithm which satisfies a regret upper bound of $\mathcal{O}(\sqrt{T})$ in the same experts with memory framework as Merhav *et al.* (2002). Anava *et al.* (2015) presents a general upper bound on the policy regret of loss functions with memory in the more general framework of OCO with memory under certain assumptions, using which algorithms with a regret upper bound of $\mathcal{O}(\sqrt{T})$ can be derived. This general upper bound can be decomposed into components and we design an algorithm that achieves the a specific regret upper bound which is sub-linear in time. In fact, the order of this upper bound is as good as the order of the upper bound obtained by the OGD algorithm.

Most of the works which we had described so far consider the notion of the static regret, where the comparator is stationary. However, when the environment is changing a stronger comparator which is non-stationary becomes necessary. Such a comparator is considered in the concept of dynamic regret. Certain works consider the dynamic comparator with some constraints on the degree of change in the comparator. In fact, the classical Fixed-share algorithm of Bousquet and Warmuth (2003) and it's adaptation in Rooij and Erven (2009), and an adaptation of the weighted-majority algorithm presented in Geulen *et al.* (2010) consider a switching comparator in the experts setting where the comparator can change by atmost $\mathcal{O}(\sqrt{T})$ for establishing their corresponding regret bounds. There have also been several works in the case of dynamic regret for the memoryless OCO setting such as Hall and Willett (2013), Jadbabaie *et al.* (2015), Mokhtari *et al.* (2016), Yang *et al.* (2016), Zhang *et al.* (2017), and Shi *et al.* (2020). Many of these works such as present algorithms which have strong sub-linear regret upper of the order $\mathcal{O}(\sqrt{T})$ for convex and $\mathcal{O}(\log T)$ for strongly convex loss functions. Some of these works retain some restrictions on the dynamic comparator. Zinkevich (2003) presents a very general notion of dynamic regret for the memoryless OCO setting where the comparator sequence can be any sequence within the decision set. Zhang *et al.* (2018) and Zhao *et al.* (2020) present algorithms which satisfy $\mathcal{O}(\sqrt{T})$ dynamic regret bounds for the framework established in Zinkevich (2003). These algorithms are adapted to the OCO with memory setting in this work.

CHAPTER 3

MEMORYLESS OCO AND INVENTORY MANAGEMENT

3.1 Analysis of the Inventory Process

Let $\{\mathbf{u}_t\}_{t=1}^T$ be the demand sequence for a single good. The demand is chosen by the adversary. Let $\{\mathbf{x}_t\}_{t=1}^T$ denote the inventory level at every step. The learner makes decisions $\{\mathbf{y}_t\}_{t=1}^T$ which represent the amount of goods procured to replenish the inventory. A practical limit on the purchases would pertain to the capacity of transporting goods. Let the transporting capacity be C. Thus, the decision set $\mathcal{Y} \subseteq \mathbb{R}$ is $[0, C]$.

The inventory evolution process as described before is,

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t \tag{3.1}$$

$$\mathbf{x}_1 = 0$$

The loss function in this case would be the cost of purchasing goods, cost of storing goods and the cost associated with delaying the demand. Also, we assume that the new stock is added to inventory before sales everyday. Assuming that unit cost of the good is c,its holding cost is h per unit and its delay cost is d per unit, the one-step reward can be expressed as,

$$f_t(\mathbf{y}_t) = c\mathbf{y}_t + \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t)) \tag{3.2}$$

The total loss thus is the summation of all the one-step losses accumulated within a horizon. In this case it is,

$$\begin{aligned} F_T(\{\mathbf{y}_t\}_{t=1}^T) &= \sum_{t=1}^T f_t(\mathbf{y}_t) \\ &= c\sum_{t=1}^T \mathbf{y}_t + \sum_{t=1}^T \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t)) \end{aligned} \tag{3.3}$$

The policy generated by the learner/algorithm $\mathcal{A}$ in this case is $\pi = \{\mathbf{y}_t\}_{t=1}^{T}$. We always define the regret for a particular learner's decisions.

3.1.1 Regret and Memoryless Nature

Consistent with the definition of static regret for the OCO formulation we define the regret of an algorithm $\mathcal{A}$ with decisions $\{\mathbf{y}_t\}_{t=1}^{T}$ as,

$$R_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(\mathbf{y}_t) - \min_{\mathbf{y}\in\mathcal{Y}} \sum_{t=1}^{T} f_t(\mathbf{y})$$

For ease of notation, we simply refer to the above term by R_T since is embedded that a regret is defined for an algorithm. We consider the memoryless setting in the sense that $\mathbf{x}_t$ is generated by the adversary. What this means is that the sequence of $\mathbf{x}_t$ is assumed to be an inherent part of the loss function. So for the purpose of regret, the same sequence $\mathbf{x}_t$ is generated for both the learner and the comparator. However, there is no restriction on the adversary with respect to how it generates this sequence. In fact, the adversary needn't disclose the way $\mathbf{x}_t$ is generated. So as far as the algorithm is concerned irrespective of the way $\mathbf{x}_t$ is generated, the sequence is independent of its decisions. This means that recursion 3.1 is not unrolled in the loss represented in equation 3.2 and $\mathbf{x}_t$ is assumed to be a wholly generated by the adversary. The sequence could, however, be generated as $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t$ but this is known only to the adversary and the adversary simply reveals $\mathbf{x}_t$ while abstracting how it is generated. Although this means that there is an abstraction of the inventory evolution process (3.1), it is a good starting point for our analysis. The decision thus taken are in a sense instantaneous, i.e. decisions made given an inventory level irrespective of the influence of previous actions. When we consider the case of OCO with memory we will deal with a more original version of the problem where the inventory evolution process is known to the learner.

This can be described as,

for $t = 1, ..., T$ **do**

1: the learner chooses the amount of goods to be purchased $\mathbf{y}_t \in \mathcal{Y}$ to replenish the inventory;

2: the adversary chooses the demand for the goods $\mathbf{u}_t$ and the inventory level $\mathbf{x}_t$;

3: the learner suffers a cost of $f_t(\mathbf{y}_t) = c\mathbf{y}_t + \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t))$;

end for

3.1.2 Analysis of the step-wise loss function

The step-wise loss function is a piecewise linear function. In fact, it can be written as,

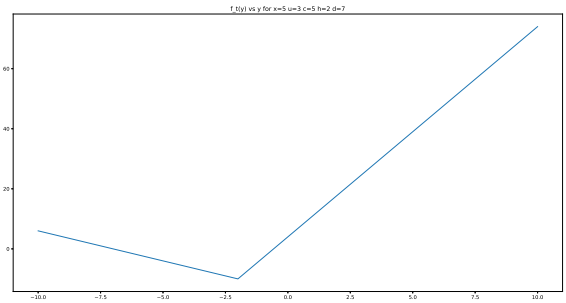
$$\begin{aligned} f_t(\mathbf{y}_t) &= c\mathbf{y}_t + \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t)) \\ &= \max(c\mathbf{y}_t + h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), c\mathbf{y}_t - d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t)) \\ &= \max((c+h)\mathbf{y}_t + h(\mathbf{x}_t - \mathbf{u}_t), (c-d)\mathbf{y}_t - d(\mathbf{x}_t - \mathbf{u}_t)) \end{aligned}$$

Note that the functions over which $\max$ is taken are linear. An important assumption we make is that the delay-cost is greater than the purchase cost per unit item, i.e $d > c$. Since, maximum taken over a finite number of linear functions results in a convex function (Boyd and Vandenberghe (2004)), clearly the loss function per step is a convex function over $\mathbf{y}_t$. Additionally, due to our assumption that $c - d < 0$, one of these linear functions has a non-positive slope while the other has a non-negative slope. In fact, the point of intersection of these linear functions is $\mathbf{y}_t = \mathbf{u}_t - \mathbf{x}_t$ and this is where the change of slope occurs. Also, the slopes ($c + h$ and $c - d$ are the same for any step t and the only change in the functions over different steps is the point where there is a change of slope. In figure 3.1, we consider the different cases of the loss function under different settings of the parameters. Note the negative slope for $\mathbf{y} < \mathbf{u} - \mathbf{x}$ if $d > c$ which is the case of our interest.

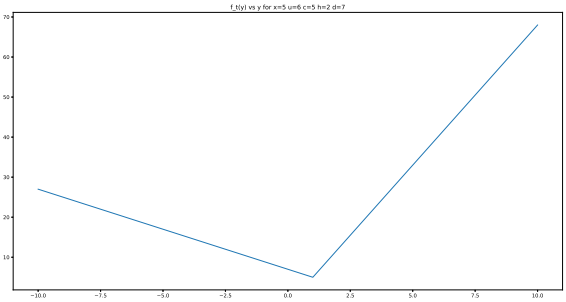
3.2 Optimality of Base-Stock Policy under MDP Setting

Before investigating the performance of online algorithms on the inventory management problem, we analyse the optimal policy of the same under a finite-horizon Markov Decision Process setting. Bertsekas (2000) establish the optimality of a base-stock policy under such settings based on the dynamic programming algorithm.

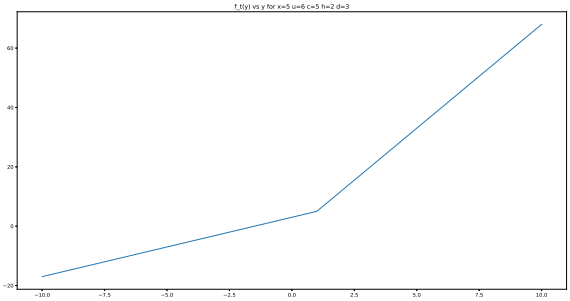
We will slightly abuse the notation used in **?**in order to match with that used previously. However, the underlying structure of the problem remains the same. The



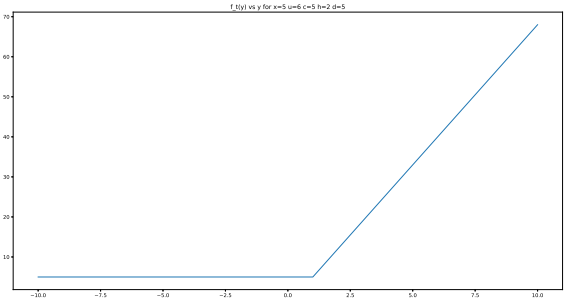
(a) $d > c$ and $x - u > 0$



(b) $d > c$ and $x - u < 0$



(c) $d < c$



(d) $d = c$

Figure 3.1: Different cases of the single step memoryless loss function under different settings of the underlying parameters

inventory evolves as,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{y}_k - \mathbf{u}_k$$

with $\mathbf{x}_k, \mathbf{y}_k$ and $\mathbf{u}_k$ having the same meaning as before. The decision to be made is the purchase amount $\mathbf{y}_k$ at each step. The stochasticity lies in the environment which selects the demand $\mathbf{u}_k$. However, the probability distribution of these demands are known. The loss function is represented with a slight change of notation as,

$$f_k(\mathbf{y}_k) = c\mathbf{y_k} + r(\mathbf{x}_k + \mathbf{y}_k - \mathbf{u}_k)$$

where,$r((x) = \max(h\mathbf{x}, -d\mathbf{x})$ which represents the holding or the delay cost. The total cost is thus,

$$\sum_{k=1}^{T} f_k(\mathbf{y}_k) = \sum_{k=1}^{T} (c\mathbf{y_k} + r(\mathbf{x}_k + \mathbf{y}_k - \mathbf{u}_k))$$

The goal is the minimisation of the total expected cost,

$$\mathbb{E}\big(\sum_{k=1}^{T} f_k(\mathbf{y}_k)\big) = \mathbb{E}\big(\sum_{k=1}^{T} (c\mathbf{y_k} + r(\mathbf{x}_k + \mathbf{y}_k - \mathbf{u}_k))\big)$$

An optimal policy which minimises the total expected cost can be derived from the application of the dynamic programming algorithm. Application of the dynamic programming results in the following definition of the cost-to-go,

$$J_{T+1}(\mathbf{x}_{T+1}) = 0$$
$$J_k(\mathbf{x}_k) = \min_{\mathbf{y}_k \geq 0} \left[c\mathbf{y}_k + H(\mathbf{x}_k + \mathbf{y}_k) + \mathbb{E}\big\{ J_{k+1}(\mathbf{x}_k + \mathbf{y}_k - \mathbf{u}_k \big\} \right]$$

where, $H(\mathbf{x}) = \mathbb{E}\big[r(\mathbf{x} - \mathbf{u}_k)\big]$ and the expectations are taken over the probability distribution of $\mathbf{u}_k$. NOte that the minimisers in the DP equation yield the optimal policy. The function $r(\mathbf{x} - \mathbf{u}_k)$ is convex in $\mathbf{x}$. In function H, the expectation is taken over the probability distribution of $\mathbf{u}_k$ and since operation preserves convexity, the function H is convex as well. Using $\mathbf{z}_k = \mathbf{x}_k + \mathbf{y}_k$, the DP equation is re-written as,

$$J_k(\mathbf{x}_k) = \min_{\mathbf{z}_k \geq \mathbf{x}_k} G_k(\mathbf{z}_k) - c\mathbf{x}_k \tag{3.4}$$

where, $G_k(\mathbf{z}) = c\mathbf{z} + H(\mathbf{z}) + \mathbb{E}[J_{k+1}(\mathbf{z} - \mathbf{u}_k)]$, is a convex function. The proof that G_k is convex is slightly involved and hence we do not present the same here. However,

using the property of convexity, an interesting optimal policy is derived. Defining,

$$S_k = \arg\min_{\mathbf{z}\in\mathbb{R}} G_k(\mathbf{z})$$

it is noted that if $\mathbf{x}_k < S_k$, in view of the constraint $\mathbf{z}_k \geq \mathbf{x}_k$ in the minimisation in Eq. 3.4, the minimisation occurs at $\mathbf{z_k} = S_k$. However, if $\mathbf{x}_k \geq S_k$, the minimisation in Eq. 3.4, the minimiser is $\mathbf{z}_k = \mathbf{x}_k$ due to the convexity of G_k. Since, $\mathbf{z}_k = \mathbf{x}_k + \mathbf{y}_k$, the above minimisers are written in terms of $\mathbf{y}_k$ as, $\mathbf{y}_k = S_k - \mathbf{x}_k$ when $\mathbf{x}_k < S_k$ and $\mathbf{y}_k = 0$ when $\mathbf{x}_k \geq S_k$. Thus the optimal policy $\mu_k^*$is expressed as,

$$\mu_k^*(\mathbf{x_k}) = \begin{cases} S_k - \mathbf{x_k} & \mathbf{x_k} < S_k \\ 0 & \mathbf{x}_k \geq S_k \end{cases} \tag{3.5}$$

The existence of S_k is guaranteed by the convexity of G_k and the property

$$\lim_{|y|\to\infty} G_k(y) = \infty$$

which is trivial to prove. Such a policy where a purchase is made when the inventory level is below a particular threshold is known as the base-stock policy. The purchases are made such that the inventory level reaches the threshold. In this case the threshold at step k is S_k and this threshold is called the base-stock. Even though this policy was derived under an MDP setting where the goal was to minimise the total expected cost, we explore this interesting policy in combination with online learning algorithms where there are guarantees with respect to their worst-case performance. As we have described earlier, the goals of online learning algorithms need not be to minimise the total expected costs. In fact, the probability distribution of the demand amounts are unknown to the learner in an online learning setup which was required by the dynamic programming algorithm to derive the base-stock policy.

3.3 Base-Stock Policy under the experts setting

3.3.1 Setup

It would be an interesting consideration of the base-stock policy under the prediction with experts setting. Essentially, here the different experts represent different levels of the base-stock. An algorithm under this setting has to make a decision based on the received expert advice. A learner in the experts setting makes its base-stock decisions as,

decision space $\mathcal{S}$, set of experts $\mathcal{E}$, inventory space $\mathcal{X}$, demand space $\mathcal{U}$

for $t = 1, ..., T$ **do**

1: the adversary chooses the inventory level $\mathbf{x}_t \in \mathcal{X}$

2: the expert advice of the base-stock $\left\{S_{E,t} \in \mathcal{S} : E \in \mathcal{E}\right\}$ is chosen and revealed to the learner

3: the learner chooses the base-stock level $S_t \in \mathcal{S}$;

4: the adversary chooses the demand $\mathbf{u}_t \in \mathcal{U}$ and reveals the loss

$$l_t(S) = c\max(S - \mathbf{x}_t, 0) + \max\left(h(\max(S, \mathbf{x}_t) - \mathbf{u}_t), -d(\max(S, \mathbf{x}_t) - \mathbf{u}_t)\right) \quad (3.6)$$

5: the learner suffers a loss of $l_t(S_t)$; and each expert E incurs a loss of $l_t(S_{E,t})$;

end for

Note that although the loss represents the same cost associated with purchasing, storing and delaying goods, the loss is now with respect to the base-stock level and not the purchase amount. The purchase made to replenish inventory under a base-stock S and inventory level $\mathbf{x}_t$ is $\max(S - \mathbf{x}_t, 0)$. It is trivial to note that the inventory level after the purchase and demand $\mathbf{u}_t$ has been met under this base-stock is $\max(S, \mathbf{x}_t) - \mathbf{u}_t$. These have been embedded in the loss function l_t. The static regret defined in the $\mathcal{A}$. This is defined as,

$$R_T(\mathcal{A}) = \sum_{t=1}^{T} l_t(S_t) - \min_{E \in \mathcal{E}} \sum_{t=1}^{T} l_t(S_{E,t}) \quad (3.7)$$

where, S_t is the decision made by $\mathcal{A}$ at step t. We would require that the regret grow

slower than T, the time horizon, such that,

$$\lim_{T\to\infty} \frac{R_T(\mathcal{A})}{T} = 0$$

Such guarantees are obtained when the regret is upper bounded by a sub-linear function of T. We now consider a well-known algorithm under the experts setting, called the **Hedge** or mutliplicative weights algorithm in our base-stock inventory framework. The Hedge algorithm, as we shall present, has a strong sub-linear regret guarantee under certain conditions on the loss function.

3.3.2 Hedge Algorithm

Freund and Schapire (1997) present an algorithm called **Hedge** for the experts setting which generalises the idea of the classical weighted majority predictor. The Hedge algorithm can be derived from the Online Mirror Descent algorithm with the entropy regulariser as shown in Shalev-Shwartz (2012). The algorithm is extremely popular for the experts setting that several works have studied it's application for different problems. The Hedge algorithm has a hyper-parameter β and proceeds as follows,

Algorithm 1: Hedge Algorithm

Input: set β, decision set $\mathcal{Y}$;

Initialise: $w_{1,E} = 1 \quad \forall E \in \mathcal{E}$;

for $t = 1, 2, 3, 4...T$ **do**

each expert $E \in \mathcal{E}$ advises $\mathbf{y}_{t,E} \in \mathcal{Y}$;

set $p_{t,E} = \dfrac{w_{t,E}}{\sum_{i\in\mathcal{E}} w_{t,i}}$ for each expert $E \in \mathcal{E}$;

set $\mathbf{p}_t = [p_{t,E} \quad \forall E \in \mathcal{E}]$;

sample expert $K_t \sim$ Categorical$(\mathbf{p}_t)$ and choose action $\mathbf{y}_{t,K_t}$;

adversary chooses loss l_t;

each expert $E \in \mathcal{E}$ suffers loss $l_t(\mathbf{y}_{t,E})$ and algorithm suffers loss $l_t(\mathbf{y}_{t,K_t})$;

set $w_{t+1,E} = w_{t,E} \exp\left(-\beta l_t(\mathbf{y}_{t,E})\right)$ for each expert $E \in \mathcal{E}$;

end

$w_{t,E}$ is the weight for each expert E. The update to the weights at each step are multiplicative in nature. Since there is stochasticity due to the sampling of the expert by the algorithm thus, although the regret defined in Eq. 3.7 still holds, there is an

expectation involved in the first term, i.e.

$$R_T(\mathcal{A}) = \sum_{t=1}^{T} \mathbb{E}_{K\sim\mathbf{p}_t}\Big[l_t(\mathbf{y}_{t,K})\Big] - \min_{E\in\mathcal{E}} \sum_{t=1}^{T} l_t(\mathbf{y}_{t,E})$$

$\mathbf{p}_t$ is the same probability vector at step t as defined in the algorithm.

Theorem 1 *Let* $\mathbf{l_t}$ *denote the loss vector* $\mathbf{l_t} = [(l_t(\mathbf{y}_{t,E}))] \quad \forall E \in \mathcal{E}$, $\mathbf{l_t^2}$ *denote the pointwise square losses vector i.e.* $\mathbf{l_t^2} = [(l_t(\mathbf{y}_{t,E}))^2] \quad \forall E \in \mathcal{E}$, *and* $\beta > 0$, *and assume all losses* l_t *to be non-negative. The Hedge algorithm satisfies for every expert* E^*,

$$\sum_{t=1}^{T} \mathbf{p}_t.\mathbf{l_t} - \sum_{t=1}^{T} l_t(\mathbf{y}_{t,E^*}) \leq \beta \sum_{t=1}^{T} \mathbf{p}_t.\mathbf{l_t^2} + \frac{\log(|\mathcal{E}|)}{\beta}$$

Corollary 1.1 *If* $l_t^2(\mathbf{y}) \leq G \quad \forall t = 1, 2, ...T, \quad \forall y \in \mathcal{Y}$, *then Hedge algorithm with hyper-parameter* β, *satisfies for every expert* E^*,

$$\sum_{t=1}^{T} \mathbf{p}_t.\mathbf{l_t} - \sum_{t=1}^{T} l_t(\mathbf{y}_{t,E^*}) \leq \beta GT + \frac{\log(|\mathcal{E}|)}{\beta}$$

This corollary is a direct application of the bound in Theorem 1 and using the fact that $\sum_{E\in\mathcal{E}} p_{t,E} = 1$, since $\mathbf{p}_t$ is a probability vector.

Corollary 1.2 *If* $l_t^2(\mathbf{y}) \leq G \quad \forall t = 1, 2, ...T, \quad \forall y \in \mathcal{Y}$, *then Hedge algorithm with hyper-parameter* $\beta = \sqrt{\frac{\log(|\mathcal{E}|)}{GT}}$, *satisfies the regret bound,*

$$R_T \leq 2\sqrt{G \log(|\mathcal{E}|)T}$$

This is a simple extension of corollary 1.1 by the fact that E^* in corollary 1 can be $\arg\min_{E\in\mathcal{E}} \sum_{t=1}^{T} l_t(\mathbf{y}_{t,E})$. Additionally, $\mathbb{E}_{K\sim\mathbf{p}_t}\big[l_t(\mathbf{y}_{t,K})\big] = \sum_{t=1}^{T} \mathbf{p}_t.\mathbf{l_t}$. Finally, using the above value of β results in the regret bound. The obtained regret bound is $\mathcal{O}(\sqrt{T})$ and therefore sub-linear in time.

3.3.3 Base-Stock with Hedge

With the regret guarantees and the conditions for achieving them established previously, we now attempt to apply the Hedge algorithm to the inventory problem with base-stock experts as presented in section 3.3.1. We, however, need to establish the conditions on the loss functions, and define the decision set S, the inventory set $\mathcal{X}$, and the demand set $\mathcal{U}$. We assume that the demand $\mathbf{u}_t$ is upper bounded by M, and obviously non-negative. Thus, the demand set, $\mathcal{U}$, is $[0, M]$. With this assumption, we can restrict the decision base-stock set $\mathcal{S}$ to $[0, M]$ as well, without loss of optimality as we shall show.

Let us consider 2 base-stocks $S = M$ and $S' > M$. and the corresponding loss functions as defined in Eq. 3.6, $l_t(S)$ and $l_t(S')$ are,

$$\begin{aligned} l_t(S') &= c\max(S' - \mathbf{x}_t, 0) + \max\left(h(\max(S', \mathbf{x}_t) - \mathbf{u}_t), -d(\max(S', \mathbf{x}_t) - \mathbf{u}_t)\right) \\ &= (c+h)\max(S', \mathbf{x}_t) - c\mathbf{x}_t - h\mathbf{u}_t \quad \because S' > M, \mathbf{u}_t \leq M \end{aligned}$$

The second statement follows from the property of the $\max$ function which implies that the term $\max(S', \mathbf{x}_t) - \mathbf{u}_t > 0$, and thus, holding cost is incurred rather than the delay cost. Similarly,

$$\begin{aligned} l_t(S) &= c\max(S - \mathbf{x}_t, 0) + \max\left(h(\max(S, \mathbf{x}_t) - \mathbf{u}_t), -d(\max(S, \mathbf{x}_t) - \mathbf{u}_t)\right) \\ &= (c+h)\max(S, \mathbf{x}_t) - c\mathbf{x}_t - h\mathbf{u}_t \quad \because S = M, \mathbf{u}_t \leq M \end{aligned}$$

Now we consider,

$$l_t(S') - l_t(S) = (c+h)(\max(S', \mathbf{x}_t) - \max(S, \mathbf{x}_t))$$

Now, since $S' > S$, $\max(S', \mathbf{x}_t) > \max(S, \mathbf{x}_t)$

$$\implies l_t(S') > l_t(S)$$

Thus, a base-stock of M performs better than a base-stock greater than M and hence we have justified our decision set $[0, M]$. We index the experts as $i \in \mathcal{E}$ where $\mathcal{E} = \{0, 1, 2, ...M\}$. There are $M + 1$ experts. The advises of the experts are static in the sense that $S_{t,i} = i \quad \forall t, \forall i \in \mathcal{E}$. Now, we restrict our inventory set set $\mathcal{X}$ to [-M,M]. Although it might seem non-trivial to restrict the inventory set, but as we shall show, for

the inventory process which we considered $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t$, where the purchase $\mathbf{y}_t = \max(S_t - \mathbf{x}_t, 0)$. Thus, $\mathbf{x}_{t+1} = \max(S_t, \mathbf{x}_t) - \mathbf{u}_t$. Additionally, $\mathbf{x}_1 = 0$. With a proof by contradiction, we can show that $\mathbf{x}_t \leq M$, when $\mathbf{u}_t \leq M$ and $S_t \leq M$.

Assume $\mathbf{x}_{t+1} > M$,

$$
\begin{aligned}
&\mathbf{x}_{t+1} = \max(S_t, \mathbf{x}_t) - \mathbf{u}_t \\
&\implies \max(S_t, \mathbf{x}_t) - \mathbf{u}_t > M \\
&\implies \max(S_t, \mathbf{x}_t) > M \quad \because \mathbf{u}_t \geq 0 \\
&\implies \mathbf{x}_t > M \quad \because \mathbf{x}_t \leq M
\end{aligned}
$$

This results to a recursive relation finally leading to $\mathbf{x}_1 > M$ which is a contradiction since $\mathbf{x}_1 = 0$. Hence, if the adversary generates the inventory levels according to the inventory process, then $\mathbf{x}_t \leq M$ when $\mathbf{u}_t \leq M$ and $S_t \leq M$. It is trivial to show that, $\mathbf{x}_t \geq -M$. However, the adversary isn't limited to generate the inventory level according to the process defined previously. In any case, the adversary is restricted to generate the inventory levels such that $-M \leq \mathbf{x}_t \leq M$. With these definitions of the different sets, we can now bound the loss function,

$$
\begin{aligned}
&l_t(S) = c\max(S - \mathbf{x}_t, 0) \\
&\qquad + \max\left(h(\max(S, \mathbf{x}_t) - \mathbf{u}_t), -d(\max(S, \mathbf{x}_t) - \mathbf{u}_t)\right) \\
&\geq 0 \quad \because \text{both the first and second terms are non-negative} \\
&S - \mathbf{x}_t \leq 2M \\
&\implies c\max(S - \mathbf{x}_t, 0) \leq 2cM \\
&-M \leq \max(S, \mathbf{x}_t) - \mathbf{u}_t \leq M \\
&\implies \max\left(h(\max(S, \mathbf{x}_t) - \mathbf{u}_t), -d(\max(S, \mathbf{x}_t) - \mathbf{u}_t)\right) \leq (h+d)M \\
&\therefore 0 \leq l_t(S) \leq (2c + h + d)M \\
&\implies l_t^2(S) \leq (2c + h + d)^2 M^2
\end{aligned}
$$

Thus, all the conditions on the bounds of the loss function are met. And thus, the Hedge algorithm with hyper-parameter β achieves the regret bound,

$$
R_T \leq \beta(2c + h + d)^2 M^2 T + \frac{\log(M+1)}{\beta}
$$

and with $\beta = \dfrac{1}{(2c+h+d)M}\sqrt{\dfrac{\log(M+1)}{T}}$,

$$R_T \leq 2(2c+h+d)M\sqrt{\log(M+1)T}$$

Now, we perform some simulations and compute the regret of the hedge algorithm.

Simulations

We now perform simulations of the Hedge algorithm under the memoryless inventory base-stock problem as defined earlier. It is simple to compute the regret in this case since the performance of the individual experts are tracked. We run the inventory and demand generating adversary and the Hedge algorithm with the following setting,

- The upper bound on the demand generated by environment is $M = 50$.
- The demand $\mathbf{u}_t$ is generated by a Poisson process with $\lambda = M/2 = 25$.
- The Poisson distribution is truncated at $M = 50$.
- The values of $c = 10$,$d = 15$,$h = 5$, satisfying the assumption $d > c$.
- There are 51 experts with expert advises $0, 1, 2, ...50$.
- The inventory levels at time t are generated as $\mathbf{x}_t = \max(S_{t-1}, \mathbf{x}_{t-1}) - \mathbf{u}_{t-1}$, where S_{t-1} is the decision taken by the algorithm in step $t-1$.
- However, the underlying process is not revealed to the algorithm.
- The simulation is performed for T steps and the regret is computed.
- The simulation is repeated $n = 10$ times and both the maximum and the average of the regrets computed for each iteration is plotted.
- The simulations are repeated for $T = 1, 2 \ldots 1000$

We now plot the regrets obtained from the simulation, both the average and maximum over repetitions. We also plot the hedge algorithm's decisions along with the best expert advise and the demand chosen by the environment for a single instance.

Note that regret per time rather than regret is plotted. Clearly, the empirically computed regret for the base-stock hedge algorithm is sub-linear as seen from the plots in Fig.3.2 and Fig.3.3. The upper bound of $\mathcal{O}(1/\sqrt{T})$ is also shown for better comparison. Additionally, the decision plot shows that the best expert gets sampled more often as the time progresses. The probability of sampling the best expert grows close to 1 as time steps progress.

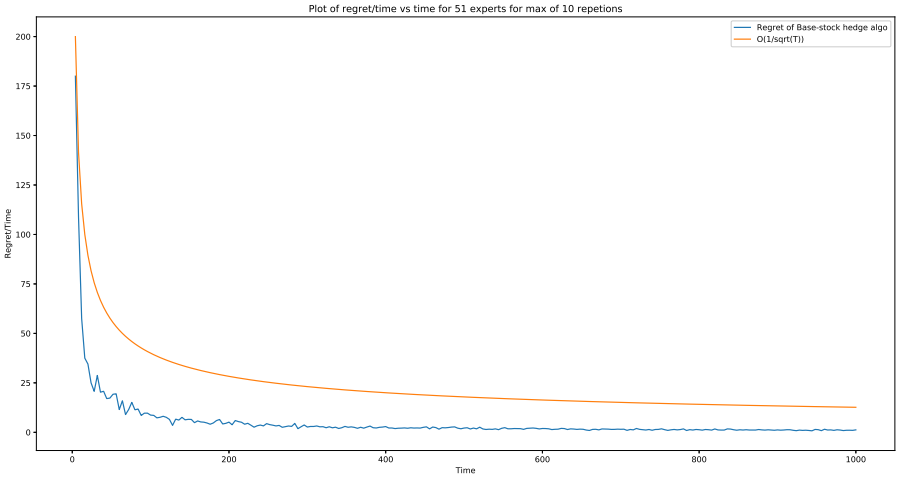


Figure 3.2: Regret/time plot for the base-stock hedge algorithm. Max over 10 repetitions is shown

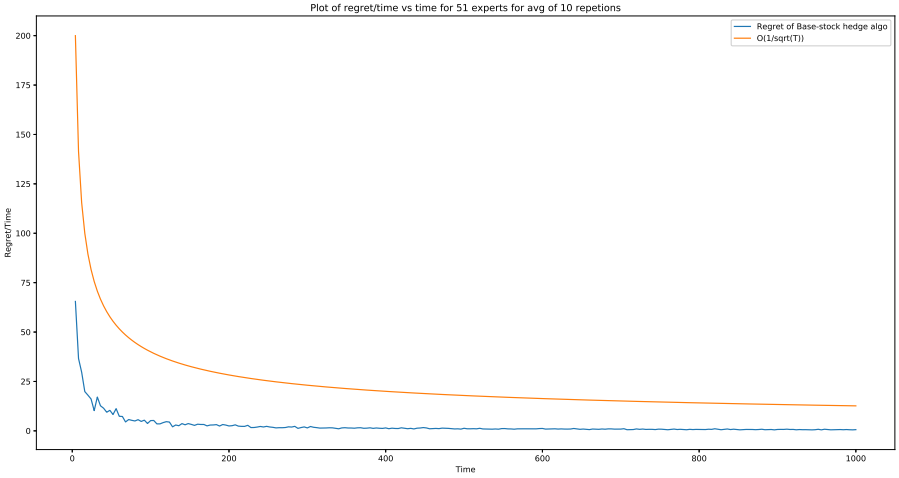


Figure 3.3: Regret/time plot for the base-stock hedge algorithm. Average over 10 repetitions is shown

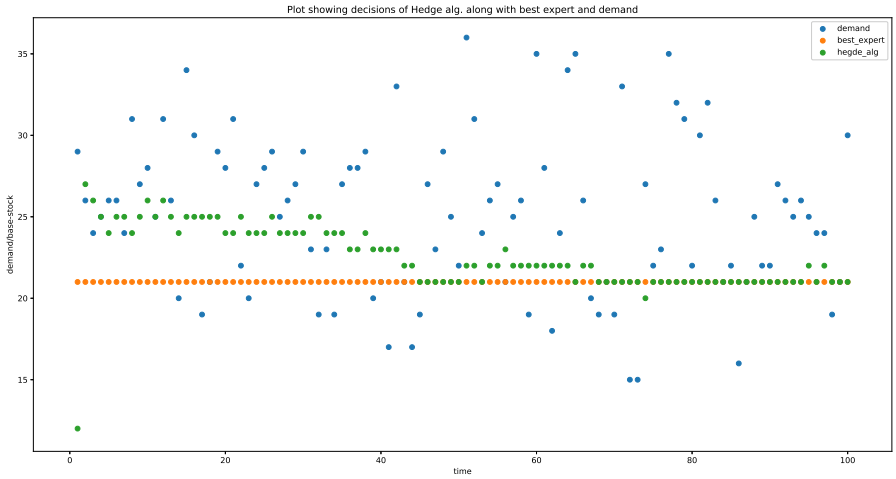


Figure 3.4: The decisions of the base-stock hedge algorithm along with the incoming demands and the best expert advise for a single instance of 100 steps.

3.4 Inventory Problem under the OCO setting

It would indeed be an interesting idea to extend the inventory problem with base-stock decisions to the Online Convex Optimisation setting. However, the convexity of the loss functions suffered by the learner is a crucial condition as we have seen in section 1.4. The convexity of the base-stock loss function defined in Eq.3.6 cannot be established. In fact, for certain values of underlying parameters, the function turns out to be non-convex. Therefore, even though we could apply OCO algorithms to the base-stock inventory problem, we will not benefit from the theoretical guarantees which are established for convex functions. Therefore, we revert back to the inventory problem with direct control on the purchases. In essence, we explore the problem as defined in section 3.1 directly without involving base-stocks.

Thus, the problem is now involves the learner's decision $\mathbf{y}_t$ chosen from the set $\mathbf{Y}$, the demand $\mathbf{u}_t$ chosen by the adversary from the set $\mathcal{U}$ and the inventory level $\mathbf{x}_t$ chosen from the set $\mathcal{X}$. As we shall explore, the guarantees of the OCO algorithms depend only on the convexity of the decision set $\mathcal{Y}$ and independent of the characteristics of the sets $\mathbf{U}$ and $\mathbf{X}$. We consider the case where $\mathcal{X} = \mathbb{R}$ and $\mathcal{U} = [0, M]$. The purchase decision $\mathbf{y}_t$ is upper bounded by C which the maximum purchase capacity and lower bounded, of course, by 0. Thus, the decision set is $[0, C]$ which is indeed convex. The process defined is exactly the same as defined in section 3.1. Additionally, as established in

section 3.1.2, the loss function defined in Eq. 3.1

$$f_t(\mathbf{y}_t) = c\mathbf{y}_t + \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t))$$

is a convex function. The convexity conditions of the OCO setting on the decision set and the loss function have been satisfied. The static regret of an algorithm $\mathcal{A}$ is defined in section 3.1.1 as,

$$R_T(\mathcal{A}) = \sum_{t=1}^{T} f_t(\mathbf{y}_t) - \min_{\mathbf{y}\in\mathcal{Y}} \sum_{t=1}^{T} f_t(\mathbf{y})$$

where $\mathbf{y}_t$ is the decision of the algorithm at step t.

3.4.1 Online Gradient Descent Algorithm

The online gradient descent (OGD) is an algorithm for the OCO framework which can be recovered from the more general Online Mirror Descent algorithm with the quadratic regulariser function. It is well-studied in several works dealing with OCO problems. The algorithm proceeds as follows,

Algorithm 2: Online Gradient Descent Algorithm

Input: set η, convex decision set $\mathcal{Y}$;

Initialise: $\mathbf{y}_1 \in \mathcal{Y}$;

for $t = 1, 2, 3, 4...T$ **do**

choose $\mathbf{y}_t$ as the decision;

adversary chooses loss function $f_t : \mathcal{Y} \mapsto \mathbb{R}$;

suffer loss $f_t(\mathbf{y}_t)$;

set $\mathbf{w}_{t+1} = \mathbf{y}_t - \eta g_t$ where $g_t \in \partial f_t(\mathbf{y}_t)$;

project $\mathbf{w}_t$ onto the set $\mathcal{Y}$ and set $\mathbf{y}_{t+1} = \Pi_{\mathcal{Y}}(\mathbf{w}_{t+1})$;

end

The OGD algorithm, as we have seen, is fairly straightforward. Note that g_t is the sub-gradient of f_t at $\mathbf{y}_t$. If the function is differentiable at $\mathbf{y}_t$, the sub-gradient is simply the gradient. The operation $\Pi_{\mathcal{Y}}$ is the projection operation defined as,

$$\Pi_{\mathcal{Y}}(\mathbf{w}) = \arg\min_{\mathbf{y}\in\mathcal{Y}} ||w - y||$$

for any defined norm $||.||$. We would use the Euclidean norm henceforth. We present an

important lemma shown by Shalev-Shwartz (2012) which helps prove a theorem which establishes the regret upper bound for the OGD under certain conditions on the loss function. We shall also use this important lemma with respect to the OGD algorithm which we shall use in the case of the inventory problem with memory considerations.

Lemma 1 *Let $f_1, f_2 \dots f_T$ be a sequence of convex loss functions such that f_t is L_t-Lipschitz with respect to $||.||_2$ norm. Let $\mathbf{y}_1, \mathbf{y}_2 \dots$ be the predictions of the OGD algorithm with learning rate η. Then,*

$$f_t(\mathbf{y}_t) - f_t(\mathbf{y}_{t+1}) \leq L_t||\mathbf{y}_t - \mathbf{y}_{t+1}|| \leq \eta L_t^2$$

The above lemma is used in combination with the following lemma 2 also proved in Shalev-Shwartz (2012) to result in the theorem which establishes the upper bound for the regret of the OGD algorithm.

Lemma 2 *Let $f_1, f_2 \dots f_T$ be a sequence of convex loss functions. Let $\mathbf{y}_1, \mathbf{y}_2 \dots$ be the predictions of the OGD algorithm with learning rate η. Then, for all $\mathbf{y} \in \mathcal{Y}$,*

$$\sum_{t=1}^{T} f_t(\mathbf{y}_t) - \sum_{t=1}^{T} f_t(\mathbf{y}) \leq \frac{1}{2\eta}||\mathbf{y}||_2^2 - \frac{1}{2\eta}||\mathbf{y}_1||_2^2 + \sum_{t=1}^{T} \big(f_t(\mathbf{y}_t) - f_t(\mathbf{y}_{t+1})\big)$$

Combining lemmas 1 and 2, we obtain the following theorem.

Theorem 2 *Let $f_1, f_2 \dots f_T$ be a sequence of convex loss functions such that f_t is L_t-Lipschitz with respect to $||.||_2$ norm. Let L be such that $\frac{1}{T}\sum_{t=1}^{T} L_t^2 \leq L^2$. Then, the online gradient descent algorithm with hyper-parameter η satisfies for all $\mathbf{y} \in \mathcal{Y}$,*

$$\sum_{t=1}^{T} f_t(\mathbf{y}_t) - \sum_{t=1}^{T} f_t(\mathbf{y}) \leq \frac{1}{2\eta}||\mathbf{y}||_2^2 + \eta T L^2$$

Corollary 2.1 *If $||\mathbf{y}||_2 \leq B \quad \forall \mathbf{y} \in \mathcal{Y}$, then OGD algorithm with hyper-parameter $\eta = \frac{B}{L\sqrt{2T}}$, satisfies the regret bound,*

$$R_T \leq BL\sqrt{2T}$$

This corollary is arrived at by letting $\mathbf{y} = \arg\min_{\mathbf{y}' \in \mathcal{Y}} \sum_{t=1}^{T} f_t(\mathbf{y}')$ in theorem 2 and

using the bounds on the norm of $\mathbf{y} \in \mathcal{Y}$. The obtained regret upper bound is $\mathcal{O}(\sqrt{T}$ in time and therefore sub-linear. We now need to establish that the inventory loss function is Lipschitz so that such a regret bound is applicable to the OGD algorithm for the inventory problem.

3.4.2 Inventory Management with Online Gradient Descent algorithm

One of the assumptions of theorem 2 are that the euclidean norm of any decision $\mathbf{y}_t$ is upper bounded. This is trivially satisfied for our decision set $\mathcal{Y} = [0, C]$ where $||\mathbf{y}_t||_2 \leq C$. We additionally need to prove that the convex function,

$$f_t(\mathbf{y}_t) = c\mathbf{y}_t + \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t))$$

is also Lipschitz. We present the following lemma from Shalev-Shwartz (2012).

Lemma 3 *Let $f : \mathcal{Y} \mapsto \mathbb{R}$ be a convex function. Then, f is L-Lipschitz over $\mathcal{Y}$ with respect to a norm $||.||$ iff for all $\mathbf{y} \in \mathcal{Y}$ and $\mathbf{g} \in \partial f(\mathbf{y})$, we have that $||\mathbf{g}||_* \leq L$ where $||.||_*$ is the dual norm.*

The lemma presented above is a direct result of the convexity of the loss function and the properties of norms. This lemma essentially implies that if all the sub-gradients of a convex function are upper bounded, then the function is G-Lipschitz, where G is the bound on the sub-gradients. We consider the euclidean norm. In our case of the inventory loss function,

$$\partial f_t(\mathbf{y}_t) = \begin{cases} c - d & \mathbf{y}_t < \mathbf{u}_t - \mathbf{x}_t \\ c + h & \mathbf{y}_t > \mathbf{u}_t - \mathbf{x}_t \\ [c - d, c + h] & \mathbf{y}_t = \mathbf{u}_t - \mathbf{x}_t \end{cases}$$

Thus, the norms of sub-gradients are upper bounded at all points by $\max(d - c, c + h) \leq h + d$. Therefore, by lemma 1, f_t is $(h + d)$-Lipschitz. The average Lipschitz coefficient is thus $h + d$.

With all the conditions on the decision set and the loss function being satisfied, the

OGD algorithm with a learning rate $\eta = \dfrac{C}{(h+d)\sqrt{2T}}$ applied to the inventory problem has a regret upper bound of,

$$R_T \leq C(h+d)\sqrt{2T}$$

Simulations

We now perform simulations of the OGD algorithm under the memoryless inventory problem as defined earlier with the optimal learning rate as defined earlier. Unlike the previous case, the best stationary policy cannot be computed directly. It can, however, be computed using linear programming. The following lemma which allows for the representation of max of linear functions is used in the computation.

Lemma 4 *The optimisation problem,*

$$\begin{aligned} \textit{minimise} f(x) = \max_{i=1,2\ldots m} (a_i^T x + b_i) \\ \textit{s.t} \quad h_j^T x + d_j \leq 0 \quad j = 1, 2 \ldots n \\ g_k^T x + d_k = 0 \quad k = 1, 2 \ldots l \end{aligned}$$

is equivalent to,

$$\begin{aligned} \textit{minimise} f(x) = t \\ \textit{s.t} \quad a_i^T x + b_i - t \leq 0 \quad i = 1, 2 \ldots m \\ h_j^T x + d_j \leq 0 \quad j = 1, 2 \ldots n \\ g_k^T x + d_k = 0 \quad k = 1, 2 \ldots l \end{aligned}$$

Additionally, the projection problem at each step of OGD can be represented as a constrained quadratic optimisation problem. We use the solvers in cvxopt library of python to compute the projection and the stationary policy.

- The demand $\mathbf{u}_t$ is generated by a Poisson process with $\lambda = 25$ truncated at $M = 50$.
- The values of $c = 10$,$d = 15$,$h = 5$, satisfying the assumption $d > c$.
- The inventory levels at time t are generated as $\mathbf{x}_t = \mathbf{x}_{t-1} poisso + \mathbf{y}_{t-1} - \mathbf{u}_{t-1}$, where $\mathbf{y}_{t-1}$ is the decision taken by the algorithm in step $t-1$.

- However, the underlying process is not revealed to the algorithm.
- The simulation is performed for T steps and the regret is computed.
- The simulation is repeated $n = 10$ times and both the maximum and the average of the regrets computed for each iteration is plotted.
- The capacity is set at $C = 25$.
- The simulations are repeated for $T = 1, 2 \ldots 1000$.

Figure 3.5: Regret/time plot for the OGD algorithm. Max over 10 repetitions is shown

Figure 3.6: Regret/time plot for the OGD algorithm. Average over 10 repetitions is shown

We now plot the time-divided regrets (regret/T) obtained from the simulation, both the average and maximum over repetitions. We also plot the OGD algorithm's decisions

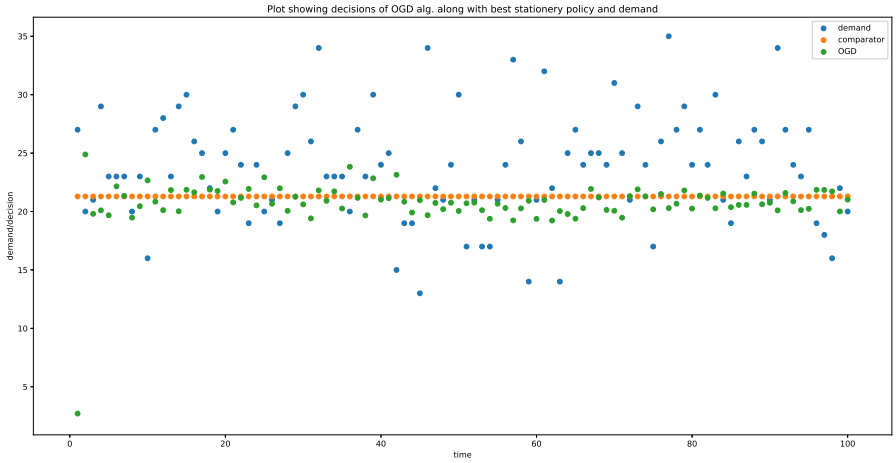


Figure 3.7: The decisions of the OGD algorithm along with the incoming demands and the best stationery policy for a single instance of 100 steps.

along with the best stationary policy and the demands chosen by the environment for a single instance. For ease of comparison, the regret upper bound is plotted in the graph as well. Thus the empirical results are consistent with the theoretical upper bounds. Note that, by increasing or decreasing the capacity, the nature of the regret plot would still remain similar. However, the policy plot might change. Also, note that the empirically computed regret is not as smooth as the one computed for the Hedge algorithm. This is due to the fact that the Hedge algorithm is limited by the expert advise, whereas the OGD algorithm can choose any decision as long as the decision set constraint is satisfied and thus shows higher volatility in the regret. Additionally, in the policy plot, the Hedge algorithm tends to converge, more or less, towards the best expert whereas in the case of OGD, although the decisions after sufficient number of step are close to the best stationary policy, there is still some variability in them.

CHAPTER 4

OCO WITH BOUNDED MEMORY AND INVENTORY MANAGEMENT

4.1 Setup of Memory in the Inventory Problem

In the formulation of the inventory management problem, we defined the single-step loss as

$$f_t(\mathbf{y}_t) = c\mathbf{y}_t + \max(h(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t), -d(\mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t))$$

for the decision $\mathbf{y}_t \in \mathcal{Y}$ and demands $\mathbf{u}_t \in \mathcal{U}$ for all $t \in [T]$, with $\mathcal{Y} = [0, C]$ and $\mathcal{U} = [0, M]$. Previously, in the memoryless analysis, we assumed that the inventory evolution was unknown to the learner. However, when the inventory evolution process $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{y}_t - \mathbf{u}_t$ is known to the learner, $\mathbf{x}_t$ itself is a function of the previous decisions and demands. Unrolling the recursion, we find that,

$$\mathbf{x}_t = \sum_{k=1}^{t-1} \mathbf{y}_k - \sum_{k=1}^{t-1} \mathbf{u}_k$$

This reveals that if the inventory evolution is known to the learner, then the single step loss at step t is actually a function of the current and all the previously made decisions $\mathbf{y}_t, \mathbf{y}_{t-1}, \dots \mathbf{y}_1$. Hence, we define the single step loss at step t as,

$$f_t(\mathbf{y}_t, \mathbf{y}_{t-1} \dots \mathbf{y}_1) = c\mathbf{y}_t + \max\Big(h\big(\sum_{k=1}^{t} \mathbf{y}_k - \sum_{k=1}^{t} \mathbf{u}_k\big), -d\big(\sum_{k=1}^{t} \mathbf{y}_k - \sum_{k=1}^{t} \mathbf{u}_k\big)\Big)$$

Such a problem, where the step loss suffered is a function of the current as well as previous decisions, is said to have memory of the previous decisions made by the learners/algorithms. However, in the inventory problem, the memory is unbounded and grows with time. Under such a problem, it is, in fact, impossible to obtain useful regret bounds for any algorithm. Thus, we slightly modify the original problem to include a

bound on the memory m. Such an assumption is practical too in that many goods expire within a limited time and cannot be stored unsold forever. The modified inventory evolution process is now,

$$\mathbf{x}_t = \sum_{k=t-m}^{t} \mathbf{y}_k - \sum_{k=t-m}^{t} \mathbf{u}_k$$

and the single step loss at step t is,

$$\begin{aligned} & f_t(\mathbf{y}_t, \mathbf{y}_{t-1} \dots \mathbf{y}_{t-m}) \\ & \quad = c\mathbf{y}_t + \max\Big(h\big(\sum_{k=t-m}^{t} \mathbf{y}_k - \sum_{k=t-m}^{t} \mathbf{u}_k\big), -d\big(\sum_{k=t-m}^{t} \mathbf{y}_k - \sum_{k=t-m}^{t} \mathbf{u}_k\big)\Big) \end{aligned} \tag{4.1}$$

The memory is now bounded by m. The Online Convex Optimisation is slightly modified to deal with such a case of bounded memory.

4.2 The Framework of Online Convex Optimisation with Memory

The classical OCO framework is slightly modified to accommodate m bounded memory as follows,

for $t = m + 1, ..., T$ **do**

1: the learner chooses $\mathbf{y}_t \in \mathcal{Y}$

2: the adversary chooses the loss $f_t : \mathcal{Y}^{m+1} \mapsto \mathbb{R}$;

3: the learner suffers a loss of $f_t(\mathbf{y}_t, \mathbf{y}_{t-1}, \dots, \mathbf{y}_{t-m})$;

end for

The notion of regret changes slightly from the case of memoeyless OCO. For the framework of OCO with memory we consider the static policy regret defined as,

$$R_T = \sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \min_{\mathbf{y} \in \mathcal{Y}} \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{y})$$

As in the case of memoryless OCO, the decision set $\mathcal{Y}$ is bounded and convex. The

loss functions $\{f_t\}_{t=1}^{T}$ are convex in memory. This means that the function

$$\tilde{f}_t(\mathbf{y}) = f_t(\mathbf{y}, \mathbf{y}, \dots \mathbf{y})$$

is convex is $\mathbf{y}$. Note that the function $\tilde{f}_t : \mathcal{Y} \mapsto \mathbb{R}$ has the single-step decision set as the domain. This definition of $\tilde{f}_t$ is very important and will be used repeatedly henceforth.

4.2.1 Online Gradient Descent Algorithm for the OCO with Memory Framework

Anava *et al.* (2015) show that when the loss function f_t in the framework of OCO with memory satisfies the following assumption,

Assumption 1 *The function $f_t : \mathcal{Y}^{m+1} \mapsto \mathbb{R}$ is L-Lipschitz i.e.,*

$$|f_t(\mathbf{x}_0, ...\mathbf{x}_m) - f_t(\mathbf{y}_0, ...\mathbf{y}_m)|^2 \leq L^2 \sum_{i=0}^{m} ||\mathbf{x}_i - \mathbf{y}_i||_2^2 \leq L^2 \Big(\sum_{i=0}^{m} ||\mathbf{x}_i - \mathbf{y}_i||_2 \Big)^2$$

then the following theorem holds,

Theorem 3 *If assumption 1 is satisfied in the OCO with memory problem, with memory bound m, then the following holds for any sequence of decisions $\{\mathbf{y}_t\}_{t=1}^{T} \in \mathcal{Y}^T$ and for all $\mathbf{y} \in \mathcal{Y}$,*

$$\sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{y}) \leq m^2 L \sum_{t=m+1}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{y})$$

Algorithm 3: Online Gradient Descent for Memory OCO (OGDM)

Input: set η, convex decision set $\mathcal{Y}$;

Initialise: $\mathbf{y}_1, \mathbf{y}_2 \dots \mathbf{y}_m \in \mathcal{Y}$;

for $t = m+1, \dots T$ **do**

choose $\mathbf{y}_t$ as the decision;

adversary chooses loss function $f_t : \mathcal{Y}^{\Updownarrow+\infty} \mapsto \mathbb{R}$;

suffer loss $f_t(\mathbf{y}_t, \dots \mathbf{y}_{t-m})$;

define $\tilde{f}_t(\mathbf{y}_t) = f_t(\mathbf{y}_t, \dots \mathbf{y}_t)$;

set $\mathbf{w}_{t+1} = \mathbf{y}_t - \eta g_t$ where $g_t \in \partial \tilde{f}_t(\mathbf{y}_t)$;

project $\mathbf{w}_t$ onto the set $\mathcal{Y}$ and set $\mathbf{y}_{t+1} = \Pi_{\mathcal{Y}}(\mathbf{w}_{t+1})$;

end

This result provides a very useful algorithm agnostic upper bound on the static policy regret. Essentially, this links the static policy regret of an OCO problem with memory with the loss function f_t to the static regret of a related memoryless OCO problem with single step loss $\tilde{f}_t$ and an additional switching cost. In some sense, this upper bound is very useful in being able to convert a problem of OCO with memory to the memoryless OCO setting. An algorithm which has a specific upper bound the memoryless OCO problem with loss function $\tilde{f}_t$ retains the upper bound with with an additional switching cost for the OCO problem with bounded memory. With some additional assumptions, the OGD algorithm applied to the unary loss function $\tilde{f}_t$ achieves a sub-linear regret bound.

This algorithm Online Gradient Descent for Memory OCO (OGDM) is described in Algorithm 3. The additional assumptions made to obtain the regret upper bounds for OGDM are,

Assumption 2 *The sub-gradient norm of the unary loss is at most G, i.e., for all* $\mathbf{y} \in \mathcal{V}$ *and* $t \in [T]$

$$||\mathbf{g}_t||_2 \leq G \quad \forall \mathbf{g}_t \in \partial f_t(\mathbf{y})$$

Assumption 3 *The domain* $\mathcal{Y}$ *is convex, closed, and satisfies* $||\mathbf{y} - \mathbf{y}'||_2 \leq B$ *for any* $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$. *Additionally, it is also assumed that* $0 \in \mathcal{Y}$

With these assumptions, invoking theorem 2 and lemma 3, gives the following bound for the OGD algorithm with learning rate η run on the sequence of unary loss

functions $\tilde{f}_t$,

$$\sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{y}) \leq \frac{1}{2\eta}||\mathbf{y}||_2^2 + \eta T G^2 \leq \frac{1}{2\eta}B^2 + \eta T G^2 \tag{4.2}$$

where the OGD decisions are $\{\mathbf{y}_t\}_{t=1}^{T} \in \mathcal{Y}^T$. This bound holds for all $\mathbf{y} \in \mathcal{Y}$. The only additional bound required is for the switching cost of the OGD algorithm and this is obtained by invoking lemma 1 and lemma 3 and arriving at the following upper bound

$$\sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 \leq \eta G T \tag{4.3}$$

Combining equations 4.2 and 4.3, the following corollary of theorem 3 is obtained,

Corollary 3.1 *Let the sequence of convex functions $f_1, f_2 \dots f_T$ of the OCO with memory bounded by m satisfy Assumption 1. Let $\tilde{f}_t(\mathbf{y}) = f_t(\mathbf{y}, \dots \mathbf{y})$ for all $\mathbf{y} \in \mathcal{Y}$ be the unary loss function at step t. Then, with assumptions 2 and 3 satisfied, the OGD algorithm run on the unary loss function (OGDM) algorithm with learning rate η satisfies,*

$$\sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{y}) \leq \frac{1}{2\eta}B^2 + (m^2 LG + G^2)\eta T$$

Setting $\eta = \sqrt{\frac{B^2}{2(G^2 + m^2 LG)T}}$, and $\mathbf{y} = \arg\min_{\mathbf{y}\in\mathcal{Y}} \sum_{t=1}^{T} \tilde{f}_t(\mathbf{y})$, the following static policy regret bound is obtained for the OGDM algorithm,

$$R_T = \sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \min_{\mathbf{y}\in\mathcal{Y}} \sum_{t=1}^{T} \tilde{f}_t(\mathbf{y}) \leq B\sqrt{2(G^2 + m^2 LG)T}$$

The regret bound is $\mathcal{O}(\sqrt{T})$ as in the case of OGD applied to the case of memoryless OCO.

4.2.2 OGDM for the Inventory Problem with bounded memory

In order for the previously established regret upper bounds to hold when OGDM is applied to the inventory problem, all we need to do is verifying that assumptions 1,2 and 3 hold.

The decision set $\mathcal{Y} = [0, C]$ trivially satisfies Assumption 3 with $B = C$. The loss function $f_t(\mathbf{y}_{m+1}, \dots \mathbf{y}_1)$ defined in Eq. 4.1 is convex in $(\mathbf{y}_{m+1}, \dots \mathbf{y}_1)$ since it is a maximum over linear functions. In order to verify that it satisfies the coordinate-wise Lipschitzness condition of Assumption 1, we define the set of sub-gradients of f_t as,

$$\partial f_t(\mathbf{y}_t, \mathbf{y}_{t-1} \dots \mathbf{y}_{t-m}) = \begin{cases} \{[c-d, d \dots d]\} & \sum_{k=t-m}^{t} \mathbf{y}_k < \sum_{k=t-m}^{t} \mathbf{u}_t \\ \{[c+h, h \dots h]\} & \sum_{k=t-m}^{t} \mathbf{y}_k > \sum_{k=t-m}^{t} \mathbf{u}_t \\ \{\lambda[c-d, d \dots d] + (1-\lambda) \quad [c+h, h \dots h] & \in [0,1]\} \\ & \sum_{k=t-m}^{t} \mathbf{y}_k = \sum_{k=t-m}^{t} \mathbf{u}_t \end{cases}$$

Now, the Euclidean norm of the sub-gradient can be trivially upper bounded by $L = ||h+d, h+d, \dots h+d||_2 = (h+d)\sqrt{m+1}$

The unary loss function $\tilde{f}_t$ is defined as,

$$\tilde{f}_t(\mathbf{y}) = f_t(\mathbf{y}, \dots \mathbf{y}) = c\mathbf{y} + \max\Big(h\big((m+1)\mathbf{y} - \sum_{k=1}^{t} \mathbf{u}_k\big), -d\big((m+1)\mathbf{y} - \sum_{k=1}^{t} \mathbf{u}_k\big)\Big)$$

and the corresponding sub-gradients are,

$$\partial \tilde{f}_t(\mathbf{y}_t) = \begin{cases} \{c - d(m+1)\} & \mathbf{y}_t < \frac{1}{m+1} \sum_{k=t-m}^{t} \mathbf{u}_t \\ \{c + h(m+1)\} & \mathbf{y}_t > \frac{1}{m+1} \sum_{k=t-m}^{t} \mathbf{u}_t \\ [c - d(m+1), c + h(m+1)] & \mathbf{y}_t = \frac{1}{m+1} \sum_{k=t-m}^{t} \mathbf{u}_t \end{cases}$$

The Euclidean norms of the sub-gradients are upper bounded by $G = (m+1)(h+d)$, and hence, Assumption 2 is satisfied. Applying the OGDM algorithm to the inventory problem with bounded memory and setting learning rate as,

$$\eta = \frac{C}{h+d}\sqrt{\frac{1}{2((m+1)^2 + m^2(m+1)^{3/2})T}}$$

results in the following upper bound for the static policy regret,

$$R_T \leq C(h+d)\sqrt{2((m+1)^2 + m^2(m+1)^{3/2})T}$$

Simulations

In order to empirically verify that the performance of the OGDM algorithm satisfies the regret upper bound, we simulate the same. As in the case of the OGD in the memoryless inventory problem, we can formulate the problem of finding the best static policy as a linear constrained optimisation problem and solve it using linear programming solvers in the cvxopt library of python. The simulation setup is as follows,

- The demand $\mathbf{u}_t$ is generated by a Poisson process with $\lambda = 25$ truncated at $M = 50$.
- The values of $c = 10$,$d = 15$,$h = 5$, satisfying the assumption $d > c$, are used.
- The simulation is done for T steps and the regret is computed.
- The simulation is repeated $n = 10$ times and both the maximum and the average of the regrets computed for each iteration is plotted.
- The capacity is set at $C = 25$.
- The simulations are performed for $m = 10, 100$
- The simulations are repeated for $T = m + 1, m + 2, \ldots 1000$

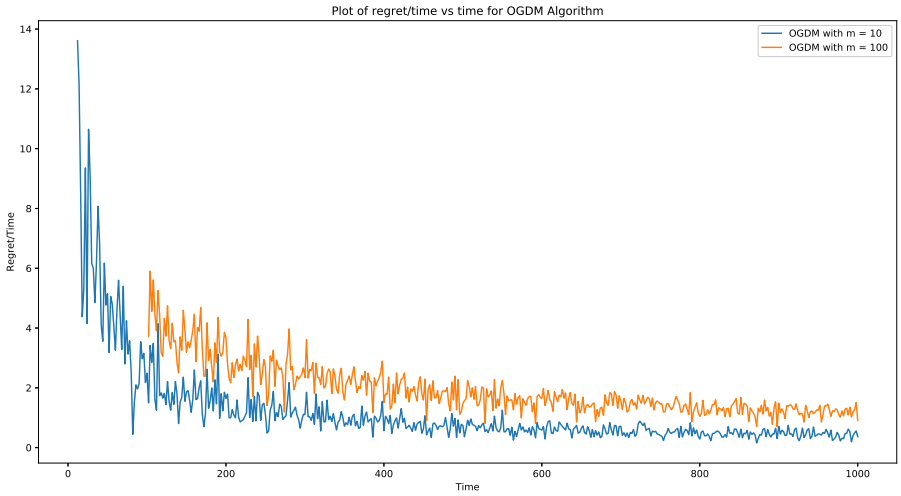


Figure 4.1: Regret/time plot for the OGDM algorithm. Max over 10 repetitions is shown. Plots for both $m = 10, m = 100$ are shown

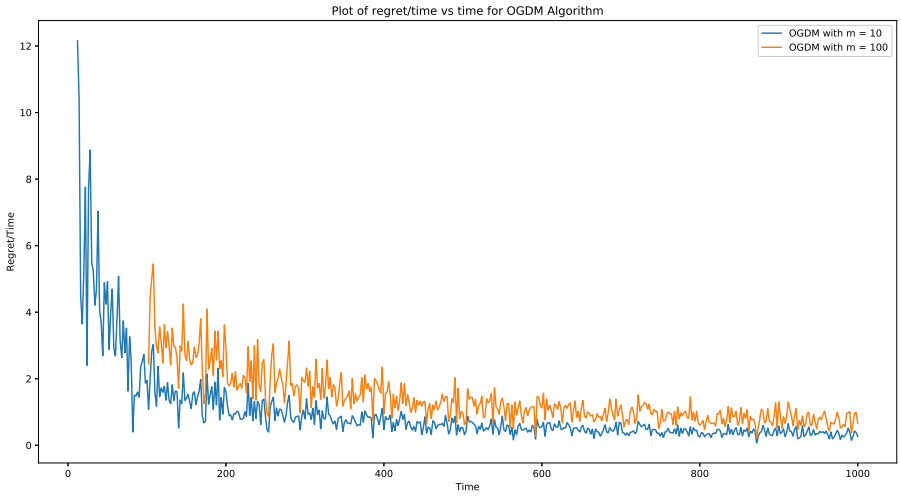


Figure 4.2: Regret/time plot for the OGDM algorithm. Average over 10 repetitions is shown. Plots for both $m = 10, m = 100$ are shown

We plot the time divided regrets (regret/T) obtained from the simulation, both the average and maximum over repetitions. We also plot the OGDM algorithm's decisions along with the best stationary policy and the demands chosen by the environment for a single instance. These regrets are obtained for various memory bounds m. We expect the regrets increase as we increase m and this is ,in fact. what we observe. The regret plots are shown in Fig. 4.1 and Fig 4.2. Clearly, as the bound on memory increases from 10 to 100, the corresponding regrets increase too. The policy of the OGDM algorithm behaves similar to the OGD algorithm although with slightly enhanced variance.

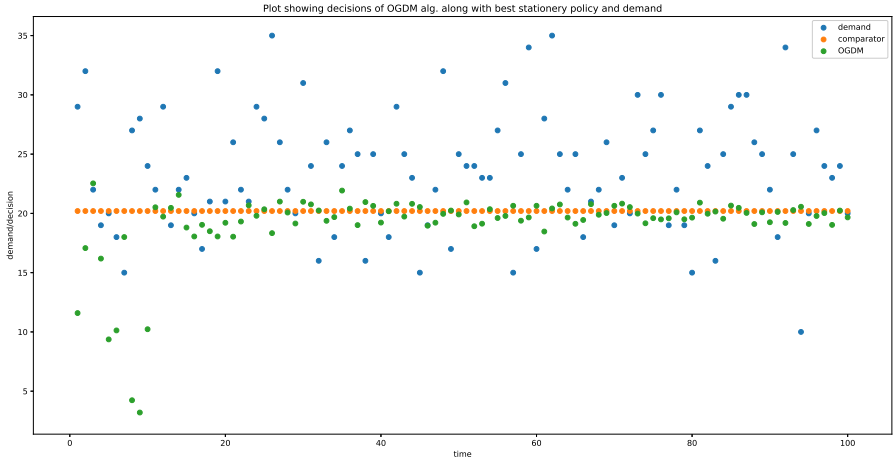


Figure 4.3: The decisions of the OGDM algorithm along with the incoming demands and the best stationery policy for a single instance of 100 steps for $m = 10$.

4.3 Adaptive Online Learning for OCO with Bounded Memory

So far, we had considered the static regret both in the case of OCO with and without memory. However, when the environment is dynamic, the notion of static regret is no longer relevant. Hence, we study the notion of dynamic regret. Zinkevich (2003) presents a version of the dynamic regret where the comparators can be any sequence of decisions within the decision set. for the memoryless setting. We adapt this notion of dynamic regret to the OCO with memory framework and define the dynamic policy regret of an algorithm that generates the decisions $\{\mathbf{y}_t\}_{t=1}^T$ as,

$$R_T^D(\mathbf{z}_T, \mathbf{z}_{T-1} \dots \mathbf{z}_1) = \sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t)$$

where the comparator sequence is $\mathbf{z}_t\}_{t=1}^T$ with $\mathbf{z}_t \in \mathcal{Y}$ for all $t \in [T]$. From this section onwards, we consider the dynamic regret for which we establish guarantees for algorithms we devise. Assuming that the loss function f_t satisfies Assumption 1 and is L-Lipschitz, and applying theorem 3 to the definition of dynamic regret, we obtain,

$$\begin{aligned} R_T^D(\mathbf{z}_T, \mathbf{z}_{T-1} \dots \mathbf{z}_{m+1}) &= \sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \\ &\leq m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \end{aligned} \tag{4.4}$$

The above upper bound represents the switching cost $\sum_{t=m+1}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2$ and the dynamic regret with respect to the memoryless loss function $\tilde{f}_t$ defined as $\sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t)$. Similar to the case of the static policy regret, we have this upper bounds maps the OCO with memory problem to the memoryless OCO setting and if we obtain an algorithm that suffers sub-linear dynamic regret with respect to the unary loss function, and has a sub-linar switching cost, then the from Eq. 4.4, the dyanmic policy regret is sub-linear as well. Hence, we consider an algorithm that suffers sub-linear dynamic regret with respect to the memoryless loss functions $\tilde{f}_t$ and adapt it to our case so that it suffers sub-linear switching cost too.

4.3.1 An Algorithm Satisfying Sub-Linear Dynamic Regret Upper Bound for the Memoryless OCO Case

Zhang *et al.* (2018) establish the following theorem which bounds the dynamic regret suffered by the OGD algorithm for the memoryless OCO framework with memoryless convex loss functions h_t at every step t.

Theorem 4 *Let $h_1, h_2 \dots h_T$ be sequences of convex loss functions which have sub-gradients whose Euclidean norms are upper bounded by G. Let the diameter of the decision set $\mathcal{Y}$ which includes $\mathbf{0}$ be B. Then, for the sequence of decisions of the OGD algorithm $\{\mathbf{y}_t\}_{t=1}^T$,*

$$\sum_{t=1}^{T} h_t(\mathbf{y}_t) - \sum_{t=1}^{T} h_t(\mathbf{z}_t) \leq \frac{1}{4\eta}(7B^2 + 4BP_T) + \frac{\eta G^2 T}{2}$$

where $\mathbf{z}_t \in \mathcal{Y}$ for all $t \in [T]$, and $P_T = \sum_{t=2}^{T} ||\mathbf{z}_t - \mathbf{z}_{t-1}||_2$

Setting $\eta = \eta^* = \sqrt{\frac{7B^2 + 4BP_T}{2G^2T}}$, we obtain a $\mathcal{O}(\sqrt{T(1+P_T)})$ dynamic regret upper bound. The work also establishes an algorithm-agnostic lower bound on the dynamic regret of $\Omega(\sqrt{T(1+P_T)})$ whose order is matched by that of the dynamic regret upper bound of OGD with optimal η^*. However, setting this optimal η^* requires the knowledge of the path length P_T to the learner a priori, which might not be the case. Thus, the work devises a set of learning rates $\mathcal{H}$ and experts $\mathcal{E}$, both uniquely indexed by $i \in [N]$ where $N = |\mathcal{H}|$. The expert i runs the OGD algorithm on the memoryless OCO setting with a learning rate $\eta_i \in \mathcal{H}$ and makes the decision $\mathbf{y}_{t,i} \in \mathcal{Y}$ at step t as suffers loss $h_t(\mathbf{y}_{t,i})$. Thus, each expert is an OGD algorithm with a specific learning rate. This OGD algorithm represented by each expert i is called the expert-algorithm. To generate the final decision from the individual expert decisions, a meta-algorithm which is the exponential weighting algorithm (Cesa-Bianchi and Lugosi (2006)) with parameter β is used on the individual experts with the loss function h_t at each step t. This proposed algorithm is named adaptive learning for dynamic environment (Ader). Essentially, this

splits the memoryless dynamic regret as,

$$\sum_{t=m=1}^{T} h_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} h_t(\mathbf{z}_t) = \Big(\sum_{t=m=1}^{T} h_t(\mathbf{y}_t) - \sum_{t=m=1}^{T} h_t(\mathbf{y}_{t,i}) \Big) + \Big(\sum_{t=m=1}^{T} h_t(\mathbf{y}_{t,i}) - \sum_{t=m+1}^{T} h_t(\mathbf{z}_t) \Big) \quad \text{for some } i \in [N] \tag{4.5}$$

the first term is the related to the regret of the exponential weighting algorithm which, when the loss function h_t satisfies $a \leq h_t(\mathbf{y}) \leq a+c$, is upper bounded by $\mathcal{O}(\sqrt{T \log(N)})$ for an appropriate $\beta = \mathcal{O}(\sqrt{\log(N)/T})$ independent of the path length. The construction of the set of learning rates $\mathcal{H}$ is such that there exists an expert i^* such that $\eta_{i^*} \in \mathcal{H}$ has an individual expert memoryless dynamic regret upper bounded by $\mathcal{O}(\sqrt{T(1+P_T)})$, the bound achieved in theorem 4. Note that the construction of the set is independent of the path length P_T. Setting $i = i^*$ in Eq. 4.5 and summing the 2 terms, we find that the dynamic regret of the overall Ader algorithm is upper bounded by $\mathcal{O}(\sqrt{T(1+P_T)})$. In the next section, we adapt the Ader algorithm to the framework of OCO with memory with some modifications to the expert and meta algorithms.

4.3.2 Adaptation of ADER algorithm to the OCO Problem with Memory

We now shift our focus back to the case of dynamic regret in the case of OCO with memory. When assumptions 1,2 and 3 are satisfied, we can combine theorems 3 and 4, with Eq. 4.3, and we obtain the following theorem,

Theorem 5 *Let the sequence of convex loss functions with bounded memory m, $f_1, \dots f_T$ satisfy Assumption 1 (L-Lipschitz). Let the unary loss functions defined as $\tilde{f}_t(\mathbf{y}) = f_t(\mathbf{y}, \dots \mathbf{y})$ satisfy Assumption 2 (sub-gradients' norms bounded by G). Let the decision set $\mathcal{Y}$ satisfy Assumption 3 (bounded convex set with diameter B). Then, if $\{\mathbf{y}_t\}_{t=1}^{T}$*

are generated by running the OGD algorithm on the unary loss functions $\tilde{f}_t$ for $t \in [T]$,

$$\begin{aligned} R_T^D(\mathbf{z}_T, \mathbf{z}_{T-1} \dots \mathbf{z}_{m+1}) &= \sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \\ &\leq m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \\ &\leq \frac{1}{4\eta}(7B^2 + 4BP_T) + \frac{\eta(2m^2LG + G^2)T}{2} \end{aligned}$$

where $P_T = \sum_{t=m+1}^{T} ||\mathbf{z}_t - \mathbf{z}_{t-1}||_2$ is the path length of the sequence of comparators $\{\mathbf{z}_t\}_{t=m+1}^{T}$.

The above bound is very similar to that of theorem 4. Similar to the previous memoryless case, setting the optimal $\eta = \eta^* = \sqrt{\frac{7B^2 + 4BP_T}{2(2m^2LG + G^2)T}}$ requires the knowledge of P_T, which may not be known to the learner a priori. Hence, we try using the Ader algorithm on the unary loss function $\tilde{f}_t$ for $t \in [T]$. Using Eq. 4.4, we bound the dynamic policy regret in an OCO problem with memory as,

$$\begin{aligned} R_T^D(\mathbf{z}_T, \mathbf{z}_{T-1} \dots \mathbf{z}_{m+1}) &= \sum_{t=m+1}^{T} f_t(\mathbf{y}_t, ..., \mathbf{y}_{t-m}) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \\ &\leq m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \end{aligned}$$

Running the Ader algorithm only guarantees that the second term, i.e. the memoryless dynamic regret is upper bounded by $\mathcal{O}(\sqrt{T(1 + P_T)})$. There is no guarantee on the corresponding switching cost of Ader. Hence we adopt a split different from the previous memoryless case. We define a set of learning rates $\mathcal{H}$ and set of experts $\mathcal{E}$ indexed by $i \in [N]$ for $N = |\mathcal{H}|$. We split the upper bound on the dynamic policy regret as,

$$\begin{aligned} & m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \\ &= \Big(m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) \\ &- m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 - \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_{t.i}) \Big) \\ &+ \Big(m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_{t.i}) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \Big) \end{aligned} \tag{4.6}$$

for some $i \in [N]$. With this split we define the following adaptation of the Ader algorithm which we name Adaptive learning for Dynamic Environment with Memory (Aderm) which consists of 2 parts, the expert algorithm and the meta algorithm.

Algorithm 4: Aderm: Meta Algorithm

Input: set of learning rates $\mathcal{H} = \{\eta_1, \eta_2 \dots \eta_N\}$ as defined in Eq. 4.7 , convex decision set $\mathcal{Y}$, set parameter β ;
Sort the step sizes in ascending order $\eta_1 \leq \eta_2 \cdots \leq \eta_N$;
Activate a set of experts $\{E_i | i \in [N]\}$ by invoking Algorithm 5 for each $\eta_i \in \mathcal{H}$;
Set $w_{1,i} = \dfrac{W}{i(i+1)}$ for $W = 1 + \frac{1}{N}$;
for $t = m+1, \dots T$ **do**
  Receive $\mathbf{y}_{t,i}$ from each expert E_i ;
  Output

$$\mathbf{y}_t = \sum_{i=1}^{N} w_{t,i} \mathbf{y}_{t,i}$$

  ;
  Adversary chooses loss function $f_t : \mathcal{Y}^{m+1} \mapsto \mathbb{R}$;
  Observe the unary loss function $\tilde{f}_t(.)$;
  Each expert suffers loss $l_t(i) = m^2 L ||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 + \tilde{f}_t(\mathbf{y}_{t,i})$;
  Update the weight of each expert as,

$$w_{t+1,i} = \frac{w_{t,i} \exp(-\beta l_t(i))}{\sum_{j=1}^{N} w_{t,j} \exp(-\beta l_t(j))}$$

  ;
  Send sub-gradient $\mathbf{g}_{t,i} \in \partial \tilde{f}_t(\mathbf{y}_{t,i})$ to each expertE_i ;
end

Algorithm 5: Aderm: Expert Algorithm

Input: learning rate η, convex decision set $\mathcal{Y}$;
Initialise: $\mathbf{y}_1^\eta, \mathbf{y}_2^\eta \dots \mathbf{y}_m^\eta \in \mathcal{Y}$;
for $t = m+1 \dots T$ **do**
  Submit $\mathbf{y}_t^\eta$ to the meta-algorithm ;
  Receive sub-gradient $\mathbf{g}_t^\eta$ from meta algorithm;
  Set $\mathbf{y}_{t+1} = \Pi_{\mathcal{Y}}(\mathbf{y}_t^\eta - \eta \mathbf{g}_t^\eta)$;
end

Before presenting the theorem which establishes the dynamic policy regret upper bound for the Aderm algorithm, we make the following assumption on the unary loss function $\tilde{f}_t$.

Assumption 4 *We assume that the unary function* $a \leq \tilde{f}_t(\mathbf{y}) \leq a + d \; \forall \mathbf{y} \in \mathcal{Y}$ *and* $\forall t \in [T]$*, for some* $a \in \mathbb{R}$ *and* $d > 0$*.*

We now present the main theorem of this section which establishes the dynamic policy regret of the Aderm algorithm.

Theorem 6 *Set*

$$\mathcal{H} = \left\{ \eta_i = 2^{i-1} B \sqrt{\frac{7}{2(G^2 + m^2 LG)T}} \middle| i = 1, \dots N \right\} \tag{4.7}$$

where $N = \lceil \frac{1}{2} \log_2(1 + 4T/7) \rceil + 1$ *and* $\beta = \sqrt{8/(9(m^2 LB + d)^2 T)}$ *in Algorithm 4 under Assumptions 1,2,3 and 4 for any comparator sequence* $\mathbf{z}_{m+1}, \mathbf{z}_{m+2} \dots \mathbf{z}_T \in \mathcal{Y}$ *with path length* $P_T = \sum_{t=m+1}^{T} ||\mathbf{z}_t - \mathbf{z}_{t-1}||_2$*, the proposed Aderm algorithm in the OCO setting with m-bounded memory satisfies,*

$$\begin{aligned} R_T^D(\mathbf{z}_T, \dots \mathbf{z}_{m+1}) \leq & \frac{3\sqrt{2m^2 LG + G^2}}{4} \sqrt{2T(7B^2 + 4BP_T)} \\ & + \frac{3(m^2 LB + d)\sqrt{2T}}{4} (1 + 2\log(k+1)) \end{aligned} \tag{4.8}$$

for $k = \lfloor \frac{1}{2} \log_2(1 + \frac{4P_T}{7B}) \rfloor + 1$.

The order of the upper bound is $\mathcal{O}(\sqrt{T(1 + P_T)})$, the same as the one obtained by the Ader algorithm in the memoryless case. The proof of this theorem is present in Appendix A and it follows a similar outline to the proof of theorem 4 presented in Zhang *et al.* (2018).

4.3.3 Application of ADERM Algorithm to the Inventory Problem

In order for the dynamic policy regret bound specified in Theorem 6 to hold when the Aderm algorithm is applied to the inventory problem with m-bounded memory, we only need to verify that Assumptions 1,2,3 and 4 are satisfied. We had already verified

that assumption 1,2 and 3 hold for the inventory problem with m-bounded memory in section 4.2.2 where the loss functions with memory f_t are L-Lipschitz, the norms of the sub-gradients of the unary loss functions $\tilde{f}_t$ are upper bounded by G and the diameter of the decision set $\mathcal{Y} = [0, C]$ is upper bounded by B for all $t \in [T]$, with $L = (h+d)\sqrt{m+1}$, $G = (m+1)(h+d)$ and $B = C$. We only need to verify that assumption 4 is satisfied by the unary loss function $\tilde{f}_t$.

$$\begin{aligned}\tilde{f}_t(\mathbf{y}) &= c\mathbf{y} + \max\Big(h\big(m\mathbf{y} - \sum_{k=t-m}^{t}\mathbf{u}_k\big), -d\big(m\mathbf{y} - \sum_{k=t-m}^{t}\mathbf{u}_k\big)\Big) \\ &\geq 0 \quad \because \mathbf{u}_t \in \mathcal{U} = [0, M]\end{aligned}$$

We also have $\mathbf{y} \leq C$ and,

$$\begin{aligned}\max\Big(h\big(m\mathbf{y} - \sum_{k=t-m}^{t}\mathbf{u}_k\big), -d\big(m\mathbf{y} - \sum_{k=t-m}^{t}\mathbf{u}_k\big)\Big) &\leq (h+d)\Big|m\mathbf{y} - \sum_{k=t-m}^{t}\mathbf{u}_k\Big| \\ &\leq m(h+d)\max(C, M) \\ &\leq m(h+d)(C+M) \\ \implies 0 \leq \tilde{f}_t(\mathbf{y}) &\leq C(c+mh+md) + M(mh+md)\end{aligned}$$

Thus, assumption 4 is satisfied by $\tilde{f}_t$. Hence, the dynamic policy regret upper bound of $\mathcal{O}(\sqrt{T(1+P_T)})$ for the Aderm algorithm established by Theorem 6 holds for the inventory problem with m-bounded memory.

Simulations

We simulate the inventory problem with m-bounded memory with the Aderm algorithm applied to it to empirically verify that it's performance satisfies the regret upper bound. In order to make the environment dynamic, we change the demand generating process at step $t = \lceil \frac{T}{2} \rceil$ by changing the rate of the Poisson process. Thus, the comparator sequence is defined as $\mathbf{z}_t = \arg\min_{\mathbf{z}\in\mathcal{Y}} \sum_{t=m+1}^{\lceil T/2 \rceil - 1} \tilde{f}_t(\mathbf{z})$ for $t < \lceil \frac{T}{2} \rceil$ and $\mathbf{z}_t = \arg\min_{\mathbf{z}\in\mathcal{Y}} \sum_{t=\lceil T/2 \rceil}^{T} \tilde{f}_t(\mathbf{z})$ for $\lceil \frac{T}{2} \rceil \leq t \leq T$. Like the previous case of OGDM, we can find the comparators by solving the minimisation problem using linear programming solvers of cvxopt library in Python. The simulation setup is as follows,

- The demand $\mathbf{u}_t$ is generated by a Poisson process with $\lambda = 25$ truncated at $M = 50$ for $t < \lceil \frac{T}{2} \rceil$.

- The demand $\mathbf{u}_t$ is generated by a Poisson process with $\lambda = 10$ truncated at $M = 50$ for $\lceil \frac{T}{2} \rceil \leq t \leq T$.
- The values of $c = 10$,$d = 15$,$h = 5$, satisfying the assumption $d > c$ are used.
- The simulation is done for T steps and the regret is computed.
- The simulation is repeated $n = 2$ times and both the maximum and the average of the regrets computed for each iteration is plotted.
- The capacity is set at $C = 25$.
- The simulations are performed for memory $m = 10$.
- The simulations are repeated for $T = m + 1, m + 2, \ldots 1000$

We plot the maximum and average regrets/time as in the previous algorithms. Additionally, we also plot the single-step regret/time evolution for a single instance of the above simulation for 1000 steps. The decisions of the Aderm algorithm along with the incoming demands and the dynamic comparator sequence is also plotted. As we can see, the regret plots in Fig. 4.4 and Fig. 4.5 are sub-linear in nature. By noticing the jump in regret, close to step 500, we can see the effect of the changing comparator at step $\lceil T/2 \rceil$ in the single-step regret evolution plot in Fig. 4.6. The change in the pattern of decisions of Aderm once the demand generating process is changed is clearly seen in Fig. 4.7. Note that the comparator is changed only once in this simulation.

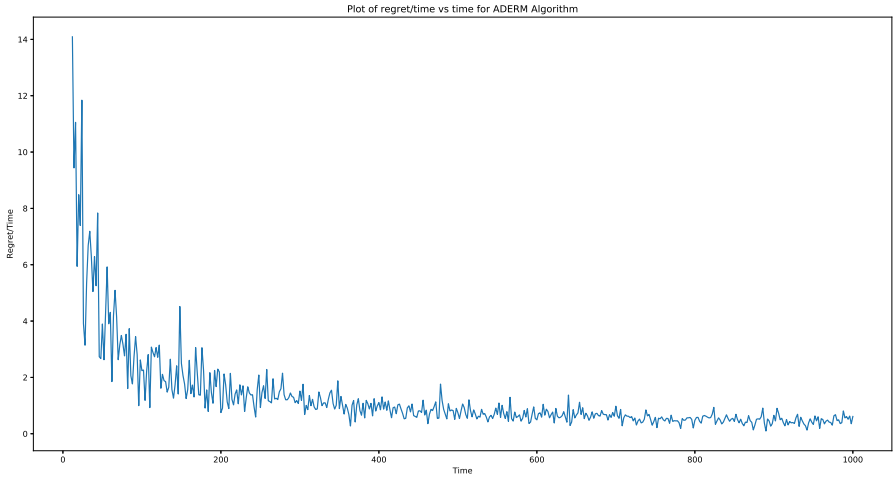


Figure 4.4: Regret/time plot for the Aderm algorithm. Max over 10 repetitions is shown.

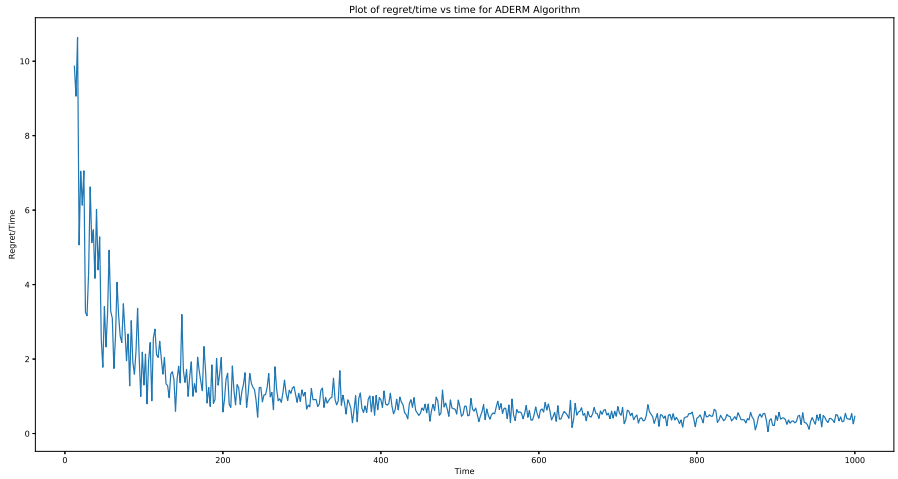


Figure 4.5: Regret/time plot for the Aderm algorithm. Average over 10 repetitions is shown.

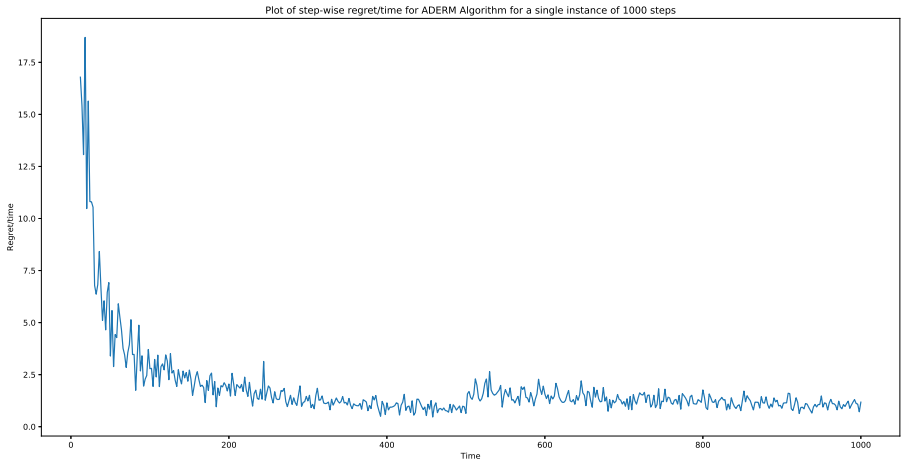


Figure 4.6: Step-wise regret/time evolution for the Aderm algorithm for a single instance of 1000 steps

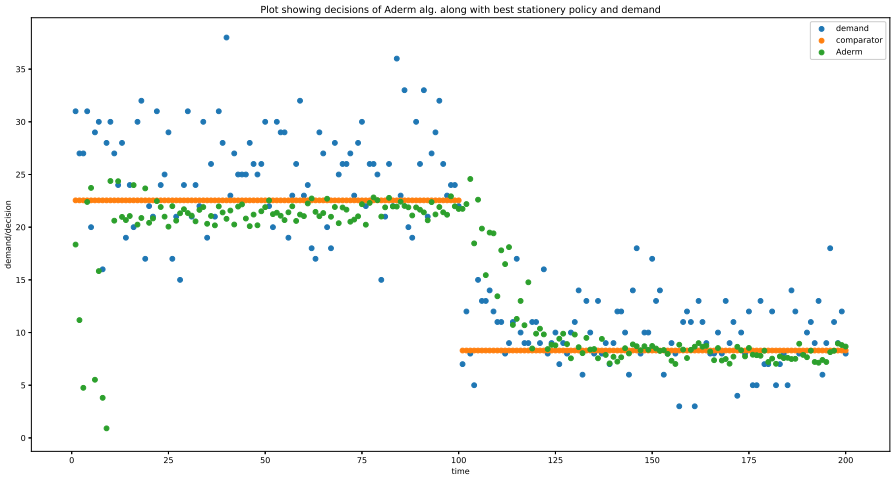


Figure 4.7: The decisions of the Aderm algorithm along with the incoming demands and the dynamic comparator sequence for a single instance of 200 steps.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this work, we had explored several frameworks used in online learning and explored the feasibility of using these frameworks in the inventory management problem, both with and without memory and to obtain sub-linear regret guarantees. These frameworks, especially the memoryless OCO setting, have been well studied in several works and hence there are several well known algorithms like the Online Gradient Descent and the Hedge Algorithm for prediction with experts with strong static regret guarantees. We had explored the application of the Hedge algorithm on the base-stock version of the inventory problem with experts, and since it satisfied the assumptions of bounded loss, the static regret of the Hedge algorithm was upper bounded by $\mathcal{O}(\sqrt{T\log(N)}$ with N base-stock experts. We, then, shifted to the OCO framework, and since the memoryless inventory problem (without base-stock control) satisfied the convexity constraints of the setting, it was an ideal candidate for applying algorithms for the OCO setting. With the norm of the sub-gradients of the memoryless loss function and the diameter of the decision set being upper bounded, running the OGD algorithm on the memoryless inventory problem resulted in a sub-linear regret upper bound of $\mathcal{O}(\sqrt{T})$. The presence of memory in the inventory problem is an indispensable factor since the current purchase decisions can have future cost implications in terms of storage or delay. Hence, we consider the inventory problem with bounded memory m, where the memory bound indicates the expiry of goods. We use the bounds derived in Anava *et al.* (2015) to conclude that the OGD algorithm run on the unary loss function (OGDM), which is derived from the loss function with memory, has static policy regret upper bounds of $\mathcal{O}(\sqrt{T})$ under certain assumptions. We verify that the inventory problem with bounded memory satisfies these assumptions and hence the OGDM has a static regret upper bound of $\mathcal{O}(\sqrt{T}$. The main contribution of Anava *et al.* (2015) was to map the OCO problem with memory to the memoryless OCO problem through the upper bound which they prove. Recognising that the environment which generates demand can be dynamic, we consider the framework of dynamic regret where the comparators need not be static and

can change with time. Zhang *et al.* (2018) proposes the Ader algorithm which has a dynamic regret upper bound of $\mathcal{O}(\sqrt{T(1+P_T)})$ for the memoryless OCO setting, where P_T is the path length of the comparator sequence. Using, the theorem of Anava *et al.* (2015) which maps OCO with memory to memoryless OCO, we adapt the Ader algorithm with minor alterations to obtain the Aderm algorithm which has dynamic policy regret upper bounds of $\mathcal{O}(\sqrt{T(1+P_T}$ matching the order of that of the Ader algorithm in the memoryless OCO setting. We tabulate the regret upper bounds under various settings.

OCO Setting / **Comparator**	**Memoryless OCO**	**OCO with Bounded Memory**
Static	OGD ($\mathcal{O}(\sqrt{T})$)	OGDM ($\mathcal{O}(\sqrt{T})$)
Dynamic	Ader ($\mathcal{O}(\sqrt{T(1+P_T)})$)	**Aderm** (**$\mathcal{O}(\sqrt{T(1+P_T)})$**)

Table 5.1: Regret upper bounds for algorithms considered under different settings of OCO

The algorithms considered along with their regret guarantees under different settings are presented in the table. The term within the brackets represents the regret upper bound of that algorithm. The algorithm in bold is the Aderm algorithm, an adaptation of the Ader algorithm of Zhang *et al.* (2018), which we formulate.

This work considers only the case of a single item in the inventory. For the multiple item case with an overall purchase capacity, it needs to be verified that the derived regret bounds hold. Since the decision set is still convex, we expect that the regret bounds only change by a constant. The case of multiple expiry duration resulting in different memory bounds for different items also needs to be considered. Another interesting case would be the adversary controlled expiry, where the expiry of the items is controlled by the adversary.

Zhang *et al.* (2018) establish a fundamental lower bound of $\Omega(\sqrt{T(1+P_T)})$ on the dynamic regret achieved in the OCO setting. As a follow-up, it would be interesting to derive such a lower bound for the dynamic policy regret for the case of OCO with bounded memory. It would be beneficial if we could adapt the improved version of Ader, proposed by Zhang *et al.* (2018) with a much less number of sub-gradient queries.

APPENDIX A

Analysis of Section 4.3.2

We prove our main theorem stated in section 4.3.2,

Theorem 6 *Set*

$$\mathcal{H} = \left\{ \eta_i = 2^{i-1} B \sqrt{\frac{7}{2(G^2 + m^2 LG)T}} \middle| i = 1, \dots N \right\} \tag{4.7}$$

where $N = \lceil \frac{1}{2} \log_2(1 + 4T/7) \rceil + 1$ *and* $\beta = \sqrt{8/(9(m^2LB + d)^2 T)}$ *in Algorithm 4 under Assumptions 1,2,3 and 4 for any comparator sequence* $\mathbf{z}_{m+1}, \mathbf{z}_{m+2} \dots \mathbf{z}_T \in \mathcal{Y}$ *with path length* $P_T = \sum_{t=m+1}^{T} ||\mathbf{z}_t - \mathbf{z}_{t-1}||_2$*, the proposed Aderm algorithm in the OCO setting with m-bounded memory satisfies,*

$$\begin{aligned} R_T^D(\mathbf{z}_T, \dots \mathbf{z}_{m+1}) \leq & \frac{3\sqrt{2m^2LG + G^2}}{4} \sqrt{2T(7B^2 + 4BP_T)} \\ & + \frac{3(m^2LB + d)\sqrt{2T}}{4} (1 + 2\log(k+1)) \end{aligned} \tag{4.8}$$

for $k = \lfloor \frac{1}{2} \log_2(1 + \frac{4P_T}{7B}) \rfloor + 1$.

Proof: From Eq. 4.4 and 4.6,

$$\begin{aligned} R_T^D(\mathbf{z}_T, \dots \mathbf{z}_{m+1}) &\leq m^2 L \sum_{t=m+2}^{T} ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \sum_{t=m=1}^{T} \tilde{f}_t(\mathbf{y}_t) - \sum_{t=m+1}^{T} \tilde{f}_t(\mathbf{z}_t) \\ &\leq \sum_{t=m=1}^{T} R_{t,i}^M + \sum_{t=m=1}^{T} R_{t,i}^E \end{aligned} \tag{A.1}$$

where,

$$R_{t,i}^M = \left(m^2 L ||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 + \tilde{f}_t(\mathbf{y}_t) - m^2 L ||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 - \tilde{f}_t(\mathbf{y}_{t.i}) \right) \tag{A.2}$$

$$R_{t,i}^E = \left(m^2 L ||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 + \tilde{f}_t(\mathbf{y}_{t.i}) - \tilde{f}_t(\mathbf{z}_t) \right) \tag{A.3}$$

for some $i \in [N]$. The term $R^M_{t,i}$ is the single step static regret suffered by the meta-algorithm with respect to expert E_i, while the term $R^E_{t,i}$ is the single-step dynamic memoryless regret suffered by the expert-algorithm i along with the switching cost. We will now consider each of these terms separately.

$$
\begin{aligned}
||\mathbf{y}_t - \mathbf{y}_{t-1}||_2 &= \Big|\Big| \sum_{j=1}^{N} w_{t,i}\mathbf{y}_{t,j} - \sum_{j=1}^{N} w_{t-1,j}\mathbf{y}_{t-1,j} \Big|\Big|_2 \\
&= \Big|\Big| \big(\sum_{j=1}^{N} w_{t,j}\mathbf{y}_{t,j} - \sum_{j=1}^{N} w_{t,j}\mathbf{y}_{t-1,j}\big) + \big(\sum_{j=1}^{N} w_{t,j}\mathbf{y}_{t-1,j} - \sum_{i=1}^{N} w_{t-1,j}\mathbf{y}_{t-1,j}\big) \Big|\Big|_2 \\
&\leq \Big|\Big| \sum_{j=1}^{N} w_{t,j}\mathbf{y}_{t,j} - \sum_{j=1}^{N} w_{t,j}\mathbf{y}_{t-1,j} \Big|\Big|_2 + \Big|\Big| \sum_{j=1}^{N} w_{t,j}\mathbf{y}_{t-1,j} - \sum_{i=1}^{N} w_{t-1,j}\mathbf{y}_{t-1,j} \Big|\Big|_2 \\
&\leq \sum_{j=1}^{N} w_{t,j}||\mathbf{y}_{t,j} - \mathbf{y}_{t-1,j}||_2 + B\sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}|
\end{aligned}
$$

Putting this is equation A.2, we get,

$$
\begin{aligned}
R^M_{t,i} \leq & m^2 L \sum_{j=1}^{N} w_{t,j}||\mathbf{y}_{t,j} - \mathbf{y}_{t-1,j}||_2 + m^2 LB \sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}| \\
& + \tilde{f}_t(\mathbf{y}_t) - (m^2 L||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 + \tilde{f}_t(\mathbf{y}_{t.i})) \\
= & m^2 LB \sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}| \\
& + m^2 L \sum_{j=1}^{N} w_{t,j}||\mathbf{y}_{t,j} - \mathbf{y}_{t-1,j}||_2 + \tilde{f}_t(\mathbf{y}_t) - (m^2 L||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 + \tilde{f}_t(\mathbf{y}_{t.i})) \\
\leq & m^2 LB \sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}| + m^2 L \sum_{j=1}^{N} w_{t,j}||\mathbf{y}_{t,j} - \mathbf{y}_{t-1,j}||_2 \\
& + \sum_{j=1}^{N} w_{t,j}\tilde{f}_t(\mathbf{y}_{t,j}) - (m^2 L||\mathbf{y}_{t,i} - \mathbf{y}_{t-1,i}||_2 + \tilde{f}_t(\mathbf{y}_{t.i})) \\
= & m^2 LB \sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}| + \sum_{j=1}^{N} w_{t,j} l_t(j) - l_t(i)
\end{aligned}
$$

where $l_t(j) = m^2 L||\mathbf{y}_{t,j} - \mathbf{y}_{t-1,j}||_2 + \tilde{f}_t(\mathbf{y}_{t,j})$

$$
\therefore \sum_{t=m+1}^{T} R^M_{t,i} \leq m^2 LB \sum_{t=m+1}^{T} \sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}| + \sum_{t=m+1}^{T} \big(\sum_{j=1}^{N} w_{t,j} l_t(j) - l_t(i)\big) \quad \text{(A.4)}
$$

The second term in the above equation is the regret of the exponentially weighted

average algorithm run with parameter β in the experts setting with loss $l_t(.)$. Since, $a \geq l_t(i) \leq a + m^2LB + d$. Zhang *et al.* (2018) present the following lemma,

Lemma 5 *If, $a \leq l_t(i) \leq a + r$ is the individual expert loss under the prediction with N experts setting, then for the exponentially weighted average forecaster,*

$$\sum_{t=m+1}^{T} \big(\sum_{j=1}^{N} w_{t,j} l_t(j) - l_t(i) \big) \leq \frac{\log(1/w_{1,i})}{\beta} + \frac{\beta T r^2}{8}$$

Using lemma 5, we establish the following regret bound,

$$\sum_{t=m+1}^{T} \big(\sum_{j=1}^{N} w_{t,j} l_t(j) - l_t(i) \big) \leq \frac{\log(1/w_{1,i})}{\beta} + \frac{\beta T (m^2LB + d)^2}{8} \tag{A.5}$$

Without loss of generality, we can subtract a from the losses $l_t(i)$ for all $i \in [N]$, and $t \in [T]$ and define $0 \leq l'_t(i) = l_t(i) - a \leq m^2LB + c$. It is trivial to verify that the obtained algorithm 4, i.e. the weights of the experts, would not change. Essentially,

$$w_{t,i} = \frac{\exp(-\beta(\sum_{k=1}^{t} l_k(i)))}{\sum_{j=1}^{N} \exp(-\beta(\sum_{k=1}^{t} l_k(j)))} = \frac{\exp(-\beta(\sum_{k=1}^{t} l'_k(i)))}{\sum_{j=1}^{N} \exp(-\beta(\sum_{k=1}^{t} l'_k(j)))}$$

With this modified loss function resulting in the same weights for the exponential weighting algorithm, Shalev-Shwartz (2012) show that we can bound the first term in the R.H.S. of Eq. A.4 by,

$$\sum_{j=1}^{N} |w_{t,j} - w_{t-1,j}| \leq \beta ||l'_t(j)||_\infty = \beta(m^2LB + d) \tag{A.6}$$

Combining Eq. A.4, A.5 and A.6, we obtain,

$$\begin{aligned} \sum_{t=m+1}^{T} R_{t,i}^{M} &\leq \frac{\log(1/w_{t,i})}{\beta} + \frac{\beta T (8m^2LB(m^2LB + d) + (m^2LB + d)^2)}{8} \\ &\leq \frac{\log(1/w_{t,i})}{\beta} + \frac{9\beta T (m^2LB + d)^2}{8} \end{aligned}$$

Choosing $\beta = \frac{1}{3(m^2LB+d)}\sqrt{\frac{8}{T}}$, we obtain,

$$\sum_{t=m+1}^{T} R_{t,i}^{M} \leq \frac{3(m^2LB+d)\sqrt{2T}}{4}\Big(1+\log\frac{1}{w_{t,i}}\Big) \tag{A.7}$$

Now, we consider the second term of Eq. A.1,

$$R_{t,i}^{E} = \Big(m^2L||\mathbf{y}_{t,i}-\mathbf{y}_{t-1,i}||_2 + \tilde{f}_t(\mathbf{y}_{t.i}) - \tilde{f}_t(\mathbf{z}_t)\Big)$$

Since expert E_i performs OGD with learning rate η_i on the unary loss functions $\tilde{f}_t$ for all $t \in [T]$, using Theorem 5,

$$\sum_{t=m+1}^{T} R_{t,i}^{E} \leq \frac{1}{4\eta_i}(7B^2+4BP_T) + \frac{\eta_i(2m^2LG+G^2)T}{2} \quad \forall i \in [N] \tag{A.8}$$

The optimal learning rate of the upper bound of Theorem 5 is,

$$\eta^* = \sqrt{\frac{7B^2+4BP_T}{2(2m^2LG+G^2)T}} \tag{A.9}$$

We need to show that there exists an $\eta_i \in \mathcal{H}$ corresponding to expert E_i such that the R.H.S of Eq. A.8 is almost minimal. Since the diameter of the decision set $\mathcal{Y}$ is bounded by B,

$$0 \leq P_t = \sum_{t=m+2}^{T} ||\mathbf{z}_t - \mathbf{z}_{t-1}||_2 \leq TB$$

Hence,

$$\frac{B}{\sqrt{(2m^2LG+G^2)}}\sqrt{\frac{7}{2T}} \leq \eta^* \leq \frac{B}{\sqrt{(2m^2LG+G^2)}}\sqrt{\frac{7}{2T}+2}$$

It is simple to verify that,

$$\min\mathcal{H} = \frac{B}{\sqrt{(2m^2LG+G^2)}}\sqrt{\frac{7}{2T}} \text{ and } \max\mathcal{H} \geq \frac{B}{\sqrt{(2m^2LG+G^2)}}\sqrt{\frac{7}{2T}+2}$$

, Therefore, for any possible value of P_T, there exists a step size $\eta_k \in \mathcal{H}$, such that

$$\eta_k = \frac{2^{i-1}B}{\sqrt{(2m^2LG+G^2)}}\sqrt{\frac{7}{2T}} \leq \eta^* \leq 2\eta_k \tag{A.10}$$

where $k = \lfloor \frac{1}{2}\log_2(1+\frac{4P_T}{7B})\rfloor + 1$. Setting $i = k$ and plugging η_k into Eq. A.8, and

using Eqs. A.9 and A.10

$$\begin{aligned}\sum_{t=m+1}^{T} R_{t,k}^{E} &\leq \frac{1}{4\eta_k}(7B^2 + 4BP_T) + \frac{\eta_k(2m^2LG + G^2)T}{2} \\ &\leq \frac{1}{2\eta^*}(7B^2 + 4BP_T) + \frac{\eta^*(2m^2LG + G^2)T}{2} \\ &= \frac{3\sqrt{2m^2LG + G^2}}{4}\sqrt{2T(7B^2 + 4BP_T)}\end{aligned} \tag{A.11}$$

Also, the initial weight of expert E_k, for $W = 1 + \frac{1}{N}$, is

$$w_{1,k} = \frac{W}{k(k+1)} \geq \frac{1}{k(k+1)} \geq \frac{1}{(i+1)^2}$$

Again setting $i = k$ and plugging the above relation into Eq. A.7, we get,

$$\sum_{t=m+1}^{T} R_{t,k}^{M} \leq \frac{3(m^2LB + d)\sqrt{2T}}{4}(1 + 2\log(k+1)) \tag{A.12}$$

Finally, setting $i = k$ in Eq. A.1 and using Eqs. A.11 and A.12,

$$\begin{aligned}R_T^D(\mathbf{z}_T, \dots \mathbf{z}_{m+!}) \leq &\frac{3\sqrt{2m^2LG + G^2}}{4}\sqrt{2T(7B^2 + 4BP_T)} \\ &+ \frac{3(m^2LB + d)\sqrt{2T}}{4}(1 + 2\log(k+1))\end{aligned} \tag{A.13}$$

for $k = \lfloor \frac{1}{2}\log_2(1 + \frac{4P_T}{7B}) \rfloor + 1$.

The above bound holds for any sequence of comparators $\mathbf{z}_T, \mathbf{z}_{T-1} \dots \mathbf{z}_{m+1} \in \mathcal{Y}$.

REFERENCES

1. **Anava, O.**, **E. Hazan**, and **S. Mannor**, Online learning for adversaries with memory: Price of past mistakes. *In* **C. Cortes**, **N. Lawrence**, **D. Lee**, **M. Sugiyama**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/38913e1d6a7b94cb0f55994f679f5956-Paper.pdf.

2. **Bertsekas, D. P.**, *Dynamic Programming and Optimal Control*. Athena Scientific, 2000, 2nd edition. ISBN 1886529094.

3. **Bhattacharjee, R.**, **S. Banerjee**, and **A. Sinha** (2020). Fundamental limits on the regret of online network-caching. *Proc. ACM Meas. Anal. Comput. Syst.*, **4**(2). URL https://doi.org/10.1145/3392143.

4. **Bousquet, O.** and **M. K. Warmuth** (2003). Tracking a small set of experts by mixing past posteriors. *J. Mach. Learn. Res.*, **3**(null), 363–396. ISSN 1532-4435.

5. **Boyd, S.** and **L. Vandenberghe**, *Convex optimization*. Cambridge university press, 2004.

6. **Cesa-Bianchi, N.** and **G. Lugosi**, *Prediction, learning, and games*. Cambridge University Press, 2006. ISBN 978-0-521-84108-5. URL https://doi.org/10.1017/CBO9780511546921.

7. **Freund, Y.** and **R. E. Schapire** (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139. ISSN 0022-0000. URL https://www.sciencedirect.com/science/article/pii/S002200009791504X.

8. **Geulen, S.**, **B. Vöcking**, and **M. Winkler**, Regret minimization for online buffering problems using the weighted majority algorithm. *In* **A. T. Kalai** and **M. Mohri** (eds.), *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*. Omnipress, 2010. URL http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf#page=140.

9. **György, A.** and **G. Neu**, Near-optimal rates for limited-delay universal lossy source coding. *In 2011 IEEE International Symposium on Information Theory Proceedings*. 2011.

10. **Hall, E. C.** and **R. M. Willett**, Dynamical models and tracking regret in online convex programming. *In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13. JMLR.org, 2013.

11. **Jadbabaie, A.**, **A. Rakhlin**, **S. Shahrampour**, and **K. Sridharan**, Online Optimization : Competing with Dynamic Comparators. *In* **G. Lebanon** and **S. V. N. Vishwanathan** (eds.), *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*. PMLR, San Diego, California, USA, 2015. URL http://proceedings.mlr.press/v38/jadbabaie15.html.

12. **Merhav, N.**, **E. Ordentlich**, **G. Seroussi**, and **M. Weinberger** (2002). On sequential strategies for loss functions with memory. *IEEE Transactions on Information Theory*, **48**(7), 1947–1958.

13. **Mokhtari, A.**, **S. Shahrampour**, **A. Jadbabaie**, and **A. Ribeiro**, Online optimization in dynamic environments: Improved regret rates for strongly convex problems. *In 2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE Press, 2016. URL https://doi.org/10.1109/CDC.2016.7799379.

14. **Rooij, S.** and **T. Erven**, Learning the switching rate by discretising bernoulli sources online. *In* **D. van Dyk** and **M. Welling** (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009. URL http://proceedings.mlr.press/v5/rooij09a.html.

15. **Shalev-Shwartz, S.** (2012). Online learning and online convex optimization. *Found. Trends Mach. Learn.*, **4**(2), 107–194. ISSN 1935-8237. URL https://doi.org/10.1561/2200000018.

16. **Shi, G.**, **Y. Lin**, **S.-J. Chung**, **Y. Yue**, and **A. Wierman**, Online optimization with memory and competitive control. *In* **H. Larochelle**, **M. Ranzato**, **R. Hadsell**, **M. F. Balcan**, and **H. Lin** (eds.), *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/ed46558a56a4a26b96a68738a0d28273-Paper.pdf.

17. **Yang, T.**, **L. Zhang**, **R. Jin**, and **J. Yi**, Tracking slowly moving clairvoyant: Optimal dynamic regret of online learning with true and noisy gradient. *In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16. JMLR.org, 2016.

18. **Zhang, L.**, **S. Lu**, and **Z.-H. Zhou**, Adaptive online learning in dynamic environments. *In* **S. Bengio**, **H. Wallach**, **H. Larochelle**, **K. Grauman**, **N. Cesa-Bianchi**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/10a5ab2db37feedfdeaab192ead4ac0e-Paper.pdf.

19. **Zhang, L.**, **T. Yangt**, **J. Yi**, **R. Jin**, and **Z.-H. Zhou**, Improved dynamic regret for non-degenerate functions. *In Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17. Curran Associates Inc., Red Hook, NY, USA, 2017. ISBN 9781510860964.

20. **Zhao, P.**, **Y.-J. Zhang**, **L. Zhang**, and **Z.-H. Zhou**, Dynamic regret of convex and smooth functions. *In* **H. Larochelle**, **M. Ranzato**, **R. Hadsell**, **M. F. Balcan**, and **H. Lin** (eds.), *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/939314105ce8701e67489642ef4d49e8-Paper.pdf.

21. **Zinkevich, M.**, Online convex programming and generalized infinitesimal gradient ascent. *In Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03. AAAI Press, 2003. ISBN 1577351894.