

國立臺灣大學電機資訊學院資訊工程學系

博士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral Dissertation

邁向實際的線上學習

Toward Realistic Online Learning

蔣兆凱

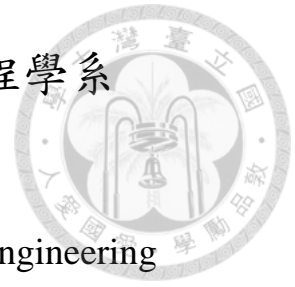
Chao-Kai Chiang

指導教授：林軒田 博士

Advisor: Hsuan-Tien Lin, Ph.D.

中華民國 103 年 1 月

January 2014





國立臺灣大學博士學位論文  
口試委員會審定書

邁向實際的線上學習

Toward Realistic Online Learning

本論文係蔣兆凱君（學號D97922013）在國立臺灣大學資訊工程學系完成之博士學位論文，於民國 102 年 12 月 10 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

林軒田

（指導教授）

林子德

顏海司鈞

林智仁

呂育道

昆及人

新永英

系主任

## 謝誌



這篇論文紀錄了我過去幾年來在 online learning 這個領域的研究成果。在完成論文的過程中，我接受了許多老師的教導，朋友的鼓勵，以及家人的扶持。在這裡藉此短文表達我對眾人的深恩的謝意。

首先，我要感謝我的指導老師，林軒田教授和呂及人博士。感謝你們以豐富的學養與知識，帶領我進入 learning theory 的美麗世界。感謝你們在我受挫、徬徨與怠惰的時候給予我激勵與方向。感謝你們親身力行，在學生面前展現最好的身教。你們專心致力於學術研究，你們拓展知識疆界的熱忱與追求真理的渴望，深深刻畫在我的研究基因之上。

我還要感謝我的口試委員，呂育道教授、林守德教授、林智仁教授及顏嗣鈞教授。感謝各位老師在我研究計畫上的建議，協助形成論文的雛型及研究問題。各位老師的建議，也讓我能夠從更開闊的角度及不同的方向檢視我的研究，讓我對這個研究領域有更深廣的認識。

另外我也要感謝許多在中研院和台大資訊系的老師、同事及朋友。因為有各位亦師亦友的陪伴及研討，讓我的研究生生活豐富且快樂。

我的感謝比起家人對我的付出是微不足道的。我要感謝我的太太黃雅惠，謝謝她一直以來對我的支持與照顧。她的照顧讓我能夠無後顧之憂地從事研究工作。我要感謝我的弟弟蔣震弘，謝謝他在生活上許多有智慧及有趣的討論和分享。身為我生命中學習歷程的頭兩位老師，我要感謝我的父母，蔣猷良先生和彭逸娟女士。沒有兩位從小至今的教導與照顧，我不可能有如今的成長。

末了，僅將此論文獻給我人生中所有的老師，包含過去曾經，現在正在，還有未來將會指導我的所有老師。感恩，感謝。

## 中文摘要



本文研究線上決策問題，其問題設定如下。玩家需要一個環境中重複決定要採取什麼行動，並得知該行動所帶來的影響的訊息。玩家希望有一個線上演算法，可以從過往的歷史中學習，並且在往後能夠作出更好的決策。我們使用後悔度來衡量一個演算法的優劣。

近來的研究著重於線上最佳化(傳統最佳化問題的變形)以及專家問題(有限數目的專家可供選擇)，因為這些問題模型可以用來模擬許多現實生活中的挑戰。本文將報告我們在這個方向上的貢獻。

首先，我們研究逐漸改變的環境。我們定義一個新測度稱為偏離度來衡量環境的變化程度。在這個新的問題模型中，我們藉由修改一個廣受研究的 FTRL 演算法來設計我們的演算法，並證明其表現能以偏離度的函數表示。因此我們的演算法在變化和緩的環境中，能夠表現得比以往的演算法更好。

接著，我們研究逐漸改變的環境，並假設玩家所能獲得的只有部分的訊息。我們希望得知在什麼樣的回饋訊息下，依舊能夠表現得和得知完整訊息的玩家一樣好。我們設計了一個新穎的抽樣機制，並且使用取得的訊息估計未知的梯度向量。我們的研究成果顯示，每回合僅需知道損益函數的兩個點的值，我們演算法的表現便能與得到所有訊息的玩家一樣好。

在第三部分中，我們允許玩家可以主動探詢他想要的部分訊息。我們提出一個新的問題模型，讓玩家可以在某個上限內探詢他所需的訊息。這個問題模型推廣以往不能允許主動探詢的問題。在這個新的問題模型下，我們設計幾乎完美的演算法並且將其表現以玩家的探詢上限的函數表示。

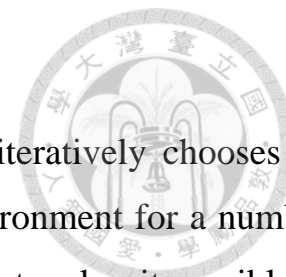
最後在第四部份，我們研究含有文意的最少訊息問題。與傳統最小訊息問題的差別在於，玩家在選擇策略之前可以得知額外的訊息。由於擁有所有訊息可以讓演算法有更好的表現，所以我們定義一種新訊息稱為虛擬獲益，並使用虛擬獲益來估計未知的訊息，協助演算法學習。我們設計並分析一個名為

LINPRUCB 的新演算法，它是目前已知最好的 LINUCB 演算法的衍伸。



關鍵字：線上學習、後悔度、查詢、偏離度、最少訊息、含文意的最少訊息

## Abstract

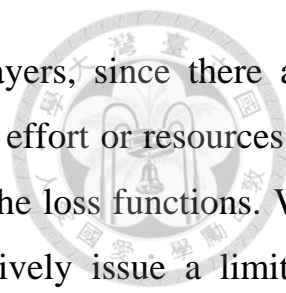


We study the *online decision problem* in which a player iteratively chooses an action and then receives certain loss information from the environment for a number of rounds. The player would like to have an online algorithm that makes it possible to learn from past experiences, make better decisions as time goes by, and achieve small regret which is defined as the difference between the total loss of the algorithm and that of the best fixed action.

Recently, a class of studies consider variants of the *Online Convex Optimization* (OCO, which is a variant of the classical convex optimization problem) and the *Prediction with Expert Advice* (PEA, which models an online decision problem in which the set of actions consists of finite number of actions) problems that can be used to model more realistic challenges. In this thesis, we will present our contributions in this direction.

In the first part, we study instances of gradually changing environments in our daily life. We define a new notion that is referred to as the *deviation* that measures the total difference between consecutive loss functions in order to describe a gradually changing environment. We show that a modification of the well understood FOLLOW THE REGULARIZED LEADER algorithm leads to regret in terms of deviation, thereby implying a small regret for environments with small deviations.

In the second part, we ask the same question in a setting where there is *partial* information. Our goal is to determine how much information is needed, in situations where the deviation constraint is in effect, in order to obtain regret bounds that are close to the regret bounds that were obtained in our previous study [28]. A novel sampling scheme is designed in order to estimate the unknown gradient information that is needed in order to apply the algorithms that are introduced in [28]. We construct two-point bandit algorithms that are able to achieve regret bounds that are close to the regret bounds from our previous study [28] that were based on the full information setting.



In the third part, we consider the possibility of *active* players, since there are scenarios where it seems possible for the player to spend some effort or resources to actively collect some intentionally selected information about the loss functions. We describe a new scenario where the player is allowed to actively issue a limited number of queries in order to obtain the loss information she needs in each round before making a decision. This scenario generalizes the previous problem models in which no query is allowed to be issued before a decision is made. We design an algorithm that achieves the regret as a function of the number of bits that can be queried in one round and provide lower bounds showing that the upper bound in general cannot be improved.

In the last part, we study the contextual bandit problem, which is different from the traditional bandit problem, the learner can access additional information about the environment (i.e., context) before making selections. Motivated by the better regret bounds that are available in settings with full information, we create a new notion called *pseudo-reward* that can be employed for making guesses about unseen rewards and develop a forgetting mechanism for handling fallacious rewards that were computed in the early rounds. Combining the pseudo-rewards and the forgetting mechanism, we propose and analyze a new algorithm called LINPRUCB that is an extension of a state-of-the-art algorithm called LINUCB.

Keywords: Online Learning, Regret, Query, Deviation, Bandit, Contextual Bandit

# 目錄



口試委員會審定書 .....	i
謝誌 .....	ii
中文摘要 .....	iii
英文摘要 .....	v
第一章 Introduction .....	1
1.1 Problem Models and Previous Works .....	1
1.2 Main Contributions .....	3
1.3 Thesis Structure .....	5
第二章 The Online Decision Problem .....	7
2.1 Relations Between Other Areas .....	8
2.2 Justification of Regret .....	8
2.3 Instances of the Online Decision Problem and Two Basic Algorithms ...	9
2.4 Recent Progress .....	12
第三章 Online Learning in Gradually Evolving Worlds .....	21
3.1 Regret Bounded by Variation .....	21
3.2 Online Learning in Gradually Evolving Worlds .....	22
3.3 Preliminaries .....	24
3.4 Meta Algorithm .....	25
3.5 Linear Loss Functions .....	28
3.6 General Convex Loss Functions .....	32
3.7 Strictly Convex Loss Functions .....	33
第四章 Beating Bandits in Gradually Evolving Worlds .....	39
4.1 Previous Works .....	39
4.2 Beating Bandits in Gradually Evolving Worlds .....	40
4.3 Meta Algorithm .....	43





4.4 Linear Loss Functions .....	47
4.5 Convex Loss Functions .....	49
4.6 Strongly Convex Loss Functions .....	52
第五章 Online Learning with Queries .....	55
5.1 Preliminaries .....	57
5.2 A Special Case .....	59
5.3 The Main Result .....	60
5.4 Lower Bounds .....	67
第六章 Pseudo-Reward for Linear Contextual Bandits .....	73
6.1 Introduction .....	73
6.2 Preliminaries .....	76
6.3 Linear Pseudo-Reward Upper Confidence Bound Algorithm .....	77
附錄 A .....	85
A.1 Proof of Lemma 3.1 in Section 3.4 .....	85
A.2 Proof of Lemma 3.2 in Section 3.4 .....	86
A.3 Proof of Lemma 3.8 in Section 3.7 .....	87
附錄 B .....	89
B.1 Proof of Lemma 4.3 in Section 4.3 .....	89
B.2 Proof of Lemma 4.8 in Section 4.5 .....	90
B.3 Proof of Lemma 4.10 in Section 4.6 .....	91
B.4 Proof of Lemma 4.11 in Section 4.6 .....	92
附錄 C .....	93
C.1 Proof of Lemma 5.1 in Section 5.1 .....	93
參考文獻 .....	97

表目錄

2.1 Instances of the online decision problem .....	10
2.2 Studies related to the OCO problem .....	13
2.3 Studies related to the PEA and the MAB problems .....	17
2.4 Online combinatorial problems .....	18





# Toward Realistic Online Learning

Chao-Kai Chiang

December 2013

Department of Computer Science and Information Engineering  
National Taiwan University  
Taipei 10617, Taiwan

**Thesis Committee:**

Dr. Chih-Jen Lin  
Dr. Hsuan-Tien Lin  
Dr. Shou-De Lin  
Dr. Chi-Jen Lu  
Dr. Yuh-Dauh Lyuu  
Dr. Hsu-Chun Yen

*A dissertation submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy.*



**Keywords:** Online Learning, Regret, Query, Deviation, Bandit, Contextual Bandit



*To my teachers;  
the past, the present, and the future.*



## Abstract

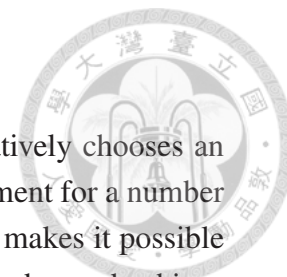
We study the *online decision problem* in which a player iteratively chooses an action and then receives certain loss information from the environment for a number of rounds. The player would like to have an online algorithm that makes it possible to learn from past experiences, make better decisions as time goes by, and achieve small regret which is defined as the difference between the total loss of the algorithm and that of the best fixed action.

Recently, a class of studies consider variants of the *Online Convex Optimization* (OCO, which is a variant of the classical convex optimization problem) and the *Prediction with Expert Advice* (PEA, which models an online decision problem in which the set of actions consists of finite number of actions) problems that can be used to model more realistic challenges. In this thesis, we will present our contributions in this direction.

In the first part, we study instances of gradually changing environments in our daily life. We define a new notion that is referred to as the *deviation* that measures the total difference between consecutive loss functions in order to describe a gradually changing environment. We show that a modification of the well understood FOLLOW THE REGULARIZED LEADER algorithm leads to regret in terms of deviation, thereby implying a small regret for environments with small deviations.

In the second part, we ask the same question in a setting where there is *partial* information. Our goal is to determine how much information is needed, in situations where the deviation constraint is in effect, in order to obtain regret bounds that are close to the regret bounds that were obtained in our previous study [28]. A novel sampling scheme is designed in order to estimate the unknown gradient information that is needed in order to apply the algorithms that are introduced in [28]. We construct two-point bandit algorithms that are able to achieve regret bounds that are close to the regret bounds from our previous study [28] that were based on the full information setting.

In the third part, we consider the possibility of *active* players, since there are scenarios where it seems possible for the player to spend some effort or resources to actively collect some intentionally selected information about the loss functions. We describe a new scenario where the player is allowed to actively issue a limited number of queries in order to obtain the loss information she needs in each round before making a decision. This scenario generalizes the previous problem models in which no query is allowed to be issued before a decision is made. We design an algorithm that achieves the regret as a function of the number of bits that can



be queried in one round and provide lower bounds showing that the upper bound in general cannot be improved.

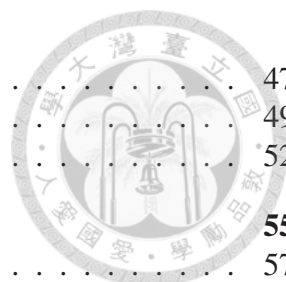
In the last part, we study the contextual bandit problem, which is different from the traditional bandit problem, the learner can access additional information about the environment (i.e., context) before making selections. Motivated by the better regret bounds that are available in settings with full information, we create a new notion called *pseudo-reward* that can be employed for making guesses about unseen rewards and develop a forgetting mechanism for handling fallacious rewards that were computed in the early rounds. Combining the pseudo-rewards and the forgetting mechanism, we propose and analyze a new algorithm called LINPRUCB that is an extension of a state-of-the-art algorithm called LINUCB.





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Models and Previous Works . . . . .	1
1.2	Main Contributions . . . . .	3
1.2.1	Online Learning in Gradually Evolving Worlds . . . . .	3
1.2.2	Beating Bandits in Gradually Evolving Worlds . . . . .	3
1.2.3	Online Learning with Queries . . . . .	4
1.2.4	Pseudo-Reward for Linear Contextual Bandits . . . . .	4
1.3	Thesis Structure . . . . .	5
<b>2</b>	<b>The Online Decision Problem</b>	<b>7</b>
2.1	Relations Between Other Areas . . . . .	8
2.2	Justification of Regret . . . . .	8
2.3	Instances of the Online Decision Problem and Two Basic Algorithms . . . . .	9
2.3.1	The GRADIENT DESCENT Algorithm . . . . .	10
2.3.2	The MULTIPLICATIVE WEIGHTS UPDATE Algorithm . . . . .	11
2.4	Recent Progress . . . . .	12
2.4.1	Online Convex Optimization . . . . .	12
2.4.2	Prediction with Expert Advice and Multi-Armed Bandit . . . . .	16
<b>3</b>	<b>Online Learning in Gradually Evolving Worlds</b>	<b>21</b>
3.1	Regret Bounded by Variation . . . . .	21
3.2	Online Learning in Gradually Evolving Worlds . . . . .	22
3.3	Preliminaries . . . . .	24
3.4	Meta Algorithm . . . . .	25
3.5	Linear Loss Functions . . . . .	28
3.5.1	Online Linear Optimization Problem . . . . .	28
3.5.2	Prediction with Expert Advice . . . . .	30
3.6	General Convex Loss Functions . . . . .	32
3.7	Strictly Convex Loss Functions . . . . .	33
<b>4</b>	<b>Beating Bandits in Gradually Evolving Worlds</b>	<b>39</b>
4.1	Previous Works . . . . .	39
4.2	Beating Bandits in Gradually Evolving Worlds . . . . .	40
4.3	Meta Algorithm . . . . .	43



4.4	Linear Loss Functions . . . . .	47
4.5	Convex Loss Functions . . . . .	49
4.6	Strongly Convex Loss Functions . . . . .	52
<b>5</b>	<b>Online Learning with Queries</b>	<b>55</b>
5.1	Preliminaries . . . . .	57
5.2	A Special Case . . . . .	59
5.3	The Main Result . . . . .	60
5.3.1	The Algorithm $ALG_2$ . . . . .	61
5.3.2	Proof of Theorem 5.2 . . . . .	63
5.3.3	Proof of Lemma 5.2 . . . . .	64
5.3.4	Proof of Claim 5.1 . . . . .	65
5.4	Lower Bounds . . . . .	67
5.4.1	Proof of Lemma 5.3 . . . . .	70
<b>6</b>	<b>Pseudo-Reward for Linear Contextual Bandits</b>	<b>73</b>
6.1	Introduction . . . . .	73
6.1.1	Previous Works . . . . .	74
6.1.2	Our Approach . . . . .	75
6.2	Preliminaries . . . . .	76
6.2.1	The LINUCB Algorithm . . . . .	76
6.3	Linear Pseudo-Reward Upper Confidence Bound Algorithm . . . . .	77
6.3.1	The Pseudo-Reward Scheme . . . . .	77
6.3.2	Handling Biased Terms . . . . .	78
6.3.3	The LINPRUCB Algorithm . . . . .	78
6.3.4	The SupLinPRUCB Algorithm . . . . .	79
<b>A</b>	<b>Deferred Proofs in Chapter 3</b>	<b>85</b>
A.1	Proof of Lemma 3.1 in Section 3.4 . . . . .	85
A.2	Proof of Lemma 3.2 in Section 3.4 . . . . .	86
A.3	Proof of Lemma 3.8 in Section 3.7 . . . . .	87
<b>B</b>	<b>Deferred Proofs in Chapter 4</b>	<b>89</b>
B.1	Proof of Lemma 4.3 in Section 4.3 . . . . .	89
B.2	Proof of Lemma 4.8 in Section 4.5 . . . . .	90
B.3	Proof of Lemma 4.10 in Section 4.6 . . . . .	91
B.4	Proof of Lemma 4.11 in Section 4.6 . . . . .	92
<b>C</b>	<b>Deferred Proofs in Chapter 5</b>	<b>93</b>
C.1	Proof of Lemma 5.1 in Section 5.1 . . . . .	93
	<b>Bibliography</b>	<b>97</b>



# List of Tables

2.1	Instances of the online decision problem. . . . .	10
2.2	Studies related to the OCO problem. . . . .	13
2.3	Studies related to the PEA and the MAB problems. . . . .	17
2.4	Online combinatorial problems. . . . .	18





# Chapter 1

## Introduction

Many situations in daily life involve repeated decisions in unknown and changing environments. Examples include trading stocks, commuting to work, routing in a network, forecasting the weather, playing games, etc. These are the types of problems that are studied in the Online Learning community and that motivate the study of the *online decision problem* [21] in which a player iteratively chooses an action and then receives certain loss information from the environment for a number of rounds. The player would like to have an online algorithm that makes it possible to learn from past experiences, make better decisions as time goes by, and keep the total accumulated loss small <sup>1</sup>. The standard notion for evaluating online algorithms is called *regret*, which is defined as the difference between the total loss of the algorithm and that of the best fixed action. The goal of the player is to find an algorithm that achieves the minimum level of regret.

### 1.1 Problem Models and Previous Works

There are several ways to specify an online decision problem. In the beginning, studies considered the most general cases in order to build up the fundamental understanding and to discover important properties of the online decision problem. The two most basic and important problems that have been studied extensively are the *Online Convex Optimization* (OCO) problem [68] and the *Prediction with Expert Advice* (PEA) problem [6]. In OCO, the set of actions is a convex set with a bounded diameter, the loss of each action is defined by a convex function with a bounded gradient, and the feedback information that the player receives is the entire loss function. In

<sup>1</sup>One can also consider the feedback information as reward, and the goal is to maximize the total reward. Moreover, as explained in Remark 2.1, in this thesis we only consider loss information rather than reward. The results in this thesis can be extended easily to scenarios where feedback is considered as reward information.

PEA, the number of actions is finite, the loss function is an arbitrary function with bounded function values that can be represented by a loss vector with each entry being the loss from the corresponding action, and the feedback information is the entire loss vector.

Once a fundamental understanding of the OCO and the PEA problems was established, studies started considering variants that could be used to model more realistic challenges. For instance, different types of problems can be modeled by different classes of loss functions. In different environments, the player may receive different kinds of information. Further, there maybe specific patterns among loss functions that can be utilized to improve the algorithms. Moreover, there can be other variants of OCO and PEA that are derived to model other natural scenarios. Thus we list the following four approaches that are related to this thesis. In the first approach, [46] and [42] studied specific classes of loss functions and proposed new algorithms for achieving better regret bounds.

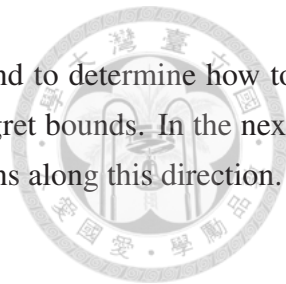
In the second approach, researchers considered specific patterns in the sequences of the loss functions where the observed loss values for each action were close to the corresponding mean. These approaches try to describe the properties that obtain between the loss functions in order to model more natural and realistic problems where the loss functions are generated by benign systems (or natural environments) rather than by adversaries (studied in worse case analysis). The resulting property is subsequently used to obtain a smaller regret bound. The studies of [41] and [42] in this approach showed that better regret bounds could be achieved for several benign environments, and their improved results could be applied to the online portfolio management problem.

In the third approach, researchers transitioned to more challenging and natural scenarios that are referred to as partial feedback settings where the feedback in each round is no longer the entire loss function or loss vector, but only the loss from a subset of actions. The studies of [3, 4, 43] showed that even in situations where the feedback information is limited, there are still algorithms that can achieve small regret bounds.

In the last approach, researchers studied variants of the OCO and PEA problems. For instance, in the *contextual bandit* problem [65], the player is allowed to have access to additional information from the environment before making a decision. Another variant studied extensively is the *selective sampling* [23, 25] problem which is intrinsically an online binary classification problem, but the player has to issue a query to obtain feedback information for learning.

The high level goal of the previous studies were to define suitable problems for modeling realistic learning scenarios, to determine what types of additional information (such as specific classes of loss functions, special properties between loss functions, or actively queried informa-

tion) from different models were helpful during the learning process, and to determine how to exploit this information and use it to construct algorithms with small regret bounds. In the next section, we will continue this high level goal and present our contributions along this direction.



## 1.2 Main Contributions

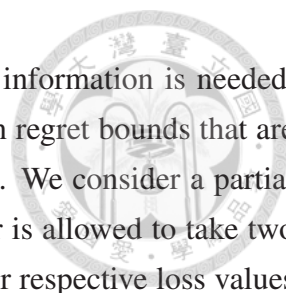
In this thesis, we make four contributions that propose new problem models and develop new algorithms that bring us one step toward the understanding of realistic online learning scenarios.

### 1.2.1 Online Learning in Gradually Evolving Worlds

The approaches that are mentioned above are not able to model certain classes of natural environments in which the loss functions change gradually from time to time. There are many instances of gradually changing environments in our daily life. Examples include the weather system and the stock market. The prevailing weather conditions and the stock prices at one moment are usually correlated with the conditions at the next moment and the differences from one moment to the next are usually small. Drastic and abrupt changes only occur sporadically. Thus, we define a new notion that is referred to as the *deviation* that measures the total difference between consecutive loss functions in order to describe a gradually changing environment. Inspired by the natural property of gradually changing loss functions, we show that a simple, but clever, modification of the FOLLOW THE REGULARIZED LEADER algorithm leads to regret in terms of deviation, thereby implying a small regret for environments with small deviations (when the changes in the loss functions are mild). We use a meta-algorithm that is a variant of the MIRROR DESCENT algorithm [14, 59] to unify all of our algorithms. The meta algorithm is based on the notion of Bregman divergence with respect to a regularization function. We also derive different algorithms for different situations by simply combining the meta algorithm with different choices for the regularization function.

### 1.2.2 Beating Bandits in Gradually Evolving Worlds

Next, we ask the same question in a setting where there is partial information. That is, for the OCO problem, when the loss functions satisfy the deviation constraint and the player receives only partial or bandit information, can we design algorithms that achieve small regret bounds? The works of [36, 47, 62] showed that a lack of information about the loss functions affects



the resulting regret bounds. Thus, our goal is to determine how much information is needed, in situations where the deviation constraint is in effect, in order to obtain regret bounds that are close to the regret bounds that were obtained in our previous study [28]. We consider a partial information setting, the two-point bandit setting of [4], where the player is allowed to take two actions, instead of just one, in a given round and is allowed to know their respective loss values and the average of the loss values is counted as the loss for that round. We answer the previous question affirmatively. A novel sampling scheme is designed in order to estimate the unknown gradient information that is needed in order to apply the algorithms that are introduced in [28]. With a sampling scheme like this in hand, we construct two-point bandit algorithms that are able to achieve regret bounds that are close to the regret bounds from our previous study [28] that were based on the full information setting <sup>2</sup>.

### 1.2.3 Online Learning with Queries

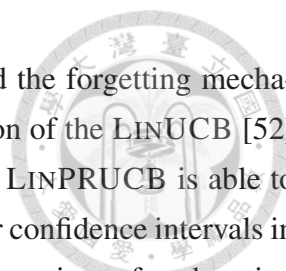
In all of the situations that are mentioned above, the player seems to be passive and seems to have no control over the information from the loss functions. However, there are scenarios where it seems possible for the player to spend some effort or resources to actively collect some intentionally selected information about the loss functions. This observation has inspired a new branch of approaches that consider the possibility of *active* players. We describe a new scenario in [27] where the player is allowed to actively issue a limited number of queries in order to obtain the loss information she needs in each round before making a decision. This scenario generalizes the previous approaches in which no query is allowed to be issued before a decision is made. There are two contributions in this study. The first is an algorithm that achieves the regret as a function of the number of bits that can be queried in one round and the second is a set of lower bounds that establish that the upper bound in general cannot be improved.

### 1.2.4 Pseudo-Reward for Linear Contextual Bandits

In the contextual bandit problem, which is different from the traditional bandit problem [50], the learner can access additional information about the environment (i.e., context) before making selections. Motivated by the better regret bounds that are available in settings with full information, we create a new notion called *pseudo-reward* that can be employed for making guesses about unseen rewards and develop a forgetting mechanism for handling fallacious rewards that

<sup>2</sup>In the full information setting, the feedback information in each round is the entire loss function.





were computed in the early rounds. Combining the pseudo-rewards and the forgetting mechanism, we propose a new algorithm called LINPRUCB that is an extension of the LINUCB [52] algorithm. This new algorithm has the following two advantages. First, LINPRUCB is able to update the linear models of all actions in each round and results in sharper confidence intervals in the earlier rounds as compared to LINUCB. Second, it computes the uncertainty of each action that will be used for exploration during the model updating stage, allowing for flexibility and quick selections of actions. To the best of our knowledge, these interesting ideas, which involve making guesses about unseen rewards and moving the exploration from the action selection stage to the modeling stage, have not been studied seriously before.

### **1.3 Thesis Structure**

In Chapter 2, the online decision problem and its four variants will be reviewed. A review of the recent progress related to the variants will be included as well. This chapter will provide a global view of the connections between the various results and our studies that have been derived from the online decision problem.

We will present our four studies in the following chapters. The study that was mentioned in Subsection 1.2.1 will be the topic of Chapter 3. In Chapter 4, we will discuss our study that was mentioned in Subsection 1.2.2 in detail. The topic of Chapter 5 is our study that was mentioned in Subsection 1.2.3. Finally, the study that was mentioned in Subsection 1.2.4 will be discussed thoroughly in Chapter 6.





## Chapter 2

# The Online Decision Problem

This chapter discusses the online decision problem. It will begin with a formal definition of the online decision problem. The relations between online learning, other philosophies of learning, and the philosophy of regret minimization will be clarified in Section 2.1. In Section 2.3, the four basic instances of the online decision problem and two fundamental algorithms will be reviewed. Finally, we will cover recent progress related to the online decision problem in Section 2.4. This will include studies that have been derived from the basic problems that are described in Section 2.3 and from our studies as well, in order to provide a global view of the positions and contributions of our studies and to point out future directions. Let us begin with a formal definition of the online decision problem.

The online decision problem is an abstract framework that covers many online problems and can be formulated as follows. There is a player facing an unknown environment who has to make iterative decisions for  $T$  rounds in the following way: In each round,  $t \in \{1, 2, \dots, T\}$ ,

1. The environment secretly chooses a loss function  $f_t : \mathcal{X} \rightarrow \mathbb{R}$  that assigns each action in  $\mathcal{X}$  a loss.
2. The player decides an action  $x_t$  to play from a feasible set  $\mathcal{X} \subseteq \mathbb{R}^n$  based on her decision strategy.
3. After taking the action  $x_t$ , the player receives certain loss information from the environment, and then
4. The player uses the loss information obtained in this round to adjust her decision strategy for the next round.

The player would like to have an online algorithm that can learn from the past, make better decisions as time goes by, and keep the total accumulated loss small. The standard notion for



evaluating such an online algorithm is called *regret*. It is defined as

$$\sum_{t=1}^T f_t(x_t) - \arg \min_{\pi \in \mathcal{X}} \sum_{t=1}^T f_t(\pi),$$

and is the difference between the total loss of the algorithm and that of the best fixed action  $\pi \in \mathcal{X}$ . The goal of an online algorithm is to minimize regret.

## 2.1 Relations Between Other Areas

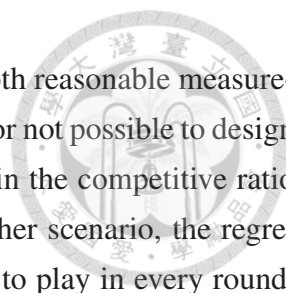
In the field of online learning, the learning examples are presented in an online fashion (i.e., one example per round). This characteristic makes online learning different from batch learning because the input in batch learning is a pool of training examples that are given at the beginning of the learning process.

Furthermore, it is assumed that there exists an unknown model that generates examples for learning and testing in batch learning. Thus, in batch learning, the goal is to learn the hidden target model. On the other hand, in online learning there is no such an assumption for an unknown model. Therefore, in online learning, the performance of an online algorithm is compared to an off-line algorithm and is measured by the regret. However, online learning and batch learning are closely related. For example, there are online algorithms that can be used to solve batch learning problems (e.g., classification with perceptron).

Online learning is also different from reinforcement learning because the concepts of *state* is considered in reinforcement learning. In an instance of reinforcement learning, the player is transited from one state to another according to an unknown Markov process. In different states, the player's action sets and loss functions will be different, and a different action will lead to a different next state as well. On the other hand, an instance of online learning can be viewed as a special case of reinforcement learning where there is only one single state and one single action set.

## 2.2 Justification of Regret

There is more than one way to measure the performance of an online algorithm. Different from the regret, the *competitive ratio* is the ratio between the total loss of the algorithm and the total loss from the best actions in each round (the loss in each round is the loss of the best action in



the corresponding round). Although regrets and competitive ratios are both reasonable measurements, studies of competitive ratios showed that many problems are hard or not possible to design algorithms to achieve satisfying ratios. This is because the comparator in the competitive ratio is allowed to choose the best action in each round. If we consider another scenario, the regret minimization problem, in which the comparator can only fix one action to play in every round, then many problems were proved to have average regret bounds converging to zero as the number of rounds grows larger, showing certain sense of learnability. Furthermore, the study of regret bounds leads to the discovery of the following interesting connections. An online learning algorithm is called a no-regret algorithm if its average regret converges to zero as  $T$  increases. Study in [16] showed that there are no-regret algorithms for the PEA problem (which will be defined in Section 2.3) that can be used to achieve the minimax value for a zero-sum game and provided a proof for von Neumann's minimax theorem. They also showed that no-regret algorithms can be used to find a  $\epsilon$ -correlated equilibrium.

## 2.3 Instances of the Online Decision Problem and Two Basic Algorithms

We can define different learning problems by specifying different types of loss functions and feasible sets and by considering the different amounts of loss information that a player can obtain. We list four basic and important instances of the online decision problem in Table 2.1.

If the loss function  $f_t$  is a convex function, the set of actions  $\mathcal{X}$  is a convex body, and the player's feedback information is the entire loss function  $f_t$  (which is referred to as the *full* information setting), then the problem is called the *online convex optimization* (OCO). If the player only knows the loss for the chosen action  $f_t(x_t)$  (which is referred to as the *bandit* information setting), then the problem is called OCO with bandit feedback (OCOFB). If the set of actions  $\mathcal{X}$  is a finite set and the loss function  $f_t$  is an arbitrary function, then the problem with full information feedback is called *prediction with expert advice* (PEA). The PEA with bandit feedback is called the *multi-armed bandit* (MAB) problem.

From the player's perspective, it is clear that one is more likely to take better actions if one has more information about the loss functions. On the other hand, different environments generate different types of loss functions and release different amounts of loss information to the player. Therefore, different problem models are required in order to formulate different types of loss functions and different amounts of loss information so that one can design algorithms that

	$f_t$ : a convex function $\mathcal{X}$ : a convex body	$f_t$ : an arbitrary function $\mathcal{X}$ : a finite set
full information	online convex optimization (OCO)	prediction with expert advice (PEA)
bandit information	OCO with bandit information (OCOBF)	multi-armed bandit (MAB)

Table 2.1: Instances of the online decision problem.

achieve low regret bounds or provide lower bounds for these problems.

**Remark 2.1.** Note that we can denote a reward function using a concave function to formulate an OCO problem whose goal is to maximize the total reward. Further, the problem of maximizing the total reward can be converted into a problem of minimizing the total loss by simply putting a negative sign before the concave reward function. Thus, for a clearer and simpler presentation, from now on we only focus on problems that consider convex loss functions and aim to minimize the total loss.

In the following subsections, we will review two important and optimal algorithms, the GRADIENT DESCENT and the MULTIPLICATIVE WEIGHTS UPDATE, for the OCO and PEA problems respectively.

### 2.3.1 The GRADIENT DESCENT Algorithm

For the OCO problem, [68] designed the GRADIENT DESCENT algorithm that is shown in Algorithm 1 achieving the optimal regret of order  $O(|\mathcal{X}|G\sqrt{T})$  when the diameter of the feasible set  $\mathcal{X}$  has a constant upper bound  $|\mathcal{X}|$ , and for all  $t$  and  $x \in \mathcal{X}$ , the  $L_2$ -norm of the gradient of  $f_t(x)$  is upper bounded by a constant  $G$ .

---

#### Algorithm 1 GRADIENT DESCENT

---

- 1: Initially, set  $\eta = \frac{|\mathcal{X}|}{G\sqrt{T}}$  and let  $x_1 = (0, \dots, 0)^\top$ .
  - 2: In round  $t \in \{1, \dots, T\}$ :
    - 2(a): Play  $x_t$ .
    - 2(b): Receive  $f_t(\cdot)$  and compute  $y_{t+1} = x_t - \eta \nabla f_t(x_t)$ .
    - 2(c): Update  $x_{t+1} = \Pi_{\mathcal{X}}(y_{t+1})$ .
- 

The GRADIENT DESCENT algorithm plays an action  $x_t$  in round  $t$  that is computed by loss functions  $f_1(\cdot), \dots, f_{t-1}(\cdot)$  that were gathered in the previous rounds. After taking an action,

the loss function  $f_t(\cdot)$  is revealed. The algorithm computes the action for the next round  $x_{t+1}$  by moving one step in the opposite direction of the gradient of  $f_t(\cdot)$  with a step size  $\eta$ . If this process results in a non-feasible action (i.e.,  $y_{t+1} \notin \mathcal{X}$ ), then the algorithm uses the projection  $\Pi_{\mathcal{X}}(y_{t+1})$ , which is  $\arg \min_{x \in \mathcal{X}} \|x - y_{t+1}\|_2$ , to compute the closest point in  $\mathcal{X}$  to find  $x_{t+1}$ .

The GRADIENT DESCENT algorithm is based on the property of the gradient. The gradient of the loss function  $f_t(\cdot)$  at point  $x_t$  gives the direction of the fastest ascent starting from  $x_t$ . Thus, in order to minimize the loss that is suffered, it makes sense to choose a point that is located at the minimum value of the convex loss function. The negative gradient,  $-\nabla f_t(x_t)$ , provides a possible direction for moving toward the point that is at the minimum value, and a carefully chosen step size  $\eta$  prevents the algorithm from overreacting. A step size that is too large may result in a point that is far from the minimum.

The GRADIENT DESCENT algorithm and the idea behind it have inspired the development of other more general or sophisticated algorithms such as the MIRROR DESCENT [14], the ONLINE NEWTON STEP [45], and many other successful algorithms. For more information about the GRADIENT DESCENT algorithm, please refer to an excellent survey in [40].

### 2.3.2 The MULTIPLICATIVE WEIGHTS UPDATE Algorithm

Assume there are  $N$  actions in the feasible set  $\mathcal{X}$  (i.e.,  $\mathcal{X} = \{1, 2, \dots, N\}$ ). The loss function  $f_t$  in round  $t$  is an  $N$ -dimensional vector and its  $i$ th entry  $f_t(i)$  represents the loss of the action  $i$ . In each round, the MULTIPLICATIVE WEIGHTS UPDATE algorithm maintains a weight  $w_t(i)$  for each action  $i \in \{1, 2, \dots, N\}$ . On seeing the loss values  $f_t(1), f_t(2), \dots, f_t(N)$ , the algorithm updates the weight of action  $i$  using  $w_{t+1}(i) = w_t(i)e^{-\eta f_t(i)}$ , where  $w_t(i)$  is the weight and  $f_t(i)$  is the loss for action  $i$  in round  $t$ .

---

#### Algorithm 2 MULTIPLICATIVE WEIGHTS UPDATE

---

- 1: Initially, set  $\eta = \sqrt{\frac{\ln N}{T}}$  and set the initial weight vector  $w_1 = (\frac{1}{N}, \dots, \frac{1}{N})^\top$ .
  - 2: In round  $t \in \{1, \dots, T\}$ :
    - 2(a): Select an action according to  $p_t$ .
    - 2(b): Receive  $f_t$  and update  $w_{t+1}(i) = w_t(i)e^{-\eta f_t(i)}$  for each  $i \in \{1, 2, \dots, N\}$ .
    - 2(c): Compute  $p_{t+1}(i) = \frac{w_{t+1}(i)}{\sum_{j=1}^N w_{t+1}(j)}$ .
- 

For the PEA problem which has  $N$  actions to choose from and  $T$  rounds to play, the MULTIPLICATIVE WEIGHTS UPDATE algorithm achieves a regret of  $O(\sqrt{T \ln N})$ . The bound is, in fact, as tight as a matching lower bound of  $\Omega(\sqrt{T \ln N})$  as can be shown (see e.g. [21]). The intu-

ition that forms the basis of the MULTIPLICATIVE WEIGHTS UPDATE algorithm is very simple: When an action  $i$  leads to a large loss  $f_t(i)$  in round  $t$ , then in the next round the update scheme reduces the weight for that action heavily. For more information about the MULTIPLICATIVE WEIGHTS UPDATE algorithm, please refer to the excellent survey in [49].



## 2.4 Recent Progress

Next, we will summarize the existing studies that are related to this thesis. In Section 2.4.1, we will discuss studies that are related to the OCO and the OCOBF problems that are originally listed in the middle column of Table 2.1. Then, in Section 2.4.2, we will discuss studies that are related to the PEA and the MAB problems that are originally listed in the third column of Table 2.1. We will also discuss studies that extend the PEA and the MAB problems to the online combinatorial problems in this section.

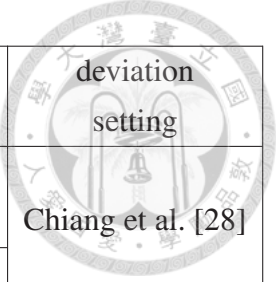
### 2.4.1 Online Convex Optimization

Studies that are related to the online convex optimization (OCO) problem and this thesis are summarized in Table 2.2. These problems are motivated by two factors. The first one is the pattern between the loss functions and the second one is the amount of feedback information. For instance, early studies considered the most general assumptions, such as assumptions that the loss functions could be arbitrary and possibly chosen in an adversarial way. Thus, we list studies that head in this direction in the “adversarial setting” column in Table 2.2. A new row for “partial information” has been added because this concept allows researchers to define different, but reasonable, feedback information settings other than the full or bandit information settings.

The most general and simplest setting for the OCO problem was studied in [68]. In [68], the loss functions are convex and the player receives the entire loss function after the player has made a decision. The main contribution of [68] is the classic GRADIENT DESCENT algorithm that achieves an optimal regret of  $O(\sqrt{T})$  (minor parameters are omitted). Later, restricted sets of loss functions were studied to determine if knowledge of specific properties of the loss functions was helpful for achieving smaller regret bounds. The study of [45] showed that a smaller regret of order  $O(\ln T)$  (some minor parameters are omitted) was achievable when the loss functions satisfied certain strongly convex properties.

However, natural environments are not always adversarial and the loss functions sometimes follow patterns that can be exploited in order to achieve smaller regret bounds. These observa-





	adversarial setting	variation setting	deviation setting
full information	linear and convex functions: Zinkevich [68]	linear functions: Hazan and Kale [41]	Chiang et al. [28]
	strictly convex functions: Hazan et al. [45]	strictly convex functions: Hazan and Kale [42]	
partial information	Agarwal et al. [4]		Chiang et al. [29]
bandit information	linear functions: Dani et al. [33] Abernethy et al. [3] Abernethy and Rakhlin [1] Bubeck et al. [19]	linear functions: Hazan and Kale [43]	
	convex functions: Flaxman et al. [36] Saha and Tewari [62]		
	strongly convex functions: Jamieson et al. [47]	strongly convex functions: lower bounded by Jamieson et al. [47]	

Table 2.2: Studies related to the OCO problem.

tions have inspired a new branch of studies that consider the patterns between loss functions. Studies that are headed in this direction are listed in the “variation setting” and the “deviation setting” columns in Table 2.2.

The first problem that tried to model natural environments more accurately was proposed by [24]. Study of [41] (listed in the first entry of the variation setting column in Table 2.2) was the first one that proposed a notion called *variation* that described a kind of pattern between loss functions. The variation  $V$  was defined as  $\sum_{t=1}^T \|f_t - \mu\|_2^2$ , where  $\mu = \sum_{t=1}^T f_t / T$  was the average of the loss functions. One can view the variation as the sum of the “similarity” between each loss function and the average of all loss functions. They studied the PEA problem and the online linear optimization problem (OCO with each  $f_t$  being linear) for the full information setting and assumed that the loss functions satisfied the variation constraint  $V$ . Their results extended the previous results that considered arbitrary sequences of loss functions. Their algorithms showed

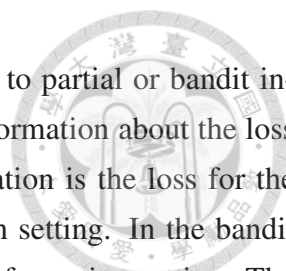
that it is possible to achieve regret bounds of order  $O(\sqrt{V})$  for a sequence of  $T$  loss functions with variation  $V$ . Note that the bounds are independent of the number of rounds  $T$ . Moreover, the  $O(\sqrt{V})$  bound is much better than the  $O(\sqrt{T})$  bound in the original adversarial setting when the loss functions have variations that are strictly smaller than the number of rounds (i.e.,  $V = o(T)$ ). When the loss functions are generated without any pattern, the variation  $V$  can be of order  $O(T)$ , and thus the  $O(\sqrt{V})$  bound recovers the  $O(\sqrt{T})$  bound that is found in [68].

After studying the PEA and OCO with linear functions, a restricted set of loss functions was studied using the variation setting. For example, in the *Portfolio Management* problem [32], the player has to design an investment strategy for accumulating wealth from each investment round. In this problem, all reward functions have certain pattern; meanwhile, each loss function is negative logarithmic exhibiting certain strongly convex property. The study of [42] (listed in the second entry of the variation setting column in Table 2.2) proposed an algorithm for achieving a regret of  $O(\ln Q)$ , where  $Q$  is the variation of a sequence of loss functions in the portfolio management problem.

According to the definition of the variation, a small variation  $V$  means that most of the loss functions center around some fixed loss function  $\mu$ . This seems to model a stationary environment where the loss functions are produced based on a fixed distribution. Therefore, we are interested in a more general scenario, where the environment is evolving in a gradual way. For example, the weather conditions or the stock prices at one moment may be correlated with the conditions or the prices at next moment and the differences are usually small. Abrupt changes usually occur only sporadically. In order to model this, we introduce a new measure for the loss functions that is called *deviation*. It is defined as

$$D_p = \sum_{t=1}^T \max_{x \in \mathcal{X}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_p^2.$$

We construct a meta algorithm for the PEA and the OCO problems with different classes of convex functions and show that this algorithm achieves regret bounds in terms of  $D_p$ . As one can show that  $D_2 = O(V)$  and  $D_1 = O(Q)$ , our results extend the results of [41, 42, 45, 68]. We also unify all of our algorithms into a meta-algorithm that is as a type of the mirror descent algorithm that was developed by [59] and [14]. The analysis of the meta algorithm provides a framework for analyzing many of the existing algorithms in Abernethy et al. [3], Audibert and Bubeck [7], Bubeck et al. [19], Freund and Schapire [37], Zinkevich [68]. This study (listed in the first entry of the deviation setting column in Table 2.2) will be discussed thoroughly in Chapter 3.



After completing the full information row in Table 2.2, we advance to partial or bandit information settings in which the player only receives partial or bandit information about the loss function. A player receives *bandit* information if the feedback information is the loss for the chosen action rather than the entire loss function in the full information setting. In the bandit setting, one can ask the same questions that are brought up in the full information setting. The bandit setting appears to be much more challenging. The major challenge is the tradeoff between exploitation (exploiting the best action that is known by far) and exploration (attempting to discover new actions that are better than the current best one). Next, we will review the positive and negative results from several related studies that consider partial or bandit settings.

For convex loss functions, [36] introduced an algorithm that estimates the gradient of a convex loss function and achieves a regret of order  $O(T^{3/4})$ . If the loss functions were linear, then [3] combined the mirror descent algorithm with self-concordant barrier functions in order to achieve a regret of  $O(N\sqrt{T\ln T})$ . Later, [62] combined the ideas of [36] and [3] in order to obtain a regret of order  $O(T^{2/3})$  for smooth convex functions. In [7], the authors established a regret of order  $O(\sqrt{NT})$  for the MAB problem. There are also other helpful studies, such as [1] that provided high probability bounds and [19] that improved the regret that is found in [3]. Like [41], [43] also considered loss functions with small variations in the bandit setting. Their result extended the work of [3] (listed to the left of Hazan and Kale [43] in Table 2.2) that considered arbitrary sequence loss functions and the work of [41] (listed above [43] in Table 2.2) that assumed full information feedback at the same time.

The studies of [36, 47, 62] (listed in the last two entries of the adversarial setting column in Table 2.2) showed that lack of information about the loss functions affects the regret bounds. Even worse, the  $\Omega(\sqrt{T})$  lower bound for the strongly convex loss functions by [47] was a surprisingly negative result that showed that in bandit setting it is impossible to achieve the  $O(\ln T)$  regret that is possible in full information settings. Thus, [4] studied multi-point bandit problems that allow the player to take multiple actions in one round. Likewise, we were also motivated to determine how much information is needed in situations, where the deviation constraint holds, in order to obtain regret bounds close to the ones from our previous study [28] that assumes full information feedback.

In our study [29], we consider a specific type of partial information setting, the two-point bandit setting of [4], in which the player can take two actions in a given round, instead of just one, and obtain their respective loss values, and the average loss is counted as the loss of that round. We answer the previous question affirmatively. That is, we provide two-point bandit algorithms that achieve regret bounds that are close to the regret bounds that are found in [28].

The dependencies on  $D_2$  in our regret bounds match those of [28] in the full information setting.

Compared to the regrets of [4] in the two-point bandit setting, we recover their results in the extreme case where  $D_2 = \Omega(T)$ , but our regret bounds are much smaller when  $D_2$  is much smaller than  $T$ . The contrast to regrets in the (one-point) bandit setting is even sharper, as the regret of [62] for convex functions is substantially higher than ours. For strongly convex functions, there is actually an  $\Omega(\sqrt{D_2})$  lower bound<sup>1</sup> that is exponentially higher than our upper bound. Moreover, all of our algorithms are simple and efficient, which again demonstrates the power of two-point bandit algorithms. Finally, since our algorithms are based on the full information algorithms of [28], we inherit two important properties. First, all our algorithms can be derived from a single meta algorithm. Second, the regret bounds can all be analyzed in a single framework. We will discuss this study (listed in the second entry of the deviation setting column in Table 2.2) thoroughly in Chapter 4.


## 2.4.2 Prediction with Expert Advice and Multi-Armed Bandit

Existing studies of the Prediction with Expert Advice (PEA) and the Multi-Armed Bandit (MAB) problems that are related to this thesis are summarized in Table 2.3. The PEA problem is studied by [56] and [37] in adversarial settings and [6] presented an extensive survey.

For the MAB approach, [12] studied the adversarial MAB in which the loss values are generated in an adversarial way. On the other hand, [10] studied the stochastic MAB in which the loss values were generated by an unknown stochastic model. An extensive survey of [18] collects and discusses several types of MAB approaches. In [1], the adversarial MAB problem was studied from a viewpoint of OCO and improved the result of [12]. Recently, [20] studied the stochastic MAB approach with additional information. They assumed that the value of the best arm and a lower bound for the gap between the value of the best arm and that of the second best arm were given to the player in advance.

There are also studies that define new feedback information for extending the MAB approach. The stochastic and adversarial bandit problems are studied in [7]. They considered four types of feedback information, namely, full information, bandit information, label efficient information (the player needs to issue a query in order to obtain the loss values for each action), and bandit label efficient information (the player needs to issue a query in order to obtain the loss of the chosen action). They developed a meta algorithm for the bandit problems and a unified analysis for the corresponding algorithms. In [58], the feedback information is modeled using a graph. In

<sup>1</sup>Such a lower bound can be easily modified from that of [47].




	stochastic or adversarial setting	deviation setting
full information	Littlestone and Warmuth [56] Freund and Schapire [37] Arora et al. [6] Hazan and Kale [41] Audibert and Bubeck [7]	Chiang et al. [28]
partial or bandit information	Auer et al. [12] Auer et al. [10] Abernethy and Rakhlin [1] Audibert and Bubeck [7] Mannor and Shamir [58] Bubeck and Cesa-Bianchi [18] Bubeck et al. [20]	
queried information	Chiang and Lu [27]	

Table 2.3: Studies related to the PEA and the MAB problems.

particular, after choosing an action, the loss values of the chosen action and every neighbor that that were connected by an edge were revealed to the player. This approach included the PEA and the MAB approaches as special cases.

There are studies that extend the PEA and the MAB problems to combinatorial problems. We list the existing studies in Table 2.4. The major issue in the combinatorial problems is the size of the feasible set. For instance, in the shortest path problem, the number of paths is exponential in the number of edges in the graph. As are result, the size of the feasible set is too large and it is not easy to design an efficient algorithm that achieves a small regret bound. Computationally efficient algorithms for the combinatorial approaches under the full information setting were introduced in [64] and [48]. On the other hand, [13] and [22] studied the combinatorial approaches under the bandit information setting. A better regret bound that is not improvable, in general, for the feasible set  $\mathcal{X} \subseteq \{0, 1\}^n$  was proved in [22]. Computationally efficient implementations for certain interesting feasible sets were also presented.

In [8], the authors considered the feasible set  $\mathcal{X} \subseteq \{0, 1\}^n$  and assumed that the loss function  $f_t \in [0, +\infty)^n$  in each round is a vector. They studied a new type of feedback, called the semi-



	adversarial setting	deviation setting
full information	Takimoto and Warmuth [64] Kalai and Vempala [48] Audibert et al. [8]	
partial or bandit information	Awerbuch and Kleinberg [13] Cesa-Bianchi and Lugosi [22] Audibert et al. [8]	

Table 2.4: Online combinatorial problems.

bandit setting, in which the player receives the entries  $f_t(i)\{x_t(i) = 1\}$  where  $i = 1, \dots, d$  and  $\{\cdot\}$  is the indicator function. For instance, for the routing problem, the semi-bandit setting allows the player to know the cost of each edge in the chosen path. They also considered the full and bandit information settings. In the full information setting, the feedback information is  $f_t$ . This can be used to model situations where the player knows the cost of each edge in the graph after choosing a path. In the bandit setting, the feedback is the inner product  $x_t^\top f_t$  that is used to model the case where the player only knows the cost of the chosen path.

In all of the problem settings that are mentioned above, the player takes a passive role in the environment and has no control over the information from the loss functions. However, there are scenarios where it seems that it is possible for the player to spend some effort or resources to actively collect some intentionally selected information about the loss functions. This observation has inspired a new branch of problems that consider the possibility of *active* players.

In our work [27], we study a modified prediction with expert advice (PEA) problem in the following way. In each round, we gave the player a  $B$ -bit budget that allowed her to choose a query for  $B$  bits of information about the loss vector before choosing her action. We assume that each loss value was represented by a  $K$ -bit string and that distinct loss values differ by at least  $\delta$ . This approach has the original PEA problem as a special case, when  $B = 0$ . On the other hand, when  $B = NK$ , one can achieve a zero regret because one has a large enough budget to determine the whole loss vector and choose the best action in each round.

The interesting cases occur when the values of  $B$  lie in the middle. With a limited number of queries, where should one spend them? Moreover, what does the regret look like as a function of the budget bound  $B$ ? The regret which our algorithm achieves depends on  $B$  in the following way. Before  $B$  approaches the bound  $B_1 = NK/2$ , the regret remains at  $O(\sqrt{T \ln N})$ . After

$B$  passes the bound  $B_1$  but before it approaches the bound  $B_2 = NK/2 + 3K/2 - 1$ , there is a noticeable drop of the regret to  $O(\sqrt{(T \ln N)/N})$ . Finally, after  $B$  passes the bound  $B_2$ , the regret takes a dramatic drop to  $(N \ln N)/\delta$ , which is independent of  $T$ . We also provide a lower bound to show that the regret is in general cannot be improved. This study (listed in the last entry of the adversarial setting column in Table 2.3) will be discussed thoroughly in Chapter 5.

There is another scenario that can be viewed as a variant of the multi-armed bandit (MAB) in which the player knows certain information *before* making her decision. For instance, consider online advertising. When a user logs into a website, the advertising algorithm is able to obtain certain information (such as habits, preferences, etc.) about the user, and then display an advertisement based on the observed information. Examples like this have inspired the study of the *contextual bandit* problem [65] in which the information that is revealed or gathered before a decision is modeled using a new notion that is referred to as *context*. For many applications, the contextual bandit problem is more realistic than the classical bandit problem of [50]. Examples where it is more appropriate include advertising, recommendations, and other Web applications [52, 55, 61].

Motivated by the better regret bounds in full information settings, we develop a new scheme called *pseudo-rewards* for making guesses about unseen rewards. Furthermore, we develop a forgetting mechanism for handling the imprecisely estimated rewards that are computed in the early iterations. Combining the pseudo-rewards and the forgetting mechanism, we propose a new algorithm called LINPRUCB that is an extension of the LINUCB [52] algorithm. The LINPRUCB algorithm is able to update the linear models that are associated with all actions in each iteration. This results in sharper confidence intervals in the earlier iterations as compared to LINUCB. Further, it computes the uncertainty for each action that will be used in the exploration during the model updating stage, allowing for flexibility in fast action selection. To the best of our knowledge, these interesting ideas, which involve making guesses about unseen rewards and moving the exploration from the action selection stage to the modeling stage, have not been studied seriously before. We will discuss this work thoroughly in Chapter 6.

We have focused on some specific directions for the online decision problem in this thesis. There have been many other wonderful studies that have also focused on it and as a result, it has grown into a rich topic with contributions coming from several fields, such as machine learning, algorithms design, and statistics. More information can be found in survey papers, such as [6, 16] or the book [21]. A sample of more recent works includes [2, 5, 15, 34, 35, 38, 39, 44, 57, 66].







## Chapter 3

# Online Learning in Gradually Evolving Worlds

As we mentioned in Subsection 2.4.1, the classic online convex optimization problem considers the most general setting in which the sequence of loss functions could be arbitrary and possibly chosen in an adversarial way. However, the environments around us may not always be adversarial, and the loss functions may have some patterns which can be exploited for achieving a smaller regret. This observation inspires new ideas to consider scenarios in which the sequence of loss functions have specific properties that can be used to model some non adversarial environment and see if such properties can be used to develop algorithms achieving small regret.

In this chapter we will first review several related works that consider certain properties in a sequence of loss functions in Section 3.1. Then, we will present the main course of this chapter, our work [28] that considers a more general scenario in Section 3.2 to Section 3.7.

### 3.1 Regret Bounded by Variation

The work [41] studied the online linear optimization problem, in which each loss function is linear and can be seen as a vector  $f_t \in [0, 1]^N$ . The authors considered the case in which the loss functions have a small variation, defined as  $V = \sum_{t=1}^T \|f_t - \mu\|_2^2$ , where  $\mu = \sum_{t=1}^T f_t / T$  is the average of the loss functions. Note that if the variation of a sequence of loss functions  $V$  is small (say,  $V \ll T$ ), meaning most of the loss function  $f_t$  center around some fixed loss function  $\mu$ , then in this kind of environment, a smart algorithm should achieve regret in terms of variation  $V$  rather than the time horizon  $T$ . For this problem, they showed that a simple modification, by

setting the step size  $\eta$  to be  $\Theta\left(\frac{1}{\sqrt{V}}\right)$ , to the GRADIENT DESCENT algorithm can achieve a regret of order  $O\left(\sqrt{V}\right)$ .

Another work studying the similar scenario is the work of [42]. They considered the *portfolio management problem* in which each loss function has the form  $f_t(x) = -\ln \langle v_t, x \rangle$  with  $v_t \in [\delta, 1]^N$  for some constant  $\delta \in (0, 1)$ , and they showed how to achieve a regret of  $O(N \log Q)$ , with  $Q = \sum_{t=1}^T \|v_t - \mu\|_2^2$  and  $\mu = \sum_{t=1}^T v_t / T$ . Note that according to their definition, a small  $Q$  also means that most of the loss functions center around some fixed loss function  $\mu$ , which is similar to the scenario that is modeled by variation  $V$ .

However, both results above seem to model a stationary environment, in which all the loss functions are produced according to some fixed distribution. Hence we consider a more general model for a gradually changing environment.

## 3.2 Online Learning in Gradually Evolving Worlds

In the work [28], we are interested in a more general scenario, in which the environment may be evolving but in a somewhat gradual way. For example, the weather condition or the stock price at one moment may have some correlation with the next and their difference is usually small, while abrupt changes only occur sporadically. To model this, we introduce a new measure, which we call  $L_p$ -deviation, for the loss functions, defined as

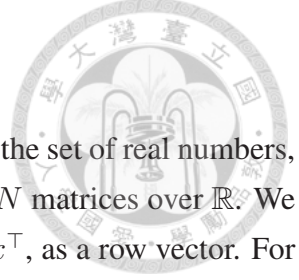
$$D_p = \sum_{t=1}^T \max_{x \in \mathcal{X}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_p^2, \quad (3.1)$$

using the convention that  $f_0$  is the all-0 function. For linear functions, this definition becomes  $D_p = \sum_{t=1}^T \|f_t - f_{t-1}\|_p^2$ , and it can be shown that  $D_2 \leq O(V)$  while there are loss functions with  $D_2 \leq O(1)$  and  $V = \Omega(T)$ . Thus, one can argue that our constraint of a small deviation is strictly easier to satisfy than that of a small variation in [41]. For the portfolio management problem, a natural measure of deviation is  $\sum_{t=1}^T \|v_t - v_{t-1}\|_2^2$ , and one can show that  $D_2 \leq O(N) \cdot \sum_{t=1}^T \|v_t - v_{t-1}\|_2^2 \leq O(NQ)$ , so one can again argue that our constraint is easier to satisfy than that of [42].

In this paper, we consider loss functions with such deviation constraints and obtain the following results. First, for the online linear optimization problem, we provide an algorithm which, when given loss functions with  $L_2$ -deviation  $D_2$ , can achieve a regret of  $O(\sqrt{D_2})$ . This is in fact optimal as a matching lower bound can be shown. Since  $D_2 \leq O(TN)$ , we immediately recover the result of [68]. Furthermore, as discussed before, since one can upper-bound  $D_2$  in

terms of  $V$  but not vice versa, our result is arguably stronger than that of [41]; interestingly, our analysis even looks simpler than theirs. Next, for the prediction with expert advice problem, we provide an algorithm such that when given loss functions with  $L_\infty$ -deviation  $D_\infty$ , it achieves a regret of  $O(\sqrt{D_\infty \ln N})$ , which is also optimal with a matching lower bound. Note that since  $D_\infty \leq O(T)$ , we also recover the  $O(\sqrt{T \ln N})$  regret bound of [37], but our result seems incomparable to that of [41]. Finally, we provide an algorithm for the online convex optimization problem studied by [46], in which the loss functions are strictly convex. Our algorithm achieves a regret of  $O(N \ln T)$  which matches that of an algorithm in [46], and when the loss functions have  $L_2$ -deviation  $D_2$ , for a large enough  $D_2$ , and satisfy some smoothness condition, our algorithm achieves a regret of  $O(N \ln D_2)$ . This can be applied to the portfolio management problem considered by [42] as the corresponding loss functions in fact satisfy our smoothness condition, and we can achieve a regret of  $O(N \ln D)$  when  $\sum_{t=1}^T \|v_t - v_{t-1}\|_2^2 \leq D$ . As discussed before, one can again argue that our result is stronger than that of [42].

All of our algorithms are based on the following idea, which we illustrate using the online linear optimization problem as an example. For general linear functions, the gradient descent algorithm is known to achieve an optimal regret, which plays in round  $t$  the point  $x_t = \Pi_{\mathcal{X}}(x_{t-1} - \eta f_{t-1})$ , the projection of  $x_{t-1} - \eta f_{t-1}$  to the feasible set  $\mathcal{X}$ . Now, if the loss functions have a small deviation,  $f_{t-1}$  may be close to  $f_t$ , so in round  $t$ , it may be a good idea to play a point which moves further in the direction of  $-f_{t-1}$  as it may make its inner product with  $f_t$  (which is its loss with respect to  $f_t$ ) smaller. In fact, it can be shown that if one could play the point  $x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta f_t)$  in round  $t$ , a very small regret could be achieved, but in reality one does not have  $f_t$  available before round to compute  $x_{t+1}$ . On the other hand, if  $f_{t-1}$  is a good estimate of  $f_t$ , the point  $\hat{x}_t = \Pi_{\mathcal{X}}(x_t - \eta f_{t-1})$  should be a good estimate of  $x_{t+1}$  too. The point  $\hat{x}_t$  can actually be computed before round  $t$  since  $f_{t-1}$  is available, so our algorithm plays  $\hat{x}_t$  in round  $t$ . Our algorithms for the prediction with expert advice problem and the online convex optimization problem use the same idea. We unify all our algorithms by a meta algorithm, which can be seen as a type of mirror descent algorithm [14], using the notion of Bregman divergence with respect to some function  $\mathcal{R}$ . Then we derive different algorithms for the three different problems simply by substantiating the meta algorithm with different choices for the function  $\mathcal{R}$ . For the online linear optimization problem, the prediction with expert advice problem, and the online convex optimization problem, respectively, the algorithms we derive can be seen as modified from the gradient descent algorithm of [68], the multiplicative algorithm of [37, 56], and the online Newton step of [46], with the modification based on the idea of moving further in the direction of  $-f_{t-1}$  discussed above.



### 3.3 Preliminaries

For a positive integer  $n$ , let  $[n]$  denote the set  $\{1, 2, \dots, n\}$ . Let  $\mathbb{R}$  denote the set of real numbers,  $\mathbb{R}^N$  the set of  $N$ -dimensional vectors over  $\mathbb{R}$ , and  $\mathbb{R}^{N \times N}$  the set of  $N \times N$  matrices over  $\mathbb{R}$ . We will see a vector  $x$  as a column vector and see its transpose, denoted by  $x^\top$ , as a row vector. For a vector  $x \in \mathbb{R}^N$  and an index  $i \in [N]$ , let  $x(i)$  denote the  $i$ th component of  $x$ . For  $x, y \in \mathbb{R}^N$ , let  $\langle x, y \rangle = \sum_{i=1}^N x(i)y(i)$  and let  $\text{RE}(x||y) = \sum_{i=1}^N x(i) \ln \frac{x(i)}{y(i)}$ . All the matrices considered in this paper will be symmetric and we will assume this without stating it later. For two matrices  $A$  and  $B$ , we write  $A \succeq B$  if  $A - B$  is a positive semidefinite (PSD) matrix. For  $x \in \mathbb{R}^N$ , let  $\|x\|_p$  denote the  $L_p$ -norm of  $x$ , and for a PSD matrix  $H \in \mathbb{R}^{N \times N}$ , define the norm  $\|x\|_H$  by  $\sqrt{x^\top H x}$ . Note that if  $H$  is the identity matrix, then  $\|x\|_H = \|x\|_2$ . We will need the following simple fact.

**Proposition 3.1.** *For any  $y, z \in \mathbb{R}^N$  and any PSD  $H \in \mathbb{R}^{N \times N}$ ,  $\|y + z\|_H^2 \leq 2\|y\|_H^2 + 2\|z\|_H^2$ .*

*Proof.* By definition,

$$2\|y\|_H^2 + 2\|z\|_H^2 - \|y + z\|_H^2 = \|y\|_H^2 + \|z\|_H^2 - 2y^\top H z = \|y - z\|_H^2 \geq 0,$$

which implies that  $2\|y\|_H^2 + 2\|z\|_H^2 \geq \|y + z\|_H^2$ .  $\square$

We will need the notion of Bregman divergence and the projection according to it.

**Definition 3.1.** *Let  $\mathcal{R} : \mathbb{R}^N \rightarrow \mathbb{R}$  be a differentiable function and  $\mathcal{X} \subseteq \mathbb{R}^N$  a convex set. Define the Bregman divergence of  $x, y \in \mathbb{R}^N$  with respect to  $\mathcal{R}$  by  $\mathcal{B}^{\mathcal{R}}(x, y) = \mathcal{R}(x) - \mathcal{R}(y) - \langle \nabla \mathcal{R}(y), x - y \rangle$ . Define the projection of  $y \in \mathbb{R}^N$  onto  $\mathcal{X}$  according to  $\mathcal{B}^{\mathcal{R}}$  by  $\Pi_{\mathcal{X}, \mathcal{R}}(y) = \arg \min_{x \in \mathcal{X}} \mathcal{B}^{\mathcal{R}}(x, y)$ .*

We consider the *online convex optimization problem*, in which an online algorithm must play in  $T$  rounds in the following way. In each round  $t \in [T]$ , it plays a point  $x_t \in \mathcal{X}$ , for some convex feasible set  $\mathcal{X} \subseteq \mathbb{R}^N$ , and after that, it receives a loss function  $f_t : \mathcal{X} \rightarrow \mathbb{R}$  and suffers a loss of  $f_t(x_t)$ . The goal is to minimize its *regret*, defined as

$$\sum_{t=1}^T f_t(x_t) - \arg \min_{\pi \in \mathcal{X}} \sum_{t=1}^T f_t(\pi),$$

which is the difference between its total loss and that of the best offline algorithm playing a single point  $\pi \in \mathcal{X}$  for all  $T$  rounds. We study four special cases of this problem. The first is the *online linear optimization problem*, in which each loss function  $f_t$  is linear. The second case is the *prediction with expert advice problem*, which can be seen as a special case of the online linear optimization problem with the set of probability distributions over  $N$  actions as the feasible set

$\mathcal{X}$ . The third case is when the loss functions are convex and smooth which is a generalization of the linear optimization setting. Finally, we consider the case when the loss functions are strictly convex in the sense defined as follows.

**Definition 3.2.** For  $\beta > 0$ , we say that a function  $f : \mathcal{X} \rightarrow \mathbb{R}$  is  $\beta$ -convex, if for all  $x, y \in \mathcal{X}$ ,

$$f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \beta \langle \nabla f(y), x - y \rangle^2.$$

As shown in [46], all the convex functions considered there are in fact  $\beta$ -convex, and thus our result also applies to those convex functions.

For simplicity of presentation, we will assume throughout the paper that the feasible set  $\mathcal{X}$  is a closed convex set contained in the unit ball  $\{x \in \mathbb{R}^n : \|x\|_2 \leq 1\}$ ; the extension to the general case is straightforward.

### 3.4 Meta Algorithm

All of our algorithms in the coming sections are based on the META algorithm, given in Algorithm 3, which has the parameter  $\mathcal{R}_t$  for  $t \in [T]$ . For different types of problems, we will have different choices of  $\mathcal{R}_t$ , which will be specified later in the respective sections. Here we allow  $\mathcal{R}_t$  to depend on  $t$ , although we do not need this freedom for linear functions and general convex functions; we only need this for strictly convex functions. Note that we define  $x_{t+1}$  using  $\mathcal{R}_t$  instead of  $\mathcal{R}_{t+1}$  for some technical reason which will be discussed soon and will become clear in the proof Lemma 3.3.

---

#### Algorithm 3 META algorithm

---

- 1: Initially, let  $x_1 = \hat{x}_1 = (1/N, \dots, 1/N)^\top$ .
  - 2: In round  $t \in [T]$ :
    - 2(a): Play  $\hat{x}_t$ .
    - 2(b): Receive  $f_t$  and compute  $\ell_t = \nabla f_t(\hat{x}_t)$ .
    - 2(c): Update
 
$$x_{t+1} = \arg \min_{x \in \mathcal{X}} (\langle \ell_t, x \rangle + \mathcal{B}^{\mathcal{R}_t}(x, x_t)),$$

$$\hat{x}_{t+1} = \arg \min_{\hat{x} \in \mathcal{X}} (\langle \ell_t, \hat{x} \rangle + \mathcal{B}^{\mathcal{R}_{t+1}}(\hat{x}, x_{t+1})).$$
- 

Our META algorithm is related to the mirror descent algorithm, as it can be shown to have the following equivalent form, which will be proved in Appendix A.1.

**Lemma 3.1.** Suppose  $y_{t+1}$  and  $\hat{y}_{t+1}$  satisfy the conditions  $\nabla \mathcal{R}_t(y_{t+1}) = \nabla \mathcal{R}_t(x_t) - \ell_t$  and  $\nabla \mathcal{R}_{t+1}(\hat{y}_{t+1}) = \nabla \mathcal{R}_{t+1}(x_{t+1}) - \ell_t$ , respectively, for a strictly convex  $\mathcal{R}_t$ . Then the update in



Step 2(c) of the META algorithm is identical to

$$\begin{aligned} x_{t+1} &= \Pi_{\mathcal{X}, \mathcal{R}_t}(y_{t+1}) = \arg \min_{x \in \mathcal{X}} \mathcal{B}^{\mathcal{R}_t}(x, y_{t+1}), \\ \hat{x}_{t+1} &= \Pi_{\mathcal{X}, \mathcal{R}_{t+1}}(\hat{y}_{t+1}) = \arg \min_{\hat{x} \in \mathcal{X}} \mathcal{B}^{\mathcal{R}_{t+1}}(\hat{x}, \hat{y}_{t+1}). \end{aligned}$$

Note that a typical mirror descent algorithm plays in round  $t$  a point roughly corresponding to our  $x_t$ , while we move one step further along the direction of  $-\ell_{t-1}$  and play  $\hat{x}_t = \arg \min_{\hat{x} \in \mathcal{X}} (\langle \ell_{t-1}, \hat{x} \rangle + \mathcal{B}^{\mathcal{R}_t}(\hat{x}, x_t))$  instead. The intuition behind our algorithm is the following. It can be shown that if one could play  $x_{t+1} = \arg \min_{x \in \mathcal{X}} (\langle \ell_t, x \rangle + \mathcal{B}^{\mathcal{R}_t}(x, x_t))$  in round  $t$ , then a small regret could be achieved, but in reality one does not have  $f_t$  available to compute  $x_{t+1}$  before round  $t$ . Nevertheless, if the loss vectors have a small deviation,  $\ell_{t-1}$  is likely to be close to  $\ell_t$ , and so is  $\hat{x}_t$  to  $x_{t+1}$ , which is made possible by defining  $x_{t+1}$  and  $\hat{x}_t$  both using  $\mathcal{R}_t$ . Based on this idea, we let our algorithm play  $\hat{x}_t$  in round  $t$ .

Now let us see how to bound the regret of the algorithm. Consider any  $\pi \in \mathcal{X}$  taken by the offline algorithm. Then for a  $\beta$ -convex function  $f_t$ , we know from the definition that

$$f_t(\hat{x}_t) - f_t(\pi) \leq \langle \ell_t, \hat{x}_t - \pi \rangle - \beta \|\hat{x}_t - \pi\|_{h_t}^2, \text{ where } h_t = \ell_t \ell_t^\top, \quad (3.2)$$

while for a linear or a general convex  $f_t$ , the above still holds with  $\beta = 0$ . Thus, the key is to bound  $\langle \ell_t, \hat{x}_t - \pi \rangle$ , which is given by the following lemma. We give the proof in Appendix A.2.

**Lemma 3.2.** *Let  $S_t = \langle \ell_t - \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle$ ,  $A_t = \mathcal{B}^{\mathcal{R}_t}(\pi, x_t) - \mathcal{B}^{\mathcal{R}_t}(\pi, x_{t+1})$  and  $B_t = \mathcal{B}^{\mathcal{R}_t}(x_{t+1}, \hat{x}_t) + \mathcal{B}^{\mathcal{R}_t}(\hat{x}_t, x_t)$ . Then*

$$\langle \ell_t, \hat{x}_t - \pi \rangle \leq S_t + A_t - B_t.$$

The following lemma provides an upper bound for  $S_t$ .

**Lemma 3.3.** *Suppose  $\|\cdot\|$  is a norm, with dual norm  $\|\cdot\|_*$ , such that  $\frac{1}{2} \|x - x'\|^2 \leq \mathcal{B}^{\mathcal{R}_t}(x, x')$  for any  $x, x' \in \mathcal{X}$ . Then,*

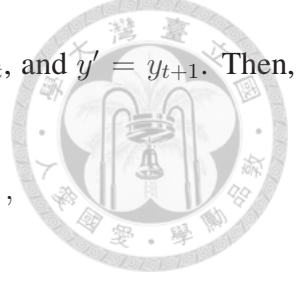
$$S_t = \langle \ell_t - \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle \leq \|\ell_t - \ell_{t-1}\|_*^2.$$

*Proof.* By a generalized Cauchy-Schwartz inequality,

$$S_t = \langle \ell_t - \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle \leq \|\ell_t - \ell_{t-1}\|_* \|\hat{x}_t - x_{t+1}\|.$$

Then we need the following proposition.

**Proposition 3.2.**  $\|\hat{x}_t - x_{t+1}\| \leq \|\nabla \mathcal{R}_t(\hat{y}_t) - \nabla \mathcal{R}_t(y_{t+1})\|_*.$



*Proof.* To simplify the notation, let  $\mathcal{R} = \mathcal{R}_t$ ,  $x = \hat{x}_t$ ,  $x' = x_{t+1}$ ,  $y = \hat{y}_t$ , and  $y' = y_{t+1}$ . Then, from the property of the norm, we know that

$$\frac{1}{2} \|x - x'\|^2 \leq \mathcal{R}(x) - \mathcal{R}(x') - \langle \nabla \mathcal{R}(x'), x - x' \rangle,$$

and also

$$\frac{1}{2} \|x' - x\|^2 \leq \mathcal{R}(x') - \mathcal{R}(x) - \langle \nabla \mathcal{R}(x), x' - x \rangle.$$

Adding these two bounds, we obtain

$$\|x - x'\|^2 \leq \langle \nabla \mathcal{R}(x) - \nabla \mathcal{R}(x'), x - x' \rangle. \quad (3.3)$$

Next, we show that

$$\langle \nabla \mathcal{R}(x) - \nabla \mathcal{R}(x'), x - x' \rangle \leq \langle \nabla \mathcal{R}(y) - \nabla \mathcal{R}(y'), x - x' \rangle. \quad (3.4)$$

For this, we need Fact A.1 in Appendix A.2. By letting  $\phi(z) = \mathcal{B}^{\mathcal{R}}(z, y)$ , we have  $x = \arg \min_{z \in \mathcal{X}} \phi(z)$ ,  $\nabla \phi(x) = \nabla \mathcal{R}(x) - \nabla \mathcal{R}(y)$ , and

$$\langle \nabla \mathcal{R}(x) - \nabla \mathcal{R}(y), x' - x \rangle \geq 0.$$

On the other hand, by letting  $\phi(z) = \mathcal{B}^{\mathcal{R}}(z, y')$ , we have  $x' = \arg \min_{z \in \mathcal{X}} \phi(z)$ ,  $\nabla \phi(x') = \nabla \mathcal{R}(x') - \nabla \mathcal{R}(y')$ , and

$$\langle \nabla \mathcal{R}(x') - \nabla \mathcal{R}(y'), x - x' \rangle \geq 0.$$

Combining these two bounds, we have

$$\langle (\nabla \mathcal{R}(y) - \nabla \mathcal{R}(y')) - (\nabla \mathcal{R}(x) - \nabla \mathcal{R}(x')), x - x' \rangle \geq 0,$$

which implies the inequality in (3.4).

Finally, by combining (3.3) and (3.4), we obtain

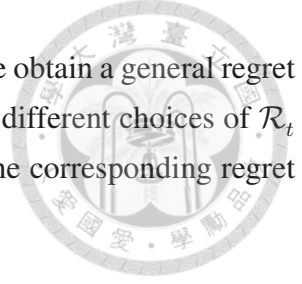
$$\|x - x'\|^2 \leq \langle \nabla \mathcal{R}(y) - \nabla \mathcal{R}(y'), x - x' \rangle \leq \|\nabla \mathcal{R}(y) - \nabla \mathcal{R}(y')\|_* \|x - x'\|,$$

by a generalized Cauchy-Schwartz inequality. Dividing both sides by  $\|x - x'\|$ , we have the proposition.  $\square$

From this proposition, we have

$$\|\hat{x}_t - x_{t+1}\| \leq \|(\nabla \mathcal{R}_t(x_t) - \ell_{t-1}) - (\nabla \mathcal{R}_t(x_t) - \ell_t)\|_* = \|\ell_t - \ell_{t-1}\|_*. \quad (3.5)$$

This is why we define  $x_{t+1}$  and  $y_{t+1}$  using  $\mathcal{R}_t$  instead of  $\mathcal{R}_{t+1}$ . Finally, by combining these bounds together, we have the lemma.  $\square$



Taking the sum over  $t$  of the bounds in Lemma 3.2 and Lemma 3.3, we obtain a general regret bound for the META algorithm. In the following sections, we will make different choices of  $\mathcal{R}_t$  and the norms for different types of loss functions, and we will derive the corresponding regret bounds.

## 3.5 Linear Loss Functions

In this section, we consider the case that each loss function  $f_t$  is linear, which can be seen as an  $N$ -dimensional vector in  $\mathbb{R}^N$  with  $f_t(x) = \langle f_t, x \rangle$  and  $\nabla f_t(x) = f_t$ . We measure the deviation of the loss functions by their  $L_p$ -deviation, defined in (3.1), which becomes  $\sum_{t=1}^T \|f_t - f_{t-1}\|_p^2$  for linear functions. To bound the regret suffered in each round, we can use the bound in (3.2) with  $\beta = 0$  and we drop the term  $B_t$  from the bound in Lemma 3.2. By summing the bound over  $t$ , we have

$$\sum_{t=1}^T f_t(\hat{x}_t) - f_t(\pi) \leq \sum_{t=1}^T S_t + \sum_{t=1}^T A_t, \quad (3.6)$$

where  $S_t = \langle f_t - f_{t-1}, \hat{x}_t - x_{t+1} \rangle$  and  $A_t = \mathcal{B}^{\mathcal{R}_t}(\pi, x_t) - \mathcal{B}^{\mathcal{R}_t}(\pi, x_{t+1})$ . In the following two sections, we will consider the online linear optimization problem and the prediction with expert advice problem, respectively, in which we will have different choices of  $\mathcal{R}_t$  and use different measures of deviation.

### 3.5.1 Online Linear Optimization Problem

In this section, we consider the online linear optimization problem, and we consider loss functions with  $L_2$ -deviation  $D_2$ . To instantiate the META algorithm for such loss functions, we choose

- $\mathcal{R}_t(x) = \frac{1}{2\eta} \|x\|_2^2$ , for every  $t \in [T]$ ,

where  $\eta$  is the learning rate to be determined later; in fact, it can also be adjusted in the algorithm using the standard doubling trick by keeping track of the deviation accumulated so far. It is easy to show that with this choice of  $\mathcal{R}_t$ ,

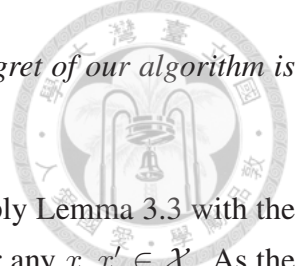
- $\nabla \mathcal{R}_t(x) = \frac{x}{\eta}$ ,  $\mathcal{B}^{\mathcal{R}_t}(x, y) = \frac{1}{2\eta} \|x - y\|_2^2$ , and  $\Pi_{\mathcal{X}, \mathcal{R}_t}(y) = \arg \min_{x \in \mathcal{X}} \|x - y\|_2^2$ .

Then, according to Lemma 3.1, the update in Step 2(c) of META algorithm becomes:

- $x_{t+1} = \arg \min_{x \in \mathcal{X}} \|x - y_{t+1}\|_2^2$ , with  $y_{t+1} = x_t - \eta f_t$ ,
- $\hat{x}_{t+1} = \arg \min_{\hat{x} \in \mathcal{X}} \|\hat{x} - \hat{y}_{t+1}\|_2^2$ , with  $\hat{y}_{t+1} = x_{t+1} - \eta f_t$ .

The regret achieved by our algorithm is guaranteed by the following.





**Theorem 3.1.** *When the  $L_2$ -deviation of the loss functions is  $D_2$ , the regret of our algorithm is at most  $O(\sqrt{D_2})$ .*

*Proof.* We start by bounding the first sum in (3.6). Note that we can apply Lemma 3.3 with the norm  $\|\cdot\| = \frac{1}{\sqrt{\eta}} \|\cdot\|_2$ , since  $\frac{1}{2} \|x - x'\|^2 = \frac{1}{2\eta} \|x - x'\|_2^2 = \mathcal{B}^{\mathcal{R}_t}(x, x')$  for any  $x, x' \in \mathcal{X}$ . As the dual norm is  $\|\cdot\|_* = \sqrt{\eta} \|\cdot\|_2$ , Lemma 3.3 gives us

$$\sum_{t=1}^T S_t \leq \sum_{t=1}^T \|f_t - f_{t-1}\|_*^2 \leq \sum_{t=1}^T \eta \|f_t - f_{t-1}\|_2^2 \leq \eta D_2.$$

Next, note that  $A_t = \frac{1}{2\eta} \|\pi - x_t\|_2^2 - \frac{1}{2\eta} \|\pi - x_{t+1}\|_2^2$ , so the second sum in (3.6) is

$$\sum_{t=1}^T A_t = \frac{1}{2\eta} (\|\pi - x_1\|_2^2 - \|\pi - x_{T+1}\|_2^2) \leq \frac{2}{\eta},$$

by telescoping and then using the fact that  $\|\pi - x_1\|_2^2 \leq 4$  and  $\|\pi - x_{T+1}\|_2^2 \geq 0$ . Finally, by substituting these two bounds into (3.6), we have

$$\sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi)) \leq \eta D_2 + \frac{2}{\eta} \leq O(\sqrt{D_2}),$$

by choosing  $\eta = \sqrt{2/D_2}$ , which proves the theorem.  $\square$

Let us make three remarks about Theorem 3.1. First, as mentioned in the introduction, one can argue that our result is strictly stronger than that of [41] as our deviation bound is easier to satisfy. This is because by Proposition 3.1, we have  $\|f_t - f_{t-1}\|_2^2 \leq 2(\|f_t - \mu\|_2^2 + \|\mu - f_{t-1}\|_2^2)$  and thus  $D_2 \leq 4V + O(1)$ , while, for example, with  $N = 1$ ,  $f_t = 0$  for  $1 \leq t \leq T/2$  and  $f_t = 1$  for  $T/2 < t \leq T$ , we have  $D_2 \leq O(1)$  and  $V \geq \Omega(T)$ . Next, we claim that the regret achieved by our algorithm is optimal. This is because a matching lower bound can be shown by simply setting the loss functions of all but the first  $r = D_2$  rounds to be the all-0 function, and then applying the known  $\Omega(\sqrt{r})$  regret lower bound on the first  $r$  rounds. Finally, our algorithm can be seen as a modification of the gradient descent (GD) algorithm of [68], which plays  $x_t$ , instead of our  $\hat{x}_t$ , in round  $t$ . Then one may wonder if GD already performs as well as our algorithm does. The following lemma provides a negative answer, which means that our modification is in fact necessary.

**Lemma 3.4.** *The regret of the GD algorithm is at least  $\Omega(\min\{D_2, \sqrt{T}\})$ .*

*Proof.* One may wonder if the GD algorithm can also achieve the same regret as our algorithm's by choosing the learning rate  $\eta$  properly. We show that no matter what the learning rate  $\eta$  the GD algorithm chooses, there exists a sequence of loss vectors which can cause a large regret. Let  $f$  be any unit vector passing through  $x_1$ . Let  $s = \lfloor 1/\eta \rfloor$ , so that if we use  $f_t = f$  for every  $t \leq s$ , each such  $y_{t+1} = x_1 - t\eta f$  still remains in  $\mathcal{X}$  and thus  $x_{t+1} = y_{t+1}$ . Next, we analyze the regret by considering the following three cases depending on the range of  $s$ .

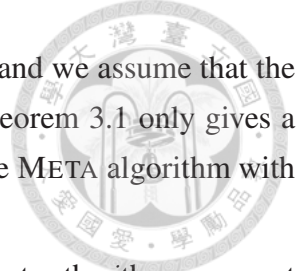
First, when  $s \geq \sqrt{T}$ , we choose  $f_t = f$  for  $t$  from 1 to  $\lfloor s/2 \rfloor$  and  $f_t = 0$  for the remaining  $t$ . Clearly, the best strategy of the offline algorithm is to play  $\pi = -f$ . On the other hand, since the learning rate  $\eta$  is too small, the strategy  $x_t$  played by GD, for  $t \leq \lfloor s/2 \rfloor$ , is far away from  $\pi$ , so that  $\langle f_t, x_t - \pi \rangle \geq 1 - t\eta \geq 1/2$ . Therefore, the regret is at least  $\lfloor s/2 \rfloor (1/2) = \Omega(\sqrt{T})$ .

Second, when  $0 < s < \sqrt{T}$ , the learning rate is high enough so that GD may overreact to each loss vector, and we make it pay by flipping the direction of loss vectors frequently. More precisely, we use the vector  $f$  for the first  $s$  rounds so that  $x_{t+1} = x_1 - t\eta f$  for any  $t \leq s$ , but just as  $x_{s+1}$  moves far enough in the direction of  $-f$ , we make it pay by switching the loss vector to  $-f$ , which we continue to use for  $s$  rounds. Note that  $x_{s+1+r} = x_{s+1-r}$  but  $f_{s+1+r} = -f_{s+1-r}$  for any  $r \leq s$ , so  $\sum_{t=1}^{2s} \langle f_t, x_t - x_1 \rangle = \langle f_{s+1}, x_{s+1} - x_1 \rangle \geq \Omega(1)$ . As  $x_{2s+1}$  returns back to  $x_1$ , we can see the first  $2s$  rounds as a period, which only contributes  $\|2f\|_2^2 = 4$  to the deviation. Then we repeat the period for  $\tau$  times, where  $\tau = \lfloor D_2/4 \rfloor$  if there are enough rounds, with  $\lfloor T/(2s) \rfloor \geq \lfloor D_2/4 \rfloor$ , to use up the deviation  $D_2$ , and  $\tau = \lfloor T/(2s) \rfloor$  otherwise. For any remaining round  $t$ , we simply choose  $f_t = 0$ . As a result, the total regret is at least  $\Omega(1) \cdot \tau = \Omega(\min\{D_2/4, T/(2s)\}) = \Omega(\min\{D_2, \sqrt{T}\})$ .

Finally, when  $s = 0$ , the learning rate is so high that we can easily make GD pay by flipping the direction of the loss vector in each round. More precisely, by starting with  $f_1 = -f$ , we can have  $x_2$  on the boundary of  $\mathcal{X}$ , which means that if we then alternate between  $f$  and  $-f$ , the strategies GD plays will alternate between  $x_3$  and  $x_2$  which have a constant distance from each other. Then following the analysis in the second case, one can show that the total regret is at least  $\Omega(\min\{D_2, T\})$ .  $\square$

### 3.5.2 Prediction with Expert Advice

In this section, we consider the prediction with expert advice problem. Now, the feasible set  $\mathcal{X}$  is the set of probability distributions over  $N$  actions, which can also be represented as  $N$ -dimensional vectors. Although this problem can be seen as a special case of that in Subsection 3.5.1 and Theorem 3.1 there also applies here, we would like to obtain a stronger result.



More precisely, now we consider  $L_\infty$ -deviation instead of  $L_2$ -deviation, and we assume that the loss functions have  $L_\infty$ -deviation  $D_\infty$ . Note that with  $D_2 \leq D_\infty N$ , Theorem 3.1 only gives a regret bound of  $O(\sqrt{D_\infty N})$ . To obtain a smaller regret, we instantiate the META algorithm with

- $\mathcal{R}_t(x) = \frac{1}{\eta} \sum_{i=1}^N x(i) (\ln x(i) - 1)$ , for every  $t \in [T]$ ,

where  $\eta$  is the learning rate to be determined later and recall that  $x(i)$  denotes the  $i$ th component of the vector  $x$ . It is easy to show that with this choice,

- $\nabla \mathcal{R}_t(x) = \frac{1}{\eta} (\ln x(1), \dots, \ln x(N))^\top$ ,  $\mathcal{B}^{\mathcal{R}_t}(x, y) = \frac{1}{\eta} \text{RE}(x||y)$ , and  $\Pi_{\mathcal{X}, \mathcal{R}_t}(y) = y/Z$  with the normalization factor  $Z = \sum_{j=1}^N y(j)$ .

Then, according to Lemma 3.1, the update in Step 2(c) of the META algorithm becomes:

- $x_{t+1}(i) = x_t(i) e^{-\eta f_t(i)} / Z_{t+1}$ , for each  $i \in [N]$ , with  $Z_{t+1} = \sum_{j=1}^N x_t(j) e^{-\eta f_t(j)}$ ,  
 $\hat{x}_{t+1}(i) = x_{t+1}(i) e^{-\eta f_t(i)} / \hat{Z}_{t+1}$ , for each  $i \in [N]$ , with  $\hat{Z}_{t+1} = \sum_{j=1}^N x_{t+1}(j) e^{-\eta f_t(j)}$ .

Note that our algorithm can be seen as a modification of the multiplicative updates algorithm [37, 56] which plays  $x_t$ , instead of our  $\hat{x}_t$ , in round  $t$ . The regret achieved by our algorithm is guaranteed by the following.

**Theorem 3.2.** *When the  $L_\infty$ -deviation of the loss functions is  $D_\infty$ , the regret of our algorithm is at most  $O(\sqrt{D_\infty \ln N})$ .*

*Proof.* We start by bounding the first sum in (3.6). Note that we can apply Lemma 3.3 with the norm  $\|\cdot\| = \frac{1}{\sqrt{\eta}} \|\cdot\|_1$ , since for any  $x, x' \in \mathcal{X}$ ,  $\frac{1}{2} \|x - x'\|^2 = \frac{1}{2\eta} \|x - x'\|_1^2 \leq \frac{1}{\eta} \text{RE}(x||x') = \mathcal{B}^{\mathcal{R}_t}(x, x')$ , by Pinsker's inequality. As the dual norm is  $\|\cdot\|_* = \sqrt{\eta} \|\cdot\|_\infty$ , Lemma 3.3 gives us

$$\sum_{t=1}^T S_t \leq \sum_{t=1}^T \|f_t - f_{t-1}\|_*^2 \leq \sum_{t=1}^T \eta \|f_t - f_{t-1}\|_\infty^2 \leq \eta D_\infty.$$

Next, note that  $A_t = \frac{1}{\eta} \text{RE}(\pi||x_t) - \frac{1}{\eta} \text{RE}(\pi||x_{t+1})$ , so the second sum in (3.6) is

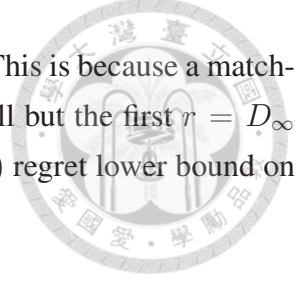
$$\sum_{t=1}^T A_t = \frac{1}{\eta} (\text{RE}(\pi||x_1) - \text{RE}(\pi||x_{T+1})) \leq \frac{1}{\eta} \ln N,$$

by telescoping and then using the fact that  $\text{RE}(\pi||x_1) \leq \ln N$  and  $\text{RE}(\pi||x_{T+1}) \geq 0$ . Finally, by substituting these two bounds into (3.6), we have

$$\sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi)) \leq \eta D_\infty + \frac{1}{\eta} \ln N \leq O\left(\sqrt{D_\infty \ln N}\right),$$

by choosing  $\eta = \sqrt{(\ln N)/D_\infty}$ , which proves the theorem.  $\square$

We remark that the regret achieved by our algorithm is also optimal. This is because a matching lower bound can be shown by simply setting the loss functions of all but the first  $r = D_\infty$  rounds to be the all-0 function, and then applying the known  $\Omega(\sqrt{r \ln N})$  regret lower bound on the first  $r$  rounds.



### 3.6 General Convex Loss Functions

In this section, we consider general convex loss functions. We measure the deviation of loss functions by their  $L_2$ -deviation defined in (3.1), which is  $\sum_{t=1}^T \max_{x \in \mathcal{X}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_2^2$ . Our algorithm for such loss functions is the same algorithm for linear functions. To bound its regret, now we need the help of the term  $B_t$  in Lemma 3.2, and we have

$$\sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi)) \leq \sum_{t=1}^T S_t + \sum_{t=1}^T A_t - \sum_{t=1}^T B_t. \quad (3.7)$$

From the proof of Theorem 3.1, we know that  $\sum_{t=1}^T A_t \leq \frac{2}{\eta}$  and  $\sum_{t=1}^T S_t \leq \sum_{t=1}^T \eta \|\ell_t - \ell_{t-1}\|_2^2$  which, unlike in Theorem 3.1, can not be immediately bounded by  $L_2$ -deviation. This is because  $\|\ell_t - \ell_{t-1}\|_2^2 = \|\nabla f_t(\hat{x}_t) - \nabla f_{t-1}(\hat{x}_{t-1})\|_2^2$ , where the two gradients are taken at different points. To handle this issue, we further assume that each gradient  $\nabla f_t$  satisfies the following  $\lambda$ -smoothness condition:

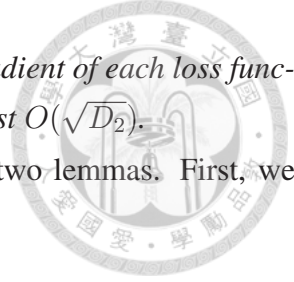
$$\|\nabla f_t(x) - \nabla f_t(y)\|_2 \leq \lambda \|x - y\|_2, \text{ for any } x, y \in \mathcal{X}. \quad (3.8)$$

We emphasize that our assumption about the smoothness of loss functions is necessary to achieve the desired bound. To see this, consider the special case of  $f_1(x) = \dots = f_T(x) = f(x)$ . If the deviation bound  $O(\sqrt{D_2})$  holds for any sequence of convex functions, then for the special case where all loss functions are identical, we will have

$$\sum_{t=1}^T f(\hat{x}_t) \leq \min_{\pi \in \mathcal{X}} \sum_{t=1}^T f(\pi) + O(1),$$

implying that  $(1/T) \sum_{t=1}^T \hat{x}_t$  approaches the optimal solution at the rate of  $O(1/T)$ . This contradicts the lower complexity bound (i.e.  $\Omega(1/\sqrt{T})$ ) for any first order optimization method [60, Theorem 3.2.1] and therefore smoothness assumption is necessary to extend our results to general convex loss functions.

Our main result of this section is the following theorem which establishes the deviation bound for general smooth convex loss functions applying META algorithm.



**Theorem 3.3.** *When the loss functions have  $L_2$ -deviation  $D_2$  and the gradient of each loss function is  $\lambda$ -smooth, with  $\lambda \leq 1/\sqrt{8D_2}$ , the regret of our algorithm is at most  $O(\sqrt{D_2})$ .*

The proof of Theorem 3.3 immediately results from the following two lemmas. First, we need the following to bound  $\sum_{t=1}^T S_t$  in terms of  $D_2$ .

**Lemma 3.5.**  $\sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_2^2 \leq 2D_2 + 2\lambda^2 \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2$ .

*Proof.*  $\|\ell_t - \ell_{t-1}\|_2^2 = \|\nabla f_t(\hat{x}_t) - \nabla f_{t-1}(\hat{x}_{t-1})\|_2^2$ , which by Proposition 3.1 is at most

$$2 \|\nabla f_t(\hat{x}_t) - \nabla f_{t-1}(\hat{x}_t)\|_2^2 + 2 \|\nabla f_{t-1}(\hat{x}_t) - \nabla f_{t-1}(\hat{x}_{t-1})\|_2^2,$$

where the second term above is at most  $2\lambda^2 \|\hat{x}_t - \hat{x}_{t-1}\|_2^2$  by the  $\lambda$ -smoothness condition. By summing the bound over  $t$ , we have the lemma.  $\square$

To eliminate the undesirable term  $2\lambda^2 \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2$  in the lemma, we use the help from the sum  $\sum_{t=1}^T B_t$ , which has the following bound.

**Lemma 3.6.**  $\sum_{t=1}^T B_t \geq \frac{1}{4\eta} \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 - O(1)$ .

*Proof.* Recall that  $B_t = \frac{1}{2\eta} \|x_{t+1} - \hat{x}_t\|_2^2 + \frac{1}{2\eta} \|\hat{x}_t - x_t\|_2^2$ , so we can write  $\sum_{t=1}^T B_t$  as

$$\begin{aligned} \frac{1}{2\eta} \sum_{t=1}^{T+1} \|x_t - \hat{x}_{t-1}\|_2^2 + \frac{1}{2\eta} \sum_{t=1}^T \|\hat{x}_t - x_t\|_2^2 &\geq \frac{1}{2\eta} \sum_{t=2}^T (\|x_t - \hat{x}_{t-1}\|_2^2 + \|\hat{x}_t - x_t\|_2^2) \\ &\geq \frac{1}{4\eta} \sum_{t=2}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2, \end{aligned}$$

by Proposition 3.1, with  $H$  being the identity matrix so that  $\|x\|_H^2 = \|x\|_2^2$ . Then the lemma follows as  $\|\hat{x}_2 - \hat{x}_1\|_2^2 \leq O(1)$ .  $\square$

According to the bounds obtained so far, the regret of our algorithm is at most

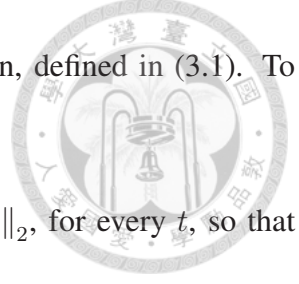
$$2\eta D_2 + 2\eta\lambda^2 \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 - \frac{1}{4\eta} \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 + O(1) + \frac{2}{\eta} \leq O\left(\eta D_2 + \frac{1}{\eta}\right) \leq O\left(\sqrt{D_2}\right),$$

when  $\lambda \leq 1/\sqrt{8\eta^2}$  and  $\eta = 1/\sqrt{D_2}$ .

### 3.7 Strictly Convex Loss Functions

In this section, we consider convex functions which are strictly convex. More precisely, suppose for some  $\beta > 0$ , each loss function is  $\beta$ -convex, so that

$$f_t(\hat{x}_t) - f_t(\pi) \leq \langle \ell_t, \hat{x}_t - \pi \rangle - \beta \|\pi - \hat{x}_t\|_{h_t}^2, \text{ where } h_t = \ell_t \ell_t^\top. \quad (3.9)$$



Again, we measure the deviation of loss functions by their  $L_2$ -deviation, defined in (3.1). To instantiate the META algorithm for such loss functions, we choose

- $\mathcal{R}_t(x) = \frac{1}{2} \|x\|_{H_t}^2$ , with  $H_t = I + \beta\gamma^2 I + \beta \sum_{\tau=1}^{t-1} \ell_\tau \ell_\tau^\top$ ,

where  $I$  is the  $N \times N$  identity matrix, and  $\gamma$  is an upper bound of  $\|\ell_t\|_2$ , for every  $t$ , so that  $\gamma^2 I \succeq \ell_t \ell_t^\top$ . It is easy to show that with this choice,

- $\nabla \mathcal{R}_t(x) = H_t x$ ,  $\mathcal{B}^{\mathcal{R}_t}(x, y) = \frac{1}{2} \|x - y\|_{H_t}^2$ , and  $\Pi_{\mathcal{X}, \mathcal{R}_t}(y) = \arg \min_{x \in \mathcal{X}} \|x - y\|_{H_t}^2$ .

Then, according to Lemma 3.1, the update in Step 2(c) of the META algorithm becomes:

- $x_{t+1} = \arg \min_{x \in \mathcal{X}} \|x - y_{t+1}\|_{H_t}^2$ , with  $y_{t+1} = x_t - H_t^{-1} \ell_t$ ,  
 $\hat{x}_{t+1} = \arg \min_{\hat{x} \in \mathcal{X}} \|\hat{x} - \hat{y}_{t+1}\|_{H_{t+1}}^2$ , with  $\hat{y}_{t+1} = x_{t+1} - H_{t+1}^{-1} \ell_t$ .

We remark that our algorithm is related to the online Newton step algorithm in [46], except that our matrix  $H_t$  is slightly different from theirs and we play  $\hat{x}_t$  in round  $t$  while they play a point roughly corresponding to our  $x_t$ . It is easy to verify that the update of our algorithm can be computed at the end of round  $t$ , because we have  $\ell_1, \dots, \ell_t$  available to compute  $H_t$  and  $H_{t+1}$ .

To bound the regret of our algorithm, note that by substituting the bound of Lemma 3.2 into (3.9) and then taking the sum over  $t$ , we obtain

$$\sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi)) \leq \sum_{t=1}^T S_t + \sum_{t=1}^T A_t - \sum_{t=1}^T B_t - \sum_{t=1}^T C_t, \quad (3.10)$$

with  $S_t = \langle \ell_t - \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle$ ,  $A_t = \frac{1}{2} \|\pi - x_t\|_{H_t}^2 - \frac{1}{2} \|\pi - x_{t+1}\|_{H_t}^2$ ,  $B_t = \frac{1}{2} \|x_{t+1} - \hat{x}_t\|_{H_t}^2 + \frac{1}{2} \|\hat{x}_t - x_t\|_{H_t}^2$ , and  $C_t = \beta \|\pi - \hat{x}_t\|_{h_t}^2$ . Then our key lemma is the following.

**Lemma 3.7.** *Suppose the loss functions are  $\beta$ -convex for some  $\beta > 0$ . Then*

$$\sum_{t=1}^T S_t + \sum_{t=1}^T A_t - \sum_{t=1}^T C_t \leq O(1 + \beta\gamma^2) + \frac{8N}{\beta} \ln \left( 1 + \frac{\beta}{4} \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_2^2 \right).$$

*Proof.* We start by bounding the first sum in (3.10). Note that we can apply Lemma 3.3 with the norm  $\|\cdot\| = \|\cdot\|_{H_t}$ , since  $\frac{1}{2} \|x - x'\|^2 = \frac{1}{2} \|x - x'\|_{H_t}^2 = \mathcal{B}^{\mathcal{R}_t}(x, x')$  for any  $x, x' \in \mathcal{X}$ . As the dual norm is  $\|\cdot\|_* = \|\cdot\|_{H_t^{-1}}$ , Lemma 3.3 gives us

$$\sum_{t=1}^T S_t \leq \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_*^2 \leq \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_{H_t^{-1}}^2.$$

Next, we bound the second sum  $\sum_{t=1}^T A_t$  in (3.10), which can be written as

$$\frac{1}{2} \|\pi - x_1\|_{H_1}^2 - \frac{1}{2} \|\pi - x_{T+1}\|_{H_{T+1}}^2 + \frac{1}{2} \sum_{t=1}^T \left( \|\pi - x_{t+1}\|_{H_{t+1}}^2 - \|\pi - x_{t+1}\|_{H_t}^2 \right).$$

Since  $\|\pi - x_1\|_{H_1}^2 = O(1 + \beta\gamma^2)$ ,  $\|\pi - x_{T+1}\|_{H_{T+1}}^2 \geq 0$ , and  $H_{t+1} - H_t = \beta h_t$ , we have

$$\sum_{t=1}^T A_t \leq O(1 + \beta\gamma^2) + \frac{\beta}{2} \sum_{t=1}^T \|\pi - x_{t+1}\|_{h_t}^2.$$



Note that unlike in the case of linear functions, here the sum does not telescope and hence we do not have a small bound for it. The last sum  $\sum_{t=1}^T C_t$  in (3.10) now comes to help. Recall that  $C_t = \beta \|\pi - \hat{x}_t\|_{h_t}^2$ , so by Proposition 3.1,

$$\frac{\beta}{2} \|\pi - x_{t+1}\|_{h_t}^2 - C_t \leq \beta \|\pi - \hat{x}_t\|_{h_t}^2 + \beta \|\hat{x}_t - x_{t+1}\|_{h_t}^2 - C_t = \beta \|\hat{x}_t - x_{t+1}\|_{h_t}^2,$$

which, by the fact that  $H_t \succeq \beta\gamma^2 I \succeq \beta h_t$  and the bound in (3.5), is at most

$$\|\hat{x}_t - x_{t+1}\|_{H_t}^2 \leq \|\ell_t - \ell_{t-1}\|_{H_t^{-1}}^2.$$

Combining the bounds derived so far, we obtain

$$\sum_{t=1}^T S_t + \sum_{t=1}^T A_t - \sum_{t=1}^T C_t \leq O(1 + \beta\gamma^2) + 2 \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_{H_t^{-1}}^2. \quad (3.11)$$

Finally, to complete our proof of Lemma 3.7, we rely on the following, which provides a bound for the last term in (3.11) and will be proved in Appendix A.3.

**Lemma 3.8.**  $\sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_{H_t^{-1}}^2 \leq \frac{4N}{\beta} \ln \left( 1 + \frac{\beta}{4} \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_2^2 \right).$

□

Note that the lemma does not use the nonnegative sum  $\sum_{t=1}^T B_t$  but it already provides a regret bound matching that in [46]. To bound  $\sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_2^2$  in terms of  $L_2$ -deviation, we again assume that each gradient  $\nabla f_t$  satisfies the  $\lambda$ -smoothness condition defined in (4.3), and we will also use the help from the sum  $\sum_{t=1}^T B_t$ . To get a cleaner regret bound, let us assume without loss of generality that  $\lambda \geq 1$  and  $\beta \leq 1$ , because otherwise we can set  $\lambda = 1$  and  $\beta = 1$  and the inequalities in (4.3) and (3.9) still hold. Our main result of this section is the following theorem.

**Theorem 3.4.** *Suppose the loss functions are  $\beta$ -convex and their  $L_2$ -deviation is  $D_2$ , with  $\beta \leq 1$  and  $D_2 \geq 1$ . Furthermore, suppose the gradient of each loss function is  $\lambda$ -smooth, with  $\lambda \geq 1$ , and has  $L_2$ -norm at most  $\gamma$ . Then the regret of our algorithm is at most  $O(\beta\gamma^2 + (N/\beta) \ln(\lambda N D_2))$ , which becomes  $O((N/\beta) \ln D_2)$  for a large enough  $D_2$ .*

*Proof.* To bound the last term in the bound of Lemma 3.7 in terms of our deviation bound  $D_2$ , we use Lemma 3.5 in Section 3.6. Combining this with (3.10), we can upper-bound  $\sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi))$  by

$$O(1 + \beta\gamma^2) + \frac{8N}{\beta} \ln \left( 1 + \frac{\beta}{2} D_2 + \frac{\beta\lambda^2}{2} \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right) - \sum_{t=1}^T B_t. \quad (3.12)$$

To eliminate the undesirable last term inside the parenthesis above, we need the help from the sum  $\sum_{t=1}^T B_t$ , which has the following bound.

**Lemma 3.9.**  $\sum_{t=1}^T B_t \geq \frac{1}{4} \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 - O(1)$ .

*Proof.* Recall that  $B_t = \frac{1}{2} \|x_{t+1} - \hat{x}_t\|_{H_t}^2 + \frac{1}{2} \|\hat{x}_t - x_t\|_{H_t}^2$ , so we can write  $\sum_{t=1}^T B_t$  as

$$\frac{1}{2} \sum_{t=2}^{T+1} \|x_t - \hat{x}_{t-1}\|_{H_{t-1}}^2 + \frac{1}{2} \sum_{t=1}^T \|\hat{x}_t - x_t\|_{H_t}^2 \geq \frac{1}{2} \sum_{t=2}^T \|x_t - \hat{x}_{t-1}\|_{H_{t-1}}^2 + \frac{1}{2} \sum_{t=2}^T \|\hat{x}_t - x_t\|_{H_{t-1}}^2$$

since  $H_t \succeq H_{t-1}$  and  $\|x_{T+1} - \hat{x}_T\|_{H_T}^2, \|\hat{x}_1 - x_1\|_{H_1}^2 \geq 0$ . By Proposition 3.1, this is at least

$$\frac{1}{4} \sum_{t=2}^T \|\hat{x}_t - \hat{x}_{t-1}\|_{H_{t-1}}^2 \geq \frac{1}{4} \sum_{t=2}^T \|\hat{x}_t - \hat{x}_{t-1}\|_I^2 = \frac{1}{4} \sum_{t=2}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2$$

as  $H_{t-1} \succeq I$  and  $I$  is the identity matrix. Then the lemma follows as  $\|\hat{x}_2 - \hat{x}_1\|_2^2 \leq O(1)$ .  $\square$

Applying this lemma to (3.12), we obtain a regret bound of the form

$$O(1 + \beta\gamma^2) + \frac{8N}{\beta} \ln \left( 1 + \frac{\beta}{2} D_2 + \frac{\beta\lambda^2}{2} W \right) - \frac{1}{4} W$$

where  $W = \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2$ . Observe that the combination of the last two terms above become negative when  $W \geq (\lambda N D_2)^c / \beta$  for some large enough constant  $c$ , as we assume  $\beta \leq 1$  and  $\lambda, D_2 \geq 1$ . Thus, the regret bound is at most  $O(\beta\gamma^2 + (N/\beta) \ln(\lambda N D_2))$ , which completes the proof of Theorem 3.4.  $\square$

An immediate application of Theorem 3.4 is to the portfolio management problem considered in [42]. In the problem, the feasible set  $\mathcal{X}$  is the  $N$ -dimensional probability simplex and each loss function has the form  $f_t(x) = -\ln \langle v_t, x \rangle$ , with  $v_t \in [\delta, 1]^N$  for some  $\delta \in (0, 1)$ . A natural measure of deviation, extending that of [42], for such loss functions is  $D = \sum_{t=1}^T \|v_t - v_{t-1}\|_2^2$ . By applying Theorem 3.4 to this problem, we have the following result.

**Corollary 3.1.** *For the portfolio management problem described above, there is an online algorithm which achieves a regret of  $O((N/\delta^2) \ln((N/\delta)D))$ .*



*Proof.* Recall that each loss function has the form  $f_t(x) = -\ln \langle v_t, x \rangle$  for some  $v_t \in [\delta, 1]^N$  with  $\delta \in (0, 1)$ , and note that  $\nabla f_t(x) = -v_t / \langle v_t, x \rangle$ . To apply Theorem 3.4, we need to determine the parameters  $\beta, \gamma, \lambda, D_2$ .

First, by a Taylor expansion, we know that for any  $x, y \in \mathcal{X}$ , there is some  $\xi_t$  on the line between  $x$  and  $y$  such that

$$f_t(x) = f_t(y) + \langle \nabla f_t(y), x - y \rangle + \frac{1}{2 \langle v_t, \xi_t \rangle^2} (x - y)^\top v_t v_t^\top (x - y),$$

where the last term above equals

$$\frac{1}{2 \langle v_t, \xi_t \rangle^2} \langle v_t, x - y \rangle^2 = \frac{\langle v_t, y \rangle^2}{2 \langle v_t, \xi_t \rangle^2} \langle \nabla f_t(y), x - y \rangle^2 \geq \frac{\delta^2}{2} \langle \nabla f_t(y), x - y \rangle^2.$$

Thus, we can choose  $\beta = \delta^2/2$ . Next, since  $\|\nabla f_t(x)\|_2 = \|v_t\|_2 / \langle v_t, x \rangle \leq \sqrt{N}/\delta$ , we can choose  $\gamma = \sqrt{N}/\delta$ . Third, note that

$$\|\nabla f_t(x) - \nabla f_t(y)\|_2 = \left\| \frac{v_t}{\langle v_t, x \rangle} - \frac{v_t}{\langle v_t, y \rangle} \right\|_2 = \frac{\|v_t\|_2 |\langle v_t, x - y \rangle|}{\langle v_t, x \rangle \langle v_t, y \rangle},$$

which by a Cauchy-Schwarz inequality is at most

$$\frac{\|v_t\|_2^2}{\langle v_t, x \rangle \langle v_t, y \rangle} \|x - y\|_2 \leq \frac{N}{\delta^2} \|x - y\|_2.$$

Thus, we can choose  $\lambda = N/\delta^2$ . Finally, note that for any  $x \in \mathcal{X}$ ,

$$\|\nabla f_t(x) - \nabla f_{t-1}(x)\|_2 = \left\| \frac{v_t}{\langle v_t, x \rangle} - \frac{v_{t-1}}{\langle v_{t-1}, x \rangle} \right\|_2 = \left\| \frac{v_t - v_{t-1}}{\langle v_t, x \rangle} + \frac{v_{t-1} (\langle v_{t-1}, x \rangle - \langle v_t, x \rangle)}{\langle v_t, x \rangle \langle v_{t-1}, x \rangle} \right\|_2,$$

which by a triangle inequality and then a Cauchy-Schwarz inequality is at most

$$\frac{\|v_t - v_{t-1}\|_2}{\langle v_t, x \rangle} + \frac{\|v_{t-1}\|_2 |\langle v_{t-1} - v_t, x \rangle|}{\langle v_t, x \rangle \langle v_{t-1}, x \rangle} \leq \frac{\|v_t - v_{t-1}\|_2}{\langle v_t, x \rangle} + \frac{\|v_{t-1}\|_2 \|x\|_2 \|v_t - v_{t-1}\|_2}{\langle v_t, x \rangle \langle v_{t-1}, x \rangle},$$

which in turn is at most  $\left(\frac{1}{\delta} + \frac{\sqrt{N}}{\delta^2}\right) \|v_t - v_{t-1}\|_2 \leq \frac{2\sqrt{N}}{\delta^2} \|v_t - v_{t-1}\|_2$ . This implies

$$\sum_{t=1}^T \max_{x \in \mathcal{X}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_2^2 \leq \sum_{t=1}^T \left(\frac{2\sqrt{N}}{\delta^2}\right)^2 \|v_t - v_{t-1}\|_2^2 \leq \left(\frac{4N}{\delta^4}\right) D.$$

Thus, we can choose  $D_2 = (4N/\delta^4)D$ . Using these bounds in Theorem 3.4, we have the corollary.  $\square$





## Chapter 4

# Beating Bandits in Gradually Evolving Worlds

In this chapter, we will extend our work [28] which considers the full information feedback setting to the partial information feedback setting. We will start by reviewing several previous works in Section 4.1, and then present the main topic of this chapter which is our work [29] from Section 4.2 to Section 4.6.

### 4.1 Previous Works

For general convex loss functions, a regret of  $O(\sqrt{nT})$  can be achieved [68], while for strongly convex loss functions, a smaller regret of  $O(n \ln T)$  becomes possible [46]. These two results, as well as many others, considered only the worst-case scenario, in which the loss functions have no pattern or are even generated in a malicious way. However, the environments we are in may not always be adversarial, so a research direction is to identify natural patterns or properties of loss functions and to design online algorithms with smaller regrets for them. For loss functions which are linear (and can be seen as vectors), Hazan and Kale [41] considered a measure called variation, defined as  $V = \sum_{t=1}^T \|f_t - \mu\|_2^2$ , where  $\mu$  is the average of the loss functions, and they provided an algorithm achieving a regret of  $O(\sqrt{V})$ . In another work, Hazan and Kale [42] considered the online portfolio management problem [32] and achieved a logarithmic regret in terms of a similar measure. Note that loss functions with small variation can be seen as basically centered around their average, which models a stationary environment with loss functions coming from some fixed distribution. Chiang et al. [28] introduced a more general measure called

deviation which models a dynamic environment that usually evolves gradually, including examples such as weather conditions and stock markets. More precisely, Chiang et al. [28] considered not only linear functions but also convex functions, and defined the deviation as

$$D_2 = \sum_{t=1}^T \max_{x \in \mathcal{K}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_2^2, \quad (4.1)$$

using the convention that  $f_0$  is the all-0 function, where  $\nabla f_\tau(x)$  denotes the gradient of  $f_\tau$  at  $x$ . With this, they provided algorithms achieving a regret of  $O(\sqrt{D})$  for convex functions and a smaller regret of  $O(n \ln D)$  for strongly convex loss functions. Since one can show that  $D \leq O(V)$  but not the other way around [28], results with regrets in terms of  $D$  are arguably stronger than those in terms of  $V$ .

## 4.2 Beating Bandits in Gradually Evolving Worlds

The bandit setting appears much more challenging. For linear functions, Abernethy et al. [3] achieved a regret of  $O(n\sqrt{\vartheta T \ln T})$  using a somewhat involved method of  $\vartheta$ -self-concordant barriers, while Bubeck et al. [19] slightly improved the regret to  $O(n\sqrt{T \ln T})$  but with an inefficient algorithm. For general convex functions, the best regret currently achieved is  $O(T^{2/3}(\ln T)^{1/3})$  by Saha and Tewari [62], which is far from the  $O(\sqrt{nT})$  regret achieved in the full-information setting. Even worse, for strongly convex functions, there is actually an  $\Omega(\sqrt{T})$  regret lower bound in the bandit setting [47], compared to the  $O(n \ln T)$  regret upper bound achievable in the full information case. For linear functions with variation  $V$ , Hazan and Kale [43] achieved a regret of  $O(n\sqrt{\vartheta V \ln T})$ , but no such result is known for convex functions or strongly convex ones. None is known either for loss functions with small deviation, even for linear functions.

Our goal is to have bandit algorithms for loss functions with small deviation, but it turns out to be difficult as we discuss next. The standard approach for designing a bandit algorithm is to run a full-information algorithm and replace the information it needs by estimated one. For loss functions with small deviation, we would like to apply this to the full-information algorithm of [28], and what it needs in round  $t$  is the gradient of the loss function at the action it plays, denoted as  $\ell_t$ . To have a bandit algorithm, a natural attempt is to replace  $\ell_t$  by an estimator  $g_t$  using bandit information, which would achieve regrets in terms of  $\sum_t \|g_t - g_{t-1}\|_2^2$ . However, this deviation of the estimated gradients can be large even when the deviation of the true gradients is small. The reason is that in most previous works, such as [1, 3, 19, 36], the estimator  $g_t$  typically takes the form of  $c_t u_t$  for some value  $c_t \in \mathbb{R}$  and some vector  $u_t$  sampled independently in each round

from a set which spans  $\mathbb{R}^n$ . As a result,  $u_t$  and  $u_{t-1}$  are very different with high probability, and so are  $g_t$  and  $g_{t-1}$ , even when  $\ell_t$  and  $\ell_{t-1}$  are close. A possible way around this is to use estimators of a different form. For linear loss functions with small variation, with loss functions centering around their average, Hazan and Kale [43] considered estimators of the form  $g_t = c_t u_t + \tilde{\mu}_t$  where  $\tilde{\mu}_t$  is an estimator of the average, and their success relies on the fact that the average can be estimated accurately with high probability by an online algorithm. This suggests us to use estimators of the form  $g_t = c_t u_t + \tilde{g}_{t-1}$  where  $\tilde{g}_{t-1}$  is an estimator of  $\ell_{t-1}$ , but it is not clear if it is possible to have an accurate estimator for each  $\ell_{t-1}$  with high probability as each loss function may only appear once. Another issue is the choice of the exploration scheme. Take that of [36] as an example. In each round, it explores randomly in a neighborhood of diameter  $\delta$  in order to get a good estimator, but this adds to the regret a term (corresponding to the length of the estimator) which is proportional to  $1/\delta^2$  as well as a term which is proportional to  $\delta$ . Then no good choice of  $\delta$  can lead to a regret characterized by  $D$  instead of by  $T$ .

To avoid some of the difficulties, we consider the relaxed two-point bandit setting of [4], in which one can play two actions, instead of just one, in a given round and get to know their respective loss values, while their average is counted as the loss of that round. In fact, such a relaxation is necessary if we want to achieve a regret comparable to that in the full-information setting for strongly convex functions [46], according to the lower bound of [47]. In such a two-point bandit setting, Agarwal et al. [4] showed that regrets close to those in the full-information setting can indeed be achieved:  $O(n^2\sqrt{T})$  for convex functions and  $O(n^2 \ln T)$  for strongly convex functions. One may wonder if their results can be generalized to having regrets characterized by the more refined measure  $D_2$ , instead of simply by  $T$ , just as those of [28] in the full-information setting.

We answer this affirmatively. That is, we provide two-point bandit algorithms which achieve regrets close to the full-information ones in [28]. For linear functions, our regret is  $O(n^{3/2}\sqrt{D_2})$ . For convex functions, our regret is  $O(n^2\sqrt{D_2 + \ln T})$ , which becomes  $O(n^2\sqrt{D_2})$  when  $D_2 \geq \Omega(\ln T)$ . For strongly convex functions, our regret is  $O(n^2(\ln(D_2 + \ln T)))$ , which becomes  $O(n^2 \ln D_2)$  when  $D_2 \geq \Omega(\ln T)$ . Note that the dependencies on  $D_2$  in our regret bounds match those of [28] in the full-information setting. Compared to the regrets of [4] in the two-point bandit setting, we recover their results in the extreme case with  $D = \Omega(T)$ , but our regrets become much smaller when  $D$  is much smaller than  $T$ . The contrast to regrets in the (one-point) bandit setting is even sharper, as the regret of [62] for convex functions is substantially higher

than ours, while for strongly convex functions, there is actually an  $\Omega(\sqrt{D})$  lower bound<sup>1</sup> which is exponentially higher than our upper bound. Moreover, all of our algorithms are simple and efficient, which again demonstrates the power of two-point bandit algorithms. Finally, as our algorithms are based on the full-information ones of [28], we inherit their nice property that all our algorithms can be derived from one single meta algorithm and their regrets can all be analyzed in one single framework.

## Preliminaries

Let  $\mathbb{N}$  denote the set of positive integers and  $\mathbb{R}$  the set of real numbers. For  $n \in \mathbb{N}$ , let  $[n]$  denote the set  $\{1, 2, \dots, n\}$  and  $\mathbb{R}^n$  the set of  $n$ -dimensional vectors over  $\mathbb{R}$ . For vectors  $x, y \in \mathbb{R}^n$ , denote the inner product of  $x$  and  $y$  by  $\langle x, y \rangle$  and the Euclidean norm of  $x$  by  $\|x\|_2$ . For a convex set  $\mathcal{X} \subseteq \mathbb{R}^n$  and some  $y \in \mathbb{R}^n$ , let  $\Pi_{\mathcal{X}}(y) = \arg \min_{x \in \mathcal{X}} \|x - y\|_2$ , which we call the projection of  $y$  onto  $\mathcal{X}$ . Let  $\{e_1, \dots, e_n\}$  be the set of standard basis for  $\mathbb{R}^n$ .

We consider the *online convex optimization problem with two-point bandit feedback*, in which an online algorithm must play  $T$  rounds in the following way. In each round  $t$ , it plays two actions  $w_t$  and  $w'_t$  from a convex feasible set  $\mathcal{K} \subseteq \mathbb{R}^n$ , and after that, it receives the loss information  $f_t(w_t)$  and  $f_t(w'_t)$  and suffers a loss of  $\frac{1}{2}(f_t(w_t) + f_t(w'_t))$  according to some convex loss function  $f_t : \mathcal{K} \rightarrow \mathbb{R}$ . The goal is to minimize the *expected regret*:

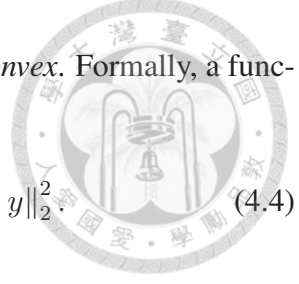
$$\mathbb{E} \left[ \sum_{t=1}^T \frac{1}{2} (f_t(w_t) + f_t(w'_t)) \right] - \min_{\pi \in \mathcal{K}} \sum_{t=1}^T f_t(\pi), \quad (4.2)$$

which is the expected total loss of the online algorithm minus that of the best offline algorithm playing a fixed action  $\pi \in \mathcal{K}$  for all  $T$  rounds, where the expectation is over the randomness of the algorithm.

As in [36], we assume that the feasible set satisfies the condition that  $r\mathbb{B} \subseteq \mathcal{K} \subseteq R\mathbb{B}$ , for some positive constants  $r \leq R$ , where  $\mathbb{B} = \{x \in \mathbb{R}^n : \|x\|_2 \leq 1\}$  is the unit ball centered at  $\mathbf{0}$ . We assume that each loss function  $f_t$  has bounded gradient  $\|\nabla f_t(x)\|_2 \leq G$  for any  $x \in \mathcal{K}$ , where  $\nabla f_t(x)$  denotes the gradient of  $f_t$  at  $x$ , and note that this implies the  $G$ -Lipschitz condition:  $|f_t(x) - f_t(y)| \leq G \|x - y\|_2$  for any  $x, y \in \mathcal{K}$ . As in previous works, we also assume that each loss function is  $\lambda$ -smooth:

$$\|\nabla f_t(x) - \nabla f_t(y)\|_2 \leq \lambda \|x - y\|_2. \quad (4.3)$$

<sup>1</sup>Such a lower bound can be easily modified from that of [47].



In addition, we will also consider loss functions which are  $H$ -strongly convex. Formally, a function  $f : \mathcal{K} \rightarrow \mathbb{R}$  is called  $H$ -strongly convex, for some  $H > 0$ , if

$$\forall x, y \in \mathcal{K} : f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{H}{2} \|x - y\|_2^2. \quad (4.4)$$

Finally, we will need the following two simple facts.

**Proposition 4.1.** (a) For  $m \in \mathbb{N}$  and  $a_1, \dots, a_m \in \mathbb{R}$ ,  $(\sum_{t=1}^m a_t)^2 \leq m \sum_{t=1}^m a_t^2$ . (b) For  $n \in \mathbb{N}$  and  $x, y \in \mathbb{R}^n$ ,  $\|x + y\|_2^2 \leq 2\|x\|_2^2 + 2\|y\|_2^2$ .

*Proof.* To prove (a), we let  $u$  be the all-1 vector and  $v$  the vector  $(a_1, \dots, a_m)$ , and by the Cauchy-Schwarz inequality, we have  $(\sum_{t=1}^m a_t)^2 = \langle u, v \rangle^2 \leq \|u\|_2^2 \|v\|_2^2 = m \sum_{t=1}^m a_t^2$ . To prove (b), simply note that  $2\|x\|_2^2 + 2\|y\|_2^2 - \|x + y\|_2^2 = \|x - y\|_2^2 \geq 0$ .  $\square$

### 4.3 Meta Algorithm

All the algorithms in the coming sections are based on the following META algorithm, given in Algorithm 4. It is in turn based on the full-information algorithm of [28], which follows the gradient descent algorithm to update  $x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta \ell_t)$  after seeing  $\ell_t$ , but plays  $\hat{x}_{t+1} = \Pi_{\mathcal{X}}(x_{t+1} - \eta \ell_t)$  instead in round  $t + 1$ , where  $\ell_t = \nabla f_t(\hat{x}_t)$ . The idea is that in the case of small deviation, one could use  $\ell_t$  as an approximation of the next  $\ell_{t+1}$ , and play  $\hat{x}_{t+1}$  which moves further in the direction of  $-\ell_t$ , and this can indeed be shown to achieve regrets in terms of the deviation  $\sum_t \|\ell_t - \ell_{t-1}\|_2^2$ . In the bandit setting, we do not have  $\ell_t$  available, and the standard approach is to feed the full-information algorithm with an estimator for  $\ell_t$  using the bandit information. An easy way to estimate  $\ell_t$  based on that of [4] is to choose a standard basis vector  $\mathbf{e}_{i_t}$  randomly, play two actions  $w_t = \hat{x}_t + \delta \mathbf{e}_{i_t}$  and  $w'_t = \hat{x}_t - \delta \mathbf{e}_{i_t}$ , compute  $v_{t,i_t} = \frac{1}{2\delta} (f_t(w_t) - f_t(w'_t))$ , and use  $\tilde{g}_t = (nv_{t,i_t})\mathbf{e}_{i_t}$  as the estimator. It can be shown that  $\mathbb{E}[\tilde{g}_t]$  is close to  $\ell_t$ . If we feed this estimator  $\tilde{g}_t$  to the algorithm of [28], we obtain regret bounds in terms of  $\sum_t \|\tilde{g}_t - \tilde{g}_{t-1}\|_2^2$ , which unfortunately may be much larger than deviation. The reason is that even when  $\|\ell_t - \ell_{t-1}\|_2^2$  is small,  $\|\tilde{g}_t - \tilde{g}_{t-1}\|_2^2 = \|(nv_{t,i_t})\mathbf{e}_{i_t} - (nv_{t-1,i_{t-1}})\mathbf{e}_{i_{t-1}}\|_2^2$  may be large if  $i_t \neq i_{t-1}$ . Thus, in our algorithm, we only follow the idea of [4] up to computing  $v_{t,i_t}$ , in our first three steps, and then we use different estimators. Our key observation is that in the regret term  $\|\ell_t - \ell_{t-1}\|_2^2$  of [28],  $\ell_t$  comes from using gradient descent to update  $x_{t+1}$ , while  $\ell_{t-1}$  comes from using it as an approximation of  $\ell_t$  to move from  $x_t$  to  $\hat{x}_t$ . Therefore, we distinguish the two different uses and compute two different estimators for them, as shown in step 4 of our algorithm, with  $g_t$  as an estimator of  $\ell_t$  which needs to have  $\mathbb{E}[g_t]$  close to  $\ell_t$ , and with  $\hat{g}_t$  as an

approximation of  $\ell_{t+1}$ . Note that  $g_t$  and  $\hat{g}_t$  computed there are obtained from  $\hat{g}_{t-1}$  by modifying only its  $i_t$ th entry, where  $\hat{g}_{\tau,i}$  denotes the  $i$ th entry of the vector  $\hat{g}_\tau$ . Then we do the the update in step 5, which can be seen as that of [28] using the estimators  $g_t$  and  $\hat{g}_t$ . The parameter  $\eta_t$  is the learning rate, which will be chosen differently for different classes of loss functions in the following sections.

---

**Algorithm 4** META algorithm

---

Let  $\mathcal{X} = (1 - \mu)\mathcal{K}$ . Let  $x_1 = \hat{x}_1 = \mathbf{0}$  and  $\hat{g}_0 = \mathbf{0}$ .

In round  $t \in [T]$ :

1: Choose  $i_t$  uniformly from  $[n]$ .

2: Play two actions  $w_t = \hat{x}_t + \delta \mathbf{e}_{i_t}$  and  $w'_t = \hat{x}_t - \delta \mathbf{e}_{i_t}$ .

3: Observe partial information  $f_t(w_t)$  and  $f_t(w'_t)$ . Let  $v_{t,i_t} = \frac{1}{2\delta} (f_t(w_t) - f_t(w'_t))$ .

4: Compute

$$g_t = n (v_{t,i_t} - \hat{g}_{t-1,i_t}) \mathbf{e}_{i_t} + \hat{g}_{t-1} \quad \text{and} \quad \hat{g}_t = (v_{t,i_t} - \hat{g}_{t-1,i_t}) \mathbf{e}_{i_t} + \hat{g}_{t-1}.$$

5: Update

$$x_{t+1} = \Pi_{\mathcal{X}}(x_t - \eta_t g_t) \quad \text{and} \quad \hat{x}_{t+1} = \Pi_{\mathcal{X}}(x_{t+1} - \eta_{t+1} \hat{g}_t).$$


---

Next, we derive a general regret bound for our algorithm. Following [4], we consider a smaller feasible set  $\mathcal{X} = (1 - \mu)\mathcal{K}$  for  $\hat{x}_t$ , with  $\mu = \delta/r$ , so that  $w_t$  and  $w'_t$  played in step 2 are feasible points in  $\mathcal{K}$ , according to Observation 3.2 of [36]. As in [4], we can choose an arbitrarily small  $\delta > 0$ , which is the advantage one can have in the two-point bandit setting<sup>2</sup>; for our purpose, any  $\delta$  such that  $\delta(\lambda GRn^2/r) \leq o(1/T)$  suffices. Similarly to [4], to bound the regret of our algorithm against  $\bar{\pi} = \arg \min_{x \in \mathcal{K}} \sum_{t=1}^T f_t(x)$ , which is the best fixed action in  $\mathcal{K}$ , it suffices to bound the regret according to the actions  $\hat{x}_t$ 's against  $\pi = \arg \min_{x \in \mathcal{X}} \sum_{t=1}^T f_t(x)$ , which is the best fixed action in  $\mathcal{X}$ . This is established by the following lemma.

**Lemma 4.1.**  $\sum_{t=1}^T \left( \frac{1}{2} (f_t(w_t) + f_t(w'_t)) - f_t(\bar{\pi}) \right) \leq \sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi)) + o(1)$ .

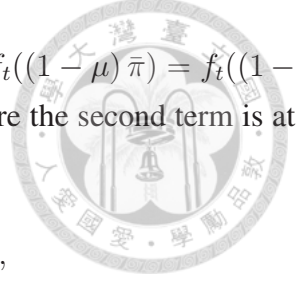
*Proof.* Observe that it suffices to prove that both  $\sum_{t=1}^T \left( \frac{1}{2} (f_t(w_t) + f_t(w'_t)) - f_t(\hat{x}_t) \right) \leq o(1)$  and  $\sum_{t=1}^T (f_t(\pi) - f_t(\bar{\pi})) \leq o(1)$  hold.

First, from the  $G$ -Lipschitz condition,  $f_t(w_t) - f_t(\hat{x}_t) \leq G \|w_t - \hat{x}_t\|_2 \leq G\delta$ , and similarly,  $f_t(w'_t) - f_t(\hat{x}_t) \leq G\delta$ . Thus

$$\sum_{t=1}^T \left( \frac{1}{2} (f_t(w_t) + f_t(w'_t)) - f_t(\hat{x}_t) \right) \leq TG\delta \leq o(1).$$

<sup>2</sup>This is because the estimator  $g_t$  now can have a bounded length, unlike the (one-point) bandit setting in which the estimator's length and consequently the regret grows proportionally to  $1/\delta^2$ .





Next, following the idea in [36], we know that as a convex function,  $f_t((1 - \mu)\bar{\pi}) = f_t((1 - \mu)\bar{\pi} + \mu\mathbf{0}) \leq (1 - \mu)f_t(\bar{\pi}) + \mu f_t(\mathbf{0}) = f_t(\bar{\pi}) + \mu(f_t(\mathbf{0}) - f_t(\bar{\pi}))$ , where the second term is at most  $\mu GR \leq o(1/T)$ . By summing over  $t$ , we have

$$\sum_{t=1}^T f_t(\pi) \leq \sum_{t=1}^T f_t((1 - \mu)\bar{\pi}) \leq \sum_{t=1}^T f_t(\bar{\pi}) + o(1),$$

where the first inequality holds since  $(1 - \mu)\bar{\pi} \in \mathcal{X}$  and  $\pi = \arg \min_{x \in \mathcal{X}} \sum_{t=1}^T f_t(x)$ . This implies that  $\sum_{t=1}^T (f_t(\pi) - f_t(\bar{\pi})) \leq o(1)$ , and we have the lemma.  $\square$

This allows us to turn our attention to bound the sum  $\sum_{t=1}^T (f_t(\hat{x}_t) - f_t(\pi))$ . Recall that  $\ell_t = \nabla f_t(\hat{x}_t)$  and let  $\ell_{t,i}$  denote the  $i$ th entry of the vector  $\ell_t$  which equals  $\nabla_i f_t(\hat{x}_t)$ , where  $\nabla_i f_t(\hat{x}_t)$  denotes the  $i$ th entry of  $\nabla f_t(\hat{x}_t)$ . Note that  $f_t(\hat{x}_t) - f_t(\pi)$  is at most  $\langle \ell_t, \hat{x}_t - \pi \rangle$  for convex  $f_t$  and at most  $\langle \ell_t, \hat{x}_t - \pi \rangle - \frac{H}{2} \|\hat{x}_t - \pi\|_2^2$  for  $H$ -strongly convex  $f_t$ . Thus, we have the following.

**Lemma 4.2.** *Let  $C_t = 0$  for convex  $f_t$  and  $C_t = \frac{H}{2} \|\hat{x}_t - \pi\|_2^2$  for  $H$ -strongly convex  $f_t$ . Then we have  $f_t(\hat{x}_t) - f_t(\pi) \leq \langle \ell_t, \hat{x}_t - \pi \rangle - C_t$ .*

Next, recall that the update rule of our algorithm can be seen as that of Chiang et al. [28] using  $g_t$  as an estimator of the gradient  $\ell_t$  and using  $\hat{g}_t$  as an approximation of  $\ell_{t+1}$ . Then we have the following from [28]; for completeness, we provide the proof in Appendix B.1.

**Lemma 4.3.** *Let  $S_t = \eta_t \|g_t - \hat{g}_{t-1}\|_2^2$ ,  $A_t = \frac{1}{2\eta_t} \|\pi - x_t\|_2^2 - \frac{1}{2\eta_t} \|\pi - x_{t+1}\|_2^2$ , and  $B_t = \frac{1}{2\eta_t} \|x_{t+1} - \hat{x}_t\|_2^2 + \frac{1}{2\eta_t} \|\hat{x}_t - x_t\|_2^2$ . Then we have  $\langle g_t, \hat{x}_t - \pi \rangle \leq S_t + A_t - B_t$ .*

To connect Lemma 4.2 with Lemma 4.3, we rely on the following lemma. Note that this justifies our use of  $g_t$  as an estimator of  $\ell_t$ .

**Lemma 4.4.**  $\mathbb{E}[\langle \ell_t, \hat{x}_t - \pi \rangle] \leq \mathbb{E}[\langle g_t, \hat{x}_t - \pi \rangle] + o(1/T)$ .

*Proof.* Let  $v_{t,i} = \frac{1}{2\delta} (f_t(\hat{x}_t + \delta \mathbf{e}_i) - f_t(\hat{x}_t - \delta \mathbf{e}_i))$  so that  $v_{t,i_t} = \frac{1}{2\delta} (f_t(w_t) - f_t(w'_t))$  and  $g_t = n(v_{t,i_t} - \hat{g}_{t-1,i_t})\mathbf{e}_{i_t} + \hat{g}_{t-1}$ .

Let us first consider any fixed choice of  $i_{[t-1]} = (i_1, \dots, i_{t-1})$ , which has  $\hat{x}_t, \ell_t = \nabla f_t(\hat{x}_t)$ , and  $\hat{g}_{t-1}$  fixed, with  $i_t$  still left random. Let  $\mathbb{E}_t[\cdot]$  denote the expectation over the random  $i_t$ , conditioned on the fixed  $i_{[t-1]}$ . Note that

$$\mathbb{E}_t[g_t] = \mathbb{E}_t[nv_{t,i_t}\mathbf{e}_{i_t}] - \mathbb{E}_t[(n\hat{g}_{t-1,i_t})\mathbf{e}_{i_t} - \hat{g}_{t-1}],$$

where the second term above is zero since  $i_t$  is chosen uniformly over  $[n]$ , and the first term above is

$$\mathbb{E}_t[nv_{t,i_t}\mathbf{e}_{i_t}] = \sum_{i=1}^n v_{t,i}\mathbf{e}_i = \sum_{i=1}^n \frac{1}{2\delta} (f_t(\hat{x}_t + \delta \mathbf{e}_i) - f_t(\hat{x}_t - \delta \mathbf{e}_i)) \mathbf{e}_i.$$

Then our goal becomes to show that the above is close to  $\ell_t = \nabla f_t(\hat{x}_t)$ , and for that it suffices to show that each  $v_{t,i}$  is close to  $\ell_{t,i} = \nabla_i f_t(\hat{x}_t)$ . Note that by Taylor's expansion,  $f_t(\hat{x}_t + \delta \mathbf{e}_i) - f_t(\hat{x}_t - \delta \mathbf{e}_i) = \langle \nabla f_t(\xi_{t,i}), 2\delta \mathbf{e}_i \rangle$  for some  $\xi_{t,i}$  on the line between  $\hat{x}_t + \delta \mathbf{e}_i$  and  $\hat{x}_t - \delta \mathbf{e}_i$ , which implies that

$$v_{t,i} = \frac{1}{2\delta} \langle \nabla f_t(\xi_{t,i}), 2\delta \mathbf{e}_i \rangle = \nabla_i f_t(\xi_{t,i}).$$

Then by the  $\lambda$ -smoothness assumption, we have

$$|\ell_{t,i} - v_{t,i}| = |\nabla_i f_t(\hat{x}_t) - \nabla_i f_t(\xi_{t,i})| \leq \|\nabla f_t(\hat{x}_t) - \nabla f_t(\xi_{t,i})\|_2 \leq \lambda \|\hat{x}_t - \xi_{t,i}\|_2 \leq \lambda \delta, \quad (4.5)$$

which implies that

$$\|\ell_t - \mathbb{E}_t [g_t]\|_2^2 = \|\ell_t - \mathbb{E}_t [nv_{t,i} \mathbf{e}_{i_t}]\|_2^2 = \sum_{i=1}^n (\ell_{t,i} - v_{t,i})^2 \leq n (\lambda \delta)^2,$$

and thus by the Cauchy-Schwarz inequality,

$$\langle \ell_t - \mathbb{E}_t [g_t], \hat{x}_t - \pi \rangle \leq \|\ell_t - \mathbb{E}_t [g_t]\|_2 \cdot \|\hat{x}_t - \pi\|_2 \leq \sqrt{n} \lambda \delta \cdot 2R \leq o(1/T).$$

Finally, let us go back to have  $i_{[t-1]} = (i_1, \dots, i_{t-1})$  randomly chosen, and let  $\mathbb{E}_{[t-1]} [\cdot]$  denote the expectation over the randomly chosen  $i_{[t-1]}$ . Then, we have the lemma as

$$\mathbb{E} [\langle \ell_t, \hat{x}_t - \pi \rangle] - \mathbb{E} [\langle g_t, \hat{x}_t - \pi \rangle] = \mathbb{E}_{[t-1]} [\langle \ell_t - \mathbb{E}_t [g_t], \hat{x}_t - \pi \rangle] \leq o(1/T).$$

□

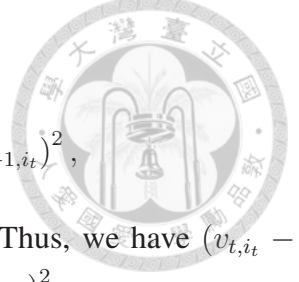
Finally, by taking expectation on the bound in Lemma 4.1 and combining the bounds in the previous three lemmas, we have the following theorem.

**Theorem 4.1.** *The expected regret of the META algorithm is*

$$\mathbb{E} \left[ \sum_{t=1}^T \left( \frac{1}{2} (f_t(w_t) + f_t(w'_t)) - f_t(\bar{\pi}) \right) \right] \leq \mathbb{E} \left[ \sum_{t=1}^T (S_t + A_t - B_t - C_t) \right] + o(1).$$

In the following sections, we will consider different classes of loss functions and instantiate the META algorithm accordingly, and then concrete regret bounds will be derived. Note that the key term in the regret bound of Theorem 4.1 is the sum of  $S_t = \eta_t \|g_t - \hat{g}_{t-1}\|_2^2$ , as it is related to the deviation according to the following lemma.

**Lemma 4.5.** *For any  $t \in [T]$ , let  $\alpha_t$  be the smallest integer such that  $0 \leq \alpha_t < t$  and  $i_\tau \neq i_t$  for any  $\alpha_t < \tau < t$ , and let  $\hat{D}_t = (\ell_{t,i_t} - \ell_{\alpha_t,i_t})^2$ . Then,  $\|g_t - \hat{g}_{t-1}\|_2^2 \leq n^2 \hat{D}_t + o(1/T)$ .*



*Proof.* Recall that

$$\|g_t - \hat{g}_{t-1}\|_2^2 = \|n(v_{t,i_t} - \hat{g}_{t-1,i_t}) \mathbf{e}_{i_t}\|_2^2 = n^2(v_{t,i_t} - \hat{g}_{t-1,i_t})^2,$$

and from the definition of  $\alpha_t$ , we know that  $\hat{g}_{t-1,i_t} = \hat{g}_{\alpha_t,i_t} = v_{\alpha_t,i_t}$ . Thus, we have  $(v_{t,i_t} - \hat{g}_{t-1,i_t})^2 = (v_{t,i_t} - v_{\alpha_t,i_t})^2$ , and we show next that it is close to  $(\ell_{t,i_t} - \ell_{\alpha_t,i_t})^2$ .

Let  $\varepsilon = |v_{t,i_t} - \ell_{t,i_t}| + |\ell_{\alpha_t,i_t} - v_{\alpha_t,i_t}|$ , and note that  $\varepsilon \leq 2\lambda\delta$  by inequality (4.5). Then we can express  $(v_{t,i_t} - v_{\alpha_t,i_t})^2$  as

$$((\ell_{t,i_t} - \ell_{\alpha_t,i_t}) + (v_{t,i_t} - \ell_{t,i_t}) + (\ell_{\alpha_t,i_t} - v_{\alpha_t,i_t}))^2 \leq (|\ell_{t,i_t} - \ell_{\alpha_t,i_t}| + \varepsilon)^2,$$

which is

$$(\ell_{t,i_t} - \ell_{\alpha_t,i_t})^2 + 2\varepsilon|\ell_{t,i_t} - \ell_{\alpha_t,i_t}| + \varepsilon^2 \leq (\ell_{t,i_t} - \ell_{\alpha_t,i_t})^2 + 8\lambda\delta G + (2\lambda\delta)^2$$

where the last two terms are both  $o(1/(Tn^2))$ . Then the lemma follows as

$$\|g_t - \hat{g}_{t-1}\|_2^2 = n^2(v_{t,i_t} - v_{\alpha_t,i_t})^2 \leq n^2(\ell_{t,i_t} - \ell_{\alpha_t,i_t})^2 + o(1/T).$$

□

Note that  $\hat{D}_t$  is related to the difference between two gradients  $t - \alpha_t$  rounds away, which is related to the deviation accumulated through those rounds. Thus, to have a small  $\hat{D}_t$ , we would like to have  $\alpha_t$  close to  $t$ . This leads us to adopt the exploration scheme of [4] but modify it to sample each  $\mathbf{e}_i$  with equal probability, so that  $t - \alpha_t$  can be shown to have a small expected value.

## 4.4 Linear Loss Functions

In this section, we consider linear loss functions. The deviation of such a sequence of loss functions according to (4.1) now turns into  $D = \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_2^2 = \sum_{t=1}^T \|f_t - f_{t-1}\|_2^2$ , where we let  $f_0 = \ell_0$  be the all-0 vector  $\mathbf{0}$ . To instantiate the META algorithm, we set  $\eta_t = \eta$  for all  $t$ , for some  $\eta$  be chosen later.

To bound the expected regret of our algorithm, we know from Theorem 4.1 that it suffices to bound

$$\mathbb{E} \left[ \sum_{t=1}^T (S_t + A_t - B_t - C_t) \right] \leq \mathbb{E} \left[ \sum_{t=1}^T S_t \right] + \mathbb{E} \left[ \sum_{t=1}^T A_t \right],$$

as  $B_t \geq 0$  and  $C_t = 0$  for linear functions. Note that with  $A_t = \frac{1}{2\eta} \|\pi - x_t\|_2^2 - \frac{1}{2\eta} \|\pi - x_{t+1}\|_2^2$ , we have by telescoping that

$$\sum_{t=1}^T A_t = \frac{1}{2\eta} \|\pi - x_1\|_2^2 - \frac{1}{2\eta} \|\pi - x_{T+1}\|_2^2 \leq \frac{R^2}{2\eta},$$

as  $\|\pi - x_1\|_2^2 \leq R^2$  and  $\|\pi - x_{T+1}\|_2^2 \geq 0$ . It remains to bound  $\mathbb{E} \left[ \sum_{t=1}^T S_t \right]$ .

We know from Lemma 4.5 that  $S_t = \eta \|g_t - \hat{g}_{t-1}\|_2^2 \leq \eta n^2 \hat{D}_t + o(1/T)$ , where  $\hat{D}_t = (\ell_{t,i_t} - \ell_{\alpha_t,i_t})^2$ , which is related to the difference between two loss functions several rounds away, instead of just between two consecutive ones as used by the definition of deviation. To bridge the gap, we need the following.

**Lemma 4.6.** *For  $t \in [T]$  and  $i \in [n]$ , let  $\rho_{t,i} = \max\{\tau_2 - \tau_1 : 0 \leq \tau_1 < t \leq \tau_2 \leq T \text{ and } i_{\tau} \neq i \text{ for any } \tau_1 < \tau < \tau_2\}$ . Then,*

$$\sum_{t=1}^T \hat{D}_t \leq \sum_{t=1}^T \sum_{i=1}^n \rho_{t,i} (\ell_{t,i} - \ell_{t-1,i})^2.$$

*Proof.* From the definition,  $\alpha_t$  is the most recent round before round  $t$  such that  $i_{\alpha_t} = i_t$ , or  $\alpha_t = 0$  if there is no such round. Then for any  $t \in [T]$  and  $i \in [n]$  such that  $i_t = i$ , we can rewrite  $\hat{D}_t = (\ell_{t,i} - \ell_{\alpha_t,i})^2$  as

$$\left( \sum_{\tau=\alpha_t+1}^t (\ell_{\tau,i} - \ell_{\tau-1,i}) \right)^2 \leq (t - \alpha_t) \sum_{\tau=\alpha_t+1}^t (\ell_{\tau,i} - \ell_{\tau-1,i})^2 = \sum_{\tau=\alpha_t+1}^t \rho_{\tau,i} (\ell_{\tau,i} - \ell_{\tau-1,i})^2,$$

where the inequality follows from Proposition 4.1(a), and the equality follows from the fact that  $\rho_{\tau,i} = (t - \alpha_t)$  for any  $\alpha_t + 1 \leq \tau \leq t$ . Therefore,

$$\sum_{t=1}^T \hat{D}_t = \sum_{i=1}^n \sum_{t:i_t=i} \hat{D}_t \leq \sum_{i=1}^n \sum_{t:i_t=i} \sum_{\tau=\alpha_t+1}^t \rho_{\tau,i} (\ell_{\tau,i} - \ell_{\tau-1,i})^2 \leq \sum_{i=1}^n \sum_{\tau=1}^T \rho_{\tau,i} (\ell_{\tau,i} - \ell_{\tau-1,i})^2,$$

where the last inequality holds since for any  $i$ , the intervals  $[\alpha_t + 1, t]$ , for  $t$  with  $i_t = i$ , have no intersection, according to the definition of  $\alpha_t$ .  $\square$

With this lemma, we can have the following, which links regret to deviation.

**Lemma 4.7.**  $\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right] \leq 2nD$ .

*Proof.* From Lemma 4.6, we know that

$$\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right] \leq \sum_{i=1}^n \sum_{t=1}^T \mathbb{E} [\rho_{t,i} (\ell_{t,i} - \ell_{t-1,i})^2] = \sum_{i=1}^n \sum_{t=1}^T \mathbb{E} [\rho_{t,i}] (\ell_{t,i} - \ell_{t-1,i})^2,$$

where the last equality follows from the fact that the gradient of a linear function does not depend on where it is taken, so each  $(\ell_{t,i} - \ell_{t-1,i})^2$  is a fixed value independent of the randomness of the expectation. It remains to bound  $\mathbb{E}[\rho_{t,i}]$ . Recall the definition of  $\rho_{t,i}$ , and suppose  $\rho_{t,i} = \tau_2 - \tau_1$  where  $0 \leq \tau_1 < t \leq \tau_2 \leq T$  and  $i_\tau \neq i$  for  $\tau_1 < \tau < \tau_2$ . Then we can write the random variable  $\rho_{t,i}$  as the sum of two random variables  $t - \tau_1$  and  $\tau_2 - t$ , and observe that both can be bounded by a geometric random variable, denoted by  $Z$ , with  $\Pr[Z = k] = (1/n)(1 - 1/n)^{k-1}$  for  $k \geq 1$  and  $\mathbb{E}[Z] = n$ ; in fact,  $t - \tau_1 = \min\{Z, t\} \leq Z$  and  $\tau_2 - t = \min\{Z - 1, T - t\} \leq Z$ . Thus,

$$\mathbb{E}[\rho_{t,i}] = \mathbb{E}[t - \tau_1] + \mathbb{E}[\tau_2 - t] \leq 2n. \quad (4.6)$$

Consequently, we have

$$\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right] \leq \sum_{i=1}^n \sum_{t=1}^T \mathbb{E}[\rho_{t,i}] (\ell_{t,i} - \ell_{t-1,i})^2 \leq 2n \sum_{i=1}^n \sum_{t=1}^T (\ell_{t,i} - \ell_{t-1,i})^2 = 2nD. \quad \square$$

Since  $\mathbb{E} \left[ \sum_{t=1}^T S_t \right] \leq \eta n^2 \mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right] + o(1) \leq 2\eta n^3 D + o(1)$ , we can conclude that the expected regret of our algorithm is at most

$$2\eta n^3 D + \frac{R^2}{2\eta} + o(1) \leq O \left( Rn^{3/2} \sqrt{D} \right),$$

by choosing  $\eta = R/\sqrt{n^3 D}$ , which gives us the following.

**Theorem 4.2.** *Suppose the loss functions are linear and have deviation  $D$ . Then the expected regret of our algorithm is at most  $O(Rn^{3/2} \sqrt{D})$ .*

## 4.5 Convex Loss Functions

In this section we consider convex loss functions. The deviation  $D$  of loss functions is measured by (4.1), which is  $\sum_{t=1}^T \max_{x \in \mathcal{K}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_2^2$ . To instantiate the META algorithm for such loss functions, we again set  $\eta_t = \eta$  for all  $t$ , for some  $\eta$  to be chosen later.

To bound the expected regret of our algorithm, we know from Theorem 4.1 that it suffices to bound

$$\mathbb{E} \left[ \sum_{t=1}^T (S_t + A_t - B_t - C_t) \right] = \mathbb{E} \left[ \sum_{t=1}^T S_t \right] + \mathbb{E} \left[ \sum_{t=1}^T A_t \right] - \mathbb{E} \left[ \sum_{t=1}^T B_t \right], \quad (4.7)$$

as  $C_t = 0$  for convex functions. As in Section 4.4, we have  $\sum_{t=1}^T A_t \leq \frac{R^2}{2\eta}$ . On the other hand, we will need the help of  $\mathbb{E} \left[ \sum_{t=1}^T B_t \right]$  here. The following lemma from [28] provides a lower bound for it; for completeness, we give the proof in Appendix B.2.

**Lemma 4.8.**  $\mathbb{E} \left[ \sum_{t=1}^T B_t \right] \geq \frac{1}{4\eta} \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] - O(1)$ .

Next, to bound  $\mathbb{E} \left[ \sum_{t=1}^T S_t \right]$ , we again turn to bound  $\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right]$ , as  $S_t \leq \eta n^2 \hat{D}_t + o(1/T)$ . Note that unlike a linear function, the gradient of a convex function now depends on where the gradient is taken, and Lemma 4.7, which works for linear functions, does not work here for convex functions. As in previous works, we assume that each  $f_t$  satisfies the  $\lambda$ -smoothness condition given in (4.3), and note that according to the discussion in Section 5 of [28], the smoothness condition is in fact necessary in order to achieve a regret bound in terms of deviation. To obtain a cleaner bound, let us assume that the parameters  $\lambda$  and  $R$  are constants, while the parameters  $T$  and  $D$  are large, with  $T, D \geq n, \lambda, R$ . Our key lemma is the following.

**Lemma 4.9.**  $\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right] \leq O(n^2 D) + O(n \ln T) \cdot \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right]$ .

*Proof.* We know from Lemma 4.6 that

$$\sum_{t=1}^T \hat{D}_t \leq \sum_{t=1}^T \sum_{i=1}^n \rho_{t,i} (\ell_{t,i} - \ell_{t-1,i})^2.$$

By definition,  $(\ell_{t,i} - \ell_{t-1,i})^2 = (\nabla_i f_t(\hat{x}_t) - \nabla_i f_{t-1}(\hat{x}_{t-1}))^2$ , which does not correspond to a term in deviation because the gradients are taken at different points. To relate it to deviation, let  $\hat{\ell}_{t-1} = \nabla f_{t-1}(\hat{x}_t)$  with  $\hat{\ell}_{t-1,i} = \nabla_i f_{t-1}(\hat{x}_t)$ , and rewrite  $(\ell_{t,i} - \ell_{t-1,i})^2$  as  $(\ell_{t,i} - \hat{\ell}_{t-1,i} + \hat{\ell}_{t-1,i} - \ell_{t-1,i})^2$ , which is at most  $2(\ell_{t,i} - \hat{\ell}_{t-1,i})^2 + 2(\hat{\ell}_{t-1,i} - \ell_{t-1,i})^2$  by Proposition 4.1(a). Then

$$\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right] \leq 2\mathbb{E} \left[ \sum_{t=1}^T \sum_{i=1}^n \rho_{t,i} (\ell_{t,i} - \hat{\ell}_{t-1,i})^2 \right] + 2\mathbb{E} \left[ \sum_{t=0}^{T-1} \sum_{i=1}^n \rho_{t,i} (\hat{\ell}_{t,i} - \ell_{t,i})^2 \right]. \quad (4.8)$$

The first expectation in (4.8) is now related to deviation since  $(\ell_{t,i} - \hat{\ell}_{t-1,i})^2 = (\nabla_i f_t(\hat{x}_t) - \nabla_i f_{t-1}(\hat{x}_t))^2$ , with the two gradients taken at the same point. However, unlike in the case of linear functions,  $(\ell_{t,i} - \hat{\ell}_{t-1,i})^2$  is now itself a random variable, which depends on the randomness of the expectation and has correlation with  $\rho_{t,i}$ . To overcome this problem, we use the upper bound

$$\left( \ell_{t,i} - \hat{\ell}_{t-1,i} \right)^2 \leq \|\nabla f_t(\hat{x}_t) - \nabla f_{t-1}(\hat{x}_t)\|_2^2 \leq D_t,$$

where  $D_t = \max_{x \in \mathcal{K}} \|\nabla f_t(x) - \nabla f_{t-1}(x)\|_2^2$  is a fixed value. Then the first expectation in (4.8) can be bounded from above by

$$\mathbb{E} \left[ \sum_{t=1}^T \sum_{i=1}^n \rho_{t,i} D_t \right] = \sum_{t=1}^T \sum_{i=1}^n \mathbb{E} [\rho_{t,i}] D_t \leq \sum_{t=1}^T \sum_{i=1}^n (2n) D_t = 2n^2 D,$$

where the first inequality uses the inequality (4.6) from Section 4.4, and the second equality uses the fact that  $D = \sum_{t=1}^T D_t$ .

The second expectation in (4.8) is slightly harder to bound. As before, the complication comes from the correlation between the two random variables  $\rho_{t,i}$  and  $(\hat{\ell}_{t,i} - \ell_{t,i})^2$ , but here we do not have a fixed upper bound for  $(\hat{\ell}_{t,i} - \ell_{t,i})^2$  which is good enough. Instead, we turn to bound  $\rho_{t,i}$ . Let  $\bar{\rho} = 4n \ln T$  and let  $Q$  denote the bad event that  $\rho_{t,i} > \bar{\rho}$  for some  $t \in [T]$  and  $i \in [n]$ , which only happens with probability

$$\Pr [Q] \leq Tn \left(1 - \frac{1}{n}\right)^{4n \ln T} \leq Tn \cdot e^{-4 \ln T} = \frac{n}{T^3}.$$

Then the second expectation in (4.8) can be expressed as

$$\Pr [\neg Q] \cdot \mathbb{E} \left[ \sum_{t=0}^{T-1} \sum_{i=1}^n \rho_{t,i} (\hat{\ell}_{t,i} - \ell_{t,i})^2 \middle| \neg Q \right] + \Pr [Q] \cdot \mathbb{E} \left[ \sum_{t=0}^{T-1} \sum_{i=1}^n \rho_{t,i} (\hat{\ell}_{t,i} - \ell_{t,i})^2 \middle| Q \right],$$

where the first term is at most

$$\Pr [\neg Q] \cdot \bar{\rho} \cdot \mathbb{E} \left[ \sum_{t=0}^{T-1} \sum_{i=1}^n (\hat{\ell}_{t,i} - \ell_{t,i})^2 \middle| \neg Q \right] \leq \bar{\rho} \cdot \mathbb{E} \left[ \sum_{t=0}^{T-1} \sum_{i=1}^n (\hat{\ell}_{t,i} - \ell_{t,i})^2 \right],$$

and the second term is at most

$$\Pr [Q] \cdot T \cdot \mathbb{E} \left[ \sum_{t=0}^{T-1} \sum_{i=1}^n (\hat{\ell}_{t,i} - \ell_{t,i})^2 \middle| Q \right].$$

Note that by the definition of  $\hat{\ell}_t$  and by the  $\lambda$ -smoothness condition,

$$\sum_{i=1}^n (\hat{\ell}_{t,i} - \ell_{t,i})^2 = \|\nabla f_t(\hat{x}_{t+1}) - \nabla f_t(\hat{x}_t)\|_2^2 \leq \lambda^2 \cdot \|\hat{x}_{t+1} - \hat{x}_t\|_2^2,$$

with  $\|\hat{x}_{t+1} - \hat{x}_t\|_2 \leq 2R$ . Thus, the second expectation in (4.8) is at most

$$\bar{\rho} \cdot \lambda^2 \cdot \mathbb{E} \left[ \sum_{t=0}^{T-1} \|\hat{x}_{t+1} - \hat{x}_t\|_2^2 \right] + \frac{n}{T^3} \cdot T^2 \lambda^2 4R^2 \leq O(n \ln T) \cdot \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] + o(1).$$

Finally, by combining the bounds for the two expectations in (4.8), we have the lemma.  $\square$

With this lemma, we obtain

$$\mathbb{E} \left[ \sum_{t=1}^T S_t \right] \leq O(\eta n^4 D) + O(\eta n^3 \ln T) \cdot \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] + o(1).$$

For some  $\eta \leq O(1/(n^2 \sqrt{D + \ln T}))$ , we can have  $O(\eta n^3 \ln T) \leq \frac{1}{4\eta}$  so that the second term above is at most  $\mathbb{E} \left[ \sum_{t=1}^T B_t \right] + O(1)$  by Lemma 4.8, and the expected regret of our algorithm, according to (4.7), can be bounded from above by

$$O \left( \eta n^4 D + \frac{1}{\eta} \right) \leq O \left( n^2 \sqrt{D + \ln T} \right).$$

As a result, we have the following theorem.



**Theorem 4.3.** *When the loss functions are convex and have deviation  $D$ , the expected regret of our algorithm is at most  $O(n^2\sqrt{D + \ln T})$ , where the hiding constant factor is a small polynomial of the constants  $\lambda$  and  $R$ .*

## 4.6 Strongly Convex Loss Functions

In this section we consider  $H$ -strongly convex functions. That is, we suppose that for some constant  $H > 0$ , each loss function  $f_t$  is  $H$ -strongly convex, so that

$$f_t(\hat{x}_t) - f_t(\pi) \leq \langle \ell_t, \hat{x}_t - \pi \rangle - \frac{H}{2} \|\pi - \hat{x}_t\|_2^2. \quad (4.9)$$

The deviation  $D$  of the loss functions is again measured by (4.1). To instantiate the META algorithm for such loss functions, now we choose the learning rate

$$\eta_t = 1 / \left( 1 + \frac{H}{2} + \frac{H}{2\gamma} \sum_{\tau=1}^{t-1} \|g_\tau - \hat{g}_{\tau-1}\|_2^2 \right),$$

with  $\gamma = 5n^2G^2$  so that  $\gamma \geq \|g_t - \hat{g}_{t-1}\|_2^2$  for any  $t \in [T]$ .<sup>3</sup> It is easy to verify that  $\eta_{t+1}$  can be computed at the end of round  $t$  for updating  $\hat{x}_{t+1}$ , as  $g_t$  and  $\hat{g}_{t-1}$  are available then.

To bound the expected regret of our algorithm, we know from Theorem 4.1 that it suffices to bound

$$\mathbb{E} \left[ \sum_{t=1}^T (S_t + A_t - B_t - C_t) \right] = \mathbb{E} \left[ \sum_{t=1}^T (S_t + A_t - C_t) \right] - \mathbb{E} \left[ \sum_{t=1}^T B_t \right],$$

where  $C_t = \frac{H}{2} \|\pi - \hat{x}_t\|_2^2$  for  $H$ -strongly convex functions. With the help of such  $C_t$ , we can reduce the regret down to only logarithmic in  $D$ . Our key lemma is the following, and we give the proof in Appendix B.3, which follows closely a similar one in [28].

**Lemma 4.10.**  $\sum_{t=1}^T (S_t + A_t - C_t) \leq \frac{4\gamma}{H} \ln \left( 1 + \frac{H}{2\gamma} \sum_{t=1}^T \|g_t - \hat{g}_{t-1}\|_2^2 \right) + O(1)$ .

From Lemma 4.5, we know that  $\|g_t - \hat{g}_{t-1}\|_2^2 \leq n^2 \hat{D}_t + o(1/T)$ . Furthermore, as in Section 4.5, we assume that each  $f_t$  satisfies the  $\lambda$ -smoothness condition given in (4.3), so we can use the upper bound for  $\mathbb{E} \left[ \sum_{t=1}^T \hat{D}_t \right]$  in Lemma 4.9. As before, to obtain a cleaner bound, we assume that the parameters  $\lambda, R, G, H$  are all constants, and the parameters  $T, D$  are large, with

<sup>3</sup>From Lemma 4.5, we know that  $\|g_t - \hat{g}_{t-1}\|_2^2 \leq n^2 \|\ell_t - \ell_{\alpha_t}\|_2^2 + o(1/T)$ , which by Proposition 4.1(b) is at most  $n^2 (2\|\ell_t\|_2^2 + 2\|\ell_{\alpha_t}\|_2^2) + o(1/T) \leq 5n^2G^2$ , as each gradient is assumed to have  $L_2$ -norm at most  $G$ .



$T, D \geq n, \lambda, R, G, H$ . Then since the logarithm function is concave,

$$\begin{aligned} \mathbb{E} \left[ \sum_{t=1}^T S_t + A_t - C_t \right] &\leq \frac{4\gamma}{H} \ln \left( 1 + \frac{H}{2\gamma} \mathbb{E} \left[ \sum_{t=1}^T \|g_t - \hat{g}_t\|_2^2 \right] \right) + O(1) \\ &\leq \frac{4\gamma}{H} \ln \left( O(n^2 D) + O(n \ln T) \cdot \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] \right), \end{aligned}$$

by Lemma 4.9. If  $\mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] \leq O(1)$ , we immediately have

$$\mathbb{E} \left[ \sum_{t=1}^T S_t + A_t - C_t \right] \leq \frac{4\gamma}{H} \ln (O(n^2 D + n \ln T)) \leq O(\gamma \ln(D + \ln T)).$$

Thus, let us assume otherwise. Then, we have

$$\begin{aligned} \mathbb{E} \left[ \sum_{t=1}^T S_t + A_t - C_t \right] &\leq \frac{4\gamma}{H} \ln \left( O(n^2 D + n \ln T) \cdot \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] \right) \\ &\leq O(\gamma \ln(D + \ln T)) + \frac{4\gamma}{H} \ln \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right]. \end{aligned}$$

The second term above may be large, and to cancel it, we rely on the following lemma, which we prove in Appendix B.4.

**Lemma 4.11.**  $\mathbb{E} \left[ \sum_{t=1}^T B_t \right] \geq \frac{1}{4} \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right] - O(1)$ .

Let  $W = \mathbb{E} \left[ \sum_{t=1}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \right]$ , and note that  $\frac{1}{4}W \geq \frac{4\gamma}{H} \ln W$  when  $W \geq (\frac{\gamma}{H})^c$  for some constant  $c$ , which implies that  $\frac{4\gamma}{H} \ln W - \frac{1}{4}W \leq O(\gamma \ln \gamma)$ . Thus, we can conclude that

$$\mathbb{E} \left[ \sum_{t=1}^T S_t + A_t - C_t - B_t \right] \leq O(\gamma \ln(D + \ln T)) + O(\gamma \ln \gamma) \leq O(n^2 \ln(D + \ln T)).$$

As a result, we have the following theorem.

**Theorem 4.4.** *When the loss functions are  $H$ -strongly convex and have deviation  $D$ , the expected regret of our algorithm is at most  $O(n^2 \ln(D + \ln T))$ , where the hiding constant factor is a small polynomial of the constants  $\lambda, R, G$ , and  $1/H$ .*





## Chapter 5

# Online Learning with Queries

In [27], we slightly modify the prediction with expert advice (PEA) problem to model an active player. We allowed the player to have the ability to query information, but with a limited budget of queries, from the loss vector before taking an action.

Our work is motivated by the following observations. In the routing-to-work problem, before deciding which route to take, a driver may first select some routes and try to collect their traffic conditions. In the stock trading problem, before deciding which stocks to trade, an investor may first select some stocks and try to do some research on their potential. In some games, before choosing the next move in a game, a player may first select some moves and try to evaluate how good they are. However, in most situations, one is unlikely to have an unlimited amount of effort or resources to collect all the information one wants; therefore, one needs to decide how to spend the limited effort or resources in an efficient way.

Our new setting consider a modified PEA problem in the following way. In each round, we give the player a  $B$ -bit budget which allows her to query  $B$  bits of her choice on the loss vector before choosing her action, where we assume that each loss value is represented by a  $K$  bits string and distinct loss values differ by at least  $\delta$ . This has the original PEA problem as a special case, when  $B = 0$ . On the other hand, when  $B = NK$ , one can achieve a zero regret since one has enough budget to figure out the whole loss vector and choose the best action in each round. The interesting case is when the value of  $B$  lies in the middle, and some questions arise. With a limited number of queries, where should one spend them? It is natural to expect that with a larger  $B$ , one can obtain more information about the loss vectors and achieve a smaller regret, but how does the regret look like as a function of the budget bound  $B$ ? We will try to answer these questions in this paper, by providing an algorithm for this problem together with lower bounds on the regret which almost match those achieved by the algorithm.

Our algorithm is based on the well-known MULTIPLICATIVE WEIGHTS UPDATE algorithm, which achieves an optimal regret for the original PEA problem. To work in our new setting, we add a step for making the queries and modify how an action is chosen in each round (while keeping the weights updated in the same multiplicative way). Instead of using the probability distribution  $p_t$  of the MULTIPLICATIVE WEIGHTS UPDATE algorithm to choose an action in each round  $t$ , we use  $p_t$  to guide our queries, and from the query result, we modify the distribution  $p_t$  by moving probabilities around among some actions. Our strategy is to use queries to find out actions with different loss values so that by moving the probabilities to actions with a smaller loss, the expected loss in that step can be reduced from that of the MULTIPLICATIVE WEIGHTS UPDATE algorithm. We start the queries on actions with larger probabilities in  $p_t$ , hoping that a larger amount of probabilities can be moved around so that a larger reduction on the loss can be achieved.

The regret which our algorithm achieves depends on the budget bound  $B$  in the following way. Before  $B$  approaches the bound  $B_1 = NK/2$ , the regret remains at

$$O\left(\sqrt{T \ln N}\right)$$

which is within the same order as that of the no-query case ( $B = 0$ ). After  $B$  passes the bound  $B_1$  but before it approaches the bound  $B_2 = NK/2 + 3K/2 - 1$ , there is a noticeable drop of the regret to

$$O\left(\sqrt{(T \ln N)/N}\right).$$

Finally, after  $B$  passes the bound  $B_2$ , the regret takes a dramatic drop to

$$(N \ln N) / \delta,$$

which is independent of  $T$ . One may see our regret bound as having two “phase transitions”, one minor and one major, at the two “critical points”  $B_1$  and  $B_2$ .

One may wonder if this interesting shape of the regret bound is just an artificial result of the particular algorithm we design. We show that it is not the case and it actually comes from the nature of the problem. We do this by providing a regret lower bound which almost matches the regret bound achieved by our algorithm. As a result, we know that unless one can query close to half of the bits in the loss vectors, these queries do not help much as they can only reduce the regret by a constant factor. Moreover, even when one can have the number of queries close to  $B_2$ , one can only reduce the regret by a factor of  $\sqrt{N}$ . On the other hand, according to our algorithm, when the budget bound exceeds  $B_2$ , the queries suddenly become extremely useful, and the regret can be made extremely small which does not even depend on  $T$ .

We consider our work as a preliminary step in the new direction of allowing queries in online learning. There are many questions that remain to be answered, and next we list three of them. First, recall that in our model, we allow an adversary to set each bit of a loss vector after receiving the corresponding query made by the online algorithm. This somewhat limits the power of the queries even though queries are allowed to be randomized. Still, we show that queries can be very powerful when their number exceeds some threshold. We would like to understand if the queries could become even more powerful when an adversary has to fix a loss vector before the online algorithm makes any query on it. Next, our algorithm is based on the specific weighted average algorithm and our regret analysis seems to rely crucially on some of its special properties. We would like to understand if it is possible to modify any existing online algorithm, instead of just the weighted average algorithm, to use queries to achieve a smaller regret. Finally, in our query model, we allow the online algorithm to obtain the information of individual bits of a loss vector, but this may not be realistic in some settings. In these settings, we would like to have more appropriate queries models which capture the kind of information one can obtain from loss vectors, and then to design algorithms which can utilize such queries to achieve small regrets.

## 5.1 Preliminaries

First, we introduce some notations which will be used in this paper. For a binary vector  $v$ , let  $\#_1(v)$  denote the number of ones in  $v$ . For a set  $\mathcal{S}$ , let  $|\mathcal{S}|$  denote the number of elements in  $\mathcal{S}$ .

Next, let us describe the original PEA problem. Suppose there is a set of  $N$  available actions and there are a total of  $T$  rounds to play. In each round  $t \in [T]$ , an online algorithm ALG chooses to play an action according to some distribution  $p_t = (p_t(1), \dots, p_t(N))$  over the  $N$  actions, where  $p_t(i)$  is the probability that ALG plays action  $i$  in round  $t$ . After that a loss vector  $f_t = (f_t(1), \dots, f_t(N)) \in [0, 1]^N$  is revealed to ALG, where  $f_t(i)$  is the loss of playing action  $i$  in round  $t$ , and ALG suffers an expected loss  $\sum_{i \in [N]} p_t(i) f_t(i)$ . The expected loss of ALG in  $T$  rounds of plays is

$$L_{\text{ALG}}^T = \sum_{t=1}^T \left( \sum_{i \in [N]} p_t(i) f_t(i) \right),$$

and we compare it with that of the best fixed action in hindsight, which is  $L_{\min}^T = \min_{i \in [N]} \sum_{t=1}^T f_t(i)$ . The goal of ALG is to minimize its regret, which is defined as

$$R_{\text{ALG}}^T = L_{\text{ALG}}^T - L_{\min}^T.$$

For this problem, there are algorithms which achieve an optimal regret of  $O(\sqrt{T \ln N})$ . Next, we describe one of them, called the MULTIPLICATIVE WEIGHTS UPDATE algorithm, denoted as  $\text{ALG}_0$ , which will be used later to build our algorithm. In each round  $t$ ,  $\text{ALG}_0$  maintains a weight vector  $w_t = (w_t(1), \dots, w_t(N))$  (initially,  $w_1 = 1^N$ ) together with the distribution  $p_t = (p_t(1), \dots, p_t(N))$  such that for each  $i \in [N]$ ,

$$p_t(i) = \frac{w_t(i)}{W_t}, \quad \text{where } W_t = \sum_{j \in [N]} w_t(j), \quad (5.1)$$

and performs the following two steps:

**Step 1:**  $\text{ALG}_0$  plays an action sampled according to the distribution  $p_t = (p_t(1), \dots, p_t(N))$ .

**Step 2:** After receiving the loss vector  $f_t$ ,  $\text{ALG}_0$  updates its weights to according to the rule that for each  $i \in [N]$ ,

$$w_{t+1}(i) = w_t(i) \cdot e^{-\eta f_t(i)}, \quad (5.2)$$

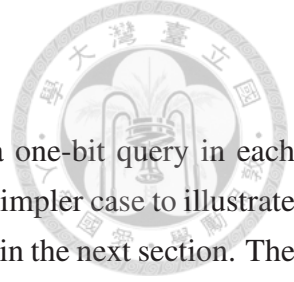
where the parameter  $\eta$  is the learning rate which one can choose.

Several ways are known for bounding the regret of this algorithm. However, for our result to work, we will use the particular one given in the following lemma, which we will prove in Section C.1. Note that it guarantees a regret of at most  $\frac{\ln N}{\eta} + T\eta$ , which is  $2\sqrt{T \ln N}$  by choosing  $\eta = \sqrt{(\ln N)/T}$ .

**Lemma 5.1.** *For any  $i_1, i_2, \dots, i_T \in [N]$ , the regret of  $\text{ALG}_0$  is at most*

$$\frac{\ln N}{\eta} + \sum_{t=1}^T \left( \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \right).$$

In this paper, we study a new setting that the online algorithm is allowed to query some information about the loss vector before choosing its action to play in each round. More precisely, in each round  $t$ , the algorithm is allowed to query  $B$  bits from the loss vector  $f_t$ . Here, we assume that each loss value  $f_t(i)$  comes from a set of at most  $2^K$  values so that we can represent each value by a  $K$ -bit string, with a smaller binary representation for a smaller loss value, and we assume furthermore that any two distinct loss values differ by at least some amount  $\delta$ . For the clarity of our presentation, we assume here that  $\delta$  (and thus  $K$ ) is a constant. We also assume that before it starts, the algorithm knows the numbers  $B$ ,  $\delta$ , and  $T$ .



## 5.2 A Special Case

In this section, we provide a simple example showing that even with a one-bit query in each round, it becomes possible to reduce the regret significantly. We use this simpler case to illustrate the basic ideas, which will be extended for the more difficult general case in the next section. The result of this section is the following.

**Theorem 5.1.** *For the special case of the online learning problem with  $N$  actions such that loss vectors are from  $\{0, 1\}^N$  and the budget bound is  $B = 1$  per round, there exists an algorithm  $\text{ALG}_1$  which achieves a regret of at most  $N \ln N$ .*

Before proving the theorem, let us first see how some partial information about a loss vector can be used to save some loss for the online algorithm. One example is that if we know  $f_t(i) > f_t(j)$  in round  $t$ , then by moving some probability  $q_i$  from playing action  $i$  to playing action  $j$ , we can save the expected loss by the amount

$$(q_i f_t(i) - q_i f_t(j)) = q_i (f_t(i) - f_t(j)) = q_i \quad (5.3)$$

since  $f_t(i), f_t(j) \in \{0, 1\}$ , which means that a larger  $q_i$  gives a larger saving. This suggests that we query the bit  $f_t(i)$  when action  $i$  is the one that we initially plan to play with the highest probability, hoping that from it we can move a large probability to some other action with a smaller loss value. Using this idea, we will design the algorithm  $\text{ALG}_1$  and analyze its regret next.

*Proof.* (of Theorem 5.1) The algorithm  $\text{ALG}_1$  is based on the weighted average algorithm  $\text{ALG}_0$  described in the previous section, but adds a query step and then modifies the distribution of actions in each round. More precisely, in round  $t$ ,  $\text{ALG}_1$  maintains a weight vector  $w_t = (w_t(1), \dots, w_t(N))$  and the distribution  $p_t = (p_t(1), \dots, p_t(N))$  defined as in (5.1), but it replaces Step 1 of  $\text{ALG}_0$  by the following:

**Step 1.1.**  $\text{ALG}_1$  queries the bit  $f_t(i_t)$  of the loss vector, where  $i_t$  is the action such that  $p_t(i_t) \geq p_t(j)$  for every  $j \in [N]$ .

**Step 1.2.**  $\text{ALG}_1$  derives the distribution  $\hat{p}_t$  from  $p_t$  by moving its probabilities in the following way:

- If  $f_t(i_t) = 0$ , then  $\text{ALG}_1$  moves all the probabilities of other actions to action  $i_t$ , so that  $\hat{p}_t(i_t) = 1$  and  $\hat{p}_t(j) = 0$  for any  $j \neq i_t$ .
- If  $f_t(i_t) = 1$ , then  $\text{ALG}_1$  moves the probability of action  $i_t$  to other actions evenly, so that  $\hat{p}_t(i_t) = 0$  and  $\hat{p}_t(j) = p_t(j) + p_t(i_t)/(N - 1)$  for any  $j \neq i_t$ .



**Step 1.3.**  $\text{ALG}_1$  plays an action sampled according to the distribution  $\hat{p}_t$ .

Next, we analyze the regret of  $\text{ALG}_1$ . We do this by comparing it with that of  $\text{ALG}_0$ , which by Lemma 5.1 is at most

$$\frac{\ln N}{\eta} + \sum_{t=1}^T \left( \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \right).$$

According to (5.3), in each round  $t$ , by moving the probabilities around, the algorithm  $\text{ALG}_1$  can reduce the loss of  $\text{ALG}_0$  by some amount  $s_t$ , such that when  $f_t(i_t) = 0$ ,

$$s_t = \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) (f_t(i) - f_t(i_t)) = \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i),$$

and when  $f_t(i_t) = 1$ ,

$$s_t \geq \sum_{i: f_t(i) \neq f_t(i_t)} \frac{p_t(i_t)}{N} (f_t(i_t) - f_t(i)) \geq \frac{1}{N} \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i),$$

since  $p_t(i_t) \geq p_t(i)$  for any  $i$ . As a result, the regret of  $\text{ALG}_1$  is at most

$$\frac{\ln N}{\eta} + \sum_{t=1}^T \left( \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) - s_t \right) \leq \frac{\ln N}{\eta} + \left( \eta - \frac{1}{N} \right) \sum_{t=1}^T \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) = N \ln N,$$

by choosing  $\eta = 1/N$ . This proves Theorem 5.1.  $\square$

### 5.3 The Main Result

In this section, we consider the general online learning problem described in Section 5.1. We will generalize the algorithm  $\text{ALG}_1$  in the previous section to the general setting, and our main result is the following theorem.

**Theorem 5.2.** *Let  $D = \max\{0, NK/2 + 3K/2 - 1 - B\}$ . Then for the general online learning problem described in Section 5.1, there exists an online algorithm  $\text{ALG}_2$ , which given a budget of  $B$  queries per round achieves a regret*

$$R_{\text{ALG}_2}^T \leq \begin{cases} (N \ln N)/\delta & \text{if } D = 0, \\ \sqrt{(8DT \ln N)/(NK)} & \text{if } D > 0, \end{cases}$$

for a large enough  $T$ .



Before proving the theorem, let us try to understand better the somewhat complicated-looking regret bound, and in particular, to see how the regret is affected by the budget bound  $B$ . First, observe that as  $B$  increases from zero, the quantity  $D$  decreases, and consequently the regret  $R_{\text{ALG}_2}^T$  decreases. This matches what one normally would expect. Next, let us take a closer look at how  $R_{\text{ALG}_2}^T$  decreases as  $B$  increases. Interestingly, the value of  $R_{\text{ALG}_2}^T$  appears to go through two “phase transitions”, one minor and one major, around  $B = NK/2$  and  $B = NK/2 + 3K/2 - 1$ , in the following sense. When  $B \leq (1 - \varepsilon)NK/2$  for any small positive constant  $\varepsilon$ ,  $R_{\text{ALG}_2}^T$  remains at

$$O\left(\sqrt{T \ln N}\right)$$

which is within the same order as that of the no-query case ( $B = 0$ ). When  $NK/2 \leq B \leq NK/2 + (1 - \varepsilon)3K/2$  for any small positive constant  $\varepsilon$ ,  $R_{\text{ALG}_2}^T$  takes a noticeable drop to

$$O\left(\sqrt{(T \ln N)/N}\right).$$

Finally, when  $B \geq NK/2 + 3K/2 - 1$ ,  $R_{\text{ALG}_2}^T$  takes a dramatic drop to

$$(N \ln N)/\delta,$$

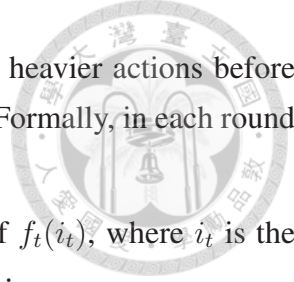
which is very small and independent of  $T$ .

Next, we proceed to prove Theorem 5.2 by providing the algorithm  $\text{ALG}_2$  and then bounding its regret in the following two subsections, respectively.

### 5.3.1 The Algorithm $\text{ALG}_2$

The algorithm  $\text{ALG}_2$  is based on the algorithm  $\text{ALG}_1$  in the previous section (which in turn is based on the weighted average algorithm  $\text{ALG}_0$ ), but it modifies Step 1.1 (for making queries) and Step 1.2 (for deriving the distribution  $\hat{p}_t$ ) in order to handle the more general case.

Consider any round  $t$ . Just as in  $\text{ALG}_1$ , we would like to use queries to find out some relationships among the losses of actions so that we can move probabilities to actions with a smaller loss. Now in the general case, which can have  $B > 1$  and  $K > 1$ , we need to decide where to spend the  $B$  bits of budget; if we spend them efficiently, we can find out more relationships. We will call an action  $i$  heavier than an action  $j$  if  $p_t(i) \geq p_t(j)$ , and we call  $i$  lighter than  $j$  otherwise. Let  $i_t$  denote the heaviest action, and our strategy is to use its loss value  $f_t(i_t)$  as a basis and to find out its relationship with  $f_t(i)$  for as many action  $i$ 's as possible. Here, we look first for a partial relationship such as  $f_t(i_t) \leq f_t(i)$  instead of an exact one such as  $f_t(i_t) = f_t(i)$  or  $f_t(i_t) < f_t(i)$ , so that we can spend as few queries as possible and still know some way to



move the probability. Following the idea in Section 5.2, we will query heavier actions before lighter ones, hoping that larger probabilities can be moved among them. Formally, in each round  $t$ ,  $\text{ALG}_2$  replaces Step 1.1 of  $\text{ALG}_1$  by the following:

**Step 1.1.** Before the  $B$ -bit budget runs out,  $\text{ALG}_2$  queries the  $K$  bits of  $f_t(i_t)$ , where  $i_t$  is the heaviest action, and then repeats the following if  $f_t(i_t) \notin \{1^K, 0^K\}$ :

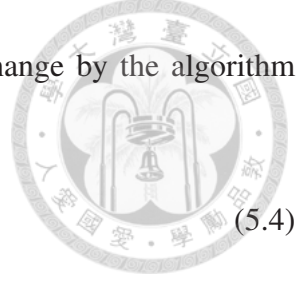
- (a)  $\text{ALG}_2$  finds the next heaviest action  $i$ .
- (b)  $\text{ALG}_2$  queries those bits of  $f_t(i)$  in those positions which have zeros in  $f_t(i_t)$  if  $f_t(i_t)$  has fewer zeros than ones, and queries the other bits of  $f_t(i)$  otherwise. (For example, if  $f_t(i_t) = 100$ , then  $\text{ALG}_2$  queries only the leftmost bit of  $f_t(i)$ .)
- (c) If any of the queried bit in  $f_t(i)$  differs from the corresponding bit in  $f_t(i_t)$ ,  $\text{ALG}_2$  queries all the remaining bits in  $f_t(i)$ .

Note that if  $f_t(i_t)$  equals  $1^K$  or  $0^K$ ,  $\text{ALG}_2$  will not make any further query on any other action  $i$  because it knows already the relationship  $f_t(i) \leq f_t(i_t)$  or  $f_t(i) \geq f_t(i_t)$ , respectively. If in Step 1.1.(b) all the queried bits match the corresponding bits in  $f_t(i_t)$ ,  $\text{ALG}_2$  knows the relationship  $f_t(i) \leq f_t(i_t)$  or  $f_t(i) \geq f_t(i_t)$  when those bits are all zeros or all ones, respectively. Otherwise (if there is a mismatch), then in Step 1.1.(c)  $\text{ALG}_2$  will query the remaining bits in  $f_t(i)$  to determine whether  $f_t(i) < f_t(i_t)$  or  $f_t(i) > f_t(i_t)$ . From such information,  $\text{ALG}_2$  can divide the  $N$  actions into six sets:  $\mathcal{I}_<$ ,  $\mathcal{I}_\leq$ ,  $\mathcal{I}_=$ ,  $\mathcal{I}_\geq$ ,  $\mathcal{I}_>$ , and  $\mathcal{I}_?$ , in the following way. If  $\text{ALG}_2$  knows  $f_t(i) < f_t(i_t)$  or  $f_t(i) > f_t(i_t)$ ,  $\text{ALG}_2$  puts action  $i$  in  $\mathcal{I}_<$  or  $\mathcal{I}_>$ , respectively. If  $\text{ALG}_2$  only knows  $f_t(i) \leq f_t(i_t)$  or  $f_t(i) \geq f_t(i_t)$ ,  $\text{ALG}_2$  puts action  $i$  in  $\mathcal{I}_\leq$  or  $\mathcal{I}_\geq$ , respectively. If  $\text{ALG}_2$  still does not know any relationship between  $f_t(i)$  and  $f_t(i_t)$  after running out the budget,  $\text{ALG}_2$  puts action  $i$  in  $\mathcal{I}_?$ . Finally, let  $\mathcal{I}_= = \{i_t\}$ .

With such information at hand,  $\text{ALG}_2$  will derive the new distribution  $\hat{p}_t$  from the distribution  $p_t$  by trying to move probabilities to actions with a smaller loss. We will say that the probabilities of some set  $\mathcal{I}$  of actions are moved to another set  $\mathcal{I}'$  of actions evenly if  $\hat{p}_t(i) = 0$  for  $i \in \mathcal{I}$  and  $\hat{p}_t(i) = p_t(i) + \sum_{j \in \mathcal{I}} p_t(j) / |\mathcal{I}'|$  for  $i \in \mathcal{I}'$ . Formally, in each round  $t$ ,  $\text{ALG}_2$  replaces Step 1.2 of  $\text{ALG}_1$  by the following:

**Step 1.2.**  $\text{ALG}_2$  derives the distribution  $\hat{p}_t$  from  $p_t$  by moving its probabilities in the following way:

- If  $\mathcal{I}_< \neq \emptyset$ ,  $\text{ALG}_2$  moves all the probabilities from  $\mathcal{I}_= \cup \mathcal{I}_\leq \cup \mathcal{I}_> \cup \mathcal{I}_\geq$  to some  $i_0 \in \mathcal{I}_<$ .
- If  $\mathcal{I}_< = \emptyset \neq \mathcal{I}_\leq$ ,  $\text{ALG}_2$  moves all the probabilities from  $\mathcal{I}_= \cup \mathcal{I}_> \cup \mathcal{I}_\geq$  to  $\mathcal{I}_\leq$  evenly.
- If  $\mathcal{I}_< = \emptyset = \mathcal{I}_\leq$ ,  $\text{ALG}_2$  moves all the probabilities from  $\mathcal{I}_> \cup \mathcal{I}_\geq$  to  $\mathcal{I}_=$ .



The other steps of the algorithm  $\text{ALG}_1$  are all inherited without change by the algorithm  $\text{ALG}_2$ , except that now  $\text{ALG}_2$  sets its learning rate as

$$\eta = \begin{cases} \delta/N & \text{if } D = 0, \\ \sqrt{(NK \ln N)/(2TD)} & \text{if } D > 0. \end{cases} \quad (5.4)$$

Next, we will show that the algorithm  $\text{ALG}_2$  indeed achieves the regret bound given in Theorem 5.2.

### 5.3.2 Proof of Theorem 5.2

We follow the analysis in Section 5.2. For each round  $t$ , let  $s_t$  denote the amount of loss  $\text{ALG}_2$  saves from that of  $\text{ALG}_0$  by moving the probabilities around (and playing according to the distribution  $\hat{p}_t$  instead of  $p_t$ ), and let

$$r_t = \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) - s_t.$$

According to Lemma 5.1 and the discussion in Section 5.2, we can bound the regret of  $\text{ALG}_2$  as

$$R_{\text{ALG}_2}^T \leq \frac{\ln N}{\eta} + \sum_{t=1}^T r_t. \quad (5.5)$$

Then we bound each  $r_t$  by the following lemma.

**Lemma 5.2.** *Let  $D = \max\{0, NK/2 + 3K/2 - 1 - B\}$ , and suppose  $\eta \leq \delta/N$ . Then for any  $t \in [T]$ ,*

$$r_t \leq \frac{2D}{NK} \eta.$$

We will prove the lemma in Subsection 5.3.3. Now let us apply it to the bound in (5.5) and consider two cases, depending on the value of  $D$ . If  $D = 0$ , by choosing  $\eta = \delta/N$ , we have

$$R_{\text{ALG}_2}^T \leq \frac{\ln N}{\eta} = \frac{N \ln N}{\delta}.$$

If  $D > 0$ , by choosing  $\eta = \sqrt{(NK \ln N)/(2TD)}$ , which is at most  $\delta/N$  for a large enough  $T$ , we have

$$R_{\text{ALG}_2}^T \leq \frac{\ln N}{\eta} + \frac{2DT}{NK} \eta = \sqrt{\frac{8DT \ln N}{NK}}.$$

This completes the proof of Theorem 5.2.



### 5.3.3 Proof of Lemma 5.2

Consider any  $t \in [T]$ , and let  $i_t$  be the heaviest action which  $\text{ALG}_2$  queries first in round  $t$ . Recall that

$$r_t = \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) - s_t,$$

where  $s_t$  is the saving of loss in round  $t$  by playing according to the probability distribution  $\hat{p}_t$  instead of  $p_t$ . Our goal is to show that

$$r_t \leq \frac{2D}{NK} \eta, \quad (5.6)$$

where  $D = \max\{0, NK/2 + 3K/2 - 1 - B\}$ . For this, we consider two cases, depending on whether or not  $|\mathcal{I}_<| = 0$ .

First, let us consider the easier case that  $|\mathcal{I}_<| \neq 0$ . In this case, the algorithm  $\text{ALG}_2$  moves the probability  $p_t(i_t)$  from the action  $i_t$  (and possibly also probabilities from other actions) to some action  $i_0 \in \mathcal{I}_<$  with  $f_t(i_0) < f_t(i_t)$ , which means that the saving of loss is

$$s_t \geq p_t(i_t)(f_t(i_t) - f_t(i_0)).$$

Since  $i_t$  is the heaviest action, we have  $p_t(i_t) \geq 1/N$ , and since distinct loss values differ by at least  $\delta$ , we have  $f_t(i_t) - f_t(i_0) \geq \delta$ . As a result, we have

$$r_t = \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) - s_t \leq \eta - \delta/N \leq 0,$$

by the assumption that  $\eta \leq \delta/N$ . Thus, the bound in (5.6) holds in this case.

Next, let us consider the more difficult case that  $|\mathcal{I}_<| = 0$ . We rely on the following claim which we will prove in Subsection 5.3.4.

**Claim 5.1.** *If  $|\mathcal{I}_<| = 0$ , then we have*

$$r_t \leq \eta \sum_{k \in \mathcal{I}_?} p_t(k) - (\delta - \eta) \sum_{j \in \mathcal{I}_>} p_t(j)$$

and  $|\mathcal{I}_?| - |\mathcal{I}_>| \leq 2D/K$ .

Recall that  $\text{ALG}_2$  queries heavier actions before lighter ones, which implies that  $p_t(k) \leq p_t(j)$  for any  $k \in \mathcal{I}_?$  and  $j \notin \mathcal{I}_?$ . Now let  $\mathcal{I}_{?1}$  be the set of the  $|\mathcal{I}_>|$  heaviest actions in  $\mathcal{I}_?$ , and let  $\mathcal{I}_{?2}$  be the set of the remaining actions in  $\mathcal{I}_?$ , so that  $\mathcal{I}_{?2}$  consists of the  $|\mathcal{I}_{?2}| \leq 2D/K$  lightest actions among all the  $N$  actions. Then we have

$$\sum_{k \in \mathcal{I}_?} p_t(k) = \sum_{k \in \mathcal{I}_{?1}} p_t(k) + \sum_{k \in \mathcal{I}_{?2}} p_t(k) \leq \sum_{j \in \mathcal{I}_>} p_t(j) + \frac{|\mathcal{I}_{?2}|}{N} \leq \sum_{j \in \mathcal{I}_>} p_t(j) + \frac{2D}{NK},$$



and substituting this into the bound in Claim 5.1, we obtain

$$\begin{aligned}
r_t &\leq \eta \left( \sum_{j \in \mathcal{I}_>} p_t(j) + \frac{2D}{NK} \right) - (\delta - \eta) \sum_{j \in \mathcal{I}_>} p_t(j) \\
&\leq \frac{2D}{NK} \eta - (\delta - 2\eta) \sum_{j \in \mathcal{I}_>} p_t(j) \\
&\leq \frac{2D}{NK} \eta,
\end{aligned}$$

since  $\delta \geq 2\eta$  by the assumption that  $\eta \leq \delta/N$ . Thus, the bound in (5.6) also holds in the case that  $|\mathcal{I}_<| = 0$ . This completes the proof of Lemma 5.2.

### 5.3.4 Proof of Claim 5.1

Assume  $|\mathcal{I}_<| = 0$ . Let us consider two cases according to the range of  $\#_1(f_t(i_t))$ , as algorithm  $\text{ALG}_2$  behaves differently in them.

**Case 1:**  $\#_1(f_t(i_t)) \leq K/2$ . In this case,  $\text{ALG}_2$  starts its queries on positions corresponding to ones in  $f_t(i_t)$ , and after finishing all the queries, each action  $i \neq i_t$  belongs to one of the three sets:  $\mathcal{I}_\geq$ ,  $\mathcal{I}_>$ , or  $\mathcal{I}_?$ . Since  $\text{ALG}_2$  moves all the probabilities from  $\mathcal{I}_\geq \cup \mathcal{I}_>$  to  $\mathcal{I}_= = \{i_t\}$ , it saves the loss of  $\text{ALG}_0$  by

$$\begin{aligned}
s_t &= \sum_{i \in \mathcal{I}_\geq: f_t(i) > f_t(i_t)} p_t(i) (f_t(i) - f_t(i_t)) + \sum_{j \in \mathcal{I}_>} p_t(j) (f_t(j) - f_t(i_t)) \\
&\geq \delta \sum_{i \in \mathcal{I}_\geq: f_t(i) > f_t(i_t)} p_t(i) + \delta \sum_{j \in \mathcal{I}_>} p_t(j),
\end{aligned} \tag{5.7}$$

since distinct loss values differ by at least  $\delta$ . On the other hand,

$$\eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \leq \eta \sum_{i \in \mathcal{I}_\geq: f_t(i) > f_t(i_t)} p_t(i) + \eta \sum_{j \in \mathcal{I}_>} p_t(j) + \eta \sum_{k \in \mathcal{I}_?} p_t(k), \tag{5.8}$$

and note that the first term in (5.8) is at most the first term in (5.7) since  $\eta \leq \delta$ . As a result,

$$\begin{aligned}
r_t &= \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) - s_t \\
&\leq \eta \sum_{j \in \mathcal{I}_>} p_t(j) + \eta \sum_{k \in \mathcal{I}_?} p_t(k) - \delta \sum_{j \in \mathcal{I}_>} p_t(j) \\
&= \eta \sum_{k \in \mathcal{I}_?} p_t(k) - (\delta - \eta) \sum_{j \in \mathcal{I}_>} p_t(j).
\end{aligned}$$



Next, let us bound  $|\mathcal{I}_?| - |\mathcal{I}_>|$ . We can assume that  $\text{ALG}_2$  does run out the budget in Step 1.1 because otherwise we have  $|\mathcal{I}_?| = 0$  and hence

$$|\mathcal{I}_?| - |\mathcal{I}_>| \leq 0 \leq \frac{2D}{K}.$$

Assuming that no budget remains and since the number of queries  $\text{ALG}_2$  spends on the actions in  $\mathcal{I}_=$ ,  $\mathcal{I}_\geq$ ,  $\mathcal{I}_>$ , and  $\mathcal{I}_?$  are at most  $K$ ,  $(K/2)|\mathcal{I}_\geq|$ ,  $K|\mathcal{I}_>|$ , and  $K - 1$ , respectively, we have

$$B \leq K + \frac{K}{2}|\mathcal{I}_\geq| + K|\mathcal{I}_>| + (K - 1).$$

On the other hand, we know that

$$N = 1 + |\mathcal{I}_\geq| + |\mathcal{I}_>| + |\mathcal{I}_?|,$$

and by combing these two bounds to remove  $|\mathcal{I}_\geq|$ , we obtain

$$|\mathcal{I}_?| - |\mathcal{I}_>| \leq \frac{2}{K} \left( \frac{NK}{2} + \frac{3K}{2} - 1 - B \right) \leq \frac{2D}{K}.$$

**Case 2:**  $\#_1(f_t(i_t)) > K/2$ . In this case,  $\text{ALG}_2$  starts its queries on positions corresponding to zeros in  $f_t(i_t)$ , and after finishing all the queries, each action  $i \neq i_t$  belongs to one of the three sets:  $\mathcal{I}_\leq$ ,  $\mathcal{I}_>$ , or  $\mathcal{I}_?$ . Since  $\text{ALG}_2$  moves all the probabilities from  $\mathcal{I}_= \cup \mathcal{I}_>$  to  $\mathcal{I}_\leq$  evenly, it saves the loss of  $\text{ALG}_0$  by

$$\begin{aligned} s_t &\geq \sum_{i \in \mathcal{I}_\leq: f_t(i) < f_t(i_t)} \frac{p_t(i_t)}{|\mathcal{I}_\leq|} (f_t(i_t) - f_t(i)) + \sum_{j \in \mathcal{I}_>} \sum_{i \in \mathcal{I}_\leq} \frac{p_t(j)}{|\mathcal{I}_\leq|} (f_t(j) - f_t(i)) \\ &\geq \frac{\delta}{N} \sum_{i \in \mathcal{I}_\leq: f_t(i) < f_t(i_t)} p_t(i) + \delta \sum_{j \in \mathcal{I}_>} p_t(j). \end{aligned} \quad (5.9)$$

On the other hand,

$$\eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \leq \eta \sum_{i \in \mathcal{I}_\leq: f_t(i) < f_t(i_t)} p_t(i) + \eta \sum_{j \in \mathcal{I}_>} p_t(j) + \eta \sum_{k \in \mathcal{I}_?} p_t(k), \quad (5.10)$$

and note that again the first term in (5.10) is at most the first term in (5.9) because  $\eta \leq \delta/N$  and  $p_t(i) \leq p_t(i_t)$  for any  $i$ . Therefore, by subtracting (5.9) from (5.10), we can obtain the same bound for  $r_t$  as in Case 1.

Furthermore, following a similar argument as in Case 1, one can show that

$$B \leq K + \frac{K}{2}|\mathcal{I}_\leq| + K|\mathcal{I}_>| + (K - 1),$$



and

$$N = 1 + |\mathcal{I}_{\leq}| + |\mathcal{I}_{>}| + |\mathcal{I}_{?}|,$$

which together give

$$|\mathcal{I}_{?}| - |\mathcal{I}_{>}| \leq \frac{2D}{K}.$$

## 5.4 Lower Bounds

In this section we provide a regret lower bound which almost matches the upper bound achieved by our algorithm. The result of this section is the following, and for the simplicity of our presentation, we assume here that  $K$  is even.

**Theorem 5.3.** *For the general online learning problem, any algorithm has a regret of at least*

- $\Omega(\sqrt{T \ln N})$ , if  $B \leq (1 - \varepsilon)NK/2$  for some constant  $\varepsilon \in (0, 1)$ .
- $\Omega(\sqrt{(T \ln N)/N})$ , if  $B \leq NK/2 - 1$ .

Here we do not attempt to prove a matching lower bound for the case which has an  $(N \ln N)/\delta$  regret upper bound in Theorem 5.2, since we consider the bound to be extremely small as it can be seen as a constant in terms of  $T$ .

The proof idea of Theorem 5.3 basically follows that for the  $\Omega(\sqrt{T \ln N})$  lower bound on the original online learning problem (see e.g. [16]). A key tool used there is a lower bound on the tail of the binomial distribution, while for our proof, we need the following bound for more general distributions.

**Lemma 5.3.** *Suppose  $\mu, \delta_1, \delta_2, \dots, \delta_n$  are constants in  $(0, 1)$ , and  $X_1, X_2, \dots, X_n$  are independent random variables such that  $\Pr[X_i = \mu - \delta_i] = \Pr[X_i = \mu + \delta_i] = 1/2$  for each  $i \in [n]$ . Let  $\lambda \geq c/\sqrt{n}$  for a large enough constant  $c$ . Then we have*

$$\Pr \left[ \sum_{i \in [n]} X_i \leq (1 - \lambda)\mu n \right] \geq e^{-O(\lambda^2 n)}.$$

We will prove the lemma in Subsection 5.4.1, and now let us proceed to prove Theorem 5.3.

*Proof.* (of Theorem 5.3)

Consider any algorithm ALG. We would like to show the existence of a sequence of  $T$  loss vectors from which ALG suffers a large regret. We prove its existence by the probabilistic method.

We will generate the  $T$  loss vectors in some probabilistic way. Let us see the the  $T$  loss vectors as an  $N \times T$  matrix, in which the entry on row  $i \in [N]$  and column  $t \in [T]$  is the

loss value of action  $i$  in round  $t$ . Here we consider  $2^K$  possible loss values in the range from 0 to  $1 - 2^{-K}$  with the natural  $K$ -bit binary representation. We would like each entry to be independently distributed and have the same expected loss  $\mu$ , for some constant  $\mu$ . This means that the expected loss of ALG in each round is exactly  $\mu$ , and the loss vectors are independent of each other. Thus a Chernoff bound shows that

$$\Pr \left[ L_{\text{ALG}}^T \leq \mu T - c\sqrt{T \ln N} \right] \leq (1/N)^{-\Omega(1)}, \quad (5.11)$$

for any constant  $c > 0$ . To make ALG spend as many queries as possible on an entry without figuring out its relationship with  $\mu$ , we choose  $\mu$  to have the binary representation  $(01)^{K/2}$ , which has alternating zeros and ones.

Then in each round, we answer queries and sample entries in the following way. For each query to some bit of an entry, we answer with the corresponding bit in  $\mu$ . After answering all the queries, some bits of the loss vector have now been fixed, and some remain free. For any entry with two adjacent bits which have not been fixed (and the corresponding two bits in  $\mu$  must have different values), we make the entry *uncertain* for ALG as follows: set those two bits of the entry to 00 or 10 with equal probability if they are 01 in  $\mu$  and set them to 01 or 11 with equal probability if they are 10 in  $\mu$ . All the other bits are then fixed as those in  $\mu$ . In this way, each entry indeed has expected value  $\mu$  and is independent from others (although some have a fixed value  $\mu$ ), and we can see each uncertain entry as a random variable satisfying the condition in Lemma 5.3. For the clarity of our presentation, we assume here that  $\mu$ , each  $\delta_i$ , and  $\delta$  are constants, but it is not hard to derive the dependence of them in our bounds. Next, we analyze the regret by considering two cases depending on the range of  $B$ .

**Case 1:**  $B \leq (1 - \varepsilon)NK/2$  for some constant  $\varepsilon \in (0, 1)$ . Since it takes at least  $K/2$  queries to an entry to avoid its uncertainty (otherwise, it must miss some adjacent bits), there must be at least  $\varepsilon N$  actions whose corresponding entries in the loss vector are left uncertain after running out the budget in each round. Thus, the total number of uncertain entries in the matrix is at least  $\varepsilon NT$ , which implies the existence of a collection  $S$  of at least  $\varepsilon N/2$  actions (rows) each of which has uncertain entries in  $\varepsilon T/2$  rounds (columns). This is because otherwise the total number of uncertain entries in the matrix is less than  $(\varepsilon N/2)T + N(\varepsilon T/2) = \varepsilon NT$ , a contradiction.

Now consider any action in  $S$  and the  $n = \varepsilon T/2$  rounds in which it has uncertain entries (fixing any additional uncertain entries to  $\mu$ ). By applying Lemma 5.3 on those rounds, one can show that the accumulated loss of that action in those rounds is at most  $(1 - \lambda)\mu n$  with probability





at least

$$e^{-O(\lambda^2 n)} \geq e^{-\ln |S|} = 1/|S|,$$

for some  $\lambda = \Theta(\sqrt{(\ln |S|)/n})$ , and when this happens, its total loss in  $T$  rounds is at most  $(1 - \lambda)\mu n + \mu(T - n) \leq \mu T - \Omega(\sqrt{T \ln N})$ . Therefore, the probability that some action in  $S$  has such a total loss is at least

$$1 - (1 - 1/|S|)^{|S|} > 1/2.$$

Finally, by combing this and the bound in (5.11) with a small enough constant  $c$ , we can conclude that  $R_{\text{ALG}}^T \geq \Omega(\sqrt{T \ln N}) - c\sqrt{T \ln N} = \Omega(\sqrt{T \ln N})$  with probability more than  $1/2 - (1/N)^{-\Omega(1)} > 0$ . This implies the existence of a sequence of  $T$  loss vectors from which the algorithm ALG suffers such a large regret.

**Case 2:**  $B \leq NK/2 - 1$ . Following the same reasoning as in Case 1, one can show that there must be at least one uncertain entry in each round (column) and thus the total number of uncertain entries in the matrix is at least  $T$ . Next, we consider the following two subcases.

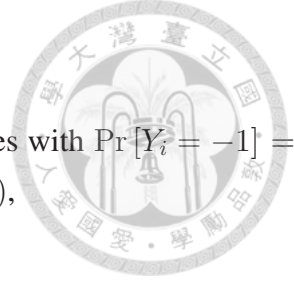
The first subcase is that some action has uncertain entries in  $n = (T \ln N)/N$  rounds. In this subcase, by applying Lemma 5.3 on those rounds, one can show that the accumulated loss of that action in those  $n$  rounds is at most  $(1 - \lambda)\mu n$  with probability at least

$$e^{-O(\lambda^2 n)} \geq \Omega(1),$$

for some  $\lambda = \Theta(\sqrt{1/n})$ , and when that happens, the total loss of that action is at most  $(1 - \lambda)\mu n + \mu(T - n) = \mu T - \Omega(\sqrt{(T \ln N)/N})$ .

The second subcase is that there exists a collection  $S$  of at least  $N/(2 \ln N)$  actions each of which has uncertain entries in  $n = T/(2N)$  rounds. In this subcase, we can choose  $\lambda = \Theta(\sqrt{(\ln |S|)/n})$  and follow the argument in Case 1 to show that some action in  $S$  has a total loss of at most  $(1 - \lambda)\mu n + \mu(T - n) = \mu T - \Omega(\sqrt{(T \ln N)/N})$  with probability more than  $1/2$ .

We claim that one of the two subcases must happen because otherwise the total number of uncertain entries in the matrix is less than  $(N/(2 \ln N))((T \ln N)/N) + N(T/(2N)) = T$ , a contradiction. Therefore, together with (5.11), we know that there exists a sequence of loss vectors such that  $R_{\text{ALG}}^T \geq \Omega(\sqrt{(T \ln N)/N})$ .  $\square$



### 5.4.1 Proof of Lemma 5.3

Let  $Y = (Y_1, Y_2, \dots, Y_n)$  be a sequence of independent random variables with  $\Pr[Y_i = -1] = \Pr[Y_i = 1] = 1/2$  for each  $i \in [n]$ , and it is known that for any  $\alpha \in (0, 1)$ ,

$$\Pr \left[ \sum_{i \in [n]} Y_i \leq -\alpha n \right] \geq 2^{-O(\alpha^2 n)}, \quad (5.12)$$

which can be shown using the Stirling formula. Note that each random variable  $X_i$  has the same distribution as  $\mu + \delta_i Y_i$ , and thus

$$\Pr \left[ \sum_{i \in [n]} X_i \leq (1 - \lambda)\mu n \right] = \Pr \left[ \sum_{i \in [n]} (\mu + \delta_i Y_i) \leq (1 - \lambda)\mu n \right] = \Pr \left[ \sum_{i \in [n]} \delta_i Y_i \leq -\lambda\mu n \right].$$

Let  $\bar{\delta} = \sum_{i \in [n]} \delta_i / n$  and let  $\gamma = \lambda\mu / \bar{\delta}$  so that  $\lambda\mu n = \gamma \bar{\delta} n$ . Let  $A$  denote the event that

$$\sum_{i \in [n]} \delta_i Y_i \leq -\gamma \bar{\delta} n,$$

and our goal now becomes to bound  $\Pr[A]$ . For this, we consider another related event, denoted as  $B$ , that

$$\sum_{i \in [n]} Y_i \leq -2\gamma n,$$

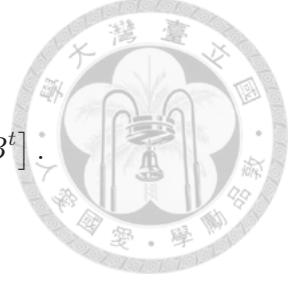
and we know from (5.12) that  $\Pr[B] \geq 2^{-O(\gamma^2 n)}$ . Observe that in the simpler case when all the  $\delta_i$ 's are the same and thus equal to  $\bar{\delta}$ , event  $B$  implies event  $A$  so that we have  $\Pr[A] \geq \Pr[B]$ . However, when these  $\delta_i$ 's are different, event  $B$  does not necessarily imply event  $A$ , so  $\Pr[A]$  may not be as large as  $\Pr[B]$  in general. Still, we will show that  $\Pr[A]$  is in fact almost as large as  $\Pr[B]$ . One approach is to use the inequality that

$$\Pr[A] \geq \Pr[A \wedge B] = \Pr[B] \cdot \Pr[A | B],$$

and show that  $\Pr[A | B]$  is large. However, it turns out to require some tedious calculation to bound  $\Pr[A | B]$ , so we take a slightly different approach.

Let us decompose the event  $B$  into several disjoint events in the following way. For any integer  $t \leq n$ , let  $B^t$  be the event that exactly  $t$  of the  $n$  random variables  $Y_1, Y_2, \dots, Y_n$  have the value 1, or equivalently

$$\sum_{i \in [n]} Y_i = 2t - n.$$



Since  $2t - n \leq -2\gamma n$  if and only if  $t \leq (1/2 - \gamma)n$ , we have

$$B = \bigvee_{t \leq (1/2 - \gamma)n} B^t \quad \text{and} \quad \Pr[B] = \sum_{t \leq (1/2 - \gamma)n} \Pr[B^t].$$

Then we use the following bound:

$$\Pr[A] \geq \Pr \left[ A \wedge \left( \bigvee_{t \leq (1/2 - \gamma)n} B^t \right) \right] = \sum_{t \leq (1/2 - \gamma)n} \Pr[A | B^t] \cdot \Pr[B^t], \quad (5.13)$$

so it suffices to show that each  $\Pr[A | B^t]$  is large. Let us fix any integer  $t \leq (1/2 - \gamma)n$ , and we next show that  $\Pr[A | B^t] \geq 1/2$  by proving that  $\Pr[\neg A | B^t] \leq 1/2$ .

Observe that the distribution of  $Y = (Y_1, Y_2, \dots, Y_n)$  conditioned on  $B^t$  is the same as that of sampling uniformly from those strings in  $\{-1, 1\}^n$  with exactly  $t$  number of 1 in them, and let  $Z^t = (Z_1^t, Z_2^t, \dots, Z_n^t)$  denote such a conditional distribution. Then we have

$$\Pr[\neg A | B^t] = \Pr \left[ \sum_{i \in [n]} \delta_i Z_i^t > -\gamma \bar{\delta} n \right], \quad (5.14)$$

which we will bound using the second moment method. Note that all the random variables  $Z_1^t, Z_2^t, \dots, Z_n^t$  have the same distribution and thus the same expected value, which we denote by  $\beta$ , and it is easy to show that  $\beta \leq -2\gamma$ . Furthermore, any two of the random variables are “negatively correlated” in the following sense.

**Claim 5.2.** For any distinct  $i, j \in [n]$ ,  $\mathbb{E}[Z_i^t Z_j^t] \leq \mathbb{E}[Z_i^t] \cdot \mathbb{E}[Z_j^t]$ .

We will prove the claim later. Now observe that the probability in (5.14) equals

$$\Pr \left[ \sum_{i \in [n]} \delta_i (Z_i^t - \beta) > -\gamma \bar{\delta} n - \beta \sum_{i \in [n]} \delta_i \right] \leq \Pr \left[ \sum_{i \in [n]} \delta_i (Z_i^t - \beta) > \gamma \bar{\delta} n \right],$$

since  $-\gamma \bar{\delta} n - \beta \sum_{i \in [n]} \delta_i \geq -\gamma \bar{\delta} n + 2\gamma \bar{\delta} n = \gamma \bar{\delta} n$ . Then the probability above is at most

$$\Pr \left[ \left( \sum_{i \in [n]} \delta_i (Z_i^t - \beta) \right)^2 > (\gamma \bar{\delta} n)^2 \right] \leq \frac{\mathbb{E} \left[ \left( \sum_{i \in [n]} \delta_i (Z_i^t - \beta) \right)^2 \right]}{(\gamma \bar{\delta} n)^2}$$

by Markov inequality, and the numerator above equals

$$\sum_{i, j \in [n]} \delta_i \delta_j \mathbb{E} \left[ (Z_i^t - \beta) (Z_j^t - \beta) \right] = \sum_{i, j \in [n]} \delta_i \delta_j (\mathbb{E}[Z_i^t Z_j^t] - \beta^2).$$

Note that when  $i \neq j$ , we have  $\mathbb{E} [Z_i^t Z_j^t] - \beta^2 \leq 0$  by Claim 5.2, and when  $i = j$ , we have  $\mathbb{E} [Z_i^t Z_j^t] - \beta^2 \leq \mathbb{E} [Z_i^t Z_j^t] = 1$ . Combining all these bounds together, we have

$$\Pr [\neg A \mid B^t] \leq \frac{\sum_{i \in [n]} \delta_i^2}{\gamma^2 \bar{\delta}^2 n^2} \leq \frac{\sum_{i \in [n]} \delta_i}{\gamma^2 \bar{\delta}^2 n^2} = \frac{1}{\gamma^2 \bar{\delta} n} \leq \frac{1}{2}, \quad (5.15)$$

since  $\gamma^2 \bar{\delta} n = \lambda^2 \mu^2 n / \bar{\delta} \geq c^2 \mu^2 / \bar{\delta} \geq 2$ , for a large enough constant  $c$ .

Finally, by substituting the bound of (5.15) into (5.13), we have

$$\Pr [A] \geq \sum_{t \leq (1/2 - \gamma)n} \left(1 - \frac{1}{2}\right) \cdot \Pr [B^t] = \frac{1}{2} \cdot \Pr [B],$$

and then by applying (5.12) to bound  $\Pr [B]$ , we obtain

$$\Pr [A] \geq 2^{-O(\gamma^2 n)} = 2^{-O(\lambda^2 n)},$$

as  $\gamma = \lambda \mu / \bar{\delta} = \Theta(\lambda)$ . Thus, to finish the proof of Lemma 5.3, it remains to prove Claim 5.2, which we do next.

*Proof.* (of Claim 5.2) Fix any distinct  $i, j \in [n]$ . Note that we have

$$\Pr [Z_j^t = 1 \mid Z_i^t = 1] = \frac{t-1}{n-1} \leq \frac{t}{n} = \Pr [Z_j^t = 1],$$

which implies that

$$\mathbb{E} [Z_j^t \mid Z_i^t = 1] = 2 \Pr [Z_j^t = 1 \mid Z_i^t = 1] - 1 \leq 2 \Pr [Z_j^t = 1] - 1 = \mathbb{E} [Z_j^t],$$

and we also have

$$\Pr [Z_j^t = -1 \mid Z_i^t = -1] = \frac{(n-t)-1}{n-1} \leq \frac{n-t}{n} = \Pr [Z_j^t = -1],$$

which implies that

$$\mathbb{E} [Z_j^t \mid Z_i^t = -1] = 1 - 2 \Pr [Z_j^t = -1 \mid Z_i^t = -1] \geq 1 - 2 \Pr [Z_j^t = -1] = \mathbb{E} [Z_j^t].$$

As a result, we have

$$\begin{aligned} \mathbb{E} [Z_i^t Z_j^t] &= \Pr [Z_i^t = 1] \cdot \mathbb{E} [Z_j^t \mid Z_i^t = 1] - \Pr [Z_i^t = -1] \cdot \mathbb{E} [Z_j^t \mid Z_i^t = -1] \\ &\leq \Pr [Z_i^t = 1] \cdot \mathbb{E} [Z_j^t] - \Pr [Z_i^t = -1] \cdot \mathbb{E} [Z_j^t] \\ &= \mathbb{E} [Z_i^t] \cdot \mathbb{E} [Z_j^t]. \end{aligned}$$

□



## Chapter 6

# Pseudo-Reward for Linear Contextual Bandits

In this chapter we study a variant of the multi-armed bandit (MAB) problem which is formulated to model a class of realistic problems in which the player in the beginning of each round is able to access information about the environment before taking an action. We will start with an introduction in Section 6.1. In Section 6.2, we will review the LINUCB) algorithm [31] which is closely related to our main algorithm. Finally in Section 6.3 we will introduce the main algorithm, LINPRUCB, and analyze its regret.

### 6.1 Introduction

We study the *contextual bandit problem* [65], or known as the  $k$ -armed bandit problem with context [11], in which a learner needs to interact iteratively with the external environment as follows. During each iteration, the learner is asked to select an action from a set of candidate actions, and in the bandit setting, only the reward for the selected action is revealed to the learner as feedback. Different from the traditional bandit problem [51], the major distinction is that in the contextual bandit problem, the learner can access additional information, called *context*, about the environment before making selections. The context is used to encode the state of the environment and helps the learner to make better selections.

For instance, consider an online advertising system operated by a contextual bandit algorithm. For each user visit (iteration), the advertising algorithm (contextual bandit algorithm) is provided with the known properties about the user (context), and an advertisement (action) from a pool

of relevant advertisements (set of candidate actions) is selected and displayed to that user. The company's income (reward) can be calculated based on whether the user clicked the selected advertisement and the value of the advertisement itself.

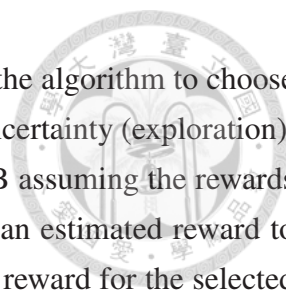
From the previous example we know that the contextual bandit problem is more realistic than the traditional bandit problems for many applications, such as advertising, recommendations, and other Web application [53, 54, 61]. Therefore, the learner would like to have an algorithm to maximize the cumulative reward. The performance of an algorithm is measured by its *regret*, which is the difference between the cumulative reward of the best strategy in hindsight and that of the algorithm. The goal of the algorithm is to minimize the regret, since smaller regret implies larger cumulative reward.

Since only the reward of chosen action is revealed to the algorithm, the major challenge in the contextual bandit problem is to strike a balance between exploitation and exploration. This is because exploitation selects the seamlessly most rewarding actions without gathering information on other uncertain and potentially more rewarding actions, while on the other hand, exploration selects the more uncertain actions at the expense of lowering the short-term reward. Many remarkable algorithms are based on finding clever ways to resolve this challenge.

### 6.1.1 Previous Works

GREEDY is the most naive algorithm that always exploits but can be viewed as a base for other successful contextual bandit algorithms. Another group of algorithms use randomized approaches to balance exploration and exploitation. The first one is  $\epsilon$ -GREEDY [67] developed from reinforcement learning that uses a tiny fraction, say  $\epsilon$ , to explore an action uniformly at random. The second one is THOMPSONSAMPLING [26]. Different from  $\epsilon$ -GREEDY whose exploration is done randomly during action selection, THOMPSONSAMPLING embeds the intent of exploration during the model updating stage.

Moreover, there are other sophisticated algorithms focus on more strategic ways of implementing exploration in order to reach a better balance between exploration and exploitation. The Upper Confidence Bound (UCB) studied in Auer [9] is arguably one of the most popular families of contextual bandit algorithms. As the rewards are generated by an unknown stochastic process, UCB cleverly uses confidence intervals as uncertainty measurements to balance exploration and exploitation. The uncertainty term represents the amount of information that has been received for the candidate action and decreases as more information is gathered during the iterations. UCB computes the estimated reward and the uncertainty term for each action, and uses the



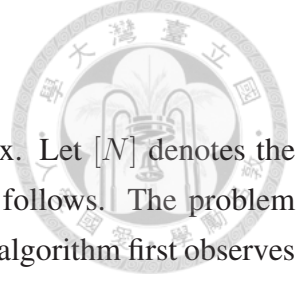
sum of these two values to guide the selection. The combination allows the algorithm to choose either an action with high estimated reward (exploitation) or with high uncertainty (exploration). Linear UCB (LINUCB) proposed in Chu et al. [31] is a member of UCB assuming the rewards are generated by a linear model, and it takes the confidence interval of an estimated reward to form the uncertainty term that is used during action selection. When the reward for the selected action is revealed, LINUCB updates only the linear model associated with that action with the new example.

### 6.1.2 Our Approach

Despite LINUCB’s success, two concerns remain unaddressed. First, LINUCB only uses the received reward and the context to update the internal model associated with the selected action. This makes LINUCB slow at gathering examples during the early iterations. Second, the selection time for LINUCB, which includes the calculation of the uncertainty term after a context is given, can be long. The long selection time may not be acceptable for certain applications. For instance, in online advertisement scenarios, a shorter selection time is highly preferred. In this paper, we propose a novel algorithm, LINPRUCB, that improves LINUCB by addressing the two concerns above.

Motivating by the better regret bound in the full information setting, we develop a new scheme called *pseudo-reward* to make a guess at each unseen reward. Furthermore, we use a forgetting mechanism to handle imprecise estimated rewards computed in the early iterations. Combining the pseudo-rewards and the forgetting mechanism, we first introduce the main algorithm, LINPRUCB, which is an extension of the LINUCB algorithm. Our new algorithm has two advantages that can be used to resolve the previous two concerns. First, LINPRUCB is able to update the linear models associated with all of the actions in each iteration and result in faster reward-gathering in the earlier iterations as compared to LINUCB. Second, it computes the uncertainty of each action which will be used in the exploration in the model updating stage, allowing a flexibility of fast action selection. To our best knowledge, these interesting ideas of guessing unseen rewards and moving exploration in action selection stage to exploration in model stage has not been seriously studied before this work. We show that LINPRUCB achieves a  $\tilde{O}(\sqrt{dT})^1$  regret. Besides, LINPRUCB also shows competitive performance in real world data [30].

<sup>1</sup>The notion  $\tilde{O}$  hides constants and logarithmic factor of the total time  $T$ , the number of actions  $K$  and a failure rate  $\delta$ .



## 6.2 Preliminaries

We use boldface  $\mathbf{w}$  to denote a column vector and  $\mathbf{Q}$  to denote a matrix. Let  $[N]$  denotes the set  $\{1, 2, \dots, N\}$ . The contextual bandit problem can be modeled as follows. The problem consists of  $T$  iterations of decision making. In each iteration  $t$ , a learning algorithm first observes a context  $\mathbf{x}_t \in \mathbb{R}^d$  from the environment, and has to select an action  $a_t$  from the action set  $[K]$ . After choosing an action, the algorithm receives the corresponding reward  $r_{t,a_t} \in \mathbb{R}$  from the environment. The algorithm will learn from this feedback information in order to make better decisions in the future iterations. In this paper we assume the reward is generated by the following linear model. That is, for each action  $a \in [K]$ , there is an unknown vector  $\mathbf{u}_a \in \mathbb{R}^d$  with  $\|\mathbf{u}_a\|_2 = 1$  such that for any context  $\mathbf{x}_t$ , the reward  $r_{t,a}$  is a random variable with expectation  $\mathbb{E}[r_{t,a}] = \mathbf{u}_a^\top \mathbf{x}_t$ . For total  $T$  iterations, the performance of an algorithm is measured by the regret, which is defined as

$$\text{regret}(T) = \sum_{t=1}^T r_{t,a_t^*} - \sum_{t=1}^T r_{t,a_t},$$

where  $a_t^* = \arg \max_{a \in [K]} \mathbf{u}_a^\top \mathbf{x}_t$  is the action selected by the optimal strategy in hindsight.

### 6.2.1 The LINUCB Algorithm

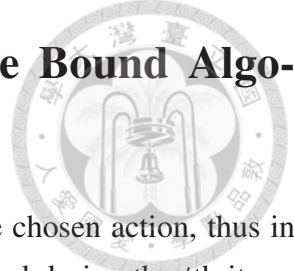
The LINUCB [53] is one of the standard approaches to solve the contextual bandit problem. The idea of LINUCB is to exploit according to the estimated reward and to explore according to the uncertainty. To compute the estimated reward, the algorithm maintains  $\mathbf{w}_{t,a}$  as a current estimation of  $\mathbf{u}_a$ . The estimation

$$\mathbf{w}_{t,a} = (\lambda \mathbf{I}_d + \mathbf{X}_{t,a}^\top \mathbf{X}_{t,a})^{-1} (\mathbf{X}_{t,a}^\top \mathbf{r}_{t,a}) \quad (6.1)$$

is computed by the ridge regression, where  $\mathbf{X}_{t,a}$  is the matrix that contains all rows  $\mathbf{x}_{\tau,a}^\top$  satisfying  $1 \leq \tau \leq t$  and  $a_\tau = a$ , and  $\mathbf{r}_{t,a}$  is the vector that contains all the corresponding  $r_{\tau,a}$  as the components. With  $\mathbf{w}_{t,a}$  in hand, the algorithm computes the estimated reward of the action  $a$  with respect to the context  $\mathbf{x}_t$  by  $\mathbf{w}_{t,a}^\top \mathbf{x}_t$ . Next, LINUCB computes  $\alpha c_{t,a} = \alpha \sqrt{\mathbf{x}_t^\top \mathbf{A}_{t,a}^{-1} \mathbf{x}_t}$ , where  $\mathbf{A}_{t,a} = (\lambda \mathbf{I}_d + \mathbf{X}_{t,a}^\top \mathbf{X}_{t,a})$  and  $\alpha > 0$ . It can be shown that  $\alpha c_{t,a}$  bounds the confidence interval of  $\mathbf{w}_{t,a}^\top \mathbf{x}_t$  from above, thus can be used to measure the uncertainty. In iteration  $t$ , LINUCB selects an action by the following rule

$$a_t = \arg \max_{a \in [K]} (\mathbf{w}_{t,a}^\top \mathbf{x}_t + \alpha c_{t,a}). \quad (6.2)$$





## 6.3 Linear Pseudo-Reward Upper Confidence Bound Algorithm

In the bandit setting, the only feedback information is the reward of the chosen action, thus in LINUCB only the weight vector  $\mathbf{w}_{t,a_t}$  of the selected action  $a_t$  is updated during the  $t$ th iteration. On the other hand, from the analysis of the LINUCB algorithm, we notice that LINUCB can achieve better regret if LINUCB is given the corresponding reward of each action. This observation inspires us to think of ways to guess the unseen rewards, and use them to design a new algorithm that has better performance. In this approach, we will face two major challenges. The first one is how to estimate the unseen rewards. Further, when we collect more data as time goes by, the estimated rewards in the early iterations may become irrelevant or even disturbing to the learning process. Thus the second challenge is how do we handle the unwanted estimations. In the following we will introduce two schemes to resolve the challenges and describe the main algorithm.

### 6.3.1 The Pseudo-Reward Scheme

We denote our guess at the unseen reward  $r_{t,a}$  by  $p_{t,a}$  and call it the *pseudo-reward*. For any action  $a$  and context  $\mathbf{x}_t$ , we defined the corresponding pseudo-reward as

$$p_{t,a} = \mathbf{w}_{t,a}^\top \mathbf{x}_t + \beta \sqrt{\mathbf{x}_t^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x}_t}. \quad (6.3)$$

The first term  $\mathbf{w}_{t,a}^\top \mathbf{x}_t$  is the estimated reward computed by  $\mathbf{w}_{t,a}$ , and the second one is an uncertainty computed by a carefully designed matrix  $\hat{\mathbf{Q}}_{t,a}$  which will be specified latter in (6.4) and a parameter  $\beta > 0$ . Note that if one naively define the pseudo-reward as  $\mathbf{w}_{t,a}^\top \mathbf{x}_t$ , the model  $\mathbf{w}_{t,a}$  is trained on its own output and has no improvement. In contrast, including the uncertainty  $\beta \sqrt{\mathbf{x}_t^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x}_t}$ , our pseudo-reward  $p_{t,a}$  enjoys the following properties. First, the uncertainty allows us to trade variance due to few examples in LINUCB with bias which is caused by the pseudo-reward in our algorithm. Second, including the uncertainty tilts the model towards the unseen actions and hence guides future exploration better. Further, the uncertainty term in the pseudo-reward provides flexibility to explore not only during action selection but also model updating.

With pseudo-rewards in hand, we can use the matrix  $\bar{\mathbf{X}}_{t,a}$  to collect those instances with no feedback as its rows, assuming there are  $\ell_{t,a}$  of them, and let  $\mathbf{p}_{t,a}$  be the column vector containing the corresponding pseudo-rewards as its entries. Recall that the matrix  $\mathbf{X}_{t,a}$  contains those

instances with feedback as its rows and the vector  $\mathbf{r}_{t,a}$  contains the corresponding rewards as its entries. Thus, we can update the model by

$$\mathbf{w}_{t+1,a} = \arg \min_{\mathbf{w} \in \mathbb{R}^d} \left( \lambda \|\mathbf{w}\|^2 + \|\mathbf{X}_{t,a} \mathbf{w} - \mathbf{r}_{t,a}\|^2 + \|\bar{\mathbf{X}}_{t,a} \mathbf{w} - \mathbf{p}_{t,a}\|^2 \right).$$



### 6.3.2 Handling Biased Terms

Note that the pseudo-rewards remain in the regression model permanently, even though some of them, especially the earlier ones, may be rather inaccurate. That is, the beginning pseudo-rewards, which promote exploration, may become irrelevant or even misleading later on, so we may not want them in later iterations to have the same effect as the more recent, and perhaps more accurate, pseudo-rewards.

In order to make the pseudo-rewards working properly, we use a forgetting mechanism to deal with the above situation. It puts emphases on more recent pseudo-rewards and their contexts than older ones, by using a forgetting parameter  $\eta \in [0, 1]$  to control how fast it forgets previous pseudo-rewards; the smaller the parameter  $\eta$  is, the faster it forgets. Let  $\hat{\mathbf{X}}_{t,a}$  be the  $\ell_{t,a} \times d$  matrix with its  $i$ th row being that of  $\bar{\mathbf{X}}_{t,a}$  multiplied by the factor  $\eta^{\ell_{t,a}-i}$ ; similarly, let  $\hat{\mathbf{p}}_{t,a}$  be the  $\ell_{t,a}$ -dimensional vector with its  $i$ th entry being that of  $\mathbf{p}_{t,a}$  multiplied by  $\eta^{\ell_{t,a}-i}$ . Then our update becomes

$$\mathbf{w}_{t+1,a} = \arg \min_{\mathbf{w} \in \mathbb{R}^d} \left( \lambda \|\mathbf{w}\|^2 + \|\mathbf{X}_{t,a} \mathbf{w} - \mathbf{r}_{t,a}\|^2 + \|\hat{\mathbf{X}}_{t,a} \mathbf{w} - \hat{\mathbf{p}}_{t,a}\|^2 \right),$$

which has the solution

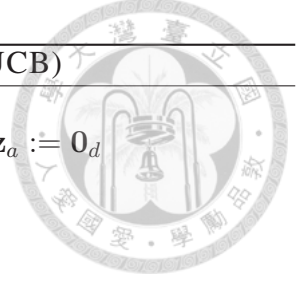
$$\mathbf{w}_{t+1,a} = \hat{\mathbf{Q}}_{t,a}^{-1} \left( \mathbf{X}_{t,a} \mathbf{r}_{t,a} + \hat{\mathbf{X}}_{t,a} \hat{\mathbf{p}}_{t,a} \right), \text{ where } \hat{\mathbf{Q}}_{t,a} = \lambda I_d + \mathbf{X}_{t,a}^\top \mathbf{X}_{t,a} + \hat{\mathbf{X}}_{t,a}^\top \hat{\mathbf{X}}_{t,a}. \quad (6.4)$$

### 6.3.3 The LINPRUCB Algorithm

We can use the pseudo-rewards and the forgetting mechanism mentioned above to upgrade the LINUCB to our main algorithm, the LINPRUCB algorithm. Similar to LINUCB, LINPRUCB chooses an action in iteration  $t$  by

$$a_t = \arg \max_{a \in [K]} \left( \mathbf{w}_{t,a}^\top \mathbf{x}_t + \alpha \hat{c}_{t,a} \right),$$

where  $\hat{c}_{t,a} = \sqrt{\mathbf{x}_t^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x}_t}$  and  $\alpha > 0$  is a parameter. The major distinction is the loop in Step 7 that updates the weight  $\mathbf{w}_{t+1,a}$  for each action. For the selected action, its corresponding context vector and actual reward is updated in Step 9 as LINUCB does. For actions whose rewards are




---

**Algorithm 5** Linear Pseudo-Reward Upper Confidence Bound (LINPRUCB)
 

---

```

1: inputs:  $\alpha, \beta, \eta > 0$ 
2: initialize:  $\mathbf{w}_{1,a} := \mathbf{0}_d$ ,  $\hat{\mathbf{Q}}_{1,a}^{-1} := \mathbf{I}_d$ ,  $\hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a := \mathbf{X}^\top \mathbf{X}_a := \mathbf{0}_{d \times d}$ ,  $\hat{\mathbf{z}}_a := \mathbf{z}_a := \mathbf{0}_d$ 
3: for  $t = 1$  to  $T$  do
4:   observe  $\mathbf{x}_t$ 
5:   select  $a_t = \arg \max_{a \in [K]} \mathbf{w}_{t,a}^\top \mathbf{x}_t + \alpha \sqrt{(\mathbf{x}_t^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x}_t)}$ 
6:   receive reward  $r_{t,a_t}$ 
7:   for  $a \in [K]$  do
8:     if  $a = a_t$  then
9:        $\mathbf{X}^\top \mathbf{X}_a := \mathbf{X}^\top \mathbf{X}_a + \mathbf{x}_t \mathbf{x}_t^\top$ 
10:       $\mathbf{z}_a := \mathbf{z}_a + \mathbf{x}_t r_{t,a_t}$ 
11:     else
12:        $p_{t,a} := \mathbf{w}_{t,a}^\top \mathbf{x}_t + \beta \sqrt{(\mathbf{x}_t^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x}_t)}$ 
13:        $\hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a := \eta \hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a + \mathbf{x}_t \mathbf{x}_t^\top$ 
14:        $\hat{\mathbf{z}}_a := \eta \hat{\mathbf{z}}_a + \mathbf{x}_t p_{t,a}$ 
15:     end if
16:      $\hat{\mathbf{Q}}_{t+1,a} := \mathbf{I}_d + \mathbf{X}^\top \mathbf{X}_a + \hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a$ 
17:      $\mathbf{w}_{t+1,a} := \hat{\mathbf{Q}}_{t+1,a}^{-1} (\mathbf{z}_a + \hat{\mathbf{z}}_a)$ 
18:   end for
19: end for

```

---

unknown, LINUCB computes the pseudo-rewards of unseen rewards in Step 11, and then the forgetting mechanism kicks in. Finally in Step 14 computes the new weight  $\mathbf{w}_{t+1,a}$  shown in (6.4) for the next iteration.

It is ideal to provide a theoretical regret bound to justify the performance of our algorithm. However, our LINPRUCB inherits a property of LINUCB that it is difficult to analyze due to the dependency between the context vectors and rewards. Similar to LINUCB of Chu et al. [31] and LINREL of Auer [9], in the next section, we are going to construct a variant of the LINPRUCB algorithm, called SUPLINPRUCB, and analyze its regret as a remedy.

### 6.3.4 The SupLinPRUCB Algorithm

The SUPLINREL algorithm and the subroutine LINREL introduced by [9] provides a clever way to handle the dependency issue. To analyze the regret of LINUCB, [31] follows the idea and the analysis of [9] to construct the SUPLINUCB algorithm, the BASELINUCB subroutine, and the analysis showing their SUPLINUCB algorithm achieves a  $\tilde{O}(\sqrt{dT})$  regret with high probability. Therefore in this subsection, we also follow the similar idea of [9] to construct an algorithm called SUPLINPRUCB and a subroutine called BASELINPRUCB.




---

**Algorithm 6** SUPLINPRUCB
 

---

1: initialize:  $S := \ln T$ ,  $\Psi_1^s := \emptyset$  for all  $s \in [S]$   
 2: **for**  $t = 1$  **to**  $T$  **do**  
 3:   Observe  $\mathbf{x}_t$ .  
 4:    $s := 1$  and  $\hat{A}_1 := [K]$   
 5:   **repeat**  
 6:     Use BASELINPRUCB with  $\Psi_t^s$  and  $\mathbf{x}_t$  to compute  $\text{ucb}_{t,a}^s$  and  $\text{width}_{t,a}^s$  for all  $a \in \hat{A}_s$ .  
 7:     **if**  $\text{width}_{t,a}^s > 2^{-s}$  for some  $a \in \hat{A}_s$  **then**  
 8:       Choose action  $a_t := a$  and update:  
        $\Psi_{t+1}^s := \Psi_t^s \cup \{t\}$  and  $\Psi_{t+1}^{s'} := \Psi_t^{s'}$  for all  $s' \neq s$ .  
 9:     **else if**  $\text{width}_{t,a}^s \leq 1/\sqrt{T}$  for all  $a \in \hat{A}_s$  **then**  
 10:       Choose action  $a_t := \arg \max_{a \in \hat{A}_s} \text{ucb}_{t,a}^s$ , and set  $\Psi_{t+1}^s := \Psi_t^s$  for all  $s \in [S]$ .  
 11:     **else**  
 12:       Update  
       
$$\hat{A}_{s+1} := \left\{ a \in \hat{A}_s \mid \text{ucb}_{t,a}^s \geq \max_{a' \in \hat{A}_s} \text{ucb}_{t,a'}^s - 2^{1-s} \right\}$$
  
       and  $s := s + 1$ .  
 13:     **end if**  
 14:     **until** an action  $a_t$  is found.  
 15:   **end for**

---

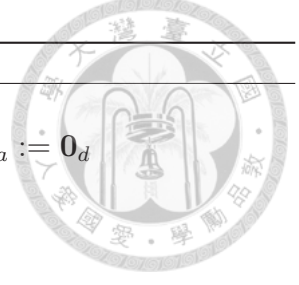
SUPLINPRUCB is designed to solve the dependent issue of context vectors and rewards, thus it is almost identical to SUPLINREL in [9] and SUPLINUCB in [31]. On the other hand, the BASELINPRUCB subroutine is different from LINREL in [9] and BASELINPRUCB in [31], since BASELINPRUCB is modified from LINPRUCB and will be used to justify our idea of pseudo reward. The performance of SUPLINPRUCB is shown in the following theorem.

**Theorem 6.1.** *With probability  $1 - \delta$ , SUPLINPRUCB achieves a regret of  $\tilde{O}(\sqrt{dKT})$ , where  $\tilde{O}$  hides the polynomials of  $\rho = 3/\sqrt{1-\eta}$  and  $\nu = \ln(TK/\delta)$ .*

To prove the theorem, we only need to follow the analysis of SUPLINREL in [9] and replace several critical steps that are related to our BASELINPRUCB subroutine. Let us denote the set of indices that do not belong to any  $\Psi_{T+1}^s$  by  $\Psi^0$  which equals  $[T] \setminus \bigcup_{s=1}^S \Psi_{T+1}^s$  and denote the subset of  $\Psi_t^s$  that collects all  $\tau$  such that  $a_\tau = a$  by  $\Psi_{t,a}^s$  (in other words,  $\Psi_t^s = \bigcup_{a=1}^K \Psi_{t,a}^s$ ). By definition, the expected regret is  $\sum_{t=1}^T (\mathbb{E}[r_{t,a_t^*}] - \mathbb{E}[r_{t,a_t}])$ , which can be regrouped as

$$\sum_{s=1}^S \sum_{a=1}^K \sum_{t \in \Psi_{T+1,a}^s} (\mathbb{E}[r_{t,a_t^*}] - \mathbb{E}[r_{t,a_t}]) + \sum_{t \in \Psi^0} (\mathbb{E}[r_{t,a_t^*}] - \mathbb{E}[r_{t,a_t}]). \quad (6.5)$$

Note that the second term in (6.5) can be easily bounded by  $2\sqrt{T}$  according to Step 9 of SUPLINPRUCB, and the the critical term in (6.5) is the first term which we bound as follows.




---

**Algorithm 7** BASELINPRUCB
 

---

```

1: inputs:  $\Psi_t = \{\sigma_1, \dots, \sigma_{|\Psi_t|} | \sigma_1 < \dots < \sigma_{|\Psi_t|}\} \subseteq [t-1]$ ,  $\beta, \eta, \nu, \rho > 0$ 
2: initialize  $\mathbf{w}_{1,a} := \mathbf{0}_d$ ,  $\hat{\mathbf{Q}}_{1,a} := \mathbf{I}_d$ ,  $\hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a := \mathbf{X}^\top \mathbf{X}_a := \mathbf{0}_{d \times d}$ ,  $\hat{\mathbf{z}}_a := \mathbf{z}_a := \mathbf{0}_d$ 
3: observe context  $x_t$ 
4: for  $a \in [K]$  do
5:   for  $\tau = 1$  to  $|\Psi_t|$  do
6:     if  $a_{\sigma_\tau} = a$  then
7:        $\mathbf{X}^\top \mathbf{X}_a := \mathbf{X}^\top \mathbf{X}_a + \mathbf{x}_{\sigma_\tau} \mathbf{x}_{\sigma_\tau}^\top$ 
8:        $\mathbf{z}_a := \mathbf{z}_a + \mathbf{x}_{\sigma_\tau} r_{\sigma_\tau, a}$ 
9:     else
10:       $p_{\sigma_\tau, a} := \mathbf{w}_{\tau, a}^\top \mathbf{x}_{\sigma_\tau} + \beta \sqrt{(\mathbf{x}_{\sigma_\tau}^\top \hat{\mathbf{Q}}_{\tau, a}^{-1} \mathbf{x}_{\sigma_\tau})}$ 
11:       $\hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a := \eta \hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a + \mathbf{x}_{\sigma_\tau} \mathbf{x}_{\sigma_\tau}^\top$ 
12:       $\hat{\mathbf{z}}_a := \eta \hat{\mathbf{z}}_a + \mathbf{x}_{\sigma_\tau} p_{\sigma_\tau, a}$ 
13:    end if
14:     $\hat{\mathbf{Q}}_{\tau+1, a} := (\mathbf{I}_d + \mathbf{X}^\top \mathbf{X}_a + \hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a)$ 
15:     $\mathbf{w}_{\tau+1, a} := \hat{\mathbf{Q}}_{\tau+1, a}^{-1} (\mathbf{z}_a + \hat{\mathbf{z}}_a)$ 
16:  end for
17:   $\text{width}_{t, a} := \alpha \hat{c}_{t, a} := (1 + \nu + \rho) \sqrt{\mathbf{x}_t^\top \hat{\mathbf{Q}}_{|\Psi_t|+1, a}^{-1} \mathbf{x}_t}$ 
18:   $\hat{r}_{t, a} := \mathbf{w}_{|\Psi_t|+1, a}^\top \mathbf{x}_t$ 
19:   $\text{ucb}_{t, a} := \hat{r}_{t, a} + \alpha \hat{c}_{t, a}$ 
20: end for

```

---

In order to bound  $\mathbb{E}[r_{t, a_t^*}] - \mathbb{E}[r_{t, a_t}]$ , we need a lemma similar to the Lemma 15 in [9]. This can be achieved if we have the following two lemmas. Lemma 6.1 is the key difference in the analysis, and it shows if the rewards indexed by the input set  $\Psi_t$ , which is computed by SUPLINPRUCB, are independent, then  $\mathbf{w}_{|\Psi_t|+1, a}$  computed by BASELINPRUCB has small deviation bound with high probability. In Lemma 6.2, we provides the independence Lemma 6.1 needs.

**Lemma 6.1** (Similar to Lemma 9 in [9]). *Let the set of time indices  $\Psi_t \subseteq [t-1]$  be the input of BASELINPRUCB. Assume for any fixed sequence of  $x_\tau$ ,  $\tau \in \Psi_t$ , the rewards  $r_{\tau, a_\tau}$  are independent random variables with means  $\mathbf{x}_\tau^\top \mathbf{u}_{a_\tau}$ . Then with probability  $1 - \delta/T$ , we have  $|\mathbf{x}_t^\top \mathbf{w}_{t, a} - \mathbf{x}_t^\top \mathbf{u}_a| \leq (1 + \nu + \rho) \hat{c}_{t, a}$  for all  $a \in [K]$ , where  $\nu = O(\ln(TK/\delta))$  and  $\rho = 3/\sqrt{1 - \eta}$ .*

*Proof.* For notational convenience, let us drop all the subscripts involving  $t$  and  $a$ . By definition,

$$\begin{aligned}
|\mathbf{x}^\top \mathbf{w} - \mathbf{x}^\top \theta| &= \left| \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} (\mathbf{X} \mathbf{r} + \hat{\mathbf{X}} \hat{\mathbf{p}}) - \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} (\lambda \mathbf{I}_d + \mathbf{X}^\top \mathbf{X} + \hat{\mathbf{X}}^\top \hat{\mathbf{X}}) \theta \right| \\
&= \left| \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \mathbf{X} (\mathbf{r} - \mathbf{X} \theta) - \lambda \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \theta + \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \hat{\mathbf{X}} (\hat{\mathbf{p}} - \hat{\mathbf{X}} \theta) \right| \\
&\leq \left| \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \mathbf{X}^\top (\mathbf{r} - \mathbf{X} \theta) \right| + \left| \lambda \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \theta \right| + \left| \mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \hat{\mathbf{X}}^\top (\hat{\mathbf{p}} - \hat{\mathbf{X}} \theta) \right|. \quad (6.6)
\end{aligned}$$

The first two terms in (6.6) together can be bounded by  $(1 + \nu)\hat{c}$  with probability  $1 - \delta/T$  using arguments similar to those in Chu et al. [31]. The third term arises from our use of pseudo-rewards, which is an additional cost we need to suffer but BASELINUCB does not, and this is where our forgetting mechanism comes to help. By the Cauchy-Schwarz inequality, this term is at most  $\|\mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \hat{\mathbf{X}}\| \|\hat{\mathbf{p}} - \hat{\mathbf{X}}\theta\|$ , where one can show that  $\|\mathbf{x}^\top \hat{\mathbf{Q}}^{-1} \hat{\mathbf{X}}\| \leq \hat{c}$  using a similar argument as in Chu et al. [31]. Since the  $i$ th entry of the vector  $\hat{\mathbf{p}} - \hat{\mathbf{X}}^\top \theta$  by definition is at most  $3\eta^{\ell-i}$ , we have  $\|\hat{\mathbf{p}} - \hat{\mathbf{X}}^\top \theta\| \leq 3\sqrt{\sum_{i \leq \ell} \eta^{2(\ell-i)}} \leq 3/\sqrt{1-\eta} = \rho$ . By combining these bounds together, we have the lemma.  $\square$

**Lemma 6.2** (Lemma 14 in [9]). *For any  $s \in [S]$ ,  $t \in [T]$ , and for any fixed sequence of context vectors  $\mathbf{x}_\tau$  with  $\tau \in \Psi_t^s$ , the rewards  $r_{\tau, a_\tau}$ ,  $\tau \in \Psi_t^s$ , are independent random variables with  $\mathbb{E}[r_{\tau, a_\tau}] = \mathbf{u}_a^\top \mathbf{x}_\tau$ .*

*Proof.* Suppose a time index  $t$  is added into a set  $\Psi_t^s$  for some  $s$  in Step 8 of SUPLINPRUCB. When  $t$  is the first time index added into  $\Psi_t^s$ , the lemma clearly holds. On the other hand, for  $\Psi_t^s \neq \emptyset$ , when the event happens, it only depends on  $\text{ucb}_{t,a}^{s'}$  and  $\text{width}_{t,a}^{s'}$  for  $s' < s$ , and on  $\text{width}_{t,a}^s$ . Note that  $\text{ucb}_{t,a}^{s'}$  and  $\text{width}_{t,a}^{s'}$  depend on the context vectors  $\mathbf{x}_j$  and rewards  $r_{j,a_j}$  for  $j \in \bigcup_{s' < s} \Psi_t^{s'}$ , and  $\text{width}_{t,a}^s$  depends merely on context vectors  $\mathbf{x}_t$  and  $\mathbf{x}_j$  for  $j \in \Psi_t^s$ . This implies  $r_{t,a_t}$  is independent of  $r_{t',a_{t'}}$  for all  $t' \in \Psi_t^s$  and  $t' < t$ , and thus implies the lemma.  $\square$

Combining the bound shown in Lemma 6.1 with the action selection scheme in SUPLINPRUCB, we can establish a lemma similar to Lemma 15 in [9], from which we have  $\mathbb{E}[r_{t,a_t^*}] - \mathbb{E}[r_{t,a_t}] \leq 2^{3-s}$  for any  $t, s$ , and  $a \in \hat{A}_s$  with high probability. This further implies the sum  $\sum_{t \in \Psi_{T+1,a}^s} (\mathbb{E}[r_{t,a_t^*}] - \mathbb{E}[r_{t,a_t}])$  in (6.5) is at most  $2^{3-s} |\Psi_{T+1,a}^s|$ .

Next, we need the following lemma to further bound  $2^{3-s} |\Psi_{T+1,a}^s|$ .

**Lemma 6.3** (Similar to Lemma 16 in [9]). *For any  $s \in [S]$ ,*

$$2^{3-s} |\Psi_{T+1,a}^s| \leq 5(1 + \nu + \rho) \sqrt{d |\Psi_{T+1,a}^s|}.$$

*Proof.* Recall in Step 14 of BASELINPRUCB,  $\hat{\mathbf{Q}}_{t,a} = \mathbf{I}_d + \mathbf{X}^\top \mathbf{X}_a + \hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a$  is a positive definite matrix since  $\mathbf{X}^\top \mathbf{X}_a$  consists of  $\mathbf{x}_\tau \mathbf{x}_\tau^\top$  with  $\tau \in \Psi_{t,a}^s$ , and  $\hat{\mathbf{X}}^\top \hat{\mathbf{X}}_a$  consists of  $\mathbf{x}_\tau \mathbf{x}_\tau^\top$  with  $\tau \in \Psi_t^s \setminus \Psi_{t,a}^s$ . Further,  $\hat{\mathbf{Q}}_{t,a} \succeq \mathbf{M}_{t,a} = \mathbf{I}_d + \mathbf{X}^\top \mathbf{X}_a$  implies  $\hat{\mathbf{Q}}_{t,a}^{-1} \preceq \mathbf{M}_{t,a}^{-1}$ , and thus  $\mathbf{x}^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x} \leq \mathbf{x}^\top \mathbf{M}_{t,a}^{-1} \mathbf{x}$  for all  $\mathbf{x} \in \mathbb{R}^d$ . Applying Lemma 3 in [31], we have  $\sum_{t \in \Psi_{T+1,a}^s} \sqrt{\mathbf{x}_t^\top \mathbf{M}_{t,a}^{-1} \mathbf{x}_t} \leq$

$5\sqrt{d|\Psi_{T+1,a}^s| \ln |\Psi_{T+1,a}^s|}$  for any  $s$  and  $a$ . Hence, we have

$$\begin{aligned}
\sum_{t \in \Psi_{T+1,a}^s} \text{width}_{t,a}^s &= \sum_{t \in \Psi_{T+1,a}^s} (1 + \nu + \rho) \sqrt{\mathbf{x}_t^\top \hat{\mathbf{Q}}_{t,a}^{-1} \mathbf{x}_t} \\
&\leq (1 + \nu + \rho) \sum_{t \in \Psi_{T+1,a}^s} \sqrt{\mathbf{x}_t^\top \mathbf{M}_{t,a}^{-1} \mathbf{x}_t} \\
&\leq 5(1 + \nu + \rho) \sqrt{d|\Psi_{T+1,a}^s| \ln |\Psi_{T+1,a}^s|}
\end{aligned}$$



On the other hand, by Steps 7 and 8 of SUPLINPRUCB,

$$\sum_{t \in \Psi_{T+1,a}^s} \text{width}_{t,a}^s \geq 2^{-s} |\Psi_{T+1,a}^s|.$$

Putting the upper and lower bounds of  $\sum_{t \in \Psi_{T+1,a}^s} \text{width}_{t,a}^s$  together we have the lemma.  $\square$

From the discussion above, we know that the expected regret of SUPLINPRUCB which is represented by (6.5), can be bounded by  $c \sum_{s=1}^S (\nu + \rho) \sqrt{d|\Psi_{T+1,a}^s|} + 2\sqrt{T}$  for some constant  $c$ . Therefore, following the same steps in the proof of Theorem 6 in [9], we obtain Theorem 6.1.







# Appendix A

## Deferred Proofs in Chapter 3

### A.1 Proof of Lemma 3.1 in Section 3.4

The lemma follows immediately from the following known fact (see e.g. [14, 63]); we give the proof for completeness.

**Proposition A.1.** *Suppose  $\mathcal{R}$  is strictly convex and differentiable, and  $y$  satisfies the condition  $\nabla\mathcal{R}(y) = \nabla\mathcal{R}(u) - \ell$ . Then*

$$\arg \min_{x \in \mathcal{X}} (\langle \ell, x \rangle + \mathcal{B}^{\mathcal{R}}(x, u)) = \arg \min_{x \in \mathcal{X}} \mathcal{B}^{\mathcal{R}}(x, y).$$

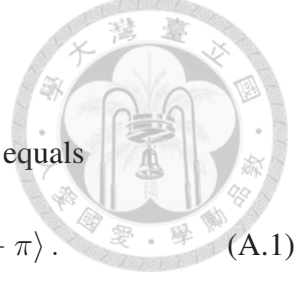
*Proof.* Since  $\mathcal{R}$  is strictly convex, the minimum on each side is achieved by a unique point. Next, note that  $\mathcal{B}^{\mathcal{R}}(x, y) = \mathcal{R}(x) - \mathcal{R}(y) - \langle \nabla\mathcal{R}(y), x - y \rangle = \mathcal{R}(x) - \langle \nabla\mathcal{R}(y), x \rangle + c$ , where  $c = -\mathcal{R}(y) + \langle \nabla\mathcal{R}(y), y \rangle$  does not depend on the variable  $x$ . Thus, using the condition that  $\nabla\mathcal{R}(y) = \nabla\mathcal{R}(u) - \ell$ , we have

$$\begin{aligned} \arg \min_{x \in \mathcal{X}} \mathcal{B}^{\mathcal{R}}(x, y) &= \arg \min_{x \in \mathcal{X}} (\mathcal{R}(x) - \langle \nabla\mathcal{R}(u) - \ell, x \rangle) \\ &= \arg \min_{x \in \mathcal{X}} (\langle \ell, x \rangle + \mathcal{R}(x) - \langle \nabla\mathcal{R}(u), x \rangle). \end{aligned}$$

On the other hand,  $\mathcal{B}^{\mathcal{R}}(x, u) = \mathcal{R}(x) - \mathcal{R}(u) - \langle \nabla\mathcal{R}(u), x - u \rangle = \mathcal{R}(x) - \langle \nabla\mathcal{R}(u), x \rangle + c'$ , where  $c' = -\mathcal{R}(u) + \langle \nabla\mathcal{R}(u), u \rangle$  does not depend on the variable  $x$ . Thus, we have

$$\arg \min_{x \in \mathcal{X}} (\langle \ell, x \rangle + \mathcal{B}^{\mathcal{R}}(x, u)) = \arg \min_{x \in \mathcal{X}} (\langle \ell, x \rangle + \mathcal{R}(x) - \langle \nabla\mathcal{R}(u), x \rangle) = \arg \min_{x \in \mathcal{X}} \mathcal{B}^{\mathcal{R}}(x, y).$$

□



## A.2 Proof of Lemma 3.2 in Section 3.4

Let us write  $\langle \ell_t, \hat{x}_t - \pi \rangle = \langle \ell_t, \hat{x}_t - x_{t+1} \rangle + \langle \ell_t, x_{t+1} - \pi \rangle$  which in turn equals

$$\langle \ell_t - \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle + \langle \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle + \langle \ell_t, x_{t+1} - \pi \rangle. \quad (\text{A.1})$$

To bound the second and third terms in (A.1), we rely on the following.

**Proposition A.2.** *Suppose  $\ell \in \mathbb{R}^n$ ,  $v = \arg \min_{x \in \mathcal{X}} (\langle \ell, x \rangle + \mathcal{B}^{\mathcal{R}}(x, u))$ , and  $w \in \mathcal{X}$ . Then*

$$\langle \ell, v - w \rangle \leq \mathcal{B}^{\mathcal{R}}(w, u) - \mathcal{B}^{\mathcal{R}}(w, v) - \mathcal{B}^{\mathcal{R}}(v, u).$$

*Proof.* We need the following well-known fact; for a proof, see e.g. pages 139–140 of [17].

**Fact A.1.** *Let  $\mathcal{X} \subseteq \mathbb{R}^n$  be a convex set and  $x = \arg \min_{z \in \mathcal{X}} \phi(z)$  for some continuous and differentiable function  $\phi : \mathcal{X} \rightarrow \mathbb{R}$ . Then for any  $w \in \mathcal{X}$ ,  $\langle \nabla \phi(x), w - x \rangle \geq 0$ .*

Let  $\phi$  be the function defined by  $\phi(x) = \langle \ell, x \rangle + \mathcal{B}^{\mathcal{R}}(x, u)$ . Since  $\mathcal{X}$  is a convex set and  $v$  is the minimizer of  $\phi(x)$  over  $x \in \mathcal{X}$ , it follows from Fact A.1 that  $\langle \nabla \phi(v), w - v \rangle \geq 0$ . Since  $\nabla \phi(v) = \ell + \nabla \mathcal{R}(v) - \nabla \mathcal{R}(u)$ , we have  $\langle \ell, v - w \rangle \leq \langle \nabla \mathcal{R}(v) - \nabla \mathcal{R}(u), w - v \rangle$ . Then, by the definition of Bregman divergence, we obtain

$$\begin{aligned} \mathcal{B}^{\mathcal{R}}(w, u) - \mathcal{B}^{\mathcal{R}}(w, v) - \mathcal{B}^{\mathcal{R}}(v, u) &= -\langle \nabla \mathcal{R}(u), w - v \rangle + \langle \nabla \mathcal{R}(v), w - v \rangle \\ &= \langle \nabla \mathcal{R}(v) - \nabla \mathcal{R}(u), w - v \rangle. \end{aligned}$$

As a result, we have

$$\langle \ell, v - w \rangle \leq \langle \nabla \mathcal{R}(v) - \nabla \mathcal{R}(u), w - v \rangle = \mathcal{B}^{\mathcal{R}}(w, u) - \mathcal{B}^{\mathcal{R}}(w, v) - \mathcal{B}^{\mathcal{R}}(v, u).$$

□

From Proposition B.2 and the definitions of  $\hat{x}_t$  and  $x_{t+1}$ , we have

$$\langle \ell_{t-1}, \hat{x}_t - x_{t+1} \rangle \leq \mathcal{B}^{\mathcal{R}_t}(x_{t+1}, x_t) - \mathcal{B}^{\mathcal{R}_t}(x_{t+1}, \hat{x}_t) - \mathcal{B}^{\mathcal{R}_t}(\hat{x}_t, x_t), \text{ and} \quad (\text{A.2})$$

$$\langle \ell_t, x_{t+1} - \pi \rangle \leq \mathcal{B}^{\mathcal{R}_t}(\pi, x_t) - \mathcal{B}^{\mathcal{R}_t}(\pi, x_{t+1}) - \mathcal{B}^{\mathcal{R}_t}(x_{t+1}, x_t). \quad (\text{A.3})$$

Combining the bounds in (A.1), (A.2), (A.3) together, we have the lemma.



### A.3 Proof of Lemma 3.8 in Section 3.7

We need the following lemma from [46]:

**Lemma A.1.** *Let  $u_t \in \mathbb{R}^N$ , for  $t \in [T]$ , be a sequence of vectors. Define  $V_t = I + \sum_{\tau=1}^t u_\tau u_\tau^\top$ .*

*Then,*

$$\sum_{t=1}^T u_t^\top V_t^{-1} u_t \leq N \ln \left( 1 + \sum_{t=1}^T \|u_t\|_2^2 \right).$$

To prove our Lemma 3.8, first note that for any  $t \in [T]$ ,

$$H_t = I + \beta \gamma^2 I + \beta \sum_{\tau=1}^{t-1} \ell_\tau \ell_\tau^\top \succeq I + \beta \sum_{\tau=1}^t \ell_\tau \ell_\tau^\top \succeq I + \frac{\beta}{2} \sum_{\tau=1}^t (\ell_\tau \ell_\tau^\top + \ell_{\tau-1} \ell_{\tau-1}^\top),$$

since  $\gamma^2 I \succeq \ell_t \ell_t^\top$  and  $\ell_0$  is the all-0 vector. Next, we claim that

$$\ell_\tau \ell_\tau^\top + \ell_{\tau-1} \ell_{\tau-1}^\top \succeq \frac{1}{2} (\ell_\tau - \ell_{\tau-1}) (\ell_\tau - \ell_{\tau-1})^\top.$$

This is because by subtracting the right-hand side from the left-hand side, we have

$$\frac{1}{2} \ell_\tau \ell_\tau^\top + \frac{1}{2} \ell_\tau \ell_{\tau-1}^\top + \frac{1}{2} \ell_{\tau-1} \ell_\tau^\top + \frac{1}{2} \ell_{\tau-1} \ell_{\tau-1}^\top = \frac{1}{2} (\ell_\tau + \ell_{\tau-1}) (\ell_\tau + \ell_{\tau-1})^\top \succeq 0.$$

Thus, with  $K_t = I + \frac{\beta}{4} \sum_{\tau=1}^t (\ell_\tau - \ell_{\tau-1}) (\ell_\tau - \ell_{\tau-1})^\top$ , we have  $H_t \succeq K_t$  and  $K_t^{-1} \succeq H_t^{-1}$ . This implies that

$$\sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_{H_t^{-1}}^2 \leq \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_{K_t^{-1}}^2 = \frac{4}{\beta} \sum_{t=1}^T \left\| \sqrt{\frac{\beta}{4}} (\ell_t - \ell_{t-1}) \right\|_{K_t^{-1}}^2,$$

which by Lemma A.1 is at most  $\frac{4N}{\beta} \ln \left( 1 + \frac{\beta}{4} \sum_{t=1}^T \|\ell_t - \ell_{t-1}\|_2^2 \right)$ .





## Appendix B

# Deferred Proofs in Chapter 4

### B.1 Proof of Lemma 4.3 in Section 4.3

We remark that our proof is a simplification of the more general one given in [28] with  $\mathcal{R}_t(x) = \frac{1}{2\eta_t} \|x\|_2^2$ .

Let us rewrite  $\langle g_t, \hat{x}_t - \pi \rangle$  as  $\langle g_t, \hat{x}_t - x_{t+1} \rangle + \langle g_t, x_{t+1} - \pi \rangle$  which equals

$$\langle g_t - \hat{g}_{t-1}, \hat{x}_t - x_{t+1} \rangle + \langle \hat{g}_{t-1}, \hat{x}_t - x_{t+1} \rangle + \langle g_t, x_{t+1} - \pi \rangle. \quad (\text{B.1})$$

The first term above is at most  $\|g_t - \hat{g}_{t-1}\|_2 \|\hat{x}_t - x_{t+1}\|_2$ , by the Cauchy-Schwarz inequality, which is at most  $\eta_t \|g_t - \hat{g}_{t-1}\|_2^2 = S_t$  by the next proposition.

**Proposition B.1.**  $\|\hat{x}_t - x_{t+1}\|_2 \leq \eta_t \|g_t - \hat{g}_{t-1}\|_2$ .

*Proof.* Let  $\phi(w) = \|w - (x_t - \eta_t \hat{g}_{t-1})\|_2^2$  so that  $\hat{x}_t = \arg \min_{w \in \mathcal{X}} \phi(w)$ . Then by the optimality criterion in convex optimization (see pages 139–140 of [17]), we have  $\langle \nabla \phi(\hat{x}_t), x_{t+1} - \hat{x}_t \rangle \geq 0$ , with  $\nabla \phi(\hat{x}_t) = 2(\hat{x}_t - (x_t - \eta_t \hat{g}_{t-1}))$ , which implies that

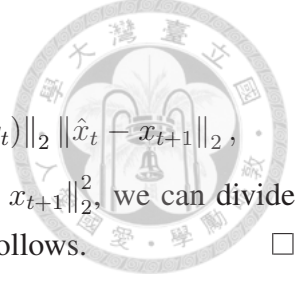
$$\langle \hat{x}_t - (x_t - \eta_t \hat{g}_{t-1}), x_{t+1} - \hat{x}_t \rangle \geq 0.$$

Similarly, by letting  $\phi(w) = \|w - (x_t - \eta_t g_t)\|_2^2$  so that  $x_{t+1} = \arg \min_{w \in \mathcal{X}} \phi(w)$ , we have

$$\langle x_{t+1} - (x_t - \eta_t g_t), \hat{x}_t - x_{t+1} \rangle \geq 0.$$

Adding these two inequalities together, we have

$$\langle (\hat{x}_t - x_{t+1}) + \eta_t (\hat{g}_{t-1} - g_t), x_{t+1} - \hat{x}_t \rangle \geq 0,$$



which implies that

$$\langle \hat{x}_t - x_{t+1}, \hat{x}_t - x_{t+1} \rangle \leq \langle \eta_t (\hat{g}_{t-1} - g_t), x_{t+1} - \hat{x}_t \rangle \leq \|\eta_t (\hat{g}_{t-1} - g_t)\|_2 \|\hat{x}_t - x_{t+1}\|_2,$$

by the Cauchy-Schwarz inequality. As  $\langle \hat{x}_t - x_{t+1}, \hat{x}_t - x_{t+1} \rangle = \|\hat{x}_t - x_{t+1}\|_2^2$ , we can divide both sides of the inequality above by  $\|\hat{x}_t - x_{t+1}\|_2$ , and the proposition follows.  $\square$

To bound the other two terms in (B.1), we need the following.

**Proposition B.2.** *Suppose  $\eta > 0$ ,  $g \in \mathbb{R}^n$ ,  $u \in \mathcal{X}$ , and  $v = \arg \min_{x \in \mathcal{X}} \|x - (u - \eta g)\|_2$ . Then for any  $w \in \mathcal{X}$ ,*

$$\langle g, v - w \rangle \leq \frac{1}{2\eta} (\|w - u\|_2^2 - \|w - v\|_2^2 - \|v - u\|_2^2).$$

*Proof.* Let  $\phi(x) = \|x - (u - \eta g)\|_2^2$  so that  $v = \arg \min_{x \in \mathcal{X}} \phi(x)$ . Then from the optimality criterion,  $\langle \nabla \phi(v), w - v \rangle \geq 0$ , with  $\nabla \phi(v) = 2(v - (u - \eta g)) = 2((v - u) + \eta g)$ , which implies that  $\langle g, v - w \rangle \leq \frac{1}{\eta} \langle v - u, w - v \rangle$ . By a straightforward calculation,  $\langle v - u, w - v \rangle = \frac{1}{2}(\|w - u\|_2^2 - \|w - v\|_2^2 - \|v - u\|_2^2)$ , and the proposition follows.  $\square$

From Proposition B.2, we have

$$\begin{aligned} \langle \hat{g}_{t-1}, \hat{x}_t - x_{t+1} \rangle &\leq \frac{1}{2\eta_t} (\|x_{t+1} - x_t\|_2^2 - \|x_{t+1} - \hat{x}_t\|_2^2 - \|\hat{x}_t - x_t\|_2^2) \text{ and} \\ \langle g_t, x_{t+1} - \pi \rangle &\leq \frac{1}{2\eta_t} (\|\pi - x_t\|_2^2 - \|\pi - x_{t+1}\|_2^2 - \|x_{t+1} - x_t\|_2^2). \end{aligned}$$

Adding the two inequalities above, we get that  $\langle \hat{g}_{t-1}, \hat{x}_t - x_{t+1} \rangle + \langle g_t, x_{t+1} - \pi \rangle$  is at most

$$\frac{1}{2\eta_t} (\|\pi - x_t\|_2^2 - \|\pi - x_{t+1}\|_2^2) - \frac{1}{2\eta_t} (\|x_{t+1} - \hat{x}_t\|_2^2 + \|\hat{x}_t - x_t\|_2^2) = A_t - B_t.$$

Combining this with  $\langle g_t - \hat{g}_{t-1}, \hat{x}_t - x_{t+1} \rangle \leq S_t$  derived before, we have the lemma.

## B.2 Proof of Lemma 4.8 in Section 4.5

Recall that  $B_t = \frac{1}{2\eta} \|x_{t+1} - \hat{x}_t\|_2^2 + \frac{1}{2\eta} \|\hat{x}_t - x_t\|_2^2$ , so

$$\begin{aligned} \sum_{t=1}^T B_t &= \frac{1}{2\eta} \|\hat{x}_1 - x_1\|_2^2 + \frac{1}{2\eta} \sum_{t=2}^T (\|x_t - \hat{x}_{t-1}\|_2^2 + \|\hat{x}_t - x_t\|_2^2) + \frac{1}{2\eta} \|x_{T+1} - \hat{x}_T\|_2^2 \\ &\geq \frac{1}{2\eta} \sum_{t=2}^T (\|x_t - \hat{x}_{t-1}\|_2^2 + \|\hat{x}_t - x_t\|_2^2) \\ &\geq \frac{1}{4\eta} \sum_{t=2}^T \|\hat{x}_t - \hat{x}_{t-1}\|_2^2 \end{aligned}$$

by Proposition 4.1(b). Then the lemma follows as  $\|\hat{x}_1 - \hat{x}_0\|_2^2 \leq R^2 \leq O(1)$ .



### B.3 Proof of Lemma 4.10 in Section 4.6

The lemma follows immediately from the following two lemmas.

**Lemma B.1.**  $\sum_{t=1}^T (A_t - C_t) \leq \sum_{t=1}^T S_t + O(1)$ .

*Proof.* Note that  $\sum_{t=1}^T A_t$  can be rearranged as

$$\frac{1}{2\eta_1} \|\pi - x_1\|_2^2 + \sum_{t=1}^T \left( \frac{1}{2\eta_{t+1}} - \frac{1}{2\eta_t} \right) \|\pi - x_{t+1}\|_2^2 - \frac{1}{2\eta_{T+1}} \|\pi - x_{T+1}\|_2^2. \quad (\text{B.2})$$

The first term above is at most  $(1 + \frac{H}{2}) R^2 = O(1)$ , and let us drop the last term. For the second term, note that  $\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} = \frac{H}{2\gamma} \|g_t - \hat{g}_{t-1}\|_2^2 \leq \frac{H}{2}$  since  $\gamma \geq \|g_t - \hat{g}_{t-1}\|_2^2$ , and moreover,  $\frac{1}{2} \|\pi - x_{t+1}\|_2^2 = \frac{1}{2} \|\pi - \hat{x}_t + \hat{x}_t - x_{t+1}\|_2^2 \leq \|\pi - \hat{x}_t\|_2^2 + \|\hat{x}_t - x_{t+1}\|_2^2$  by Proposition 4.1(b). Thus, with  $C_t = \frac{H}{2} \|\pi - \hat{x}_t\|_2^2$ , we obtain

$$\sum_{t=1}^T (A_t - C_t) \leq \sum_{t=1}^T \frac{H}{2} \|\hat{x}_t - x_{t+1}\|_2^2 + O(1).$$

Since  $\frac{H}{2} \leq \frac{1}{\eta_t}$  and  $\|\hat{x}_t - x_{t+1}\|_2^2 \leq \eta_t^2 \|\hat{g}_{t-1} - g_t\|_2^2$  by the update rule of  $\hat{x}_t$  and  $x_{t+1}$  and by Proposition B.1, we have

$$\sum_{t=1}^T (A_t - C_t) \leq \sum_{t=1}^T \eta_t \|g_t - \hat{g}_{t-1}\|_2^2 + O(1) = \sum_{t=1}^T S_t + O(1),$$

which proves the lemma.  $\square$

**Lemma B.2.**  $\sum_{t=1}^T S_t \leq \frac{2\gamma}{H} \ln \left( 1 + \frac{H}{2\gamma} \sum_{t=1}^T \|g_t - \hat{g}_{t-1}\|_2^2 \right)$ .

*Proof.* Recall that  $S_t = \eta_t \|g_t - \hat{g}_{t-1}\|_2^2$  where  $\eta_t = 1 / \left( 1 + \frac{H}{2} + \frac{H}{2\gamma} \sum_{\tau=1}^{t-1} \|g_\tau - \hat{g}_{\tau-1}\|_2^2 \right)$ . Let  $V_0 = 1$  and  $V_t = 1 + \frac{H}{2\gamma} \sum_{\tau=1}^t \|g_\tau - \hat{g}_{\tau-1}\|_2^2$  for  $t \geq 1$ . Note that  $\eta_t \leq \frac{1}{V_t}$  since  $\gamma \geq \|g_t - \hat{g}_{t-1}\|_2^2$ . This implies that

$$S_t = \eta_t \|g_t - \hat{g}_{t-1}\|_2^2 = \eta_t \frac{2\gamma}{H} (V_t - V_{t-1}) \leq \frac{2\gamma}{H} \left( 1 - \frac{V_{t-1}}{V_t} \right) \leq \frac{2\gamma}{H} \ln \frac{V_t}{V_{t-1}},$$

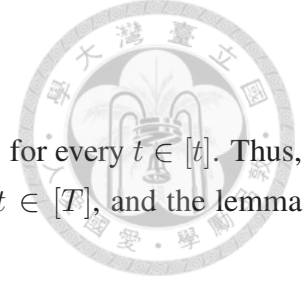
where the last inequality holds since for any two real numbers  $a > b > 0$ ,  $1 - \frac{b}{a} \leq \ln \frac{a}{b}$ . Therefore, by summing over  $t$ , we have

$$\sum_{t=1}^T S_t = \sum_{t=1}^T \eta_t \|g_t - \hat{g}_{t-1}\|_2^2 \leq \frac{2\gamma}{H} \sum_{t=1}^T \ln \frac{V_t}{V_{t-1}} = \frac{2\gamma}{H} \ln \frac{V_T}{V_0} = \frac{2\gamma}{H} \ln V_T.$$

$\square$

## B.4 Proof of Lemma 4.11 in Section 4.6

Recall that  $B_t = \frac{1}{2\eta_t} \|x_{t+1} - \hat{x}_t\|_2^2 + \frac{1}{2\eta_t} \|\hat{x}_t - x_t\|_2^2$ , and note that  $\eta_t \leq 1$  for every  $t \in [T]$ . Thus, if we let  $\eta = 1$ , then  $B_t \geq \frac{1}{2\eta} \|x_{t+1} - \hat{x}_t\|_2^2 + \frac{1}{2\eta} \|\hat{x}_t - x_t\|_2^2$  for every  $t \in [T]$ , and the lemma then follows from Lemma 4.8 (which works for any  $\eta > 0$ ).







# Appendix C

## Deferred Proofs in Chapter 5

### C.1 Proof of Lemma 5.1 in Section 5.1

Recall the update rule from (5.2) that for any  $t \in [T]$  and  $i \in [N]$ ,

$$w_{t+1}(i) = w_t(i) \cdot e^{-\eta f_t(i)},$$

and recall  $W_t = \sum_{i \in [N]} w_t(i)$ . Following a standard analysis of the weighted average algorithm (see e.g. [16, 21]), we have

$$\ln \frac{W_{T+1}}{W_1} = \ln \frac{\sum_{i \in [N]} e^{-\eta \sum_{t \in [T]} f_t(i)}}{N} \geq \ln \frac{e^{-\eta L_{\min}^T}}{N} = -\eta L_{\min}^T - \ln N,$$

and moreover

$$\ln \frac{W_{T+1}}{W_1} = \sum_{t=1}^T \ln \frac{W_{t+1}}{W_t} = \sum_{t=1}^T \ln \sum_{i \in [N]} \frac{w_t(i) \cdot e^{-\eta f_t(i)}}{W_t} = \sum_{t=1}^T \ln \sum_{i \in [N]} p_t(i) e^{-\eta f_t(i)}.$$

Then to get the specific bound of the lemma, we rely on the following claim.

**Claim C.1.** *Suppose  $\eta \in [0, 1/2]$ ,  $p(i), f(i) \in [0, 1]$  for any  $i \in [N]$ , and  $\sum_{i \in [N]} p(i) = 1$ . Then for any  $\pi \in [N]$ ,*

$$\ln \sum_{i \in [N]} p(i) e^{-\eta f(i)} \leq -\eta \sum_{i \in [N]} p(i) f(i) + \eta^2 \sum_{i: f(i) \neq f(\pi)} p(i).$$

We will prove the claim later. Now assuming it and by combining it with the bounds before,

we have

$$\begin{aligned}
-\eta L_{\min}^T - \ln N &\leq \ln \frac{W_{T+1}}{W_1} \\
&\leq \sum_{t=1}^T \left( -\eta \sum_{i \in [N]} p_t(i) f_t(i) + \eta^2 \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \right) \\
&= -\eta L_{\text{ALG}_0}^T + \sum_{t=1}^T \left( \eta^2 \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \right),
\end{aligned}$$

which implies that

$$L_{\text{ALG}_0}^T - L_{\min}^T \leq \frac{\ln N}{\eta} + \sum_{t=1}^T \left( \eta \sum_{i: f_t(i) \neq f_t(i_t)} p_t(i) \right).$$

Thus, to complete the proof of Lemma 5.1, it remains to prove Claim C.1, which we will do next.

*Proof.* Consider the function  $\Phi$  on  $x = (x(1), \dots, x(N)) \in [0, 1]^N$  defined by

$$\Phi(x) = \ln \sum_{i \in [N]} p(i) e^{-\eta x(i)}.$$

Then our goal is to bound the value of  $\Phi$  at the point  $u = (f(1), f(2), \dots, f(N))$ . Using Taylor's theorem, by expanding  $\Phi$  at the point  $u' = (f(\pi), f(\pi), \dots, f(\pi))$ , we have

$$\Phi(u) = \Phi(u') + \sum_{i \in [N]} \frac{\partial \Phi(u')}{\partial x(i)} (f(i) - f(\pi)) + \frac{1}{2} \sum_{i, j \in [N]} \frac{\partial^2 \Phi(v)}{\partial x(i) \partial x(j)} (f(i) - f(\pi))(f(j) - f(\pi)), \quad (\text{C.1})$$

for some  $v \in [0, 1]^N$ . Since  $\sum_{i \in [N]} p(i) = 1$ , the first term in (C.1) is

$$\Phi(u') = \ln \sum_{i \in [N]} p(i) e^{-\eta f(\pi)} = -\eta f(\pi).$$

It remains to bound the other two terms in (C.1).

Let  $h(x) = \sum_{i \in [N]} p(i) e^{-\eta x(i)}$  so that  $\Phi(x) = \ln h(x)$ , and let  $g_i(x) = \frac{\partial h(x)}{\partial x(i)} = -\eta p(i) e^{-\eta x(i)}$ , for  $i \in [N]$ . Then it is not hard to show that

$$\frac{\partial \Phi(x)}{\partial x(i)} = \frac{g_i(x)}{h(x)} \quad \text{and} \quad \frac{\partial^2 \Phi(x)}{\partial x(i) \partial x(j)} = \begin{cases} \frac{-\eta g_i(x)}{h(x)} - \left( \frac{g_i(x)}{h(x)} \right)^2 & \text{if } i = j, \\ -\frac{g_i(x) g_j(x)}{h^2(x)} & \text{if } i \neq j. \end{cases}$$

Using this, the second term in (C.1) can be written as

$$\sum_{i \in [N]} \frac{g_i(u')}{h(u')} (f(i) - f(\pi)) = \sum_{i \in [N]} (-\eta p(i)) (f(i) - f(\pi)) = \eta f(\pi) - \eta \sum_{i \in [N]} p(i) f(i),$$





while the third term in (C.1) can be written as

$$\begin{aligned}
& \frac{1}{2} \sum_{i \in [N]} \left( \frac{-\eta g_i(v)}{h(v)} - \left( \frac{g_i(v)}{h(v)} \right)^2 \right) (f(i) - f(\pi))^2 \\
& - \sum_{1 \leq i < j \leq N} \frac{g_i(v) g_j(v)}{h^2(v)} (f(i) - f(\pi))(f(j) - f(\pi)) \\
& = \frac{1}{2} \sum_{i \in [N]} \left( \frac{-\eta g_i(v)}{h(v)} \right) (f(i) - f(\pi))^2 - \frac{1}{2} \left( \sum_{i \in [N]} \frac{g_i(v)}{h(v)} (f(i) - f(\pi)) \right)^2 \\
& \leq \frac{1}{2} \sum_{i \in [N]} \left( \frac{-\eta g_i(v)}{h(v)} \right) (f(i) - f(\pi))^2 \\
& \leq \eta^2 \sum_{i \in [N]} p(i) (f(i) - f(\pi))^2,
\end{aligned}$$

where the last line follows from the fact that with  $\eta \in [0, 1/2]$ ,

$$\frac{-\eta g_i(v)}{h(v)} = \frac{\eta^2 p(i) e^{-\eta v(i)}}{\sum_{i \in [N]} p(i) e^{-\eta v(i)}} \leq \frac{\eta^2 p(i) e^0}{e^{-1/2}} \leq 2\eta^2 p(i).$$

Finally, by combining all these bounds together, we have

$$\begin{aligned}
\Phi(u) & = -\eta f(\pi) + \eta f(\pi) - \eta \sum_{i \in [N]} p(i) f(i) + \eta^2 \sum_{i \in [N]} p(i) (f(i) - f(\pi))^2 \\
& \leq -\eta \sum_{i \in [N]} p(i) f(i) + \eta^2 \sum_{i \in [N]: f(i) \neq f(\pi)} p(i),
\end{aligned}$$

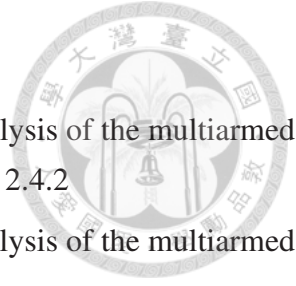
by using the fact that  $(f(i) - f(\pi))^2 \leq 1$  when  $f(i) \neq f(\pi)$ . This proves Claim C.1. □





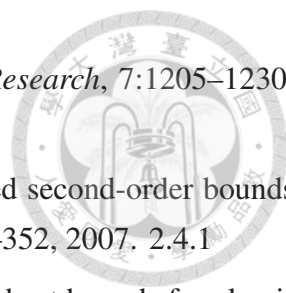
## Bibliography

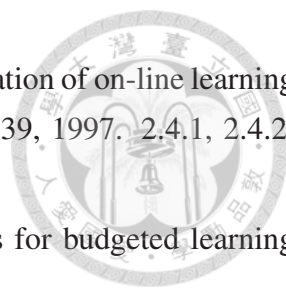
- [1] Jacob Abernethy and Alexander Rakhlin. Beating the adaptive bandit with high probability. In *COLT*, 2009. 2.4.1, 2.4.1, 2.4.2, 2.4.2, 4.2
- [2] Jacob Abernethy, Peter L. Bartlett, and Alexander Rakhlin. Multitask learning with expert advice. In *COLT*, pages 484–498, 2007. 2.4.2
- [3] Jacob Abernethy, Elad Hazan, and Alexander Rakhlin. Competing in the dark: An efficient algorithm for bandit linear optimization. In *COLT*, pages 263–274, 2008. 1.1, 2.4.1, 2.4.1, 4.2
- [4] Alekh Agarwal, Ofer Dekel, and Lin Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pages 28–40, 2010. 1.1, 1.2.2, 2.4.1, 2.4.1, 4.2, 4.3, 4.3, 4.3
- [5] Dana Angluin, James Aspnes, Jiang Chen, and Lev Reyzin. Learning large-alphabet and analog circuits with value injection queries. *Machine Learning*, 72(1–2):113–138, 2008. 2.4.2
- [6] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012. 1.1, 2.4.2, 2.4.2
- [7] Jean-Yves Audibert and Sébastien Bubeck. Regret bounds and minimax policies under partial monitoring. *Journal of Machine Learning Research*, 11:2785–2836, 2010. 2.4.1, 2.4.2, 2.4.2
- [8] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. Minimax policies for combinatorial prediction games. *Journal of Machine Learning Research — Proceedings Track*, 19: 107–132, 2011. 2.4.2, 2.4.2
- [9] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002. 6.1.1, 6.3.3, 6.3.4, 6.3.4, 6.3.4, 6.3.4, 6.1,



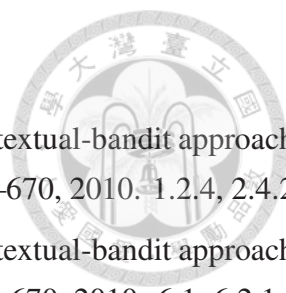
6.2, 6.3.4, 6.3, 6.3.4

- [10] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002. 2.4.2, 2.4.2
- [11] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002. 6.1
- [12] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002. 2.4.2, 2.4.2
- [13] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. Comput. Syst. Sci.*, 74(1):97–114, 2008. 2.4.2, 2.4.2
- [14] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003. 1.2.1, 2.3.1, 2.4.1, 3.2, A.1
- [15] Shai Ben-David, Dávid Pál, and Shai Shalev-Shwartz. Agnostic online learning. In *COLT*, 2009. 2.4.2
- [16] A. Blum and Y. Mansour. Learning, regret minimization, and equilibria. In *Algorithmic Game Theory*. Cambridge University Press, 2007. 2.2, 2.4.2, 5.4, C.1
- [17] Stephen Boyd and Lieven Vandenbergh. *Convex Optimization*. Cambridge University Press, New York, 2004. ISBN 0521833787. A.2, B.1
- [18] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012. 2.4.2, 2.4.2
- [19] Sébastien Bubeck, Nicolò Cesa-Bianchi, and Sham M. Kakade. Towards minimax policies for online linear optimization with bandit feedback. *Journal of Machine Learning Research — Proceedings Track*, 23:41.1–41.14, 2012. 2.4.1, 2.4.1, 4.2
- [20] Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Bounded regret in stochastic multi-armed bandits. In *COLT*, 2013. 2.4.2, 2.4.2
- [21] Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, 2006. 1, 2.3.2, 2.4.2, C.1
- [22] Nicolò Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78(5):1404–1422, 2012. 2.4.2, 2.4.2
- [23] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective

- 
- sampling for linear classification. *Journal of Machine Learning Research*, 7:1205–1230, 2006. 1.1
- [24] Nicolò Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2–3):321–352, 2007. 2.4.1
- [25] Nicolò Cesa-Bianchi, Claudio Gentile, and Francesco Orabona. Robust bounds for classification via selective sampling. In *ICML*, page 16, 2009. 1.1
- [26] Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson sampling. In *NIPS*, pages 2249–2257, 2011. 6.1.1
- [27] Chao-Kai Chiang and Chi-Jen Lu. Online learning with queries. In *SODA*, pages 616–629, 2010. 1.2.3, 2.4.2, 2.4.2, 5
- [28] Chao-Kai Chiang, Tianbao Yang, Chia-Jung Lee, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. Online optimization with gradual variations. *Journal of Machine Learning Research — Proceedings Track*, 23:6.1–6.20, 2012. (document), 1.2.2, 2.4.1, 2.4.1, 2.4.2, 3, 3.2, 4, 4.1, 4.1, 4.2, 4.3, 4.3, 4.5, 4.5, 4.6, B.1
- [29] Chao-Kai Chiang, Chia-Jung Lee, and Chi-Jen Lu. Beating bandits in gradually evolving worlds. In *COLT*, 2013. 2.4.1, 2.4.1, 4
- [30] Ku-Chun Chou, Chao-Kai Chiang, Hsuan-Tien Lin, and Chi-Jen Lu. *Pseudo-Reward for Linear Contextual Bandits*. 6.1.2
- [31] Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. *Journal of Machine Learning Research — Proceedings Track*, 15:208–214, 2011. 6, 6.1.1, 6.3.3, 6.3.4, 6.3.4, 6.3.4
- [32] Thomas Cover. Universal portfolios. *Mathematical Finance*, 1:1–19, 1991. 2.4.1, 4.1
- [33] Varsha Dani, Thomas P. Hayes, and Sham Kakade. The price of bandit information for online optimization. In *NIPS*, 2007. 2.4.1
- [34] Eyal Even-Dar, Michael Kearns, Yishay Mansour, and Jennifer Wortman. Regret to the best vs. regret to the average. *Machine Learning*, 72(1–2):21–37, 2008. 2.4.2
- [35] Eyal Even-Dar, Robert Kleinberg, Shie Mannor, and Yishay Mansour. Online learning for global cost functions. In *COLT*, 2009. 2.4.2
- [36] Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *SODA*, pages 385–394, 2005. 1.2.2, 2.4.1, 2.4.1, 4.2, 4.2, 4.3, 4.3

- 
- [37] Yoav Freund and Robert E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. 2.4.1, 2.4.2, 3.2, 3.5.2
- [38] Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *STOC*, pages 104–113, 2007. 2.4.2
- [39] András György, Gábor Lugosi, and György Ottucsák. On-line sequential bin packing. *Journal of Machine Learning Research*, 11:89–109, 2010. 2.4.2
- [40] Elad Hazan. *Efficient algorithms for online convex optimization and their applications*. PhD thesis, Princeton, NJ, USA, 2006. AAI3223851. 2.3.1
- [41] Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. In *COLT*, pages 57–68, 2008. 1.1, 2.4.1, 2.4.1, 2.4.2, 3.1, 3.2, 3.5.1, 4.1
- [42] Elad Hazan and Satyen Kale. On stochastic and worst-case models for investing. In *NIPS*, pages 709–717, 2009. 1.1, 2.4.1, 2.4.1, 3.1, 3.2, 3.7, 4.1
- [43] Elad Hazan and Satyen Kale. Better algorithms for benign bandits. In *SODA*, pages 38–47, 2009. 1.1, 2.4.1, 2.4.1, 4.2
- [44] Elad Hazan and C. Seshadhri. Adaptive algorithms for online decision problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(088), 2007. 2.4.2
- [45] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *COLT*, pages 499–513, 2006. 2.3.1, 2.4.1, 2.4.1
- [46] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *J. Comput. Syst. Sci.*, 69(2–3):169–192, 2007. 1.1, 3.2, 3.3, 3.7, 3.7, 4.1, 4.2, A.3
- [47] Kevin G. Jamieson, Robert D. Nowak, and Benjamin Recht. Query complexity of derivative-free optimization. In *NIPS*, 2012. 1.2.2, 2.4.1, 2.4.1, 1, 4.2, 1
- [48] Adam Tauman Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005. 2.4.2, 2.4.2
- [49] Satyen Kale. *Efficient algorithms using the multiplicative weights update method*. PhD thesis, Princeton, NJ, USA, 2007. AAI3286120. 2.3.2
- [50] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985. 1.2.4, 2.4.2
- [51] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances*



- 
- in Applied Mathematics*, 6(1):4–22, 1985. 6.1
- [52] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010. 1.2.4, 2.4.2
- [53] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010. 6.1, 6.2.1
- [54] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, pages 297–306, 2011. 6.1
- [55] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *WSDM*, pages 297–306, 2011. 2.4.2
- [56] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994. 2.4.2, 3.2, 3.5.2
- [57] Gábor Lugosi, Omiros Papaspiliopoulos, and Gilles Stoltz. Online multi-task learning with hard constraints. In *COLT*, 2009. 2.4.2
- [58] Shie Mannor and Ohad Shamir. From bandits to experts: On the value of side-observations. In *NIPS*, pages 684–692, 2011. 2.4.2, 2.4.2
- [59] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Nauka Publishers, Moscow, 1978. 1.2.1, 2.4.1
- [60] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization)*. Springer Netherlands, 1 edition, 2004. 3.6
- [61] Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for taxonomies: A model-based approach. In *SDM*, 2007. 2.4.2, 6.1
- [62] Ankan Saha and Ambuj Tewari. Improved regret guarantees for online smooth convex optimization with bandit feedback. *Journal of Machine Learning Research — Proceedings Track*, 15:636–642, 2011. 1.2.2, 2.4.1, 2.4.1, 4.2
- [63] Nati Srebro, Karthik Sridharan, and Ambuj Tewari. On the universality of online mirror descent. In *NIPS*, pages 2645–2653, 2011. A.1
- [64] Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003. 2.4.2, 2.4.2
- [65] Chih-Chun Wang, Sanjeev R. Kulkarni, and H. Vincent Poor. Bandit problems with side

- observations. *IEEE Trans. Automat. Contr.*, 50(3):338–355, 2005. 1.1, 2.4.2, 6.1
- [66] Manfred K. Warmuth and Dima Kuzmin. Online variance minimization. *Machine Learning*, 87(1):1–32, 2012. 2.4.2
- [67] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King’s College, Cambridge, UK, 1989. 6.1.1
- [68] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003. 1.1, 2.3.1, 2.4.1, 2.4.1, 3.2, 3.5.1, 4.1