博士論文

Subgradient-based Online and Stochastic Learning with Biases (劣勾配ベースのバイアス考慮型オンライン学習および確率的学習)

大岩 秀和

Abstract

Machine learning is a framework to make machines learn from data. By taking advantage of useful information in observed data, machine learning enables informative knowledge to be extracted from the data. This extracted knowledge can be used to analyze and predict unobserved data with great accuracy. A large variety of massive-scale data has recently become available for analysis in many fields along with the recent progress of gathering, storing, and processing data in machines. The increase of the data size contributes to the improvement of the predictive performance both theoretically and empirically; therefore, the demand to deal with massive data has been increasing in an efficient way. However, developing novel machine learning algorithms for massive data analysis is a challenging task due to many practical difficulties, such as achieving a short computational time with a small memory size to derive a predictive model. To solve these problems, subgradient-based online and stochastic learning frameworks have been emerging to systematically utilize massive data. These frameworks process a bunch of data one by one and iteratively update predictive models through the subgradient information with respect to the currently received datum. This incremental update procedure realizes an efficient massive data analysis with a simpler implementation and faster computation with a smaller memory space than other machine learning frameworks.

In this thesis, we find that several biases make the original performance measures of online and stochastic learning algorithms obsolete. We find out two biases in this thesis: a truncation bias and a human cognitive bias. These biases originate from involvements of data and humans, which are crucial for applying these incremental learning algorithms to practical applications. We first show how these biases affect the original measures by using toy examples and subjective experiments. These results indicate that the standard performance measures become inappropriate to achieve the original objectives of online and stochastic learning due to the existences of these biases. In the result, state-of-the-art incremental learning algorithms are critically deteriorated even though these algorithms are guaranteed to derive the optimal solution in the conventional performance measures. To solve this difficulty, in this thesis, we reformulate objectives in online and stochastic learning by integrating the effects of these biases and develop algorithms for new framework while maintaining advantages of incremental learning procedures. We theoretically prove the effectiveness of our proposed algorithms under new objectives. In addition, we experimentally show the superiority of our algorithms by using several datasets.

Contents

Chapter 1

Introduction

Machine learning is a framework to make machines learn from data. It is used to analyze observed data and to efficiently take advantage of information extracted from these data. Moreover, machine learning enables machines to gain knowledge and predict unobserved data. Machine learning is a sub-field of artificial intelligence, and more broadly, computer science. Furthermore, there are many research fields closely related to machine learning, including ones from theoretical perspectives, such as optimization [1], statistics [2], and game theory [3], to applications, including natural language processing, computer vision, bioinformatics, and data mining. Machine learning can thus be viewed as an interdisciplinary research field.

A large amount of data has recently become available for data analysis in research and business because of recent advances in the technologies and theories of computer science. In particular, recent progress in data storage, data indexing, data retrieval, network, and computational resources has led to the generation and use of massive amounts of data. For example, about fifteen petabytes of genome data is being generated every year [4], and more than one hundred hours worth of video was uploaded to Youtube, a video-sharing website, every minute in 2014[*1]. The analysis of massive data could achieve breakthroughs in many fields. For these reasons, the applicability of massive data analyses has broadened [5], and both researchers and practitioners are now seeking methods to deal with massive data in an efficient way [6].

We devote most of this thesis to the problems of supervised learning based on convex optimization. In particular, we introduce an incremental learning framework based on a subgradient-based update procedure with linear learners. This incremental learning framework enables us to solve supervised learning problems even if the size of data is very

[*1] `https://www.youtube.com/yt/press/statistics.html`

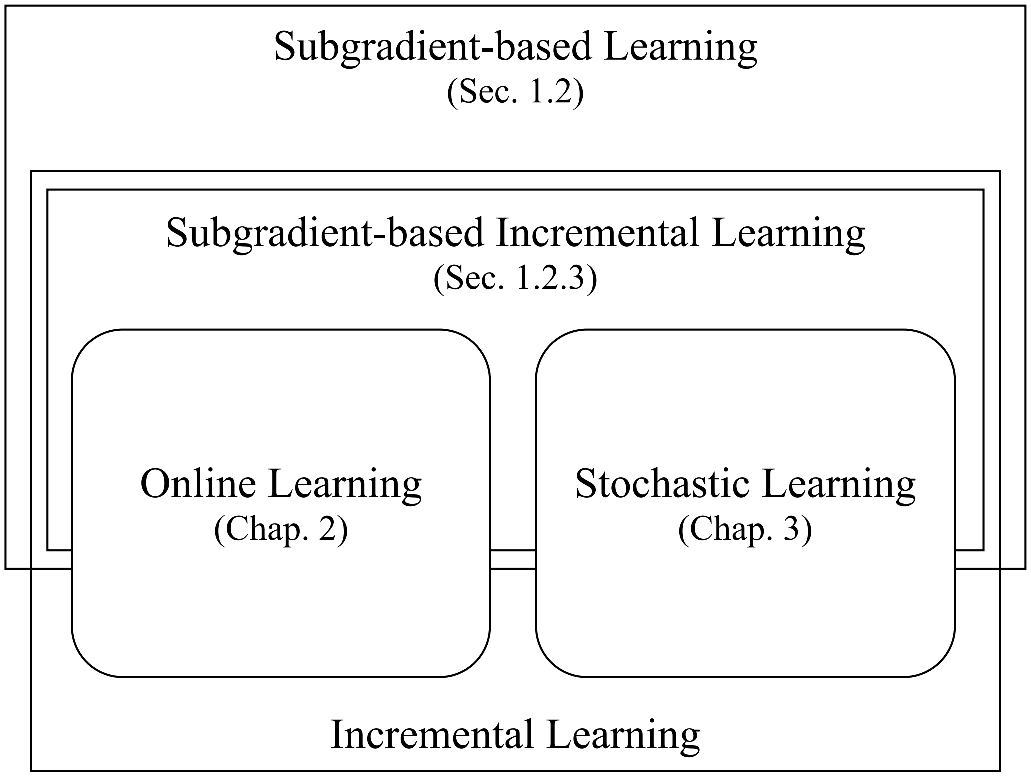

Fig. 1.1. Categorization of subgradient-based online and stochastic learning

large. The framework has proved to be effective at massive data analysis by overcoming many limiting aspects, such as computational complexity and memory constraints, and has been shown to have excellent predictive performance in practical applications [7, 8]. Because of these advantages, practitioners in many fields have utilized subgradient-based incremental learning with linear learners for massive data analysis [9, 10]. Variants of this framework exist in online learning and stochastic learning. They use sequential parameter update procedures and their predictive models are iteratively updated through the use of subgradient information retrieved from one datum each round. In this chapter, we will describe the basic framework of supervised learning and subgradient-based algorithms. After that, we will give a short introduction to subgradient-based incremental learning algorithms. Finally, we will overview the contributions of this thesis. Figure 1.1 illustrates the categorization of these frameworks.

1.1 Supervised Learning

In this thesis, we mainly focus on supervised learning based on convex optimization. In this setting, our objective is to learn an appropriate hypothesis function to predict an output from the corresponding input measures. We would like to extract significant information implicitly contained in each datum as outputs. Such outputs can be seen in the training data; however, they are unseen in the test data. To predict the outputs of test data with high probability, we would like to determine relationships between inputs and outputs that would be helpful in making predictions.

There are two major supervised learning problems: classification and regression. In classification problems, we would like to predict qualitative outputs (i.e., labels); famous examples include digit recognition from images [11], sentiment analysis from review texts [12], and news article categorization [13]. In regression problems, we would like to predict quantitative output values; for example, forecasts of future energy demand [14]. Although many other supervised learning problems exist, such as learning to rank problems [15] and structured prediction problems [16], we will mainly focus on classification and regression problems in this thesis.

We denote an input in a D-dimensional vector form $\mathbf{x}$. This input vector is taken from a closed convex input space $\mathcal{X} \subset \mathbb{R}^D$. Here, $\mathbb{R}^D$ is a D-dimensional Euclidean space. Each dimension of the input space corresponds to one feature, and each component of the input vectors represents a feature value. In most cases, input vectors are naturally defined. For example, some researches utilize gray-scale pixel values as features in the digit recognition task. It is important to generate a better representation of input vectors to improve predictive ability. We will not discuss this aspect in any detail; many studies have devised ways to generate better vector representations for various applications [17, 18].

An output is denoted by a scalar y. It is taken from an output space $\mathcal{Y}$. In classification problems, $\mathcal{Y}$ consists of a set of class labels. In the binary classification case, $\mathcal{Y} = \{-1, 1\}$. On the other hand, in regression problems, $\mathcal{Y}$ becomes a subset of a one-dimensional Euclidean space. Each datum is a pair consisting of an input vector and an output, which is represented by $(\mathbf{x}, y)$.

In supervised learning, our goal is to derive the best hypothesis function mapping from an input space to an output space. A hypothesis function is defined as $h : \mathcal{X} \to \mathcal{Y}$, and we denote the set of hypotheses by $\mathcal{H}$.

1.1.1 Linear Learner

We use a set of linear learners as a class of hypothesis functions. In the linear learner setting, machine learning algorithms use a linear weight vector $\mathbf{w} \in \mathcal{W} \subset \mathbb{R}^D$, where $\mathcal{W}$ is a closed convex set of D-dimensional weight vectors. Algorithms with linear learners make predictions by computing the inner product between weight vectors and input vectors. For example, in regression problems, a hypothesis function is simply defined as follows:

$$h(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle \ , \tag{1.1}$$

where $\langle \mathbf{a}, \mathbf{b} \rangle$ is the notation for expressing the inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$. In (1.1), the inner product of the weight vectors and input vectors is the predicted output values. A hypothesis function in binary classification problems is defined as

$$h(\mathbf{x}) = \mathrm{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle\right) \ , \tag{1.2}$$

where $\mathrm{sgn} : \mathbb{R} \to \{-1, 1\}$ is defined as

$$\mathrm{sgn}(a) = \begin{cases} -1 & a < 0 \\ 1 & \text{otherwise} \end{cases} \ . \tag{1.3}$$

regarding a scalar $a \in \mathbb{R}$. If the inner product is less than 0, the hypothesis outputs -1; otherwise, it outputs 1.

Linear learners have several advantages. First, as we will show in the following subsections, subgradients of loss functions can be efficiently computed in most cases because of their simplicity. Fast derivation of subgradients is the key to efficiently applying subgradient-based learning algorithms to massive amounts of data. Second, linear learner models make it easy to analyze predictive models. Linear learners explicitly possess the weights of each feature. We can obtain insights about how each feature affects predictions by investigating these weight vectors, though each weight value does not represent an independent influence on the predictions. Analyzability is important for continuously improving learning models. For example, linear learner models enable us to determine the truncation bias and to validate the effect of our algorithms through the analysis of linear weight vectors. We show the details in Chapter 4.

For these reasons, we will focus on linear learner models in this thesis.

1.1.2 Loss Functions

Loss functions are used to evaluate the predictive ability of weight vectors. They penalize weight vectors according to the extent to which the predictions are wrong. Loss functions are defined as $\ell : \mathcal{W} \to \mathbb{R}_+$, where $\mathbb{R}_+$ is a subset of the 1-dimensional Euclidean space where the value is non-negative. The worse the predictions with the corresponding weight vector are, the larger the output value becomes. A large variety of loss functions have been proposed for various purposes, for example, the hinge loss function,

$$\ell(\mathbf{w}) = \max\left(0, 1 - y\langle \mathbf{w}, \mathbf{x}\rangle\right) \ , \tag{1.4}$$

the logistic loss function,

$$\ell(\mathbf{w}) = \log\left(1 + \exp\left(-y\langle \mathbf{w}, \mathbf{x}\rangle\right)\right) \ , \tag{1.5}$$

and the squared loss function,

$$\ell(\mathbf{w}) = \left(y - \langle \mathbf{w}, \mathbf{x}\rangle\right)^2 \ . \tag{1.6}$$

The hinge loss and the logistic loss functions are used to evaluate the predictive performances of binary classifications. These loss functions are surrogate losses of the 0/1 loss function. The 0/1 loss function evaluates whether a prediction is correct or not. It is defined as $\mathbb{1}[\![h(x) \neq y]\!]$, where $\mathbb{1}[\![\cdot]\!]$ is an indicator function such that

$$\mathbb{1}[\![a]\!] = \begin{cases} 1 & a \text{ is true} \\ 0 & \text{otherwise} \end{cases} \ . \tag{1.7}$$

Squared loss functions are used in regression problems. To explicitly represent the relationship between the loss functions and the datum $(\mathbf{x}, y)$, we sometimes denote the loss functions by $\ell(\cdot; (\mathbf{x}, y))$. In general, the algorithms attempt to derive weight vectors for the minimization of the loss functions with respect to the data of interest. However, the definition of the objective function changes depending on the problem setting (for example, online or stochastic learning). Furthermore, biases are important components in defining objective functions for achieving the goal in question. As a result, the best hypothesis differs according to the objective.

We assume that the loss functions are convex with respect to the weight vectors. Furthermore, we assume that the loss functions can be rewritten by using a function $\hat{\ell} : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+$ that satisfies

$$\ell(\mathbf{w}) = \hat{\ell}(\langle \mathbf{w}, \mathbf{x}\rangle, y) \ . \tag{1.8}$$

Most of the major loss functions satisfy these assumptions, such as the hinge loss, the logistic loss, and the squared loss. These assumptions make it possible to use subgradient-based update procedures to learn linear predictive models effectively.

1.1.3 Regularization

Regularization is a technique to force certain penalties on hypotheses. Here, regularization terms, such as the L_2 norm of weight vectors, are added to the objective functions. These terms measure the complexity of hypotheses in general and penalize complicated hypotheses so as to avoid them. A large variety of regularization methods have been proposed to ensure that predictive models have some structure [19].

L_2-regularization is a major example. It consists of computing the L_2 norms of weight vectors and is defined as:

$$\Phi(\mathbf{w}) = \lambda\|\mathbf{w}\|_2 , \tag{1.9}$$

where we denote the L_p norm of a vector $\mathbf{a}$ by $\|\mathbf{a}\|_p$ and $\lambda \in \mathbb{R}_+$ is a regularization parameter to adjust the importance ratio between the L_2-regularization and the original objective functions. By penalizing the L_2-norm of the weight vectors, L_2-regularization simplifies the weight vectors and reduces the noise included in the data. Sparsity-inducing regularization methods have recently gained attention in the field of machine learning for many reasons. These methods are detailed in Section 4.1.1 of Chapter 4.

1.2 Subgradient-based Learning

Subgradient-based learning algorithms have been often used for solving supervised learning problems because of their computational efficiency and theoretical guarantee. These algorithms iteratively update their parameters and utilize a first-order approximation of the whole objective functions or part of the objective functions to update the weight vectors in each round. Convexity plays an important role in analyzing the theoretical properties of subgradient-based algorithms.

In the next subsection, we review several notable characteristics regarding convexity. After that, we introduce the basic subgradient-based learning framework and give an overview of subgradient-based incremental learning.

1.2.1 Convex Analysis

We formally introduce the notion of convexity in this subsection. A set $\mathcal{A} \subset \mathbb{R}^D$ is said to be convex if it obeys

$$\forall t \in [0,1],\ \forall \mathbf{a}, \mathbf{b} \in \mathcal{A}, \quad t\mathbf{a} + (1-t)\mathbf{b} \in \mathcal{A} \ . \tag{1.10}$$

Let $\mathrm{dom}(f)$ be the domain of the function f. A function f is said to be convex if it satisfies

$$\forall t \in [0,1],\ \forall \mathbf{a}, \mathbf{b} \in \mathrm{dom}(f), \quad f\left(t\mathbf{a} + (1-t)\mathbf{b}\right) \leq tf(\mathbf{a}) + (1-t)f(\mathbf{b}) \ , \tag{1.11}$$

where $\mathrm{dom}(f)$ is convex. We will assume that the loss functions are convex as defined above with respect to the weight vectors. Therefore, the loss functions $\ell : \mathcal{W} \to \mathbb{R}_+$ satisfy the following inequality:

$$\forall t \in [0,1],\ \mathbf{w}_1, \mathbf{w}_2 \in \mathcal{W}, \quad \ell\left(t\mathbf{w}_1 + (1-t)\mathbf{w}_2\right) \leq t\ell(\mathbf{w}_1) + (1-t)\ell(\mathbf{w}_2) \ . \tag{1.12}$$

When the loss function ℓ is differentiable, we denote its gradient vector with respect to a weight vector $\mathbf{w}$ by

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{w}) \ . \tag{1.13}$$

As we are using linear learner models, the gradient vectors can be reformulated through assumption (1.8) as follows:

$$\frac{\partial}{\partial \mathbf{w}} \ell(\mathbf{w}) = \frac{\partial}{\partial \hat{y}} \hat{\ell}(\hat{y}, y)\mathbf{x} \ , \tag{1.14}$$

where $\hat{y} = \langle \mathbf{w}, \mathbf{x} \rangle$. Therefore, the gradient vectors can be efficiently calculated when $\frac{\partial}{\partial \hat{y}} \hat{\ell}(\hat{y}, y)$ is easily computed. Fortunately, $\frac{\partial}{\partial \hat{y}} \hat{\ell}(\hat{y}, y)$ can be efficiently computed for some of the most well-known loss functions, such as the logistic loss and the squared loss. This characteristic helps to speed up the gradient calculation especially when the input vectors are sparse. We describe this in detail in Section 1.5.1.

Even if the loss functions are convex, they may be non-differentiable at some points. For example, there are non-differentiable points in the hinge loss function due to its non-smoothness. We can use subgradients as an alternative to gradients in these cases. A subgradient of a function f at $\mathbf{a} \in \mathrm{dom}(f)$ is defined by a vector $\nabla f(\mathbf{a})$ such that

$$\forall \mathbf{b} \in \mathrm{dom}(f), \quad f(\mathbf{b}) \geq f(\mathbf{a}) + \langle \nabla f(\mathbf{a}), \mathbf{b} - \mathbf{a} \rangle \ . \tag{1.15}$$

The subdifferential at a point is defined as the set of all subgradients at that point. We denote a subdifferential of f at $\mathbf{a}$ as $\partial_{\mathbf{a}} f(\mathbf{a})$. When f is convex, at least one subgradient exists at any interior point in the domain of the function. This is true even if a function f is non-differentiable. The existence of subgradients of convex functions has been proved.

Proposition 1. *[20, Proposition B.24 (b)] Let a function f be convex. Then, for any interior point in the domain of f, the subdifferential of f is a nonempty, convex, and compact set.*

Note that the property of (1.14) holds with respect to the subgradients when the function is convex. To simplify the following discussion, we will use the notation $\nabla f(\mathbf{a})$ to denote both the gradient and subgradient of a function f at a vector $\mathbf{a}$.

Convexity plays an important role in solving the minimization problem over convex functions. A crucial component is the equality of the local optimum and the global optimum.

Proposition 2. *[20, Proposition 1.1.2] Let a function f be convex and the domain of f be convex. If a point $\mathbf{a}$ in the domain is a local minimum of f, then $\mathbf{a}$ is the global minimum of f.*

This property guarantees that a search for the local minimum is sufficient for solving the minimization problem over convex problems because the local minimum is the global minimum. The convex formulation makes it easy to solve supervised learning problems.

The optimality of the convex functions can be evaluated through their subgradients. Before proceeding to the next proposition, let us define argmin f to be the point or set of points at which the function f becomes a minimum.

Proposition 3. *[20, Proposition B.24 (f)] Let a function f be convex and* $\mathrm{dom}(f)$ *be a closed convex set. Then, if and only if $\mathbf{a}^* \in \mathrm{dom}(f)$ becomes an optimal vector such that*

$$\mathbf{a}^* \in \operatorname*{argmin}_{\mathbf{a} \in \mathrm{dom}(f)} f(\mathbf{a}) \,, \tag{1.16}$$

there is a subgradient $\nabla f(\mathbf{a}^*)$ such that the following inequality is satisfied.

$$\forall \mathbf{b} \in \mathrm{dom}(f), \quad \langle \nabla f(\mathbf{a}^*), \mathbf{a}^* - \mathbf{b} \rangle \leq 0 \,. \tag{1.17}$$

From this proposition, we can see that the optimality of weight vectors can be evaluated from the subgradients. This proposition is often used in the analysis of online and stochastic learning algorithms.

1.2.2 Subgradinet-based Learning Framework

In this subsection, we introduce the subgradient-based learning framework for solving supervised learning problems as convex optimization problems. We denote the objective function as $f : \mathcal{W} \to \mathbb{R}_+$. We assume that f is convex. The procedure of subgradient-based learning is as follows:

1. **Initialization:** Start at round $t = 1$. Initialize the weight vector $\mathbf{w}_1 \in \mathcal{W}$.
2. **Subgradient:** At round t, calculate the subgradient of the whole objective function or parts of the objective function at the t-th weight vector $\mathbf{w}_t$.
3. **Update:** Update the weight vector by using the subgradient and derive the $t+1$-th weight vector $\mathbf{w}_{t+1}$.
4. Increase the round number t until the weight vectors converge.

A crucial component of subgradient-based learning algorithms is how to update this weight vector through the subgradient information. There are numerous work categorized as the subgradient-based learning algorithms to minimize the function f. The basic one is the subgradient method over the unconstrained domain [21]. The update procedure of the subgradient method is as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t) \ , \tag{1.18}$$

where $\eta_t > 0$ is the t-th step size and $\nabla f(\mathbf{w}_t)$ is the subgradient vector of the objective function f at the t-th weight vector $\mathbf{w}_t$. The subgradient vector indicates the direction that maximizes the first-order Taylor approximation of the objective function. Therefore, the weight vectors move in the direction which minimizes the first-order approximation of the objective function in steps of magnitude of η_t. When objective functions are differentiable, the subgradient method reduces to the gradient descent method over the unconstrained domain.

The definition of the step size plays an important role in convergence analyses of solutions given by the subgradient method [20, 22]. The basic analysis is one proving that the subgradient method can obtain the optimal function value for diminishing step sizes such that

$$\sum_{k=1}^{\infty} \eta_t = \infty \quad \sum_{k=1}^{\infty} \eta_t^2 < \infty, \ . \tag{1.19}$$

Theorem 1. *[23] Here, let us assume that an objective function $f : \mathcal{W} \to \mathbb{R}_+$ is convex* *and has a finite optimal value $f^* = \min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w})$. In addition, we assume that f satisfies*

$$\forall \mathbf{a}, \mathbf{b} \in \mathrm{dom}(f), \quad |f(\mathbf{a}) - f(\mathbf{b})| \leq L\|\mathbf{a} - \mathbf{b}\|_2 \ . \tag{1.20}$$

Then, the following inequality is satisfied for $\{\mathbf{w}_t\}_{t=1:T+1}$ derived by the subgradient method.

$$\min_{t=1,\ldots,T} f(\mathbf{w}_t) - f^* \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|_2^2 + G^2 \sum_{t=1}^{T} \eta_t^2}{2\sum_{t=1}^{T} \eta_t} \ . \tag{1.21}$$

If we set $\eta_t = O(1/\sqrt{t})$, the convergence rate becomes $O(1/\sqrt{t})$. In addition, this method achieves a linear convergence rate for strongly convex objective functions, which means it is $O(\rho^T)$, where $0 < \rho < 1$ [23].

There are many extensions of the subgradient method. The major ones are the accelerated methods [24], the smoothing approximation for non-smooth objective functions [25], mirror descent algorithms [26, 27], and primal-dual subgradient schemes [28].

1.2.3 Subgradient-based Incremental Learning

Subgradient-based learning algorithms are very effective ways of dealing with most convex machine learning problems; however, they become inefficient in massive data analysis. Let us denote the data size by N and the dimension size by D. When the objective function consists of the sum of loss functions over data, the computational cost of the subgradient of the objective function becomes $O(ND)$. If N and D are very big, we need to perform the heavy subgradient computation many times. Furthermore, when the data size exceeds the memory size, algorithms need to load data from disk to memory in each iteration. Therefore, the data loading time becomes a dominant factor in the batch learning framework [29]. In such cases, the huge computational cost of optimization in each round makes batch subgradient-based learning algorithms very slow.

To avoid this difficulty, we introduce the subgradient-based incremental learning framework. Incremental learning algorithms do not utilize the whole data in each round; they only utilize one datum per round. To update the parameters, we construct an objective function with respect to the current datum and compute a subgradient vector. Therefore, in the incremental learning setting, the computational cost in each round becomes independent of the whole data size. Because of this advantage and numerous theoretical and experimental results, incremental learning algorithms are seen as effective tools by the machine learning community.

Now, let us introduce the general incremental learning setting based on subgradient information. The basic parameter update procedure is as follows:

Algorithm 1 Incremental Learning Framework

```
Initialize $\mathbf{w}_1 = \mathbf{0} \in \mathcal{W}$
for $t = 1, \ldots$ do
    Receive the $t$-th datum $(\mathbf{x}_t, y_t) \in \mathcal{X} \times \mathcal{Y}$
    Calculate a subgradient for $(\mathbf{x}_t, y_t)$ and $\mathbf{w}_t$
    Update the weight vector and derive $\mathbf{w}_{t+1} \in \mathcal{W}$
end for
```

1. **Initialization:** Start at round $t = 1$. Initialize the weight vector $\mathbf{w}_1 \in \mathcal{W}$.
2. **Datum:** At round t, algorithms receive the t-th datum $(\mathbf{x}_t, y_t) \in \mathcal{X} \times \mathcal{Y}$.
3. **Subgradient:** Calculate a subgradient by using the t-th weight vector $\mathbf{w}_t$ and the t-th datum.
4. **Update:** Update $\mathbf{w}_t$ by using the subgradient and derive the $t+1$-th weight vector $\mathbf{w}_{t+1}$.
5. Increase the round number t. Continue this process until the weight vectors converge or no data remains.

Algorithm 1 summarizes the parameter update procedure of the incremental subgradient-based learning framework.

The incremental subgradient-based learning has recently gained attention in the context of online learning and stochastic learning because of its simple and fast parameter update procedure, memory efficiency, ease of re-learning, and adaptability to streaming data. These two fields deal with different problem settings and objectives. Online and stochastic learning algorithms are based on mathematical and statistical theories, such as linear algebra, convex analysis, duality of convex functions, and optimization methods [30]. This thesis focuses on the frameworks of these two fields, the details of which are in Chapter 2 and 3.

1.3 Subgradient-based Incremental Learning with Biases

To employ the incremental learning framework in real applications, the sort of data and human behavior need to be considered. For example, data has several characteristics, and sometimes these characteristics affect the predictive performance. In addition, systems including machine learning algorithms are usually evaluated by humans. Figure 1.2 illustrates the relationship between subgradient-based incremental learning algorithms with data and human behavior.

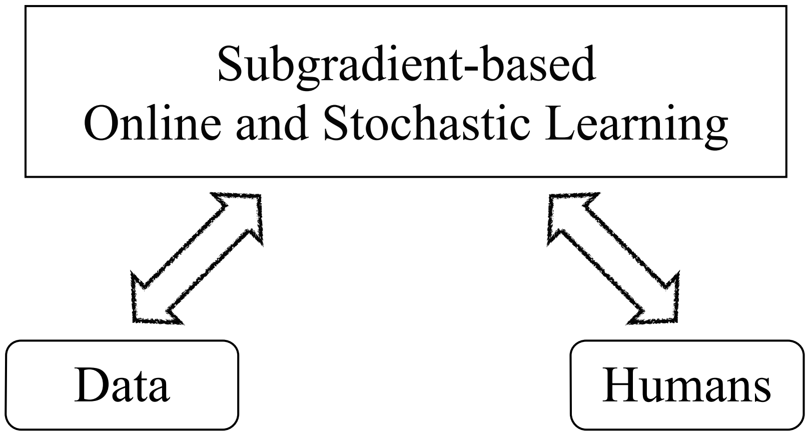

Fig. 1.2. Online and Stochastic Learning with Biases

As described in the previous sections, subgradient-based incremental learning is very efficient on massive data, but that it has biases originating from the data and human behavior. In this thesis, we introduce two such biases: truncation bias and human cognitive bias. These biases are unique to the incremental learning setting and threaten the effectiveness and applicability of its algorithms. We need to devise ways to alleviate the biases if incremental learning algorithms are to be effective.

Several distinct objective functions have been proposed in the online and stochastic learning frameworks, such as the regret and expected loss. The biases mentioned above make it inappropriate to optimize these objective functions. As a result, they critically degrade the performance of even state-of-the-art subgradient-based incremental learning algorithms that have a strong theoretical guarantee on the original performance measures. Figure 1.3 illustrates the categorization of these two biases.

In this thesis, we first show how these biases affect the original performance measures and the performance of conventional learning algorithms. Then, by analyzing the properties of these biases, we establish new frameworks to integrate these biases into the objective functions. We derive new performance measures for incremental learning algorithms and new subgradient-based incremental algorithms for optimizing the new objective functions. The difficulties are that these objective functions have to internalize the effects of the biases and the new algorithms have to maintain the advantages of the subgradient-based incremental learning framework.

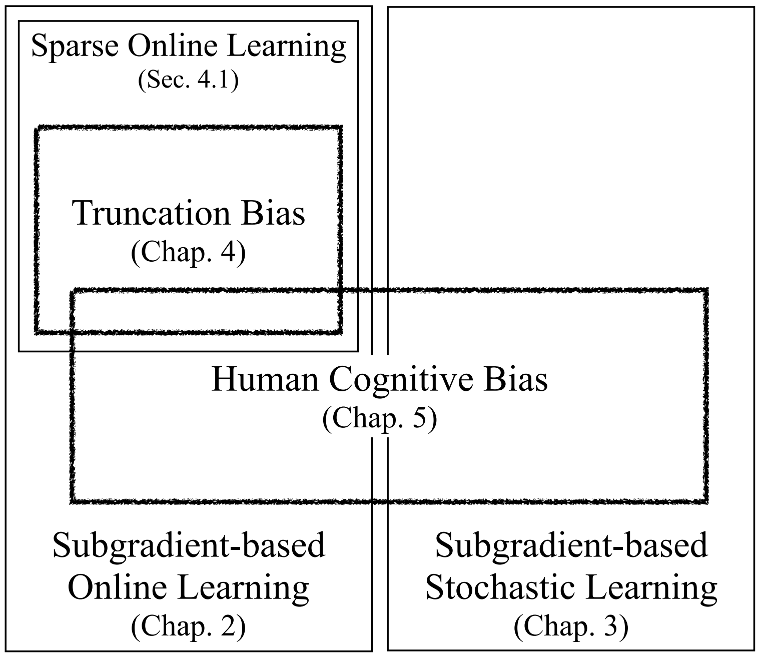

Fig. 1.3. Categorization of Online and Stochastic Learning with Biases

1.4 Our Contributions

The focus of this thesis is on the design, analysis, and solution of the bias problem in the incremental learning settings. These biases appear when we apply subgradient-based incremental learning algorithms to the practical problems.

Our main contributions are as follows:

- We summarize the online and stochastic learning frameworks in terms of their skeletons, objectives, and recent progress.
- We find that two unique biases happen in the subgradient-based incremental learning setting: truncation bias and human cognitive bias. These biases critically degrade the predictive ability of learning algorithms even if they have theoretical guarantees as regards the conventional objectives.
- This phenomenon has a bad effect on subgradient-based incremental learning algorithms in practical learning tasks. We analyze the effects of these biases by using toy examples and subjective experiments. The results indicate that due to these biases, the original objective functions in online and stochastic learning are inappropriate as algorithm performance measures.

- We reformulate new objective functions to be used as performance measures of subgradient-based online and stochastic learning algorithms by integrating these bias effects. Then, we develop new online and stochastic algorithms that have strong theoretical guarantees on the new objective functions and yet maintain the advantages of subgradient-based incremental learning algorithms.
- We conduct experiments on several real datasets, proving the effectiveness of our proposed algorithms.

The constitution of this thesis is described below. In the following section, we introduce the notation and several of the mathematical tools used in this thesis. In Chapter 2, we introduce the online learning framework. Online learning is a variant of the incremental learning framework. This framework does not put any assumptions on the sequence of data. This sequential procedure efficiently deals with a large variety of problem settings, such as streaming data environments, the never-ending learning setting, and adversarial data generation. First, we mathematically formulate the problem setting and parameter update procedure of the subgradient-based online learning framework. Then, we introduce a performance measure called regret and describe several state-of-the-art algorithms. Furthermore, we show several extensions and related work.

In Chapter 3, we introduce the stochastic learning framework. Stochastic learning is also a variant of the incremental learning framework. In this setting, algorithms iteratively update parameters in order to learn an appropriate hypothesis function from data sampled from a specific distribution. We mathematically formulate the problem setting and parameter update procedure of subgradient-based stochastic learning. Then, we introduce the expected loss as a performance measure of this setting. In addition, we show several approaches to giving a bound to the expected loss, such as online to batch conversion using the relationship with regret minimization in online learning and expected loss minimization in stochastic learning. These approaches are very important when deriving efficient stochastic learning algorithms. After that, we show several subgradient-based stochastic learning methods with strong theoretical guarantees.

In Chapter 4, we consider the relationship between the characteristics of data and subgradient-based incremental learning algorithms and analyze the truncation bias. As background, we describe online learning with sparsity-inducing regularization. These frameworks have been used to derive a compact hypothesis function by adding sparsity-inducing regularization terms to the objective functions. Sparse online learning algorithms are advantageous for massive data analysis because their compact model utilizes memory usage efficiently and speeds predictions in the test phase. First, we overview the sparsity-inducing regularization methods and sparse online learning algorithms. After that, we point out that the heterogeneity of feature occurrence frequencies and feature value ranges causes a truncation bias. Through the effect of this bias, several features tend to be removed from the optimal solution in the original performance measure of sparse online learning even if they are helpful for prediction. We show how the truncation bias affects the predictive performance by using toy examples. Then, we develop a new regularization framework, named "Feature-aware Regularization", to overcome this bias in an online manner. This new regularization term integrates information on all previous subgradients of loss functions into a compact format and dynamically adjusts the sparsity-inducing effects with respect to each feature. As a result of this reformulation, the objective function also changes when there is no pre-processing, and we can derive several efficient subgradient-based sparse online learning algorithms. We theoretically prove that our algorithms maintain their theoretical advantages and empirically demonstrate their superiority in terms of prediction performance and sparseness of the predictive models. The work presented in this chapter is based on several previous studies [31, 32, 33].

In Chapter 5, we consider the relationship between human evaluations and subgradient-based incremental learning algorithms. As a result, we find there is a cognitive bias. Incremental learning algorithms iteratively update predictive models; therefore, they sometimes make incorrect predictions on data that were correctly classified in the past. We find that such inconsistent prediction behavior adversely affects the performance evaluations of humans who are susceptible to cognitive bias. These phenomena occur in applications with human users, such as in news recommendation services. In this chapter, we describe a number of cognitive biases as background and show how they affect the incremental learning setting. After that, we show the effect of such inconsistent behavior through a subjective experiment. The results indicate that the original objective functions in the incremental learning setting do not match the human evaluation of the incremental learning algorithms. Accordingly, we formalize new frameworks by internalizing this bias into their performance measures so that they match the evaluations. We develop new subgradient-based incremental learning algorithms that have strong theoretical guarantees without having to know the history of past predictions. This means that the new algorithms can maximize the performance evaluations of humans. Finally, we demonstrate the superiority of our algorithm by conducting several experiments. A large portion of this work can be found in [34].

In Chapter 6, we summarize the discussion of this thesis and review our contributions.

1.5 Preliminaries

First, let us briefly introduce the notation used in this thesis. Scalars are denoted in lowercase italic letters, e.g., α, and the absolute value of a scalar is $|\alpha|$. Vectors are in lowercase bold letters, such as $\mathbf{a}$, and the i-th element of the vector $\mathbf{a}$ is denoted by $a^{(i)}$. Matrices are in uppercase bold letters, e.g., $\mathbf{A}$, and the (i, j)-th element of the matrix $\mathbf{A}$ is described by $A^{(i,j)}$. $\mathbf{I}$ is the identity matrix. When the matrix $\mathbf{A} - \mathbf{B}$ is positive semi-definite, we write $\mathbf{A} \succeq \mathbf{B}$. We write a first-order derivative of a function f (or a subgradient of a function f at a vector $\mathbf{a}$ by $\nabla f(\mathbf{a})$ and a second-order derivative of the function f at a vector $\mathbf{a}$ by $\nabla^2 f(\mathbf{a})$ (it is known as the Hessian matrix). A set of scalars, vectors, or functions, such as $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_T\}$, is sometimes given an abbreviated form, e.g., $\{\mathbf{a}_i\}_{i=1:T}$. Table 1.1 at the end of this chapter summarizes the notation introduced in this chapter. After that, we introduce important mathematical formulations that are used in the derivation and analyses of subgradient-based online and stochastic learning.

1.5.1 Sparse Vector

If most of the elements in a vector are zero, this vector is called sparse. Input vectors are sparse in many tasks, such as the bag-of-word vector representations in natural language processing [35] and computer vision tasks [36]. In the following discussion, we find that sparseness helps to reduce the computational time and memory usage of calculations. A sparse vector $\mathbf{a}$ can be represented by using a set of non-zero elements in an id-value pair form, i.e., $S(\mathbf{a}) = \{(\mathrm{id}_1, v_1), (\mathrm{id}_2, v_2), \ldots\}$, where id_i is an i-th non-zero feature ID and v_i is the corresponding feature value. For a vector to possess a sparse representation, we only need to store the non-zero feature information, that is, the number of non-zero elements denoted by $|S(\mathbf{a})|$. This representation is efficient as far as memory usage goes.

Such a sparse representation of vectors also helps to speed up some of the calculations. In particular, sparse vectors speed up the calculation of the inner product between sparse vectors and dense vectors. The inner product calculation between a sparse vector $\mathbf{a}$ and a dense vector $\mathbf{b}$ can be reformulated as

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a^{(i)} \cdot b^{(i)} = \sum_{(\mathrm{id}_j, v_j) \in S(\mathbf{a})} v_j \cdot b^{(\mathrm{id}_j)} \,. \tag{1.22}$$

Zero elements in a sparse vector $\mathbf{a}$ have no effect on the inner products. We can see from the above reformulation that the inner products are calculated by using only non-zero elements in $\mathbf{a}$. As a result, the computational complexity of the inner product can be reduced to $|S(\mathbf{a})|$ from the number of feature dimensions by efficiently utilizing a sparse representation. Similarly, vector-vector additions and subtractions, and scalar-vector multiplications become faster by means of sparse representation. For example, scalar-vector multiplications $\lambda\mathbf{a}$ can be represented as $S(\lambda\mathbf{a}) = \{(\mathrm{id}_1, \lambda v_1), (\mathrm{id}_2, \lambda v_2), \dots\}$, when $\mathbf{a}$ is a sparse vector. Therefore, we only need to change the values of the non-zero elements in $\mathbf{a}$. The computational complexity depends on $|S(\mathbf{a})|$. Zero value features can be ignored when computing the products of scalars and vectors.

When the input vectors or weight vectors are very sparse, it becomes important to speed up of these calculations (e.g., when a predicted value $\hat{y}$ is computed by taking the inner product of the input vectors and weight vectors). Here, sparse representations are used in most machine learning and optimization software.

1.5.2 Lipschitzness, Smoothness, and Strong Convexity

We introduce some of the important properties of functions, such as smoothness, strong convexity, and Lipschitzness of functions. These are important factors when it comes to defining the difficulty of a problem and for deriving better theoretical results [37, 23]. The discussion of this section mainly follows the presentation in Chapter 12 of [38].

A function $f : \mathrm{dom}(f) \to \mathbb{R}$ is called closed when its level set $\{\mathbf{a} : f(\mathbf{a}) \leq \alpha\}$ is closed and convex for any scalar $\alpha \in \mathbb{R}$. For any function $f : \mathrm{dom}(f) \to \mathbb{R}$ and a norm $\|\cdot\|$, we call a function f L-Lipschitz continuous with respect to the norm $\|\cdot\|$ if it satisfies

$$\forall \mathbf{a}, \mathbf{b} \in \mathrm{dom}(f), \quad |f(\mathbf{a}) - f(\mathbf{b})| \leq L\|\mathbf{a} - \mathbf{b}\| \ , \tag{1.23}$$

for a constant $L > 0$. A differentiable function $f : \mathrm{dom}(f) \to \mathbb{R}$ is said to be α-smooth with respect to a norm $\|\cdot\|$ when it holds that

$$\forall \mathbf{a}, \mathbf{b} \in \mathrm{dom}(f), \quad \langle \nabla f(\mathbf{a}) - \nabla f(\mathbf{b}), \mathbf{a} - \mathbf{b} \rangle \leq \alpha \|\mathbf{a} - \mathbf{b}\|^2 \ , \tag{1.24}$$

for a constant $\alpha > 0$. In other words, a function f is α-smooth when its gradient ∇f is α-Lipschitz with respect to the norm $\|\cdot\|$. The α-smoothness of a convex differentiable function f with respect to a norm $\|\cdot\|$ can be transformed to the upper bound formula of the function value as follows:

$$\forall \mathbf{a}, \mathbf{b} \in \mathrm{dom}(f), \quad f(\mathbf{a}) \leq f(\mathbf{b}) + \langle \nabla f(\mathbf{b}), \mathbf{a} - \mathbf{b} \rangle + \frac{\alpha}{2}\|\mathbf{a} - \mathbf{b}\|^2 \ . \tag{1.25}$$

Similarly, a differentiable function $f : \mathrm{dom}(f) \to \mathbb{R}$ is said to be β-strongly convex with respect to a norm $\|\cdot\|$ if we have

$$\forall \mathbf{a}, \mathbf{b} \in \mathrm{dom}(f), \quad f(\mathbf{a}) \geq f(\mathbf{b}) + \langle \nabla f(\mathbf{b}), \mathbf{a} - \mathbf{b} \rangle + \frac{\beta}{2} \|\mathbf{a} - \mathbf{b}\|^2 , \tag{1.26}$$

for a scalar $\beta > 0$. The theoretical performance of learning algorithms can be improved by utilizing these properties in many cases when the objective functions or a subset of them have these properties.

1.5.3 Legendre Function and Bregman Divergence

We introduce *Legendre* functions [3] and the Bregman divergence [39]. A *Legendre* function $\psi : \mathrm{dom}(\psi) \to \mathbb{R}$ is a function having the following properties:

- $\mathrm{dom}(\psi) \subset \mathbb{R}^D$ is nonempty, and its interior is convex;
- ψ is strictly convex with continuous gradients throughout the interior of $\mathrm{dom}(\psi)$;
- the norm of gradients converges to $+\infty$ when a sequence of inputs approaches the boundary of $\mathrm{dom}(\psi)$.

By using a *Legendre* function ψ, we define the Bregman divergence between two vectors. The Bregman divergence is a natural way of defining the distance between two vectors through a *Legendre* function ψ. The Bregman divergence between two vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$B_\psi(\mathbf{a}, \mathbf{b}) = \psi(\mathbf{a}) - \psi(\mathbf{b}) - \langle \nabla \psi(\mathbf{b}), \mathbf{a} - \mathbf{b} \rangle . \tag{1.27}$$

The Bregman divergence can be viewed as the difference between the *Legendre* function value and its first-order approximation value around $\mathbf{b}$ at point $\mathbf{a}$. For example, the squared Euclidean distance $B_\psi(\mathbf{a}, \mathbf{b}) = \frac{1}{2}\|\mathbf{a} - \mathbf{b}\|_2^2$ is a famous example of the Bregman divergence when we set $\psi(\mathbf{a}) = \frac{1}{2}\|\mathbf{a}\|_2^2$. The Bregman divergence provides a large variety of distance definitions depending on the problem structure, especially the weight vector space. This property has been used in the derivation of a large variety of online learning algorithms and their analyses.

The Bregman divergence has many interesting properties. First, it is always non-negative because ψ is convex. When $\mathbf{a} = \mathbf{b}$, $B_\psi(\mathbf{a}, \mathbf{b}) = 0$. Furthermore, when ψ is α-smooth with respect to a norm $\|\cdot\|$, it satisfies

$$B_\psi(\mathbf{a}, \mathbf{b}) \leq \frac{\alpha}{2} \|\mathbf{a} - \mathbf{b}\|^2 , \tag{1.28}$$

and when the *Legendre* function is β-strongly convex with respect to a norm $\|\cdot\|$, the following inequality is satisfied:

$$B_\psi(\mathbf{a},\mathbf{b}) \geq \frac{\beta}{2}\|\mathbf{a}-\mathbf{b}\|^2 , \tag{1.29}$$

Furthermore, we can see that the next equality is satisfied from the simple algebra of Bregmen divergence [3]. For all $\mathbf{a},\mathbf{b},\mathbf{c} \in \mathrm{dom}(\psi)$,

$$B_\psi(\mathbf{a},\mathbf{b}) + B_\psi(\mathbf{b},\mathbf{c}) = B_\psi(\mathbf{a},\mathbf{c}) + (\mathbf{a}-\mathbf{b})\left(\nabla\psi(\mathbf{c}) - \nabla\psi(\mathbf{b})\right) . \tag{1.30}$$

1.5.4 Duality

In this section, we introduce the notion of duality of norms and functions. Duality is very important when analyzing the theoretical aspects of subgradient-based online and stochastic learning algorithms. First, we define the dual norm $\|\cdot\|_*$ of a specific norm $\|\cdot\|$:

$$\|\mathbf{b}\|_* = \max_{\mathbf{a}}\{\langle\mathbf{a},\mathbf{b}\rangle : \|\mathbf{a}\| \leq 1\} . \tag{1.31}$$

As examples, we can immediately verify that the dual norm of the L_2 norm is itself. The dual norm of the L_1 norm, that is, $\|\mathbf{a}\|_1 = \sum_i |a^{(i)}|$, is the L_∞ norm, that is, $\|\mathbf{a}\|_\infty = \max_i |a^{(i)}|$. Moreover, the dual norm of the L_∞ norm is the L_1 norm. From this definition, we immediately see that the dual norm of a dual norm is the original norm. The following inequality regarding the dual norm is satisfied for some $\alpha > 0$,

$$\langle\mathbf{a},\mathbf{b}\rangle \leq \|\mathbf{a}\|\|\mathbf{b}\|_* \leq \frac{\alpha}{2}\|\mathbf{a}\| + \frac{1}{2\alpha}\|\mathbf{b}\|_* . \tag{1.32}$$

After that, we introduce the Fenchel conjugate as the dual notion of a function. The Fenchel conjugate f^* of a function $f : \mathrm{dom}(f) \to \mathbb{R}$ is defined as:

$$f^*(\mathbf{b}) = \max_{\mathbf{a}\in\mathrm{dom}(f)} \left(\langle\mathbf{a},\mathbf{b}\rangle - f(\mathbf{a})\right) . \tag{1.33}$$

If f is closed and convex, the Fenchel conjugate of f^* is f [40]. The Fenchel-Young inequality is derived from the definition of the Fenchel conjugate:

$$\forall\mathbf{a} \in \mathrm{dom}(f), \quad f^*(\mathbf{b}) \geq \langle\mathbf{a},\mathbf{b}\rangle - f(\mathbf{a}) . \tag{1.34}$$

This equality holds under the following condition.

Lemma 1. *[41] Let f be a closed and convex function and $\nabla f(\mathbf{a})$ be any subgradient of a function f at $\mathbf{a}$. Then, if we set $\mathbf{b} = \nabla f(\mathbf{a})$, we have $f(\mathbf{a}) + f^*(\mathbf{b}) = \langle\mathbf{a},\mathbf{b}\rangle$.*

Furthermore, Fenchel duality gives an interesting relation between strong convexity and smoothness.

Lemma 2. *[42] Assume f is a closed and β-strongly convex function with respect to a norm $\|\cdot\|$. Then f^* is $1/\beta$-smooth with respect to the dual norm $\|\cdot\|_*$. The converse is also true.*

Let us make some observations on the relation between Fenchel conjugate and *Legendre* functions. First, from the definition of *Legendre* functions indicating that they are closed and convex functions, we can see that the Fenchel conjugate of ψ^* becomes ψ when ψ is a *Legendre* function. The gradient of a dual *Legendre functions* has a special property:

Lemma 3. *[3] For all Legendre functions ψ, $\nabla\psi^* = (\nabla\psi)^{-1}$.*

Table. 1.1. General Notation

λ	scalar
$\|\lambda\|$	absolute value
$\mathrm{sgn}(\cdot)$	sign of a real number
$\mathbf{a}$	vector
$a^{(i)}$	i-th entry of vector $\mathbf{a}$
$\mathbf{A}$	matrix
$A^{(i,j)}$	(i,j)-th entry of matrix $\mathbf{A}$
$\mathbf{I}$	identity matrix
$\langle\cdot,\cdot\rangle$	inner product of vectors
$\\|\cdot\\|_p$	L_p norm
$\\|\cdot\\|_*$	dual norm
$\mathrm{dom}(f)$	domain of function f
$\mathrm{argmin}\, f$	point or a set of points for minimizing function f
$\partial_{\mathbf{w}} f(\mathbf{w})$	subdifferential of function f at $\mathbf{w}$
$\nabla f(\mathbf{w})$	(sub)gradient vector of f with respect to $\mathbf{w}$
$\nabla^2 f$	second-order derivative
$f^*(\cdot)$	Fenchel conjugate of function f
$B_\psi(\cdot,\cdot)$	Bregman divergence
$\psi(\cdot)$	*Legendre* function
$\mathbf{x}$	input vector
$\mathbf{x}_i, \mathbf{x}_t$	i-th or t-th input vector
y	output value
y_i, y_t	i-th or t-th output value
$(\mathbf{x}, y)$	datum
$\mathbb{R}^D$	D-dimensional Euclidean space
$\mathbb{R}_+$	one-dimensional non-negative Euclidean space
$\mathcal{X}$	input space
$\mathcal{Y}$	output space
$h(\cdot)$	hypothesis $(\mathcal{X} \to \mathcal{Y})$
$\mathcal{H}$	hypothesis space
$\mathbf{w}$	weight vector
$\mathcal{W}$	weight vector space
$\ell(\cdot), \ell(\cdot; (\mathbf{x}, y))$	loss function $(\mathcal{W} \to \mathbb{R}_+)$

Chapter 2

Online Learning

In the batch learning setting, we have to calculate subgradients over all data in each round when we apply subgradient-based learning algorithms, therefore, the computational complexity per round becomes linear with the number of data. As a result, parameter update procedures become very costly when the data size is quite large. Furthermore, when the data size exceeds the memory size, loading data from disk to memory is needed in each round. These data loading time becomes a dominant factor in the batch learning framework due to slow disk scan [43, 29]. For these reasons, many batch algorithms cannot derive a global optimal solution within a reasonable amount of time when we deal with massive data. The optimization in batch learning requires some sort of reformulation to attain an exact solution [29]. These problems frequently occur when we deal with streaming data [44] or never-ending learning setting [45].

As a novel example of subgradient-based incremental learning framework, online learning becomes an emerging tool for learning from massive data. Online learning algorithms are based on incremental learning procedures where predictions and parameter updates take place each time algorithms receive one datum. Online learning algorithms use only one datum per round to update parameters. After updating, algorithms can discard this datum from the memory because this datum will not be used in the next round. This learning procedure is advantageous for massive data analysis because the computational cost of one-time update becomes independent of the data size and they require a small memory space needed to store the information of only one datum. For these reasons, online learning framework has been applied to many applications, for example, the mail importance labeling [9] and online advertisement replacement optimization [10].

The constitution of this chapter is as follows: First, we introduce the basic framework of subgradient-based online learning in Section 2.1. In Section 2.2, we mathematically formalize the objective of subgradient-based online learning, the regret minimization. After that, we introduce several major subgradient-based online learning algorithms in Section 2.3. For many advantages of subgradient-based online learning settings, many batch learning algorithms have been transformed into online ones. In addition, we show several extensions for the major subgradient-based online learning algorithms to solve difficulties originating from the nature of online learning in Section 2.4. In Section 2.5, we introduce the mistake bound as an alternative performance measure of online learning algorithms for classification problems. In addition, we show several algorithms to pursue to achieve this objective. In Section 2.6, we summarize the discussion of this chapter.

2.1 Subgradient-based Online Learning

In this section, we introduce the basic scheme of subgradient-based online learning. This framework follows the problem setting of online convex programming proposed by Zinkevich [46]. Subgradient-based online learning algorithms perform sequential prediction and parameter update according to the following procedure:

1. **Initialization:** Start at round $t = 1$. Set an initial weight vector $\mathbf{w}_1 \in \mathcal{W}$. The weight vector space is restricted in a closed convex set $\mathcal{W} \subset \mathbb{R}^D$.
2. **Evaluation:** At round t, unveil the t-th loss function $\ell_t : \mathcal{W} \to \mathbb{R}_+$ and incur the cost $\ell_t(\mathbf{w}_t)$. Loss functions ℓ_t are convex with respect to weight vectors.
3. **Subgradient:** Receive the subgradient of the t-th loss function with respect to the current weight vector $\mathbf{w}_t$. The subgradient vector is denoted by $\nabla \ell_t(\mathbf{w}_t)$.
4. **Update:** Update $\mathbf{w}_t$ through the subgradient and derive the $t+1$-th weight vector $\mathbf{w}_{t+1}$.
5. Increase the round number t and repeat this procedure until stopping criterion is satisfied or data are exhausted.

This procedure is based on the subgradient-based incremental learning framework introduced in Section 1.2.3. In each round, we receive the t-th loss function and its subgradient with respect to the current weight vector in this problem setting. Algorithms update weight vectors through subgradients. Algorithm 2 summarizes the basic scheme of subgradient-based online learning algorithms. It is important to note that the generation of a sequence of loss functions does not have any assumption in subgradient-based online learning framework. Loss function can be generated adversarially as a response of a sequence of weight vectors.

Algorithm 2 Subgradient-based Online Learning

```
Initialize $\mathbf{w}_1 \in \mathcal{W}$
for $t = 1, \ldots$ do
    Receive $t$-th loss function $\ell_t$
    Evaluate the weight vector $\mathbf{w}_t$ through the loss function $\ell_t$
    Calculate a subgradient $\nabla \ell_t(\mathbf{w}_t)$
    Update weight vector and obtain $\mathbf{w}_{t+1} \in \mathcal{W}$
end for
```

In most supervised learning tasks, the above framework is specialized as follows: In each round, we receive one datum $(\mathbf{x}_t, y_t)$. A received datum is a pair of D-dimensional input feature vector $\mathbf{x}_t$ taken from a closed convex set $X \subset \mathbb{R}^D$ and the corresponding output value $y_t \in \mathcal{Y}$. By using this definition of data, loss functions are defined as $\ell_t(\cdot) = \ell(\cdot; (\mathbf{x}_t, y_t))$.

2.2 Objective: Regret Minimization

In the standard online learning setting, the main objective is to minimize the cumulative loss function values evaluated per round. In this chapter, we assume that objective functions consist of only loss functions. We discuss how to induce the additional objective to the loss minimization problem in Section 4.1. When we deal with the loss minimization problem without any additional objective (such as sparsity-inducing regularization), algorithms pursue to minimize the sum of loss functions.

In summary, the goal of online learning algorithms is equivalent to the minimization of accumulated loss function values. To achieve this objective, algorithms try to derive weight vectors to minimize the next coming loss function value in each round. The objective function of online learning algorithms is defined by the following formula:

$$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) \ . \tag{2.1}$$

We do not impose any assumption on the data distribution or a sequence of loss functions in the standard online learning setting. Under these conditions, it is known that there is no algorithm to upper bound this objective function because the data can be generated adversarially by seeing the derivation of weight vectors [30]. In addition, if there is no correlation between past data and future data, online learning algorithms could not learn anything clearly. Though online learning algorithms try to minimize the sum of loss function values as much as possible, the evaluation of online learning algorithms by the absolute value of (2.1) is hopeless. By adversarially choosing a sequence of loss functions, the objective function grows in a linear scale $O(T)$ for any online learning algorithms.

To overcome this difficulty, regret has been introduced as a performance evaluation indicator of online learning algorithms [3, 30]. The performance has been measured by using not an absolute evaluation criterion but a relative evaluation criterion compared with the optimal weight vector. For any sequence of loss functions $\{\ell_t\}_{t\geq 1}$, regret (or called cumulative regret) is defined as:

$$\mathrm{Regret}(T) = \sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \min_{\mathbf{u}\in\mathcal{W}} \sum_{t=1}^{T} \ell_t(\mathbf{u}) \ . \tag{2.2}$$

The regret is formalized as the difference between two terms

1. The total cost of loss functions over all rounds while we run online learning algorithms.
2. The minimal cumulative cost when we pick the optimal weight vector in hindsight. The optimal weight vector is defined by

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{u}\in\mathcal{W}} \sum_{t=1}^{T} \ell_t(\mathbf{u}) \ . \tag{2.3}$$

From the definition, we see that regret evaluates how well online learning algorithms generate a sequence of weight vectors compared with a fixed optimal weight vector. We redefine the goal of online learning algorithms as the regret minimization. In other words, we pursue to develop online learning algorithms that achieve as low regret as possible. If regret is upper bounded by a sublinear factor (denoted by $o(T)$) with respect to the number of data, regret on average converges to 0. It holds no matter what a sequence of loss functions is generated. It is true even when the data are generated in an adversarial way. The sublinear regret upper bound guarantees that a derived sequence of weight vectors does not make worse predictions than the best fixed weight vector in terms of the average loss function (2.1).

Unfortunately, it is known that the cumulative regret cannot be upper bounded in a sublinear rate when loss functions are non-convex. For example, Cover [47] shows that it is impossible to develop an online learning algorithm with a sublinear regret upper bound in binary classification setting with the 0/1 loss. Therefore, we focus on online learning for convex loss functions as online convex optimization. By using surrogate loss functions of non-convex loss, such as the hinge-loss function, we can formalize the online convex optimization problem. Convexity plays a crucial role in the derivation of online learning algorithms and analyses from perspectives of regret bound.

As noted above, no assumption is put on a sequence of loss functions and data. The data can be generated not only from an i.i.d. environment but also other complicated ones, such as the adversarial setting. Learning from streaming data sometimes need to deal with adversarial agents, for example, the spam mail filtering [9, 48]. In an adversarial setting, data are generated such that algorithms make wrong predictions in each round. Regret can be employed as an evaluation criterion of subgradient-based online learning even in an adversarial setting.

We note that several variants of regret has been proposed and analyzed in many researches [3, 49, 50]. For example, the underlying data distribution may be drifting over time in some tasks. This situation frequently occurs in the streaming data processing. When we know these phenomenon will occur in advance, we assume that the best competitor should be changed according to the concept drift. Adaptive regret [49] has been proposed to generalize the standard regret by allowing changes of the best competitor.

2.3 Subgradient-based Online Algorithms

Many researchers have designed algorithms in relation to subgradient-based online learning with linear predictors. In this section, we introduce several subgradient-based online learning algorithms and their theoretical properties.

2.3.1 Online Subgradient Method

We introduce Online Subgradient Method [46, 7] (OSM) in this section. This algorithm is one of the simplest subgradient-based online learning algorithms and an online version of Subgradinet Method introduced in Section 1.2.2. It is widely used because of its simplicity and extensive theoretical background.

The update formula of OSM in each round is defined as:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}\left(\mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t)\right) \ . \tag{2.4}$$

$\nabla \ell_t(\mathbf{w}_t)$ indicates a subgradient of the t-th loss function ℓ_t with respect to the t-th weight vector $\mathbf{w}_t$. $\Pi_{\mathcal{W}}(\cdot)$ is a projection function onto a convex set $\mathcal{W}$ such that $\Pi_{\mathcal{W}}(\mathbf{w}) = \arg\min_{\mathbf{w}' \in \mathcal{W}} \|\mathbf{w} - \mathbf{w}'\|_2$. We can see from this update formula that the weight vector is projected onto $\mathcal{W}$ if it moves to the outside of $\mathcal{W}$. $\{\eta_t\}_{t=1:T}$ is a sequence of positive learning rates. If the weight vector space is the D-dimensional Euclidean space $\mathbb{R}^D$, the projection function can be omitted. The weight vector is continuously updated according to the formula (2.4) whenever OSM receives a new loss function. OSM utilizes a first-order approximation of loss functions to update weight vectors. The first-order approximation of only one loss function contributes to smaller computational cost and memory usage, especially when these constraints are crucial concerns in the batch learning setting. From (2.4), we can see that OSM updates weight vectors for the reverse direction of the subgradient vector.

OSM is a natural generalization of online gradient descent (OGD) [46]. Though some papers utilize the term OGD to explicitly represent that this algorithm utilizes only the differential loss functions, we use the term OGD even if loss functions are non-differentiable and algorithms use subgradients by following the discussion of [46]. Therefore, OGD and OSM are exactly the same and we utilize the term OGD to indicate the algorithms in the following discussion.

Computational Cost of OGD

We analyze the computational complexity of OGD. The computational complexity of OGD depends on the calculation of projection functions and the inner part of the projection function in (2.4). We first analyze the inner part of the projection function, that is,

$$\mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t) \ . \tag{2.5}$$

The calculation of (2.5) requires the subgradient computation, the scalar-vector multiplication with respect to the subgradient, and the subtraction operation between the current weight vector and the subgradient. From the assumption (1.14), we assume that when we set $\hat{y} = \langle \mathbf{x}, \mathbf{w} \rangle$, subgradients can be computed by loss function value $\ell(\hat{y}, y)$ and input vectors $\mathbf{x}$, and $\ell(\hat{y}, y)$ is calculated within a constant time. Therefore, the cost of subgradient calculations is dominated by the calculation of the multiplication between the loss function value and input vectors, that is, a linear order with respect to the input vector dimension $O(D)$. The computational cost of other multiplications and subtractions are also $O(D)$, therefore, the computational cost of (2.5) becomes $O(D)$.

When input vectors are sparse, its computational cost is significantly reduced. From the assumption (1.14), we can derive the subgradient vector with the non-zero element size of sparse input vectors, that is, $O(S(\mathbf{x}_t))$. In addition, the subgradient vector also become sparse. Therefore, other multiplication and subtraction computations can be performed by $O(S(\mathbf{x}_t))$ as written in Section 1.5.1. Therefore, the computational cost of (2.5) becomes $O(S(\mathbf{x}_t))$ when input vectors are sparse.

The computational cost of projection function depends on the structure of convex weight spaces. In some cases, the projection step becomes computationally very expensive. For example, the projection onto the any kind of polytopes needs to solve a convex quadratic program [51]. It is very hard to solve these problems, therefore, several investigations of projection-free counterparts has been often done in some researches [52, 53]. On the other hand, it has been known that Euclidean projections onto several famous convex domains (such as ℓ_2-ball, ℓ_1-ball [54], cube, and simplex [55]) can be computed within the linear time with respect to the size of dimension. Therefore, when we adopt the simple weight space as described above, the computational cost of the projection follows $O(D)$ or less in some cases (e.g. weight vectors are sparse). In summary, the computational cost of OGD becomes $O(D)$ or less. It is important to note that the computational complexity does not depend on the data size. This advantage is crucial when we deal with massive data.

Memory Usage of OGD

We analyze OGD from the perspective of memory usage. To update parameters in each round, OGD needs to store the current weight vector in the memory and utilize only the current received datum. The used data in the previous rounds can be discarded, therefore, the memory usage does not depend on the data size. In summary, the memory usage of OGD follows the linear order of the dimension size, that is, $O(D)$. If both input vectors and weight vectors are sparse, this memory usage can be reduced.

Theoretical Analyses of OGD

From the theoretical aspect, regret is an important performance indicator of subgradient-based online learning algorithms. Ever since OGD was proposed in the context of online convex programming, the theoretical aspects of OGD has been often analyzed by several researches. First, Zinkevich [46] proved that OGD obtains a sublinear regret upper bound under practical constraints.

Theorem 2. *[46] Let a sequence of weight vectors $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to OGD's update formula (2.4). Assume that there exists $R, G \in \mathbb{R}$ such that for all $\mathbf{w}, \tilde{\mathbf{w}} \in \mathcal{W}$, $\|\mathbf{w} - \tilde{\mathbf{w}}\|_2 \leq R^2$ and for all t, $\|\nabla \ell_t(\mathbf{w}_t)\|_2 \leq G$. Then, when a sequence of step sizes $\{\eta_t\}_{t=1:T}$ is positive and non-increasing, regret is upper bounded by the following formula:*

$$\mathrm{Regret}(T) \leq \frac{R^2}{2\eta_T} + \frac{G^2}{2} \sum_{t=1}^{T} \eta_t \ . \tag{2.6}$$

For a decreasing step size, we can obtain the $O(\sqrt{T})$ regret upper bound.

Corollary 1. *Assume that the conditions defined in Theorem 2 is satisfied. When we set $\eta_t = R/G\sqrt{2t}$, the regret bound is*

$$\mathrm{Regret}(T) \leq \sqrt{2}RG\sqrt{T}\ . \tag{2.7}$$

$O(\sqrt{T})$ regret upper bound can be obtained in the case of constant step size when we know the number of rounds in advance.

Corollary 2. *When we set $\eta_t = R/G\sqrt{T}$, the regret bound is*

$$\mathrm{Regret}(T) \leq RG\sqrt{T}\ . \tag{2.8}$$

Even if algorithms do not know the round number T in advance, OGD obtains $O(\sqrt{T})$ regret bound by applying the doubling trick [30]. From this result, we see that the regret value of OGD is upper bounded by the square root of the round number. This bound guarantees that OGD obtains the same average loss as the the best fixed hypothesis does.

It is known that this upper bound rate $O(\sqrt{T})$ cannot be improved in general for arbitrary convex loss functions (e.g. see [3, 56]), that is $\Omega(\sqrt{T})$. This rate cannot be improved even if we assume that loss functions are sampled from a fixed specific distribution. However, when we assume that loss functions are twice-differentiable and strongly convex, OGD achieves $O(\log T)$ regret upper bound by appropriately setting a sequence of step sizes. When we know the convexity parameter of loss functions before we run algorithms, we can determine a sequence of step sizes to achieve $O(\log T)$ regret upper bound [56, 57].

Theorem 3. *[56] Let a sequence of weight vectors $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to OGD's update formula (2.4). Assume that there exists $G, \beta \in \mathbb{R}$ such that $\|\nabla \ell_t(\mathbf{w}_t)\|_2 \leq G$ for all t, and loss functions are β-strongly convex with respect to all weight vector $\mathbf{w} \in \mathcal{W}$. When we set $\eta_t = \frac{1}{\beta t}$, regret is upper bounded by the following inequality.*

$$\mathrm{Regret}(T) \leq \frac{G^2}{2\beta}(1 + \log T)\ . \tag{2.9}$$

In the following research [58], it is shown that $O(\sqrt{T})$ regret bound for arbitrary convex loss functions and $O(\log T)$ for strongly convex loss functions without any prior knowledge about the convexity parameter about loss functions by adding adaptive regularizations to objective functions. It is known that this upper bound rate $O(\log T)$ is the optimal rate [59]. In other words, the lower bound of regret for strongly convex loss functions is $\Omega(\log T)$.

2.3.2 Online Mirror Descent

As a generalized framework of gradient descent methods introduced in Section 1.2.2, Mirror Descent (MD) algorithms have been developed for achieving efficient subgradient-based batch learning [26, 27]. MD algorithms have been proposed to solve convex optimization problems through an iterative update procedure. The main advantage of MD compared with the gradient descent algorithms is the applicability of a general distance function based on the Bregman divergence, and as a result, MD provides a wider family of efficient subgradient-based learning algorithms compared with the gradient descent according to the structure of weight vectors, loss functions and its subgradients. As an online version of MD, Online Mirror Descent (OMD) has been developed [3, 60, 30]. OMD succeeds advantages of MD. The update formula of OMD in each round is defined as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{\eta_t \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w}\rangle + B_\psi(\mathbf{w}, \mathbf{w}_t)\right\} \ , \tag{2.10}$$

where B_ψ is the Bregman divergence defined in Section 1.5.3. OMD uses the subgradient of the t-th loss function to update the parameters. OMD solves the minimization problem defined in the parenthesis of the right-hand side to derive the next weight vector. The formula consists of two terms. The first term is the inner product between the next weight vector and the subgradient vector with respect to the current loss function. If weight vectors move to the reverse direction of the subgradient vector, the formula value becomes decreased. This motivation is similar to the subgradient method, where weight vectors are updated in the opposite direction of the current subgradient. The second term is the Bregman divergence between the next weight vector and the current weight vector. As written in Section 1.5.3, the Bregman divergence represents the divergence between two vectors. The value becomes large when the next weight vector moves far from the current weight vector. $\eta_t > 0$ is a hyper-parameter and it adjusts the trade-off of importance ratio between these two terms. In summary, OMD tries to minimize the linear approximation of the loss function while not moving too far from the current weight vector to derive the updated weight vector. This update formula is equivalently represented by using the dual notion[*1]. The definition of OMD update formula can be given by the following formula:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} B_\psi\left(\mathbf{w}, \nabla\psi^*\left(\nabla\psi(\mathbf{w}_t) - \eta_t \nabla \ell_t(\mathbf{w}_t)\right)\right) \ . \tag{2.11}$$

[*1] MD algorithms were originally developed from this dual notion and later proved the equivalence to (2.10). We first introduce the primal notion of the update formula for simplifying the discussion.

where ψ is a *Legendre* function used in the Bregman divergence (2.10) and ψ^* is its Fenchel conjugate. The equivalence of (2.10) and (2.11) is written in several previous papers (e.g. [27]). From the property of the derivative of dual *Legendre* functions written in Lemma 3, $\nabla\psi^*$ equals to $\nabla\psi^{-1}$. From this update formula, we can see that the subgradients of loss functions are used to update in the domain of Fenchel conjugate of the *Legendre* function, not the primal domain. This algorithm is called Online **Mirror** Descent because parameters are updated in the dual space by using subgradients in each step, and then algorithms generate weight vectors by mapping from parameters in the dual space to the primal space through the inverse of *Legendre* functions.

It has been proven that OGD is a special case of OMD.

Lemma 4. *[30] Let us define a Legendre function as $\psi(\mathbf{w}) = \|\mathbf{w}\|_2^2/2$ for $\mathbf{w} \in \mathcal{W}$ and otherwise $\psi(\mathbf{w}) = +\infty$. , the update formula becomes*

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \eta_t \langle \nabla\ell_t(\mathbf{w}_t), \mathbf{w}\rangle + \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|_2^2 \right\} . \tag{2.12}$$

By solving (2.12), we derive the following update formula that is equivalent to the OGD update formula.

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}\left(\mathbf{w}_t - \eta_t \nabla\ell_t(\mathbf{w}_t)\right) . \tag{2.13}$$

From Lemma 4, OMD can be viewed as a general class of OGD by replacing the Euclidean distance by the Bregman divergence.

The theoretical guarantee of OMD for regret bound is as follows:

Theorem 4. *Let a sequence of weight vectors $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to OMD's update formula (2.10). Assume that ψ is β-strongly convex with respect to a norm $\|\cdot\|$, there exists scalars $R, G \in \mathbb{R}$ such that $B_\psi(\mathbf{w}, \tilde{\mathbf{w}}) \le R^2$ for all $\mathbf{w}, \tilde{\mathbf{w}} \in \mathcal{W}$ and $\|\nabla\ell_t(\mathbf{w}_t)\|_* \le G$ for all t. When we use a positive non-increasing sequence of step sizes, that is $\eta_t \ge \eta_{t+1}$ for all t, the following inequality holds:*

$$\mathrm{Regret}(T) \le \frac{R^2}{\eta_T} + \frac{G^2}{2\beta}\sum_{t=1}^{T}\eta_t . \tag{2.14}$$

While we could not find the direct proof of this theorem when we set non-constant step sizes, we show the proof of this theorem below. First, we introduce and prove the following lemma to derive the upper bound of regret for OMD.

Lemma 5. *The conditions defined in Theorem 4. Let us define*

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{u}\in\mathcal{W}} \sum_{t=1}^{T} \ell_t(\mathbf{u}) . \tag{2.15}$$

. Then, the following inequality is satisfied.

$$\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) \le \frac{1}{\eta_t}\left(B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1})\right) + \frac{\eta_t}{2\beta}\|\nabla \ell_t(\mathbf{w}_t)\|_*^2 \,. \tag{2.16}$$

Proof. This lemma can be proved by following discussions in several previous work [27, 61]. We reformulate the left-hand side of the formula (2.16) as follows:

$$\begin{aligned}
\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) &\le \langle \mathbf{w}_t - \mathbf{w}^*, \nabla \ell_t(\mathbf{w}_t)\rangle \\
&= \langle \mathbf{w}_{t+1} - \mathbf{w}^*, \nabla \ell_t(\mathbf{w}_t)\rangle + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t)\rangle \\
&= \frac{1}{\eta_t}\langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla\psi(\mathbf{w}_t) - \nabla\psi(\mathbf{w}_{t+1}) - \eta_t \nabla \ell_t(\mathbf{w}_t)\rangle \\
&\quad + \frac{1}{\eta_t}\langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla\psi(\mathbf{w}_{t+1}) - \nabla\psi(\mathbf{w}_t)\rangle + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t)\rangle \,.
\end{aligned} \tag{2.17}$$

The first inequality is derived from the convexity of loss functions. Now, from the optimality condition introduced in Proposition 3, the first term in the last equation becomes non-positive because $\mathbf{w}_{t+1}$ is the optimal solution in (2.10). Therefore, the above inequality is reformulated as:

$$\begin{aligned}
\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) &\le \frac{1}{\eta_t}\langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla\psi(\mathbf{w}_{t+1}) - \nabla\psi(\mathbf{w}_t)\rangle + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t)\rangle \\
&= \frac{1}{\eta_t}B_\psi(\mathbf{w}^*, \mathbf{w}_t) - \frac{1}{\eta_t}B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) - \frac{1}{\eta_t}B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t)\rangle \\
&\le \frac{1}{\eta_t}B_\psi(\mathbf{w}^*, \mathbf{w}_t) - \frac{1}{\eta_t}B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) - \frac{1}{\eta_t}B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \\
&\quad + \frac{\beta}{2\eta_t}\|\mathbf{w}_t - \mathbf{w}_{t+1}\|^2 + \frac{\eta_t}{2\beta}\|\nabla \ell_t(\mathbf{w}_t)\|_*^2 \\
&\le \frac{1}{\eta_t}B_\psi(\mathbf{w}^*, \mathbf{w}_t) - \frac{1}{\eta_t}B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta_t}{2\beta}\|\nabla \ell_t(\mathbf{w}_t)\|_*^2 \,.
\end{aligned} \tag{2.18}$$

The first equality follows (1.30) and the second inequality is derived from (1.34). The last inequality follows (1.29). This lemma is proven by this formula. □

Proof of Theorem 4. The regret bound of OMD is derived by summing up this inequality derived in Lemma 5 from $t = 1$ to $t = T$. When we assume $B_\psi(\mathbf{w}, \tilde{\mathbf{w}}) \le R^2$ for all $\mathbf{w}, \tilde{\mathbf{w}} \in \mathcal{W}$ and $\|\nabla \ell_t(\mathbf{w}_t)\|_* \le G$ for all t, the following inequality is satisfied.

$$\begin{aligned}
\mathrm{Regret}(T) &\le \frac{1}{\eta_1}B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \sum_{t=1}^{T-1}\left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right)B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \\
&\qquad - \frac{1}{\eta_T}B_\psi(\mathbf{w}^*, \mathbf{w}_{T+1}) + \sum_{t=1}^{T}\frac{\eta_t}{2\beta}\|\nabla \ell_t(\mathbf{w}_t)\|_*^2 \\
&\le \frac{1}{\eta_T}R^2 + \frac{G^2}{2\beta}\sum_{t=1}^{T}\eta_t \,.
\end{aligned} \tag{2.19}$$

□

Furthermore, OMD achieves the logarithmic regret bound $O(\log T)$ for strongly convex loss functions [62]. It is known that OMD achieves a nearly optimal regret upper bound for any general convex online learning problems with an appropriate *Legendre* function choice [63], though this *Legendre* function choice is impractical for real applications due to the intractability of the update formula derivation.

The main difference between OGD and OMD is in the introduction of Bregman divergence and *Legendre* functions. Other than the L_2 norm, we can set other *Legendre* functions to match the problem structure. We introduce an example to apply OMD for regret minimization problem over the simplex, that is $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}_+^D; \sum_{i=1}^D w^{(i)} = 1\}$ [27]. In this problem setting, the negative entropy has been known as an efficient tool. The negative entropy function is defined for a vector $\mathbf{a} \in \mathbb{R}^D$,

$$\psi(\mathbf{a}) = \sum_{i=1}^{D} a^{(i)} \log a^{(i)} \ . \tag{2.20}$$

The Bregman divergence of this *Legendre* function is given by a well-known Kullback-Leibler divergence as:

$$D_\psi(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{D} a^{(i)} \log \left(\frac{a^{(i)}}{b^{(i)}} \right) \ . \tag{2.21}$$

This Bregman divergence is 1-strongly convex with respect to the $\|\cdot\|_1$ norm on the simplex [25], therefore, Theorem 4 can be applied. In this case, the dual norm is the L_∞ norm, therefore, the bound of G is calculated by the L_∞ norm. This implies that G does not explicitly depend on the dimension size compared with the L_2 norm case, and it contributes to the derivation of tighter upper bound. Furthermore, the projection onto the simplex can be easily computed in this case. Therefore, by appropriately choosing *Legendre* functions, several efficient algorithms can be derived from OMD.

2.3.3 Online Dual Averaging

In the batch learning setting, Dual Averaging (DA) methods [28] have been proposed to solve the drawback inherited in Mirror Descent algorithms. When updating weight vectors per round, algorithms only use the information derived from the current loss function through the subgradient vector. Although they are guaranteed to converge to the best regret value per round compared with the best fixed strategy, this characteristic leads to large variance of a sequence of weight vectors [64]. Especially when data contain noise or some data have characteristics different from other data, the current weight vector drastically move to the wrong direction and weight vectors become worse. To avoid this problem, Dual Averaging methods utilize all previous subgradients in an efficient way to update weight vectors. Because of its effectiveness, many extensions and theoretical analyses of DA have been presented. DA is known as a special case of more general framework, called primal-dual framework, presented b Shalev-Shwartz et al. [60]. In other ways, several researches [65, 66] analyzed a distributed online optimization problem over a network and derived the Dual Averaging based algorithms. They does not have a central node and updates parameters by using local information received from adjacent nodes. Under such circumstances, these work yields notable analyses of regret upper bounds, which are largely affected by the topology of networks.

An online version of Dual Averaging can be derived from the notion of Dual Averaging algorithms in the batch learning setting. We call this framework Online Dual Averaging (ODA) in this thesis. In ODA, we update weight vectors according to the following formula:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{\tau=1}^{t} \langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \frac{1}{\eta_t} B_\psi(\mathbf{w}, \mathbf{w}_1) \right\} , \tag{2.22}$$

where $\{\eta_t\}_{t=1:T}$ is a sequence of positive non-increasing constants. The main difference between OMD and ODA is in the first term. As written in (2.10), OMD only use the t-th subgradient vector. On the other hand, ODA algorithms utilize the summation of subgradient vectors over all rounds. Averaging subgradients contributes to improving the robustness of ODA. To explain this advantage, we assume that algorithms deal with data containing some noises. When the t-th loss function is noisy, the subgradient vector may have the different direction from the ones derived in previous rounds. In OMD, weight vectors move toward the wrong way aggressively in this round because the impact of the first term becomes big. On the other hand, ODA utilizes the average of subgradients generated from $t = 1$ to the present. Even if the t-th loss function is noisy, its effect is weakened by averaging of subgradients. Therefore, ODA algorithms work well in these situations.

The regret bound of ODA can be derived as follows:

Theorem 5. *Let us assume that the Legendre function ψ is β-strongly convex with respect to a norm $\|\cdot\|$ and there exist constants R, G such that $B_\psi(\mathbf{w}, \mathbf{w}_1) \le R^2$ for all $\mathbf{w} \in \mathcal{W}$ and $\|\nabla \ell_t(\mathbf{w}_t)\|_* \le G$ for all $t \ge 1$. When we set positive non-increasing step size $\{\eta_t\}_{t=1:T}$*

and $\eta_0 = \eta_1$, the following inequality holds:

$$\mathrm{Regret}(T) \le \frac{R^2}{\eta_T} + \frac{G^2}{2\beta}\sum_{t=1}^{T}\eta_{t-1} \ . \tag{2.23}$$

As in the OMD case, we could not find the direct proof of regret upper bound of ODA with non-constant step sizes. Therefore, we show the proof below.

Proof of Theorem 5. The proof of this theorem follows the discussion in [28, 64]. First, we define the gap sequence for each round $t \ge 1$ as follows:

$$\delta_t = \max_{\mathbf{w}\in\tilde{\mathcal{W}}}\left\{\sum_{\tau=1}^{t}\langle\nabla\ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}\rangle\right\} \ , \tag{2.24}$$

where $\tilde{\mathcal{W}} = \left\{\mathbf{w}\in\mathcal{W} | B_\psi(\mathbf{w},\mathbf{w}_1) \le R^2\right\}$. This gap value δ_t becomes the upper bound of regret from the convexity of loss functions. This is straightforwardly proved by the following inequality for all $\mathbf{w}\in\tilde{\mathcal{W}}$:

$$\delta_t \ge \sum_{\tau=1}^{t}\langle\nabla\ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}\rangle \ge \sum_{\tau=1}^{t}\left(\ell_\tau(\mathbf{w}_\tau) - \ell_\tau(\mathbf{w})\right) \ . \tag{2.25}$$

In the following, we denote $\sum_{\tau=1}^{t}\nabla\ell_\tau(\mathbf{w}_\tau)$ as $\nabla\ell_{1:t}$ for simplifying the notation in this proof. We derive the upper bound on δ_t. First, we reformulate δ_t as follows:

$$\delta_t = \sum_{\tau=1}^{t}\langle\nabla\ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \max_{\mathbf{w}\in\tilde{\mathcal{W}}}\left\{\langle\nabla\ell_{1:t}, \mathbf{w}_1 - \mathbf{w}\rangle\right\} \ , \tag{2.26}$$

To upper bound the second term, we use the following lemma.

Lemma 6. *Let us define two conjugate-type functions for each round t,*

$$U_t(\mathbf{a}) = \max_{\mathbf{w}\in\tilde{\mathcal{W}}}\langle\mathbf{a}, \mathbf{w}-\mathbf{w}_1\rangle \ , \quad V_t(\mathbf{a}) = \max_{\mathbf{w}\in\mathcal{W}}\left\{\langle\mathbf{a}, \mathbf{w}-\mathbf{w}_1\rangle - \frac{1}{\eta_t}B_\psi(\mathbf{w},\mathbf{w}_1)\right\} \ . \tag{2.27}$$

For any vector $\mathbf{a}$ in the dual space, which is the vector space of all linear functions on $\mathcal{W}$, we have

$$U_t(\mathbf{a}) \le V_t(\mathbf{a}) + \frac{R^2}{\eta_t} \ . \tag{2.28}$$

Proof. From the definition of U_t, we obtain

$$\begin{aligned}
U_t(\mathbf{a}) &= \max_{\mathbf{w}\in\tilde{\mathcal{W}}}\langle\mathbf{a}, \mathbf{w}-\mathbf{w}_1\rangle = \max_{\mathbf{w}\in\mathcal{W}}\min_{\beta\ge 0}\left\{\langle\mathbf{a}, \mathbf{w}-\mathbf{w}_1\rangle + \beta(R^2 - B_\psi(\mathbf{w},\mathbf{w}_1))\right\} \\
&\le \min_{\beta\ge 0}\max_{\mathbf{w}\in\mathcal{W}}\left\{\langle\mathbf{a}, \mathbf{w}-\mathbf{w}_1\rangle + \beta(R^2 - B_\psi(\mathbf{w},\mathbf{w}_1))\right\} \\
&\le \max_{\mathbf{w}\in\mathcal{W}}\left\{\langle\mathbf{a}, \mathbf{w}-\mathbf{w}_1\rangle + \frac{1}{\eta_t}(R^2 - B_\psi(\mathbf{w},\mathbf{w}_1))\right\} \\
&= V_t(\mathbf{a}) + \frac{R^2}{\eta_t} \ .
\end{aligned} \tag{2.29}$$

The first inequality use the standard max-min inequality [37]. In the second inequality, we set $\beta = 1/\eta_t$. □

By applying Lemma 6 to (2.26), we obtain

$$\delta_t \leq \sum_{\tau=1}^{t} \langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1 \rangle + V_t(-\nabla \ell_{1:t}) + \frac{R^2}{\eta_t} . \tag{2.30}$$

In addition, we see that V_t is differentiable and it is η_t/β-smooth with respect to the dual norm $\|\cdot\|_*$ due to the convex/smooth duality introduced in Lemma 2. We can derive the following inequality for each $t \geq 2$:

$$V_{t-1}(-\nabla \ell_{1:t}) \leq V_{t-1}(-\nabla \ell_{1:t-1}) + \langle -\nabla \ell_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_1 \rangle + \frac{\eta_{t-1}}{2\beta} \|\nabla \ell_t(\mathbf{w}_t)\|_*^2 . \tag{2.31}$$

Let us assume $\eta_0 = \eta_1$, then the above inequality is satisfied when $t = 1$ such that

$$V_0(-\nabla \ell_1(\mathbf{w}_1)) \leq V_0(\mathbf{0}) + \langle -\nabla \ell_1(\mathbf{w}_1), \mathbf{w}_1 - \mathbf{w}_1 \rangle + \frac{\eta_0}{2\beta} \|\nabla \ell_1(\mathbf{w}_1)\|_*^2 . \tag{2.32}$$

Furthermore, we obtain the following inequality for each $t \geq 1$,

$$\begin{aligned} V_{t-1}(-\nabla \ell_{1:t}) &= \max_{\mathbf{w} \in \mathcal{W}} \left\{ \langle -\nabla \ell_{1:t}, \mathbf{w} - \mathbf{w}_1 \rangle - \frac{1}{\eta_{t-1}} B_\psi(\mathbf{w}, \mathbf{w}_1) \right\} \\ &\geq \langle -\nabla \ell_{1:t}, \mathbf{w}_{t+1} - \mathbf{w}_1 \rangle - \frac{1}{\eta_{t-1}} B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_1) \\ &= \left\{ \langle -\nabla \ell_{1:t}, \mathbf{w}_{t+1} - \mathbf{w}_1 \rangle - \frac{1}{\eta_t} B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_1) \right\} + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_1) \\ &= V_t(-\nabla \ell_{1:t}) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_1) \\ &\geq V_t(-\nabla \ell_{1:t}) . \end{aligned} \tag{2.33}$$

The first inequality is derived by setting $\mathbf{w} = \mathbf{w}_{t+1}$. The second inequality is satisfied due to non-negativity of Bregman divergence and non-increasing sequence of $\{\eta_t\}_{t \geq 0}$. From these two inequalities, we can obtain the following inequality for $t \geq 1$:

$$\langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_1 \rangle \leq V_{t-1}(-\nabla \ell_{1:t-1}) - V_t(-\nabla \ell_{1:t}) + \frac{\eta_{t-1}}{2\beta} \|\nabla \ell_t(\mathbf{w}_t)\|_*^2 . \tag{2.34}$$

By summing up the above inequalities from $t = 1$ to $t = T$, we obtain

$$\sum_{t=1}^{T} \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_1 \rangle + V_T(-\nabla \ell_{1:T}) \leq V_0(\mathbf{0}) + \sum_{t=1}^{T} \frac{\eta_{t-1}}{2\beta} \|\nabla \ell_t(\mathbf{w}_t)\|_*^2 . \tag{2.35}$$

We note that $V_0(\mathbf{0}) = 0$. By combining (2.30) and (2.35), we obtain the upper bound. □

2.3.4 Follow-The-Regularized-Leader

We introduce several subgradient-based online learning algorithms so far. They have been developed in the field of the online convex optimization. From a different viewpoint, several algorithms and theories have been developed to minimize the regret defined in expert advice problem setting. The relationship between this setting and the online convex optimization has been investigated and these results have been imported to the online convex optimization problems [67, 68, 56]. As the simplest framework of these algorithms, Follow-The-Leader (FTL) [69, 3] has been developed to make a sequential prediction and parameter update. Based on the notion of FTL, several variants of FTL have been proposed and extended as online convex optimization problem solvers.

The update formula of FTL algorithms for online convex optimization problems [67, 56] is defined as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \sum_{\tau=1}^{t} \ell_\tau(\mathbf{w}) \ . \tag{2.36}$$

The intuition of this update formula is very simple. FTL derives weight vectors that minimize the sum of loss function values received until the previous round in each round. Though FTL fully utilizes the information about all previous loss functions, FTL obtains the linear regret bound $\Omega(T)$ as an upper bound in worst case [30].

To achieve a sublinear regret bound based on FTL, Follow-The-Regularized-Leader (FTRL) has been proposed as a variant of FTL. FTRL [70, 30] induces a regularization term to objective functions for stabilizing a sequence of weight vectors. FTRL is a fundamental template for online convex optimization and a number of cutting-edge algorithms have been derived from FTRL; OGD is one of famous examples. By adding a regularization term, FTRL prevents weight vectors from moving wildly from the previous one, which contributes to a sublinear regret bound. FTRL algorithms play the following parameter update procedure in each round:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \left\{ \sum_{\tau=1}^{t} \ell_\tau(\mathbf{w}) + \frac{1}{\eta} R(\mathbf{w}) \right\} \ . \tag{2.37}$$

where $R : \mathcal{W} \to \mathbb{R}$ is a convex regularization function. $\eta > 0$ is a trade-off parameter between the original FTL objective function and a regularization term.

However, the original FTRL framework cannot be applied to massive data analysis. The update formula defined in (2.37) requires the heavy computational cost and large memory space to derive the next weight vector and to store all previous loss functions. Therefore, it is very difficult to directly apply the FTRL algorithms for online convex optimization tasks. Fortunately, for online convex optimization problems, we do not need to solve the problem directly to upper bound the regret. From the convexity property of loss functions, the regret can be upper bounded by the regret of the first-order approximation of the original loss functions for any $\mathbf{u} \in \mathcal{W}$ as follows:

$$\sum_{t=1}^{T} \ell_t(\mathbf{w}_t) - \sum_{t=1}^{T} \ell_t(\mathbf{w}^*) \leq \sum_{t=1}^{T} \langle \mathbf{w}_t, \nabla \ell_t(\mathbf{u}) \rangle - \sum_{t=1}^{T} \langle \mathbf{u}, \nabla \ell_t(\mathbf{w}_t) \rangle \ . \tag{2.38}$$

From this reformulation, we can see that we derive the regret upper bound with respect to the original loss functions by deriving the regret upper bound in the linear approximated loss functions by some algorithms. Therefore, FTRL can use the first-order approximations as the loss functions in the original objective functions. The update formula is modified as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \left\{ \sum_{\tau=1}^{t} \langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \frac{1}{\eta} R(\mathbf{w}) \right\} \ . \tag{2.39}$$

When the regularization term $R(\mathbf{w})$ is β-strongly convex with respect to a norm $\|\cdot\|$ and the trade-off parameter is time-varying, this update formula is exactly the same as the ODA introduced in (2.22).

Furthermore, the regret upper bound of linear-approximated version of FTRL as follows:

Theorem 6. *[61] Let a sequence of weight vectors $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to FTRL udpate formula (2.39). Assume that R is β-strongly convex with respect to a norm $\|\cdot\|$, $\mathbf{w}_1 = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} R(\mathbf{w})$, and $R(\mathbf{w}_1) = 0$. Furthermore, we assume that there exists scalars $R, G \in \mathbb{R}$ such that $R(\mathbf{w}) \leq R^2$ for all $\mathbf{w} \in \mathcal{W}$ and $\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_* \leq G$ for all $\tau \geq 1$. Then, the following inequality is satisfied.*

$$\text{Regret}(T) \leq \frac{R^2}{\eta} + \frac{G^2 T}{\beta} \eta \tag{2.40}$$

When we set $\eta = c/\sqrt{T}$ for some constant $c \geq 0$, we obtain $O(\sqrt{T})$ regret bound. The detail analyses of FTRL has been done in several previous work from the viewpoint of primal-dual approach [41]. Other researches [71, 72] proved that OGD and OMD are instances of FTRL algorithms. By appropriate choosing R, FTRL can be reduced to OGD and OMD.

2.4 Several Extensions

In the previous sections, we review some state-of-the-art subgradient-based online learning algorithms (OGD, OMD, ODA, and FTRL). In this section, we introduce several extensions for subgradient-based online learning algorithms. These extensions make online learning algorithms more applicable and practical for real-world problems by dealing with several difficulties. Some extensions are closely related to our proposed methods introduced in later chapters from the perspectives of motivations and used techniques.

2.4.1 Adaptive Subgradient Methods for Online Learning

We first introduce the adaptive subgradient methods for subgradient-based online learning algorithms [73]. As described above, an appropriate choice of *Legendre* functions is very important for running subgradient-based online learning algorithms such as OMD and ODA. The choice of *Legendre* terms largely affects the predictive performance and convergence speed depending on the geometry of the data. However, in most cases, we cannot know characteristics of data in advance and it is very difficult to appropriately set the *Legendre* function according to the geometry of the data. To achieve a proper choice of the *Legendre* function, an adaptive tuning of *Legendre* functions is one of the desirable solutions.

Duchi et al. [73] have proposed adaptive subgradient methods, call AdaGrad, to achieve an adaptive tuning of *Legendre* functions. AdaGrad dynamically retrieves the geometry knowledge of the data extracted from all previous subgradients and reflects information to adaptive *Legendre* functions used for the next weight vector derivation. AdaGrad introduces a $D \times D$ symmetric matrix $\mathbf{H}_t \succeq 0$ and formalizes $\psi_t(\mathbf{a}) = \langle \mathbf{a}, \mathbf{H}_t \mathbf{a} \rangle$ for D-dimensional vector $\mathbf{a}$ as adaptive *Legendre* functions. $\mathbf{H}_t$ store the information about the geometry of data and adjust the *Legendre* functions according to the geometry of data. We provide the definition of $\mathbf{H}_t$ in a diagonal matrix case below. In this case, non-diagonal components of $\mathbf{H}_t$ are always zero. Diagonal components are defined by using all previously derived subgradients. For each diagonal component, algorithms calculate

$$H_t^{(i,i)} = \sqrt{\sum_{\tau=1}^{t} \left((\nabla \ell_\tau(\mathbf{w}_\tau))^{(i)} \right)^2} \,. \tag{2.41}$$

This matrix term $\mathbf{H}_t$ can be viewed as the approximation of the Hessian in second-order subgradient-based algorithms. AdaGrad has the guarantee of the data-dependent regret bound for OGD, OMD, and ODA with this diagonal symmetric matrix case [73]. This bound is provably as good as the one derived by the best choice of *Legendre* functions in hindsight. Though we note in detail in Chapter 4, the idea of AdaGrad to utilize subgradient information for capturing data characteristics is similar to the idea of our proposed methods for healing a truncation bias. We note that AdaGrad algorithms do not change the objective function itself.

2.4.2 Normalized Online Learning

In most tasks, features are generated from difference multiple sources or views. Therefore, each feature has its own scale of values and large heterogeneity of value ranges exists among features in most tasks. This heterogeneity may affect predictive abilities of learning algorithms badly without any treatment. The standard way to deal with this problem is to apply normalization or standardization as a pre-processing method. These pre-processing methods are used to improve prediction accuracy of algorithms in most batch learning settings.

In the online learning settings, regret bounds of state-of-the-art subgradient-based online learning algorithms explicitly depend on the norm of subgradient vectors and the divergence between two weight vectors. It also implies that the regret bound significantly depends on the norm of input vectors due to the assumption (1.14) and update procedures. Therefore, an appropriate feature scaling is needed to make regret upper bounds reasonable. However, it is difficult to employ pre-processing methods in the online learning setting while maintaining advantages of online learning. Data are prepared in advance or the characteristics of data could change over time in many online learning settings, therefore, we cannot calculate the exact sufficient statistics for applying pre-processing. For streaming data, we cannot access a large amount of data before running algorithms. For massive data, it is very costly to access the whole data for calculating basic statistics. To avoid these difficulties, it is better to develop the dynamic feature scaling method according to the characteristics of data seen until the previous round.

Ross et al. [74] has developed the normalized online learning method to achieve a dynamic feature normalization without pre-processing. This method is based on OGD and makes OGD invariant to feature scaling. As a result, they prove that their proposed method achieve a sublinear regret bound even when the scaling of each feature are determined adversarially. The idea of this method is closely related to the feature-aware regularization method we propose in Chapter 4. We note that this method is unclear how to induce (non-smooth) regularization terms while achieving sublinear regret bound with an adversarial feature scaling.

2.4.3 Importance-aware Update

Sometimes, OGD and other subgradient-based online learning algorithms deal with the data in which each datum or type of prediction error has different importance, such as the classification problems with asymmetric cost [75]. In these cases, we need to update parameters differently according to the importance. A famous method to achieve an update with difference importance is to update parameters by weighing step sizes at a certain scale. However, this method lacks two important properties: the invariance and safety. The invariance property is satisfied when the update with an importance weight h becomes the same result when the same data appears h times in a row. The safety property is satisfied when the relationship between the predicted output $\hat{y} = \langle \mathbf{w}, \mathbf{x} \rangle$ and the true output y does not change before and after the update. More formally, the safety property is satisfied when the following inequality holds:

$$\frac{\langle \mathbf{w}_{t+1}, \mathbf{x}_t \rangle - y_t}{\langle \mathbf{w}_t, \mathbf{x}_t \rangle - y_t} \geq 0 \ , \tag{2.42}$$

where $\langle \mathbf{w}_t, \mathbf{x}_t \rangle - y_t \neq 0$. Because the step size becomes large in a linear scale, the previous method might overshoot when the importance h becomes big. The safety property guarantees to prevent this type of overshooting. Methods without these properties sometimes make the update of online learning algorithms fragile.

To solve these difficulties, the importance-aware update method has been established [76]. This method provides a closed-form update formula with importance h for calculating the next weight vector at one time. This closed-form formula is derived from the solution of ordinary differential equation problems constructed to satisfy the invariance property. By this derivation, for many major convex loss functions such as the hinge loss and the logistic loss, it is guaranteed that this closed-form formula satisfies the above two important properties. Furthermore, proposed algorithms achieve the same regret bound as the original subgradient-based online learning algorithms.

In Chapter 5, we develop a new algorithm in which algorithms dynamically change the importance of the received datum according to the prediction result. To make our proposed algorithms more effective, we incorporate this importance-aware update scheme to our algorithms.

2.5 Mistake-Bound-based Online Learning

As discussed above, most subgradient-based online learning algorithms focus on the regret minimization as an learning objective. However, the regret minimization is not an only way as performance measures of online learning algorithms. Some researches have established other objectives as alternatives to the regret and developed several online learning algorithms to achieve these objectives. For classification problems, many online learning algorithms focus on the minimization of mistake counts as objective functions. In this section, we briefly review the mistake-bound-based online learning framework and algorithms.

2.5.1 Objective: Mistake Bound

For binary classification problems, many work focus on directly minimizing classification mistake counts while running online learning algorithms with linear learner models. In the binary classification setting, prediction mistakes are defined as $y\langle \mathbf{w}, \mathbf{x}\rangle \leq 0$. This definition is equivalent to the case where the hypothesis defined in (1.3) outputs the wrong label. Formally, the mistake count is defined as:

$$\sum_{t=1}^{T} \mathbb{1}[\![y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle \leq 0]\!] \; . \tag{2.43}$$

A notion of this mistake bound had been introduced in researches about Perceptron algorithms [77, 78]. In this setting, we would like to minimize mistake counts, and hopefully algorithms have the guarantee to bound mistake counts. As indicated in Section 2.2, the regret bound on 0/1 loss functions becomes linear with respect to the data size [47]. However, through surrogate loss functions such as the hinge-loss function, we can bound the mistake counts by using the regret upper bound. This idea is called the relative mistake bound and it has been investigated in some work [79, 41].

2.5.2 Mistake-Bound-based Algorithms

A large number of mistake-bound-based online learning algorithms has been developed for solving binary classification tasks efficiently. The oldest online learning algorithms for the minimization of mistake counts is known as the Perceptron [77]. The update formula is written as:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t + \mathbf{x}_t & y_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle \leq 0 \\ \mathbf{w}_t & \text{otherwise} \end{cases} . \tag{2.44}$$

From this update formula, we can see that the the Perceptron updates weight vectors only when the prediction over the received datum fails. When updating parameters, the Perceptron just adds an input vector to weight vectors. While the update procedure of the Perceptron is very simple, the Perceptron has the guarantee to achieve the upper bound of mistake counts [78]. Especially, in the realizable case where there exists a hypothesis in the hypothesis space such that all data are correctly classified, mistake counts is upper bounded by a fixed constant by the Perceptron. In other words, the number of mistakes does not depend on the data size. Due to its simplicity, easiness of implementation, free from hyper-parameters, and theoretical background, many variants of the Perceptron have been proposed, such as margin-Perceptron [80] and second-order Perceptron [81]. Furthermore, many extensions have been proposed, such as a robust distributed learning framework over noisy data [82].

Furthermore, online learning algorithms for the minimization of mistake upper bound has been improved by recent studies, such as Passive-Aggressive algorithms [83] and Confidence-Weighted algorithms [84, 85, 86]. Subgradient-based online learning algorithms for upper bounding mistake counts also exists [87]. They are all guaranteed to upper bound the mistake counts by a constant value in the realizable case. In particular, a family of Confidence-Weighted algorithms [84, 85, 86] is the state-of-the-art framework for online classification tasks. Their algorithms significantly improve the predictive performance and convergence speed compared with the Perceptron algorithms. The idea of Confidence-Weight algorithms is very similar to AdaGrad, in that they incorporate approximated second-order information to the update procedure of weight vectors and dynamically adjust the information extracted from data. Confidence-Weighted algorithms assume that weight vectors obey a Gaussian distribution. Algorithms update the mean and variance of the Gaussian distribution per round by using the received datum.

2.6 Chapter Summary

Online learning is a novel framework for efficiently learning from a large variety of data. In particular, online learning is effective for streaming data, massive data, data with concept drift, and the environment with adversarial data generation. From these reasons, subgradient-based online learning algorithms have been frequently used for solving these problems owing to several additional advantages derived from subgradient-based update procedure, such as its implementation simplicity, efficiency of computational cost and memory usage, and several theoretical guarantees through regret bounds. In this chapter, we introduce the basic framework of subgradient-based online learning and the regret minimization criterion as the major objective. To achieve this objective, many algorithms have been developed. We introduce several state-of-the-art subgradient-based online learning algorithms, Online Gradient Descent (OGD), Online Mirror Descent (OMD), Online Dual Averaging (ODA), and Follow-The-Regularized-Leader (FTRL). We review that these algorithms are closely related to each other and have strong characteristics.

In Chapter 4 and 5, we point out that algorithms and objectives described above have crucial oversights stemmed from several biases. To deal with these biases, we need to reformulate objective functions by integrating effects of these biases into objectives and develop new algorithms for solving these new problems.

Chapter 3

Stochastic Learning

In this chapter, we introduce stochastic learning as another important learning framework based on the incremental learning. To minimize some objective functions, stochastic learning is the framework to iteratively update parameters by using noisy but easily available estimates of the original objective function in each round. When the original objective function cannot be easily accessed or the objective function consists of massive data, stochastic learning methods realize the faster iterative optimization of the objective function by avoiding the costly evaluation and calculation.

In this chapter, we introduce the basic framework and algorithms of subgradient-based stochastic learning. First, we introduce the basic problem setting and parameter update procedure of subgradient-based stochastic learning in Section 3.1. To update weight vectors, algorithms utilize an unbiased estimator of a subgradient of the original objective function in each round. By iteratively updating parameters through unbiased estimates, algorithms pursue to generate the optimal weight vector of the original objective function. After that, in Section 3.2, we mathematically formulate the expected loss minimization as an algorithm performance measure of subgradient-based stochastic learning algorithms. Furthermore, we introduce the two main approaches to upper bound the expected loss through subgradient-based stochastic learning algorithms, the generalization bound and the online to batch conversion. Based on these theories, many state-of-the-art subgradient-based stochastic learning algorithms have been developed. In particular, the online to batch conversion shows the notable relationship between the regret bound in the online learning setting and the expected loss minimization in the stochastic learning setting. This technique enables us to convert state-of-the-art online learning algorithms to stochastic ones while the regret bound analysis can be simultaneously transformed to the expected loss bound analysis. In Section 3.3, we introduce several state-of-the-art subgradient-based stochastic learning algorithms. We show that they have novel theoretical guarantees and practical effectiveness [8]. In Section 3.4, we summarize the discussions of this chapter.

3.1 Subgradient-based Stochastic Learning

In this section, let us introduce the basic parameter update procedure of subgradient-based stochastic learning. The procedure is written as the following scheme:

1. **Initialization:** Start at round $t = 1$. Set a weight vector $\mathbf{w}_1 \in \mathcal{W}$. The weight vector space is restricted in a closed convex set $\mathcal{W} \subset \mathbb{R}^D$.
2. **Random Variable:** At round t, unveil the t-th random variable ξ_t. The random variable ξ_t is sampled from a specific distribution $\mathcal{D}$. The noisy objective function with respect to ξ_t is defined as $f(\cdot; \xi_t) : \mathcal{W} \to \mathbb{R}_+$. We assume that $f(\cdot; \xi_t)$ is convex with respect to the weight vector. The original objective function is defined as $f(\cdot) = \mathbb{E}_{\xi \sim \mathcal{D}}[f(\cdot; \xi)]$.
3. **Subgradient:** Algorithms calculate the t-th subgradient vector to be used to decrease the objective function value. The subgradient vector with respect to the t-th random variable and the t-th weight vector is defined as $\nabla f(\mathbf{w}_t; \xi_t) = \nabla f_t(\mathbf{w}_t)$. This is an unbiased estimator of a subgradient of the original objective function.
4. **Update:** Update weight vectors using the t-th subgradient vector. Algorithms derive the $t+1$-th weight vector $\mathbf{w}_{t+1}$.
5. Increase the round number t and repeat these processes until the stopping criterion is satisfied.

This procedure is based on the subgradient-based incremental learning framework. We update weight vectors according to the t-th unbiased estimator of a subgradient of the original objective function. While weight vectors are updated by using a random variable ξ_t which is stochastically sampled from a specific distribution $\mathcal{D}$, this framework is called stochastic learning. Algorithm 3 summarizes a basic scheme of subgradient-based stochastic learning algorithms.

When we deal with the supervised learning tasks, the above framework is specialized as follows: We receive one datum $\xi_t = (\mathbf{x}_t, y_t)$ as a random variable in each round. A received datum is a pair of a D-dimensional feature vector $\mathbf{x}_t$ taken from a closed convex set $X \subset \mathbb{R}^D$ and the corresponding output value $y_t \in \mathcal{Y}$. Each datum is sampled from a specific distribution $\mathcal{D}$. The t-th noisy objective function is defined as $f_t(\cdot) = f(\cdot; (\mathbf{x}_t, y_t))$.

Algorithm 3 Subgradient-based Stochastic Learning

```
Initialize $\mathbf{w}_1 \in \mathcal{W}$
for $t = 1, \dots$ do
    Receive $t$-th random variable $\xi_t \sim \mathcal{D}$
    Calculate a subgradient with respect to $\xi_t$ as $\nabla f_t(\mathbf{w}_t)$
    Update weight vector and obtain $\mathbf{w}_{t+1} \in \mathcal{W}$
end for
```

3.2 Objective: Expected Loss Minimization

The objective in the stochastic learning setting is to derive the optimal weight vector for minimizing the original objective function. To update weight vectors per round, we only utilize an unbiased estimator of a subgradient of the original objective function.

We mathematically formalize the objective in the stochastic learning setting. Let us assume that there exists a certain data distribution $\mathcal{D}$ and random variables $\{\xi_t\}_{t\geq 1}$ are i.i.d. sampled from $\mathcal{D}$. The objective function $f : \mathcal{W} \to \mathbb{R}_+$ is defined as the expectation with respect to random variables sampled from the data distribution as follows:

$$f(\mathbf{w}) = \mathbb{E}_{\xi\sim\mathcal{D}}\left[f(\mathbf{w};\xi)\right] \ . \tag{3.1}$$

The objective function is called as the expected loss. Algorithms pursue to derive the optimal weight vector $\mathbf{w}^*$ such that it becomes the minimization point of the objective function. Unlike in the online learning setting, weight vectors are evaluated through the finally derived one such as the average weight vector from round $t = 1$ to $t = T$. Other weight vectors are not used to evaluate the performance of stochastic learning algorithms in general. The optimal weight vector is defined as:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) \ , \tag{3.2}$$

and we would like to develop algorithms to derive the optimal weight vector at the end.

The expected loss minimization is the objective of subgradient-based stochastic learning. Unlike in the online learning setting, we assume that data are generated from a data distribution $\mathcal{D}$ in the stochastic learning setting. However, we could not know the exact data distribution $\mathcal{D}$ in most cases. Therefore, it is difficult not only to minimize the expected loss value, but also to evaluate the objective function value directly. To solve this difficult problem, we introduce two major approaches to upper bound the expected loss in this thesis; the generalization bound through the empirical risk minimization and the online to batch conversion through the regret bounds of online learning algorithms. The first approach uses the empirical loss to upper bound the expected loss with some kinds of complexities of the weight vector space. This theory guarantees that the empirical loss minimization derives the tightest expected loss upper bound in this view. Therefore, the empirical loss minimization becomes the crucial objective to minimize the expected loss. In Section 3.3.3, we show that several stochastic learning algorithms are very effective to minimize the empirical loss when the data size is very large. The second approach enables us to convert the regret upper bound of online learning algorithms to the expected loss upper bound by using these algorithms as stochastic ones. Through this theory, novel online learning algorithms introduced in Chapter 2 can be used for stochastic learning problems without any modification of parameter update procedures. We briefly introduce these two approaches below.

3.2.1 Generalization Bound and Empirical Risk Minimization

The first approach is the generalization bound through the statistical learning theory. We introduce the basic scheme of the generalization bound in this subsection. We note that we only focus on binary classification problems in this subsection. For other problem settings such as multiclass classification problems, there are many work to analyze the upper bound of the expected loss through the generalization bound in a different way [88].

To derive the generalization bound, we utilize the union bound through the Rademacher complexity [89]. The empirical Rademacher complexity evaluates the complexity of feasible hypothesis space regarding a set of input vectors. It is mathematically defined as:

$$\Re(\mathcal{W}; \{\xi_\tau\}_{\tau=1:T}) = \mathbb{E}_{\{\epsilon_\tau\}_{\tau=1:T} \sim \{\pm 1\}^T} \left[\sup_{\mathbf{w} \in \mathcal{W}} \frac{1}{T} \sum_{\tau=1}^{T} \epsilon_\tau h(\mathbf{x}_\tau; \mathbf{w}) \right] , \tag{3.3}$$

where $\{\epsilon_\tau\}_{\tau=1:T}$ is i.i.d. sampled $\{\pm 1\}$-valued random variables and $h(\mathbf{x}; \mathbf{w})$ is the hypothesis to predict outputs of the input vector $\mathbf{x}$ through the weight vector $\mathbf{w}$. The expectation is calculated over all $\{\epsilon_\tau\}_{\tau=1:T}$. In binary classification problems, the hypothesis function becomes $h : \mathcal{X} \to \{\pm 1\}$. In general, we use (1.3) as a hypothesis function. The value of the empirical Rademacher complexity is determined by a variety represented by instances in the weight vector space with respect to the sampled data. When a set of hypotheses contains a larger variety of weight vectors and these hypotheses can represent a large variety of output sets, the empirical Rademacher compelxity value becomes bigger.

The empirical Rademacher complexity can be used to upper bound the gap between the expected loss and the empirical loss as follows:

Theorem 7. *[38] We assume $|f(\mathbf{w}, \xi)| \leq 1$ for all $\mathbf{w} \in \mathcal{W}$ and $\xi \sim \mathcal{D}$. With probability of at least $1 - \delta$, for all $\mathbf{w} \in \mathcal{W}$, the following inequality is satisfied:*

$$f(\mathbf{w}) - \frac{1}{T}\sum_{\tau=1}^{T} f_\tau(\mathbf{w}; \xi_\tau) \leq 2\Re(\mathcal{W}; \{\xi_\tau\}_{\tau=1:T}) + 4\sqrt{\frac{2\log(4/\delta)}{T}} \ . \tag{3.4}$$

This theorem shows the relationship between the expected loss and the empirical loss, which is the average function value with respect to the sampled data $\{\xi_\tau\}_{\tau=1:T}$. The difference is upper bounded by using the empirical Rademacher complexity and a $O(1/\sqrt{T})$ term. If the empirical Rademacher complexity is less than $O(T)$, the difference between the expected loss and the empirical loss converges to 0 as a result. Therefore, when this term is bounded by $O(T)$, stochastic learning algorithms can focus on the empirical loss minimization because the average function converge to the original objective function. Therefore, the expected loss minimization can be achieved through the empirical loss minimization even if we do not know the exact structure of data distribution $\mathcal{D}$. Furthermore, the increase of data size contributes to the improvement of the generalization ability; therefore, the demand on systematically utilizing massive data has been increasing. We note that the empirical Rademacher complexity is bounded by $O(1/\sqrt{T})$ when we utilize a set of linear learners as a set of hypotheses under practical constraints [38].

3.2.2 Online to Batch Conversion

Second, we introduce the notion of the online to batch conversion. This conversion enables us to utilize regret upper bounds derived by online learning algorithms for upper bounding the expected loss. This technique guarantees that novel online learning algorithms can be converted into stochastic ones and the derived algorithms have the theoretical guarantee to derive the upper bound of the expected loss [90]. We show how the online to batch conversion is used. We introduce the online to batch conversion procedure as follows: First, we run online learning algorithms with the data $\{\xi_t\}_{t=1:T}$ produced from a specific data distribution $\mathcal{D}$. As a result, we obtain a sequence of weight vectors $\{\mathbf{w}_t\}_{t=1:T}$. Then, we derive representative weight vectors as final outputs. We generate the average weight vector $\bar{\mathbf{w}}_T$ such that

$$\bar{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t \ , \tag{3.5}$$

or randomly pick up one weight vector uniformly at random from $\{\mathbf{w}_t\}_{t=1:T}$. With respect to the average weight vector $\bar{\mathbf{w}}_T$, we obtain the meaningful expected loss upper bound by the following theorem.

Theorem 8. *[30] If the objective function f is convex and $\{\xi_t\}_{t=1:T}$ are i.i.d. sampled from a specific distribution $\mathcal{D}$. Then, the following inequality is satisfied.*

$$\mathbb{E}_{\{\xi_t\}_{t=1:T}\sim\mathcal{D}^T}\left[f(\bar{\mathbf{w}})\right] - f(\mathbf{w}^*) \leq \mathbb{E}_{\{\xi_t\}_{t=1:T}\sim\mathcal{D}^T}\left[\frac{1}{T}\sum_{\tau=1}^{T}\left(f(\mathbf{w}_t;\xi_t) - f(\mathbf{w}^*;\xi_t)\right)\right] . \tag{3.6}$$

The right-hand side of (3.6) is upper bounded by regret per datum defined in (2.2) because the regret bound does not put any assumption on the data generation. Therefore, when some online learning algorithms have sublinear regret bounds, the left-hand side of (3.6) converges to 0 with $T \to \infty$. Furthermore, this result implies that lower regret bounds lead to tighter expected loss upper bounds. Not only for the average weight vector defined in (3.5), but also for randomly chosen weight vectors, it is known that this theorem is satisfied [30].

This online to batch conversion technique can be applied to online learning algorithm with the sublinear regret upper bound we introduced in Chapter 2, such as OGD, OMD, ODA, and FTRL. These novel online learning algorithms can be automatically transformed to stochastic ones by averaging of weight vectors or random sampling of weight vectors and the derived weight vectors have theoretical guarantees.

3.3 Subgradient-based Stochastic Learning Algorithms

Many subgradient-based stochastic learning algorithms have been proposed to efficiently minimize the expected loss. To solve this problem, the convexity of unbiased estimators of a subgradient plays an important role. In addition, stochastic learning algorithms are effective to deal with massive data. Simple subgradient-based stochastic learning algorithms, such as Stochastic Subgradient Descent algorithms, have been experimentally shown to achieve better practical performances than other batch algorithms [8, 91]. In this section, we show several subgradient-based stochastic learning algorithms.

3.3.1 Stochastic Subgradient Methods

One of the simplest subgradient-based stochastic learning algorithms is Stochastic Gradient Descent (SGD) and Stochastic Subgradient Methods [92, 93]. As the same reason as the Online Gradient Descent, we use the term Stochastic Gradient Descent (SGD) as the generic name of these algorithms even when we utilize subgradients to update parameters. Basically, the parameter update procedure of SGD is exactly the same as the one of OGD and can be viewed as a stochastic version of OGD. The update formula is defined as follows:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{W}} \left(\mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t; \xi_t) \right) \ . \tag{3.7}$$

where $\{\eta_t\}_{t=1:T}$ is a sequence of step sizes and $\Pi_{\mathcal{W}}$ is a projection function onto a convex set $\mathcal{W}$ based on the Euclidean distance. As the same as OGD, SGD moves toward the direction where an unbiased estimator of the first-order approximation will be decreased in a steepest way.

SGD has a long history and many theoretical analyses on SGD have been done [92, 94, 93]. It is known [92] that a derived weight vector of SGD converges to the optimal weight vector with probability one under several conditions such that

$$\mathbb{E}_{\xi \sim \mathcal{D}} \left[\left(\nabla f(\mathbf{w}; \xi) - \nabla f(\mathbf{w}) \right)^2 | \mathbf{w} \right] < \infty \ , \tag{3.8}$$

$$\lim_{t \to \infty} \eta_t = 0 \ , \ \sum_{t=1}^{\infty} \eta_t = \infty \ , \ \sum_{t=1}^{\infty} \eta_t^2 < \infty \ . \tag{3.9}$$

SGD achieves the $O(1/\sqrt{T})$ asymptotic convergence rate when the original objective function is convex [26]. The non-aymptotic convergence analysis is recently done and $O(1/\sqrt{T})$ convergence rate has been derived [95]. Properties of computational cost and memory usage is also the same as the one of OGD defined in Section 2.3.1.

As a technique to derive robust weight vectors, the average of the iterates is known as helpful method. The average of the iterates is mathematically defined as:

$$\bar{\mathbf{w}}_T = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t \ . \tag{3.10}$$

This method is known as Polyak-Ruppert averaging [96]. This averaging method, or its polynominal-decay weighted version [95], derives the optimal convergence rate when we set an appropriate step size.

Through the online to batch conversion, we can derive the expected loss upper bound of SGD through the regret bound of OGD. The following theorem shows the relationship between these two bounds.

Theorem 9. *[90] Assume that the conditions set in Theorem 2 are satisfied. Then, the following expected loss upper bound is satisfied when we derive $\{\mathbf{w}_t\}_{t=1:T+1}$ by running OGD.*

$$\mathbb{E}_{\{\xi_t\}_{t=1:T} \sim \mathcal{D}^T} \left[f(\bar{\mathbf{w}}_T) \right] - f(\mathbf{w}^*) \leq \mathrm{Regret}(T) = \frac{2\sqrt{2}RG}{\sqrt{T}} \ , \tag{3.11}$$

where

$$\bar{\mathbf{w}}_T = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t \ . \tag{3.12}$$

Therefore, SGD achieves $O(1/\sqrt{T})$ convergence rate for the general convex objective function through the online-to-batch conversion. By applying the same technique, SGD obtains $O(\log T/T)$ convergence rate for the strongly convex case through the result of the $O(\log T)$ regret bound of OGD [56]. However, the optimal convergence rate of subgradient-based stochastic learning algorithms for strongly convex problems is known as $O(1/T)$ [97, 98]. The optimal upper bound rate of regret by subgradient-based online learning algorithms for strongly convex loss functions is known as $O(\log T)$ [98]. Therefore, there is a gap between the best convergence rate obtained by the online to batch conversion and the true optimal rate of stochastic learning algorithms. This optimal rate can be achieved through a slight modification of SGD by introducing the intra-epoch procedure [98].

Due to its simplicity, many extensional algorithms have been proposed. To reduce the computational cost of the projection onto a convex set $\mathcal{W}$, SGD with only one projection method has been developed [99]. This method guarantees the convergence toward the optimal point without the projection onto the feasible region in each round.. They prove that their developed algorithm requires only one projection onto $\mathcal{W}$ to achieve $O(1/\sqrt{T})$ convergence rate for general convex optimization problem and $O(\log T/T)$ for strongly convex problem. In addition, Hazan et al. has developed the projection-free online and stochastic learning algorithms using the Frank-Wolfe technique [100]. Though the convergence rate and regret bound becomes worse or at least the same as the standard OGD and SGD, their algorithms are quite efficient when the projection onto the weight vector space requires a large amount of computational time.

An appropriate learning rate setting of SGD is also very important to achieve a good predictive performance. To achieve an adaptive tuning of step sizes, some researches focus on how to possess the approximated information of the Hessian without any burden of computational cost and memory usage [101]. Schaul et al. has developed a method for on-the-fly estimations of the variance of the subgradients and for incorporating these information into the adaptive choice of step sizes to efficiently minimize the expected loss. Experimental results show that this auto-tuning method outperforms the basic SGD and AdaGrad [73] with respect to the predictive performance and the convergence speed.

3.3.2 Pegasos

SGD and its variants show strong effectiveness for several machine learning tasks. To show its effectiveness of the stochastic learning methods, we introduce Pegasos [102]. This algorithm is a well-known SGD-based learning framework for Support Vector Machine (SVM) [103] and fully utilize the online to batch conversion technique for deriving efficient online and stochastic learning algorithms in the same way. As a result, Pegasos provides a family of optimization methods to solve the SVM problem in many ways, such as the mini-batch, online, and stochastic learning framework. In this section, we focus on the stochastic learning variant of Pegasos.

Pegasos has been developed to solve the SVM optimization problem in a primal form efficiently. The primal objective function of the SVM is explicitly described as follows:

$$f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\max(0, 1 - y\langle\mathbf{w},\mathbf{x}\rangle)\right] . \tag{3.13}$$

where $\mathcal{D}$ is the data distribution and λ is the hyper-parameter to adjust the importance ratio between the expected loss function and the regularization term. The weight vector space $\mathcal{W}$ is set to $\mathbb{R}^D$ and the hinge loss is used as loss functions. We use $\ell(\mathbf{w}; (\mathbf{x}, y)) = \max(0, 1 - y\langle\mathbf{w},\mathbf{x}\rangle)$ in this section.

We describe the parameter update procedure of Pegasos. Pegasos receives one datum sampled from a specific distribution $\mathcal{D}$ and received the following objective function in each round t,

$$f_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) . \tag{3.14}$$

Equation (3.14) represents the objective function with respect to only the t-th datum $(\mathbf{x}_t, y_t)$.

Pegasos performs two-step update procedures; the subgradient-based parameter update step and the projection step. In the first step, we use the subgradient of f_t at $\mathbf{w}_t$ to update weight vectors. We denote its subgradient as $\nabla f_t(\mathbf{w}_t)$. Weight vectors are updated as follows and we obtain a new weight vector $\mathbf{w}_{t+\frac{1}{2}}$:

$$\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t \nabla f_t(\mathbf{w}_t) , \tag{3.15}$$

where

$$\nabla f_t(\mathbf{w}_t) = \lambda\mathbf{w}_t - \mathbb{1}[\![1 - y_t\langle\mathbf{w}_t,\mathbf{x}_t\rangle \geq 0]\!] y_t\mathbf{x}_t . \tag{3.16}$$

and $\eta_t > 0$ is the positive learning rate.

In the second step, weight vectors are projected onto the region where the optimal solution exists. $\mathbf{w}_{t+\frac{1}{2}}$ is projected onto the set B such that

$$B = \{\mathbf{w} : \|\mathbf{w}\|_2 \leq 1/\sqrt{\lambda}\} . \tag{3.17}$$

Although the weight vector space is defined as the whole Euclidean space, the optimal solution $\mathbf{w}^*$ is necessarily in the set B. This property is verified through the convexity of the objective function, the optimality condition about the subgradient of (3.13), and the norm of the subgradient of the hinge loss function. The projection can be performed by the closed-form formula. The explicit update formulation is as follows:

$$\mathbf{w}_{t+1} = \min\left\{1, \frac{1/\sqrt{\lambda}}{\|\mathbf{w}_{t+\frac{1}{2}}\|_2}\right\}\mathbf{w}_{t+\frac{1}{2}} . \tag{3.18}$$

$\mathbf{w}_{t+1}$ is obtained by scaling of $\mathbf{w}_{t+\frac{1}{2}}$. Pegasos performs these two procedures each time algorithms receive a new datum.

First, we analyze the online learning aspect of theoretical property of Pegasos. The objective function of Pegasos (3.13) is strongly convex with respect to weight vectors. It is because (3.13) consists of non-smooth convex loss functions and a strongly convex regularization term, the L_2 norm. By effectively utilizing this property, Pegasos has been proved to obtain a logarithmic scale of regret upper bound.

Lemma 7. *[102] Assume that there exists $R \in \mathbb{R}$ such that for all t, $\|\mathbf{x}_t\|_2 \leq R$. Let $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to Pegasos's update rule. When we set learning rates as $\eta_t = 1/\lambda t$, for $T \geq 3$ and for any $\mathbf{u} \in \mathcal{W}$, we have*

$$\sum_{t=1}^{T} f_t(\mathbf{w}_t) - \sum_{t=1}^{T} f_t(\mathbf{u}) \leq \frac{c(1+\log(T))}{2\lambda} , \tag{3.19}$$

where $c = (\sqrt{\lambda} + R)^2$.

By using the online to batch conversion, Pegasos achieves $O(\log T/T)$ upper bound of the expected loss.

Theorem 10. *[102] Assume that conditions used in Theorem 7 are satisfied and data $\{\xi_t\}_{t=1:T}$ are i.i.d. generated according to the data distribution $\mathcal{D}$. Then, for $T \geq 3$, we have*

$$\mathbb{E}_{\{\xi_t\}_{t=1:T}\sim\mathcal{D}^T}\left[f(\bar{\mathbf{w}})\right] \leq f(\mathbf{w}^*) + \frac{c(1+\log(T))}{2\lambda T} , \tag{3.20}$$

where $c = (\sqrt{\lambda} + R)^2$, and

$$\bar{\mathbf{w}} = \sum_{t=1}^{T} \mathbf{w}_t, \quad \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}) . \tag{3.21}$$

The above theorem guarantees that the average weight vector converges to the optimal weight vector in expectation without the direct access to the original objective function. The expected loss can be bounded through a randomly sampled weight vector from a set $\{\mathbf{w}_t\}_{t=1:T}$ [102]. In several experiments, Pegasos works very well for massive data, which means T is very large, compared with other state-of-the-art methods such as the cutting-plane method [104] and the decomposition method [105]. This result implies that subgradient-based stochastic learning algorithms are very efficient for massive data analyses as a optimization solver for SVM. Pegasos quickly optimize parameters in weight vectors due to the low computational cost and memory usage in each round. Though it is slow to fully optimize the minimization over the empirical loss compared with non-stochastic optimization methods due to slow convergence rate, however, it quickly obtains a generalized weight vector with respect to the expected loss [8].

3.3.3 Toward Linearly Convergence of SGD

From the perspective of the generalization bound analyses, efficient optimization methods for minimizing the empirical loss are also needed. To achieve this objective, the main disadvantage of subgradient-based batch learning algorithms is heavy computational cost of the full subgradient calculation over all data as discussed above. On the other hand, subgradient-based stochastic learning algorithms update weight vectors very fast in each round. However, the convergence speed to the optimal solution becomes very slow compared with the batch learning algorithms. Though stochastic learning algorithms are efficient to quickly derive a good weight vector in practice, these algorithms have not been appropriate to exactly minimize the empirical loss due to its slow convergence speed. In particular, for strongly convex objective functions, the gap between the subgradient-based batch learning algorithms and stochastic ones is significantly large. To overcome this difficulty, several recent work has developed some subgradient-based stochastic algorithms to achieve linear convergence rate for strongly convex objective functions with finite samples. These methods have both advantages: linear convergence rate of subgradient-based batch learning algorithms and fast iteration in stochastic learning algorithms by avoiding full subgradient calculation.

We redefine the problem setting. We assume that data $\{\xi_n\}_{n=1:N}$ is finite and fixed, and we pursue to minimize the empirical loss. In this problem setting, the objective function is defined as the minimization problem of the empirical loss:

$$\hat{f}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{w};\xi_n) \ . \tag{3.22}$$

The minimization over the empirical loss becomes crucial for the expected loss minimization through the generalization bound analysis introduced in Section 3.2.1. This effect becomes more effective when we deal with a large amount of data because the empirical loss converges to the expected loss as $N \to \infty$.

Roux et al. has first proposed algorithms to achieve linear convergence rate in a stochastic learning way [106]. This algorithm is named as Stochastic Average Gradient (SAG). The overview of SAG is as follows: The important difference between SGD and SAG is that SAG preserves the latest calculated subgradients of each datum in the memory and utilize them to update parameters in each round. In each round, SAG receives one datum randomly sampled from a fixed set of data and derive the subgradient with respect to the received datum at the current weight vector. Then, SAG replaces the stored subgradient with respect to the received datum by the newly calculate one. Other stored subgradients remain the same. After replacing subgradients, SAG update weight vectors by using the average of stored subgradients. In summary, SAG maintains the latest subgradients for each datum and used them in the parameter update procedure. The update formula of SAG is mathematically defined by the following formula:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \frac{\eta_t}{N}\sum_{n=1}^{N} \tilde{\mathbf{g}}_{t,n} \ , \tag{3.23}$$

where N is the size of the data and

$$\tilde{\mathbf{g}}_{t,n} = \begin{cases} \nabla f(\mathbf{w}_t;\xi_t) & \text{if } \xi_t = \xi_n \\ \tilde{\mathbf{g}}_{t-1,n} & \text{otherwise} \end{cases} \tag{3.24}$$

From this update formula, we can see that SAG can be viewed as the intermediate algorithm between the batch gradient descent methods and SGD. In place of full subgradients, SAG use the average of stored subgradients to avoid the full calculation of subgradients over all data. To improve the convergence rate, we utilize stored subgradients in addition to the currently received subgradient.

The convergence analysis guarantees that SAG has a linear convergence rate for strongly convex objective function.

Theorem 11. *[106] Let us assume that each n-th objective function $f(\cdot;\xi_n)$ is convex, differentiable, and their gradients are L-Lipschitz continuous with respect to the norm $\|\cdot\|$.*

Moreover, we assume that the objective function is μ-strongly convex with respect to the norm $\|\cdot\|$ and there exists a scalar σ^2, R such that $\sigma^2 = \|\nabla f(\mathbf{w}^*; \xi_n)\|^2/N$ where $\mathbf{w}^*$ is the unique minimizer of the empirical loss and $\|\mathbf{w}_1 - \mathbf{w}^*\| \leq R$. In this case, the following inequality is satisfied:

$$\mathbb{E}\left[\hat{f}(\bar{\mathbf{w}}_T)\right] - \hat{f}(\mathbf{w}^*) \leq \left(1 - \min\left\{\frac{\mu}{16L}, \frac{1}{8N}\right\}\right)^T \left(\hat{f}(\mathbf{w}_1) - \hat{f}(\mathbf{w}^*) + \frac{4LR^2}{N} + \frac{\sigma^2}{16L}\right) , \tag{3.25}$$

where the expectation is calculated over a sequence of data sampling from the fixed data.

From this result of the theorem, we can see that the convergence rate of SAG becomes linear. Furthermore, the computational cost per round becomes $O(D)$ because we can calculate the average subgradient by using the following equation:

$$\frac{1}{N}\sum_{n=1}^{N} \tilde{\mathbf{g}}_{t,n} = \frac{1}{N}\left(\sum_{n=1}^{N} \tilde{\mathbf{g}}_{t-1,n} + \nabla f(\mathbf{w}_t; \xi_t) - \tilde{\mathbf{g}}_{t-1,n_t}\right) , \tag{3.26}$$

where n_t is the datum ID which SAG receives at round t. Therefore, the computational cost of update procedure becomes independent of data size. On the other hand, the memory usage cause some difficulties. In the normal case, SAG have to store subgradients with respect to all data, so, it may be impractical for massive data analysis from the viewpoint of the memory size constraint. However, in the linear learner case, algorithms do not need to store subgradients explicitly and only need to store scalar values due to the assumption (1.14).

Many variants and extensions of SAG have been developed. For example, Johnson and Zhang has recently proposed the algorithm named Stochastic Variance-reduced Gradient (SVRG) [107]. This method exploits the essence to achieve a linear convergence property of subgradient-based stochastic learning algorithms for strongly convex objective function. Based on these discussions, they develop SVRG without the need to store the subgradient information in the memory. Furthermore, the proof of linear convergence property of SVRG is quite simple compared with the other competitive methods, such as SAG, the Minimization by Incremental Surrogate Optimization (MISO) [108], and the Stochastic Dual Coordinate Descent [109].

3.4 Chapter Summary

In this chapter, we review the framework and algorithms of subgradient-based stochastic learning. Stochastic learning is a novel optimization framework for solving several difficult tasks. We first introduce the stochastic learning framework. We receive a noisy but easily available estimate of the original objective function in each round. Then, algorithms calculate the subgradient with respect to the received datum and update parameters by utilizing the subgradient. After that, we introduce the objective of stochastic learning framework, the expected loss minimization. Stochastic learning algorithms focus on the derivation of the optimal weight vector such that it minimizes the expected loss. Then, we show two techniques for upper bounding the expected loss, the generalization bound and the online to batch conversion. Especially, the online to batch conversion technique ensures that novel online learning algorithms introduced in Chapter 2 can be transformed to stochastic ones and the expected loss bounds of these algorithms are automatically derived. Based on above discussions, we introduce several subgradient-based stochastic learning algorithms. The advantages and disadvantages of these algorithms are very similar to the ones of subgradient-based online learning algorithms. Furthermore, we show the recent progress about linear convergence methods based on the subgradient-based stochastic learning algorithms for strongly convex problems. These algorithms are very effective to minimize the expected loss through the minimization of the empirical loss and the generalization bound. In Chapter 5, we point out that these algorithms and objectives have crucial deficits originated from a human cognitive bias. We show that objectives and algorithms are needed to be modified and the derived ones are efficient from both theoretical and practical aspects.

Chapter 4

Feature-aware Regularization for Sparse Online Learning

We first introduce Sparse Online Learning in this chapter. Compact predictive models are desirable for efficiently solving supervised learning problems with massive data. Compact predictive models consist of a small amount of effective features, therefore, these models enable us to make predictions faster by using only a small amount of memory. To make predictive models compact, we introduce sparsity-inducing regularization methods for inducing sparsity to predictive models. Sparsity-inducing regularization methods make predictive models compact. The most famous example of sparsity-inducing regularization methods is the L_1-regularization. The L_1-regularization triggers the truncation of parameters that are ineffective for precise predictions and the minimization of the loss functions. On the other hand, online learning generally requires less computational cost and a smaller memory space than batch learning as written in Chapter 2. By combining subgradient-based online learning algorithms with these sparsity-inducing regularization techniques, online learning with sparsity-inducing regularization methods has gained a great deal of attentions as one of the most sophisticated methods for learning from massive data. We call subgradient-based online learning with sparsity-inducing regularization methods sparse online learning. Due to these advantages, sparse online learning framework has recently been analyzed and many algorithms have been developed.

We point out that conventional sparse online learning algorithms do not consider information on features in input vectors. Each feature has distinct characteristics such as its value range and occurrence frequency. These characteristics are usually not uniform in most tasks such as natural language processing and pattern recognition. It is well-known that some rare features are informative for precise prediction, therefore, several techniques have been developed to emphasize such features and retain them in the predictive model through pre-processing methods. For example, TF-IDF [110] is the state-of-the-art pre-processing method to normalize these heterogeneity and emphasize rarely occurring features for natural language processing tasks. However, to apply this type of pre-processing, machines have to load all data into the memory, sums up the occurrence counts of each feature in prior, and rewrites all feature values before we start to run machine learning algorithms. This requirement does not match the essence of online learning framework wherein data are sequentially processed and all data do not have to be prepared in advance. Therefore, pre-processing methods are difficult to be employed while preserving advantages of online learning algorithms. We point out that many unemphatic features are easily truncated without any treatment for modifying these heterogeneity, even though such features are crucial for making a precise prediction. As feature occurrence frequencies or value ranges are highly skewed in many tasks, informative features tend to be truncated beyond necessity when these features are rare or have low value range. Similar problems are indicated by [74], however, they only deal with the feature value range problem, and in addition, their developed methods cannot incorporate non-smooth regularization terms, such as sparsity-inducing regularization methods. We treat this type of bias as the truncation bias and we have thus developed extensional framework for online learning with sparsity-inducing regularization. This bias originates from the problem inside objective functions, therefore, we need to dynamically modify the objective function depending on the truncation bias.

In this chapter, we introduce the a new extensional method for sparse online learning to solve this difficulty. Our framework enables us to eliminate the bias for retaining informative features in an online setting without any pre-processing. By integrating the dynamic adjustment framework of this bias, these algorithms retain the weight of rare or low value range but informative features. Our methods enhance the conventional L_1-regularization by integrating the information on all previous subgradients of loss functions. These extensions enable online learning to automatically and dynamically adjust the intensity of each feature's truncation. This extension enables us to identify and retain infrequently occurring but informative features in an online setting. The important thing is that the optimal solution changes according to the dynamic adjustment effect even when the data are sampled from a fixed data distribution. Furthermore, we establish that our methods achieve the same order of computational complexity and regret upper bound rate as the sparse online learning algorithms. Experimental results demonstrate that our extensional algorithms outperform conventional online learning with sparsity-inducing regularization methods in many classification tasks.

The constitution of this chapter is as follows: In Section 4.1, we introduce the notion of sparse online learning, sparsity-inducing regularizations, and several state-of-the-art subgradient-based sparse online learning algorithms as the background of this chapter. In Section 4.2, we introduce the truncation bias. The truncation bias originates from the skewness of occurrence frequencies and value ranges of features. We show how these biases affect the predictive performance by using toy examples. In Section 4.3, we propose our new sparsity-inducing regularization framework, called feature-aware regularization. The key idea behind our framework is to integrate the information about feature occurrence frequencies and value ranges absolute through subgradients of loss functions into the regularization term. We show that our framework dynamically weakens truncation effects on rare or low value range features and thereby obtains a set of important features regardless of their occurrence frequency or value range. In addition, we develop a unified extension method for many subgradient-based online learning and three applications of our framework in this section. Section 4.4 discusses the upper bound of regret for our derived algorithms. We analyzed the theoretical aspects of our framework and derived the same computational cost and the same order of the regret upper bound as those for the original algorithms in this section. Section 4.5 describes several experimental results. We compared our methods with other subgradient-based sparse online algorithms and evaluated the performances. The results reveal that our framework improve the predictive performance of state-of-the-art algorithms on most datasets with a smaller size of predictive models. Section 4.6 concludes the discussion of this chapter.

4.1 Background: Sparse Online Learning

In this section, we introduce the notion of sparse online learning and several state-of-the-art algorithms. Sparse online learning has been focused on as one of the most sophisticated methods for learning from large-scale data. Subgradient-based sparse online learning have several effective advantages such as less computational time and a smaller memory space originating from subgradient-based online learning framework and compact predictive models induced by sparsity-inducing regularization methods. As described before, compact predictive models contribute to speed up predictions and reduce memory usage. However, to achieve efficient subgradient-based sparse online learning algorithms, there were many challenging tasks to obtain a sparse solution while preserving online learning's computational advantages. Consequently, many optimization methods and learning algorithms have been developed to attain an efficient modeling and framework of subgradient-based sparse online learning as a way of solving large-scale problems.

We first introduce what sparsity-inducing regularization is and how it works in Section 4.1.1. After that, we show how the objective of subgradient-based online learning changes by adding sparsity-inducing regularization terms. In the end, we introduce three state-of-the-art subgradient-based sparse online learning algorithms in later sections. In this thesis, we focus on three state-of-the-art algorithms.

- COMID (Composite Objective MIrror Descent) [111, 61];
- RDA (Regularized Dual Averaging) [64];
- FTPRL (Follow-The-Proximally-Regularized-Leader) [71, 72];

4.1.1 Sparsity-inducing Regularization

First, we introduce the basic of sparsity-inducing regularization methods and show several examples in this section. As described in Section 1.1.3, regularization is a well-known technique to obtain desirable structures in accordance with task objectives. Regularization is widely used for many reasons, such as the improvement of the generalization ability by avoiding overfitting. We integrate a regularization term into an objective function to apply regularization techniques.

Sparsity-inducing regularization is a family of regularization methods to enforce a sparse representation on weight vectors. In other words, sparsity-inducing regularization methods make a large amount of elements zero. By making the derived predictive model sparse, it is capable of reducing the computational time and required memory space for predictions because a sparse representation of weight vectors has a small amount of informative information. L_1-regularization is a simple and famous example of sparsity-inducing regularization. The L_1-regularization term is defined as follows:

$$\Phi(\mathbf{w}) = \lambda \|\mathbf{w}\|_1 \ , \tag{4.1}$$

where $\lambda > 0$ is a trade-off parameter to adjust the importance ratio between the original objective function and the sparsity-inducing regularization term. The functionality of L_1-regularization is to eliminate parameters that are insignificant for prediction on the fly. These sparsity-inducing regularization methods have an effect to automatically truncate uninformative parameters. Moreover, L_1-regularization has the side effect of preventing over-fitting to training data by eliminating non-informative features especially when the number of data is small. From these reasons, L_1-regularization is an effective technique for learning. The disadvantage of L_1-regularization is that it makes objective functions non-smooth due to the non-smoothness of the L_1 norm. Due to its non-smoothness, L_1-regularization has the sparsity-inducing effect, however, at the same time, the minimization over objective functions with L_1-regularization becomes difficult to be solved efficiently compared with smooth ones. For solving minimization problems with non-smooth sparsity-inducing regularization, several optimization methods have been proposed, such as COMID, RDA, and FTPRL. We focus on L_1-regularization as a sparsity-inducing norm in this chapter.

We note that many other sparsity-inducing regularization methods exist. Elastic net [112] is defined as a combination of the L_1 norm and the L_2 norm. This regularization method is very effective to capture all informative features for predictions even if these features are largely correlated with each other. This property contributes to the analysis of what type of features are effective for predictions. OSCAR [113] combines the L_1 norm and the pair-wise L_∞ norms. OSCAR has a stronger effect for capturing all informative features than Elastic net because of the strong grouping effect of the pair-wise L_∞ norms. There are many other sparsity-inducing regularization [114, 115] and these are used for many tasks [116, 117, 118]. In particular, we have proposed an efficient solver for convex optimization problem with L_2 and L_0 regularization terms through the dual decomposition technique [115].

4.1.2 Objective: Regret with Regularization Term

By adding a convex regularization term to objective functions, objectives in the sparse online learning setting change from the regret minimization over the sum of loss functions to the regret minimization over the sum of loss functions and regularization terms. Our main focus in sparse online learning is to derive weight vectors to minimize the next coming loss function added at regularization terms, therefore, objective functions are reformulated as follows:

$$\sum_{t=1}^{T} \left(\ell_t(\mathbf{w}) + \Phi(\mathbf{w})\right) \ , \tag{4.2}$$

where Φ is a regularization term of the form $\Phi : \mathcal{W} \to \mathbb{R}_+$ where Φ is convex in the weight vector space $\mathcal{W}$. For optimization methods over objective functions with sparsity-inducing regularization terms in the batch learning setting, proximal gradient methods have been studied in many researches [114]. These optimization methods pursue to derive the optimal weight vector by using iterative update procedures in the batch-learning setting. The idea of the proximal gradient method is to split objective functions into two parts, differentiable terms and non-differentiable (non-smooth) terms, and we optimize parameters through the first-order approximations of the former formula and the exact form of the latter formula in each iteration. Especially, when an objective function consists of the sum of convex smooth loss functions and the L_1-regularization (or other non-smooth regularization methods), proximal gradient methods are very effective to optimize these parameters [24, 119, 120]. There is a thorough survey of proximal gradient methods and their statistical properties [121].

For online learning setting, we use the regret as an objective function. The definition of regret also changes when regularization terms are integrated. A new definition of the regret in sparse online learning setting is as follows:

$$\mathrm{Regret}_{\ell+\Phi}(T) = \sum_{t=1}^{T} \left(\ell_t(\mathbf{w}_t) + \Phi(\mathbf{w}_t)\right) - \min_{\mathbf{u}\in\mathcal{W}} \sum_{t=1}^{T} \left(\ell_t(\mathbf{u}) + \Phi(\mathbf{u})\right) \ . \tag{4.3}$$

In the following subsections, we introduce three major subgradient-based online learning frameworks that have been developed to deal with these sparsity-inducing regularization methods in an exact form in the update procedure.

4.1.3 Composite Objective Mirror Descent (COMID)

It has been a challenging task to derive a sparse solution and at the same time to preserve online learning framework's advantages. Conventional subgradient-based online learning algorithms, such as OGD and OMD, cannot derive a sparse solution by simply integrating sparsity-inducing regularization methods because a first-order approximation of sparsity-inducing regularization methods does not work to clip parameters to zero. This problem originates from its update procedure, that is to say, while it uses the linear approximation of non-smooth sparsity-inducing norms, not original regularization terms, cannot maintain characteristics of sparsity-inducing regularization methods to update parameters. To overcome the problem, we have to deal with sparsity-inducing regularization without first-order approximation of these terms in the update procedure.

We first introduce Composite-Objective Mirror Descent (COMID) [61] as a method to provide an efficient way to achieve sparse online learning. This framework can be viewed as a unified framework of online subgradient methods [46], online mirror descent [3, 60, 30], and FOBOS (forward-backward splitting) [111]. COMID enables regularization terms to be induced into objective functions even if these regularization terms are non-smooth, and minimize regret with regularization terms by subgradient-based update procedures in an online way. When we set sparsity-inducing regularization as regularization terms, COMID has both advantages of online learning framework and sparsity-inducing regularization methods, such as its implementation simplicity, computation and memory efficiency, and variety of loss function and regularization term choices. The update formula of COMID is written as follows and we solve the following optimization problem in each round to derive the next weight vector.

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \{\eta_t \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w} \rangle + \eta_t \Phi(\mathbf{w}) + B_\psi(\mathbf{w}, \mathbf{w}_t)\} \ , \tag{4.4}$$

where $\nabla \ell_t(\mathbf{w}_t)$ is a subgradient of t-th loss function ℓ_t at $\mathbf{w}_t$, $\{\eta_t\}_{t \geq 1}$ is a sequence of positive non-increasing constants, and B_ψ is the Bregman divergence.

The update formula (4.4) consists of a sum of three terms. The first term indicates that the objective function value will decrease when the next weight vector moves in the reverse direction of the current subgradient. In other words, weight vectors are updated in such a way that its value decreases with respect to the first-order approximation of t-th loss function at the current weight vector. The second term is a regularization term. When we set sparsity-inducing regularization terms, it is hoped that the derived weight vector become sparse. The third term is the Bregman divergence between the current weight vector and the next one. The farther the next weight vector moves away from the current vector, the bigger the Bregman divergence becomes. By inducing this Bregman divergence, a weight vector is updated in a way that it stays close to the current weight vector. The next weight vectors are derived by minimizing the sum of these three terms. Variables $\{\eta_t\}_{t \geq 1}$ control a relative importance of these terms.

COMID can be viewed as the extension of OMD introduced in Section 2.3.2. By inducing a regularization term Φ into the right-hand side of formula (2.10), the update formula of COMID (4.4) is derived. Therefore, when we remove Φ from the update formula of COMID, COMID is reduced to OMD. The upper bound on the regret has been proven to be $O(\sqrt{T})$ by properly setting $\{\eta_t\}_{t \geq 1}$ [61]. We note that the regret is defined by (4.3), not (2.2).

To realize sparse online learning, the splitting method approach was first proposed with theoretical guarantee. COMID is a generalized version of this splitting method approach. Carpenter [122] proposed a splitting approach that combines OGD/SGD with L_1-regularization while maintaining advantages of both techniques. Duchi and Singer [111] and Langford et al. [123] generalized Carpenter's method to achieve sparse online learning based on OGD, which enables to utilize different type of regularization terms. In particular, Duchi and Singer's algorithm is called FOBOS [111]. FOBOS has been proven that FOBOS has asymptotically $O(\sqrt{T})$ regret upper bound under practical conditions.

We show the parameter update procedure of online subgradient method with L_1 regularization as the example of FOBOS. In FOBOS, the parameter update procedure consists of two steps in each round, i.e., a parameter update step and a regularization step. In the first step, weight vectors are updated to decrease the t-th loss function value by using the received t-th datum. We apply online subgradient method to update parameters for pushing down the current loss function value. In this step, weight vectors are updated as:

$$\mathbf{w}_{t+1/2} = \mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t) , \tag{4.5}$$

where $\nabla \ell_t(\mathbf{w}_t)$ is a subgradient of the t-th loss function ℓ_t with respect to $\mathbf{w}_t$ and $\{\eta_t\}_{t \geq 1}$ is a sequence of positive step sizes. We do not consider regularization terms in this step.

Then, regularization terms have been applied in the second step. In this example, we focus on the L_1-regularization case. As described before, L_1-regularization is very effective to derive sparse predictive models. At second step, we apply L_1-regularization to weight vectors. We penalize the current weight vector through sparsity-inducing regularization terms as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \left\{ \frac{1}{2} \|\mathbf{w} - \mathbf{w}_{t+1/2}\|_2^2 + \lambda \eta_{t+1/2} \|\mathbf{w}\|_1 \right\} , \tag{4.6}$$

where $\{\eta_{t+1/2}\}_{t \geq 1}$ is a sequence of learning rates used in the second step and λ is the trade-off parameter to adjust the importance between loss functions and the L_1-regularization. In this step, it truncates parameters where it is not useful for prediction in a weight vector through L_1-regularization. As we transform equation (4.6) into a closed-form solution, we obtain the following parameter update formula.

$$w_{t+1}^{(i)} = \begin{cases} 0 & \text{if } |w_{t+1/2}^{(i)}| \leq \lambda \\ \operatorname{sgn}(w_{t+1/2}^{(i)}) |w_{t+1/2}^{(i)} - \lambda \eta_{t+1/2}| & \text{otherwise} \end{cases} . \tag{4.7}$$

From formula (4.7), we can see that algorithms truncate parameters where its absolute value is less than λ without exception. By applying the regularization term explicitly as described in (4.6), the truncation effect in L_1-regularization is preserved. In summary, weight vectors are updated in a loss minimization step according to the subgradient method, and then it truncates uninformative parameters for predictions using the L_1-regularization term in this framework. In practice, these two step update formula can be summarized as one closed update formula. L_1-regularization is helpful because it derives sparse predictive models while preserving the advantages of online learning framework in FOBOS.

The connection between FOBOS and COMID can be described as follows. When we set $\eta_t = \eta_{t+1/2}$ for all t, the update formula of FOBOS with L_1-reuglarization can be described as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w} \rangle + \Phi(\mathbf{w}) + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}_t\|_2^2 \right\} . \tag{4.8}$$

Therefore, FOBOS is a special case of COMID in which the squared Euclidean distance is used as the Bregman divergence.

Although COMID is a sophisticated sparse online learning algorithm, it has a serious deficit; COMID updates parameters by utilizing only the current subgradient and forgets the information on the previous subgradients. As a result, the original version of COMID has a problem wherein a solution is often significantly affected by the last few data. This is the problem OMD-like algorithm has, and this problem is already indicated in Section 2.3.2. Unfortunately, this problem becomes more crucial in the case of sparse online learning. When the derived weight vector is strongly influenced by the recent few data, many uninformative features may become non-zero if these features appear in the recent data. This is because weight values easily move away from zero when the corresponding features are included in the last few data. For example, when we apply the L_1 norm as a regularization term, COMID tends to retain parameters that corresponding features occur recently, independent of the importance of features for predictions. To solve this difficulty, several extensions have been developed for it. After that, we will introduce Regularized Dual Averaging (RDA) as one approach to solve this problem. In another way, the truncation-cumulative version of FOBOS [124] heals the problem. The main idea of the cumulative penalty model is to keep track of the total L_1-regularization potential operations as a cumulative L_1 penalty. A cumulative L_1 penalty is used to smooth out fluctuation effects caused by the last few data. When weight values are clipped to zero for a long time, the cumulative penalty is accumulated. To move away from zero, weight values need to accumulate the subgradient value larger than the accumulated truncation penalty. In addition, this scheme not only smooths the fluctuation effect, it also suppresses noisy data.

4.1.4 Regularized Dual Averaging (RDA)

In addition to OMD, ODA has also been extended to integrate regularization terms to objective functions. This method is called Regularized Dual Averaging (RDA) [64] as an extensional framework of ODA introduced in Section 2.3.3. RDA can be viewed as a way of overcoming the weakly truncated problem of COMID by using the average subgradients, not only the subgradient with respect to the current datum. In each round, RDA solves a simple minimization problem that involves the summation of all past subgradients with respect to loss functions and the regularization term to derive the next weight vector. RDA updates parameters subject to the following equation:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \left\{ \sum_{\tau=1}^{t} \langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + t\Phi(\mathbf{w}) + \frac{1}{\eta_t} B_\psi(\mathbf{w}, \mathbf{w}_1) \right\} , \tag{4.9}$$

where $\{\eta_t\}_{t\geq 1}$ is a positive and non-increasing scalar sequence that determines the convergence properties of the algorithm.

RDA consists of a minimization problem of three terms, which are similar to COMID. The first term represents the inner product between the next weight vector and the sum of all previous subgradients. Therefore, if weight vectors are updated toward the direction which is reverse with the sum of all previous subgradients, the function value will decrease. The main difference between RDA and COMID is in this first term. Averaging by all previous subgradients alleviates the fluctuation of weight vector updates. The second term is a regularization term Φ. We assume Φ is a closed convex function. For example, L_1-regularization works as Φ to obtain a sparse weight vector. The third term is the Bregman divergence between the next weight vector and initial weight vector. In RDA, we set the initial weight vector $\mathbf{w}_1$ such that it satisfies

$$\mathbf{w}_1 \in \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \Phi(\mathbf{w}) . \tag{4.10}$$

As is similar to COMID and ODA, RDA is also guaranteed to have an $O(\sqrt{T})$ regret bound by properly setting $\{\eta_t\}_{t\geq 1}$ [64]. Similar to ODA, RDA do not need to store the previous subgradients to solve (4.9). If we have the average subgradient over all previous round, we can calculate the next average subgradient by only using the current subgradient and we can discard the current subgradient after update.

There are several extensional algorithms for RDA. In [64], an accelerated version of the RDA has been proposed to achieve the improved convergence rate for smooth loss functions. Dekel et al. [125] has proposed a variant of Dual Averaging methods for efficiently processing data distributed in many servers. Their derived algorithms have the theoretical guarantee of an asymptotically optimal regret bound for smooth convex loss functions. Lee et al. [126] has showed that RDA with L_1-regularization is able to identify a low-dimensional manifold in which the optimal solution exists. This study led to the development of a new modified variant of Dual Averaging algorithms that works faster than the original RDA by identifying low-dimensional manifold structures and efficiently utilizing them to search for the optimal solution.

4.1.5 Follow-The-Proximally-Regularized-Leader (FTPRL)

As another algorithm for sparse online learning, McMahan developed an algorithm called Follow-The-Proximally-Regularized-Leader (FTPRL) [71, 72]. FTPRL can be viewed as the extensional method of FTRL introduced in Section 2.3.4. We note that the regularization term used in Section 2.3.4 is replaced by the Bregman divergence and we newly insert the regularization term as Φ. The update formula of FTPRL can be written as the following formula:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \sum_{\tau=1}^{t} \left\{ \langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \Phi(\mathbf{w}) + \left(\frac{1}{\eta_\tau} - \frac{1}{\eta_{\tau-1}} \right) B_\psi(\mathbf{w}, \mathbf{w}_\tau) \right\} , \tag{4.11}$$

where $\{\eta_t\}_{t\geq 0}$ is positive non-increasing sequence of constants that determine the convergence property. In FTPRL, we set the initial weight vector $\mathbf{w}_1$ such that it satisfies

$$\mathbf{w}_1 \in \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \Phi(\mathbf{w}) . \tag{4.12}$$

FTPRL is reduced to FTRL by erasing the regularization term Φ. FTPRL can be viewed as an intermediate version of COMID and RDA. The crucial difference between COMID and FTPRL is in the first term. FTPRL applies the average of all previous subgradients to derive the next weight vector as in the same way as RDA. On the other hand, COMID utilizes the current subgradient vector in each round. Averaging of subgradients contributes to the improvement of the robustness of FTPRL and FTPRL heals the problem where COMID is largely affected by the recently received data.

The difference between RDA and FTPRL is in center points of the Bregman divergences (the third term). RDA sets the initial weight vector as the center point of the Bregman divergence and it is fixed at every round. In RDA, the farther the next weight vector moves away from the initial point, the bigger the value of Bregman divergence becomes. In contrast, FTPRL sets a weighted sum of the Bregman divergences in which each previous weight vectors is set as a center point. When the next weight vector stays near a weighted sum of previous weight vectors, the Bregman divergence of FTPRL will remain small in contrast to RDA.

The regret upper bound of FTPRL has been proved. It is guaranteed that FTPRL has $O(\sqrt{T})$ regret upper bound under the specific conditions of $\{\eta_t\}_{t\geq 1}$ [72]. Previous studies [71, 72] have shown FTPRL is equivalent to COMID as update procedures by appropriately setting the Bregman divergence. Table 4.1 illustrates the differences between COMID, RDA, and FTPRL.

Table. 4.1. Comparison of Subgradient-based Sparse Online Learning Algorithms

	COMID	RDA	FTPRL
Average Subgradient		√	√
Bregman Divergence	Current weight	Initial weight	Previous weights
$o(T)$ Regret bound	√	√	√

4.2 Truncation Bias

In the previous section, we introduce the overview of sparse online learning. Sparse online learning is effective for massive data analysis because of its implementation simplicity, fast computation, theoretical guarantee, and compact predictive model derived by sparsity-inducing regularization. We point out that these algorithms have the problem originated from the truncation bias in this section. As indicated above, a large skewness of feature occurrence frequency and value range exists in many tasks, such as Natural Language Processing and Pattern Recognition tasks. The truncation bias stems from these heterogeneity among features. This bias has bad effect for predictive performance of sparse online learning algorithms and we need to modify objective functions to solve this problem.

In the following subsections, we show two examples in which plain L_1-regularization causes problems in sparse online learning due to the truncation bias. In addition, we show that subgradient-based sparse online learning algorithms, such as RDA, do not take the truncation bias into account. Therefore, subgradient-based online learning algorithms with L_1-regularization terms tend to truncate some specific kinds of features even if they are crucial for making a prediction. It is because the plain L_1-regularization applies the same penalty to all features independent of occurrence counts or parameter update counts. We point out that this bias largely affects the derived predictive model, therefore appropriate treatment are needed to improve the predictive performance.

4.2.1 Feature Occurrence Frequency Problem

Suppose we will run RDA with the plain L_1-regularization as a sparsity-inducing norm. Let $\Phi(\mathbf{w}) = \lambda\|\mathbf{w}\|_1$, where λ is a positive constant for controlling the intensity of regularization, and $B_\psi(\mathbf{w}, \mathbf{w}_1) = \|\mathbf{w} - \mathbf{w}_1\|_2^2/2 = \|\mathbf{w}\|_2^2/2$ with $\mathbf{w}_1 = \mathbf{0}$. In addition, let us assume $\eta_t = 1/\sqrt{t}$. In this case, we can derive a coordinate-wise update formula:

$$w_{t+1}^{(i)} = \begin{cases} 0 & \text{if} \quad |\bar{g}_t^{(i)}| \leq \lambda \\ t\sqrt{t}\left(-\bar{g}_t^{(i)} + \mathrm{sgn}(\bar{g}_t^{(i)})\lambda\right) & \text{otherwise} \end{cases} , \tag{4.13}$$

where $\bar{\mathbf{g}}_t$ is the average value of the subgradients of the loss functions in the range from 1 to t. In other words,

$$\bar{\mathbf{g}}_t = \frac{1}{t}\sum_{\tau=1}^{t} \nabla \ell_\tau(\mathbf{w}_\tau) \; . \tag{4.14}$$

Let us apply this simple RDA to a dataset in which the occurrence rate of feature α is 1/100 and that of feature β is 1/2. Feature α inevitably becomes zero unless the average absolute value of the α-th index of the subgradients exceeds 100λ. In other words, when $\hat{g}^{(\alpha)} < 100\lambda$ is satisfied, where $\hat{g}^{(i)}$ is the average of the i-th absolute values of the subgradients by taking the i-th index to be non-zero, $w^{(\alpha)}$ always becomes zero. On the other hand, the weight of feature β is not always truncated to zero, wherein $\hat{g}^{(\beta)} \geq 2\lambda$. Therefore, if the feature occurrence frequency is highly skewed, which happens in many tasks, sparse online learning algorithms might fail to retain features that are rare but helpful for making precise predictions. The same problem occurs in the case of COMID and FTPRL with L_1-regularization.

4.2.2 Value Range Problem

The disparity in the range of features affects the truncation, too. Let us assume a dataset with two features: feature α and β. Feature α is an arbitrary feature, and feature β is the one whose value is exactly 1000 times larger than that of feature α. Although they both have the same importance when it comes to predictions, the weight of feature α is truncated faster than that of feature β when RDA learns from this dataset. That is, the following inequality is satisfied.

$$|\bar{g}^{(\alpha)}| \leq \lambda \leq |\bar{g}^{(\beta)}| \; , \tag{4.15}$$

where $\bar{g}^{(i)}$ is the average of the i-th indexes of all previous subgradients. We can easily see from formula (4.13) that the weight of feature i is truncated if the i-th index of the average subgradient is less than λ. Therefore, when inequality (4.15) is satisfied, only feature α is truncated; feature β is not. The converse phenomenon does not occur.

4.3 Feature-aware Regularization

The truncation bias largely affects the selection of informative features in weight vectors as written above. The simple solution to heal this type of bias is to apply pre-processing methods such as normalization and TF-IDF. However, in the online learning setting, it is difficult to apply pre-processing methods while maintaining advantages of online learning framework. To overcome this truncation bias problem in an online manner, we develop a feature-aware regularization method for sparse online learning without pre-processing. Our framework integrates the information obtained from all previous subgradients into a regularization term so as to adjust the intensity of the truncation and retain informative features for predictions. In this thesis, we focus on the feature-aware regularization for the L_1-regularization case.

First, we define the feature-aware matrix used in this framework. This feature-aware matrix is later used to dynamically adjust the truncation intensity for each feature. The feature-aware matrix is mathematically defined as:

$$\mathbf{R}_{t,q} = \begin{pmatrix} r_{t,q}^{(1)} & 0 & \dots & 0 \\ 0 & r_{t,q}^{(2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{t,q}^{(d)} \end{pmatrix} \quad s.t. \;\; r_{t,q}^{(i)} = \sqrt[q]{\sum_{\tau=1}^{t} \left| (\nabla \ell_\tau(\mathbf{w}_\tau))^{(i)} \right|^q} \, . \tag{4.16}$$

$r_{t,q}^{(i)}$ is the L_q norm of a vector that consists of values of the i-th index's subgradients derived in each round. $q \geq 1$ is a scalar parameter. $\mathbf{R}_{t,q}$ is a matrix consisting of $r_{t,q}^{(i)}$ of all features in a diagonal component. From the definition of (4.16), we can see that $r_{t,q}^{(i)}$ has a high probability of being a small value when parameter updates are rare or feature value ranges are small. On the other hand, when the i-th feature has large value range or frequently occurs in the input vectors, $r_{t,q}^{(i)}$ tends to become bigger.

By using this feature-aware matrix, we replace the L_1-regularization term by the feature-aware L_1-regularization term to achieve automatic adjustment of the truncation intensity. By putting this matrix inside the L_1-regularization term, the feature-aware L_1-regularization is defined as

$$\Phi_t(\mathbf{w}) = \lambda \|\mathbf{R}_{t,q}\mathbf{w}\|_1 \, , \tag{4.17}$$

where λ is a regularization parameter. Note that L_1-regularization can be restored by setting $\mathbf{R}_{t,q} = \mathbf{I}$ for all t. From the definition of (4.17), when $r_{t,q}^{(i)}$ is large, online learning algorithms with feature-aware L_1-regularization aggressively truncate the i-th component of the weight vector $w^{(i)}$. On the other hand, if $r_{t,q}^{(i)}$ has a small value, the i-th weight value tends to be non-zero. In such a case, we exert an influence on the regularization intensity through $r_{t,q}^{(i)}$, that is almost proportional to the parameter update frequency and feature value range. This effect heals the truncation bias.

We can see from the definition of $r_{t,q}^{(i)}$ that we adjust the regularization intensity according to the **subgradient** information at each index and not to the feature information. We use the information of subgradients to define feature-aware parameters. It is because weight vectors are updated according to the subgradient information and input vectors are not directly related to feature occurrence counts or feature value range. As a result, values in weight vectors tend to largely depend on subgradients other than input vectors and so, the intensity of regularization should depend on subgradient information. We will verify that subgradient-based regularization matrix outperforms feature-frequency-based definition of feature-aware matrix in experiments of Section 4.5.3.

We can set a wide variety of scalar values to q for adjusting the regularization intensity between features. We show how the parameter q affects the feature-aware parameter $r_{t,q}^{(i)}$ by using a simple numerical example. In this numerical example, we assume that components of all subgradients are limited to either 0 or 1. Figure 4.1 shows how the relationship between $r_{t,q}^{(i)}$ and the parameter update counts of feature i changes depending on the parameter q in this case. The horizontal axis is the number of occurrences in ascending order. The vertical axis is the value of $r_{t,q}^{(i)}$. This figure indicates that the smaller the value of q is, the faster $r_{t,q}^{(i)}$ becomes a large value for frequently occurring features. In terms of value range, we can easily see that $r_{t,q}^{(i)}$ becomes linearly bigger according to the value range of features regardless of the parameter q.

We note that the difference between our proposed feature-aware regularization methods and AdaGrad introduced in Section 2.4.1. The main difference between them is in our proposed feature-aware regularization methods. The truncation bias affects the objective functions themselves, therefore, tighter upper bound of regret is not helpful to heal this bias even if AdaGrad version of online learning algorithms are the best to obtain the regret upper bound. The appropriate reformulation of objective functions through adaptive feature-aware regularization methods is needed to heal this truncation bias. Our feature-aware regularization methods achieve this objective by introducing feature-aware parameters into the regularization terms and changing objective functions to derive a sequence of weight vectors according to the data structure.

Here, we apply feature-aware regularizations to COMID, RDA, and FTPRL to derive sophisticated subgradient-based sparse online learning algorithms. The derived algorithms

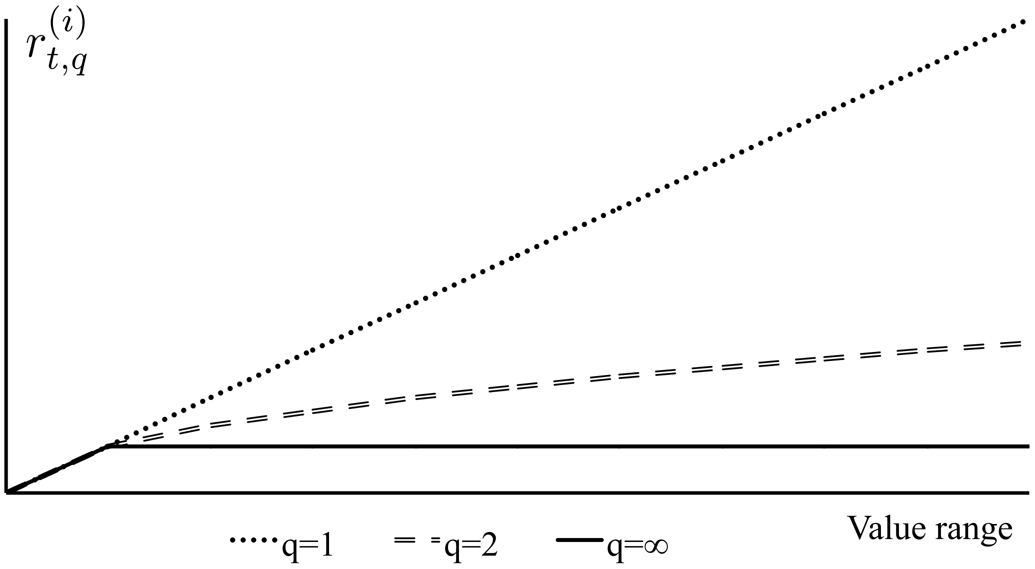

Fig. 4.1. Plot of feature-aware parameter $r^{(i)}_{t,q}$ against parameter q

are FR-COMID (Feature-aware Regularized COMID), which is an extension of COMID, FRDA (Feature-aware Regularized Dual Averaging), which is an extension of RDA, and FTPFRL (Follow-The-Proximally-Feature-aware-Regularized-Leader), which is an extension of FTPRL. To simplify our explanation in the remainder of this section, we will omit the notations on the parameter q indicating, for example, from $\mathbf{R}_{t,q}$ to $\mathbf{R}_t$ and from $r^{(i)}_{t,q}$ to $r^{(i)}_t$. In addition, for simplifying the following discussion, let us define that feature-aware intensity vector $\mathbf{r}_{t,q}$ consisting of feature-aware parameters $r^{(i)}_{t,q}$ for each component, that is, $\mathbf{r}_{t,q} = (r^{(1)}_{t,q}, r^{(2)}_{t,q}, \ldots, r^{(d)}_{t,q})$.

4.3.1 Feature-aware Regularized COMID

Let us apply feature-aware regularization methods to COMID. We name this feature-aware version of COMID as Feature-aware Regularized Composite Objective Mirror Descent (FR-COMID). The objective function of FR-COMID is defined as

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \eta_t \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w} \rangle + \eta_t \Phi_t(\mathbf{w}) + B_\psi(\mathbf{w}, \mathbf{w}_t) \right\} \ , \tag{4.18}$$

where Φ_t is defined as the feature-aware L_1-regularization (4.17). FR-COMID can be generated by replacing the fixed regularization term in the update formula of the original COMID (4.4) by feature-aware regularization term (4.17).

As a motivating example, let us derive a closed update formula in which we set the Euclidean distance as the Bregman divergence $B_\psi(\mathbf{w}, \mathbf{w}_t) = \|\mathbf{w} - \mathbf{w}_t\|_2^2/2$. The formula

(4.18) is reformulated as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \eta_t \langle \nabla \ell_t(\mathbf{w}_t), \mathbf{w} \rangle + \eta_t \lambda \|\mathbf{R}_t \mathbf{w}\|_1 + \frac{1}{2}\|\mathbf{w} - \mathbf{w}_t\|_2^2 \right\} . \tag{4.19}$$

The coordinate-wise closed-form update formula is derived by setting the derivative of right-hand side of the formula (4.19) with respect to the weight vector $\mathbf{w}$ to zero. The solution is uniquely derived because this formula is strongly convex with respect to the weight vector. This problem can be decomposed into coordinate-wise one and the equation with respect to i-th element is derived as follows:

$$\eta_t \left(\nabla \ell_t(\mathbf{w}_t)\right)^{(i)} + \eta_t \lambda r_t^{(i)} \xi_t^{(i)} + w_{t+1}^{(i)} - w_t^{(i)} = 0 , \tag{4.20}$$

where $\xi_t^{(i)}$ is the subgradient of $|w_{t+1}^{(i)}|$. The subdifferential of $|w_{t+1}^{(i)}|$ changes according to the value of $w_{t+1}^{(i)}$. $\xi_t^{(i)} = 1$ if $w_{t+1}^{(i)} > 0$, $\xi_t^{(i)} = -1$ if $w_{t+1}^{(i)} < 0$, or $\{\xi \in \mathbb{R} | -1 \le \xi \le 1\}$ if $w_{t+1}^{(i)} = 0$.

By summarizing (4.20), we can derive the coordinate-wise closed-form update formula as follows:

$$w_{t+1}^{(i)} = \begin{cases} 0 & v_t^{(i)} \le 0 \\ \operatorname{sgn}\left(w_t^{(i)} - \eta_t \left(\nabla \ell_t(\mathbf{w}_t)\right)^{(i)}\right) v_t^{(i)} & \text{otherwise} \end{cases} . \tag{4.21}$$

where $v_t^{(i)} = \left| w_t^{(i)} - \eta_t \left(\nabla \ell_t(\mathbf{w}_t)\right)^{(i)} \right| - \eta_t \lambda r_t^{(i)}$.

From this update formula, we can see that $r_t^{(i)}$ adjusts the intensity of the truncation so as to retain informative but rarely occurring features. We can see from the formula (4.21) that FR-COMID can process one datum at $O(D)$ computational cost. This computational cost is exactly the same as the original COMID. The update procedure is summarized in Algorithm 4.

Lazy Update

We additionally proved that FR-COMID allows us to truncate parameters in a lazy fashion [31]. We do not need to apply sparsity-inducing regularization to all components at all rounds. To perform exact prediction, algorithms can postpone the update of weights such that these corresponding features do not occur in the current datum; therefore, we can postpone to apply the regularization effect on these weights in each round. Such an updating scheme enables us to speed up the update procedure when the input vectors are very sparse and its dimension is very large.

We define the cumulative value of the total L_1-regularization from $\tau = 1$ to t as

$$l_t = \lambda \sum_{\tau=1}^{t} \eta_\tau . \tag{4.23}$$

Algorithm 4 Feature-aware Regularized Composite Objective Mirror Descent (FR-COMID)

```
Require:
1: $\{\eta_t\}_{t\geq 1}$ is a positive non-increasing sequence.
2: $\mathbf{w}_1 = \mathbf{0}$.
Algorithm:
1: for $t = 1, 2, \dots$ do
2:     Given a loss function $\ell_t$, compute the subgradient $\nabla\ell_t(\mathbf{w}_t)$.
3:     Derive a new weight vector $\mathbf{w}_{t+1}$ as:
       $$w_{t+1}^{(i)} = \begin{cases} 0 & v_t^{(i)} \leq 0 \\ \mathrm{sign}\left(w_t^{(i)} - \eta_t \left(\nabla\ell_t(\mathbf{w}_t)\right)^{(i)}\right) v_t^{(i)} & \text{otherwise} \end{cases}. \quad (4.22)$$
       where $v_t^{(i)} = \left|w_t^{(i)} - \eta_t \left(\nabla\ell_t(\mathbf{w}_t)\right)^{(i)}\right| - \eta_t\lambda r_t^{(i)}$.
4: end for
```

In the lazy update scheme, before the subgradient of the loss function is computed, we apply feature-aware L_1-regularization to features that occur in the current datum, i.e.,

$$w_{t+1/2}^{(i)} = \begin{cases} \max\left(0, w_t^{(i)} - (l_{t-1} - l_{s-1})r_s^{(i)}\right) & w_t^{(i)} \geq 0 \\ \min\left(0, w_t^{(i)} + (l_{t-1} - l_{s-1})r_s^{(i)}\right) & w_t^{(i)} < 0 \end{cases}, \quad (4.24)$$

where s is the most recent round number at which feature i appear in the input vector. After truncating weight vectors, we calculate a subgradient by using weight vectors derived by (4.24) and update parameters through the update formula of the online subgradient method. The computational cost of lazy-update FR-COMID can be reduced to the order of the number of non-zero elements in input vectors at the risk of the memory size for storing the value of l_s for each feature.

FR-COMID with Cumulative Penalty

We also derived the cumulative penalty version of COMID based on Tsuruoka et al. proposed model [124]. The normal COMID with the L_1-regularization has a problem where weight vectors are significantly affected by the last few instances. This is because weight values easily move away from zero when the corresponding features are used in the last few data. The main idea of the cumulative penalty model is to keep track of total penalty incurred by the regularization term. Then, we apply a cumulative feature-aware

L_1 penalty to smooth the effect of the update fluctuation and move away from zero unless the updating sum exceeds the cumulative penalty. In addition, this model can smooth the effect of the update and also suppress noisy data.

We introduce FR-COMID with the cumulative penalty model. We begin by introducing $q_t^{(i)}$ as a cumulative feature-aware L_1 penalty value of feature i at the t-th round. We initialize $q_1^{(i)} = 0$ for all i. In this model, we truncate weight vectors whose features are used in the current data as follows:

$$w_{t+1}^{(i)} = \begin{cases} \max\left(0, w_{t+1/2}^{(i)} - (r_t^{(i)} l_t + q_t^{(i)})\right) & w_{t+1/2}^{(i)} \geq 0 \\ \min\left(0, w_{t+1/2}^{(i)} + (r_t^{(i)} l_t - q_t^{(i)})\right) & w_{t+1/2}^{(i)} < 0 \end{cases} . \tag{4.25}$$

Then, we update the parameter $q_t^{(i)}$ if the feature i is non-zero in the current input vector. The update formula is defined as follows:

$$q_{t+1}^{(i)} = q_t^{(i)} + (w_{t+1}^{(i)} - w_{t+1/2}^{(i)}) . \tag{4.26}$$

We represent the lazy update form of the update formula. This formalization makes update functions simple, reduces space complexity, and enables faster calculation.

4.3.2 Feature-aware Regularized Dual Averaging

We apply the feature-aware regularization to RDA. The update formula can be rewritten as:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{\tau=1}^{t} \left(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \Phi_\tau(\mathbf{w})\right) + \frac{1}{\eta_t} B_\psi(\mathbf{w}, \mathbf{w}_1) \right\} . \tag{4.27}$$

This update formula can be generated by replacing the regularization term in RDA by the feature-aware regularization (4.17). We call this method FRDA.

We derive a closed-form update formula of the feature-aware L_1-regularization defined in (4.17). Our derivation is similar to that of Xiao [64], except for the definition of $\Phi_t(\mathbf{w})$ and corresponding terms. Let us set $B_\psi(\mathbf{w}, \mathbf{w}_1) = \|\mathbf{w} - \mathbf{w}_1\|_2^2/2$. And in this case, the update formula is reformulated as follows:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{\tau=1}^{t} \left(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \lambda \|\mathbf{R}_\tau \mathbf{w}\|_1\right) + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}_1\|_2^2 \right\} . \tag{4.28}$$

FRDA set the following restriction for any integer $t \geq 1$.

$$\mathbf{w}_1 = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \tag{4.29}$$

From this condition, we set the initial weight vector as $\mathbf{w}_1 = \mathbf{0}$.

Let $\bar{\mathbf{g}}_t$ be the average of all previous subgradients $\nabla \ell_\tau(\mathbf{w}_\tau)$ from $\tau = 1$ to $\tau = t$, and let $\bar{\mathbf{r}}_t$ be the average of all previous feature-aware regularization parameters $\mathbf{r}_\tau$ from $\tau = 1$ to $\tau = t$. Moreover, to simplify our explanation, we define the vector $\mathbf{u}_t$, each element of which satisfies the following equation:

$$u_t^{(i)} = \sum_{\tau=1}^{t} |\, (\nabla \ell_\tau(\mathbf{w}_\tau))^{(i)}\, |^q \; . \tag{4.30}$$

The parameter q is the same as the one in the definition of $r_{t,q}^{(i)}$.

By using these notations, we explain the parameter update procedure in FRDA. In each round, we updates $\mathbf{u}_t$ in a coordinate-wise as follows:

$$u_t^{(i)} = u_{t-1}^{(i)} + |\, (\nabla \ell_t(\mathbf{w}_t))^{(i)}\, |^q \; . \tag{4.31}$$

After all, we calculate the feature-aware parameters as $\mathbf{r}_t$ and $\bar{\mathbf{r}}_t$:

$$r_t^{(i)} = \sqrt[q]{u_t^{(i)}} \; , \quad \bar{\mathbf{r}}_t = \frac{t-1}{t}\bar{\mathbf{r}}_{t-1} + \frac{1}{t}\mathbf{r}_t \; . \tag{4.32}$$

The optimization problem of the right-hand side of (4.28) can be decomposed into a coordinate-wise update formula:

$$w_{t+1}^{(i)} = \operatorname*{argmin}_{w \in R} \left(t\bar{g}_t^{(i)} w + t\lambda |\bar{r}_t^{(i)} w| + \frac{1}{2\eta_t} w^2 \right) \; , \tag{4.33}$$

From the definition of $r_t^{(i)}$, we clearly have $\bar{r}_t^{(i)} \geq 0$. Thus, the optimal solution of the formula (4.33) is subject to

$$\bar{g}_t^{(i)} + \lambda \bar{r}_t^{(i)} \xi^{(i)} + \frac{1}{t\eta_t} w_{t+1}^{(i)} = 0 \; , \tag{4.34}$$

where $\xi^{(i)}$ is the subgradient of $|w_{t+1}^{(i)}|$. The value of $\xi_t^{(i)}$ changes according to the value of $w_{t+1}^{(i)}$. $\xi_t^{(i)} = 1$ if $w_{t+1}^{(i)} > 0$, $\xi_t^{(i)} = -1$ if $w_{t+1}^{(i)} < 0$, or $\{\xi \in R | -1 \leq \xi \leq 1\}$ if $w_{t+1}^{(i)} = 0$.

Therefore, we can solve the optimization problem as follows:

- If $|\bar{g}_t^{(i)}| \leq \lambda \bar{r}_t^{(i)}$, set $w_{t+1}^{(i)} = 0$ and $\xi^{(i)} = -\bar{g}_t^{(i)}/\lambda \bar{r}_t^{(i)}$. When $w_{t+1}^{(i)} \neq 0$, the formula (4.34) cannot be satisfied.
- If $\bar{g}_t^{(i)} > \lambda \bar{r}_t^{(i)} > 0$, set $w_{t+1}^{(i)} < 0$ and $\xi^{(i)} = -1$.
- If $\bar{g}_t^{(i)} < -\lambda \bar{r}_t^{(i)} < 0$, set $w_{t+1}^{(i)} > 0$ and $\xi^{(i)} = 1$.

The above solution is summarized as

$$w_{t+1}^{(i)} = \begin{cases} 0 & v_t^{(i)} \leq 0 \\ -\mathrm{sign}(\bar{g}_t^{(i)}) t\eta_t v_t^{(i)} & \text{otherwise} \end{cases} \; , \tag{4.35}$$

where we have defined $v_t^{(i)} = |\bar{g}_t^{(i)}| - \lambda \bar{r}_t^{(i)}$.

The update procedure of FRDA algorithm is summarized in Algorithm 5.

Algorithm 5 Feature-aware Regularized Dual Averaging (FRDA)

```
Require:
1: $\{\eta_t\}_{t \geq 1}$ is a positive non-increasing sequence.
2: $\mathbf{w}_1 = \mathbf{0}$, $\mathbf{u}_0 = \mathbf{0}$, $\bar{\mathbf{g}}_0 = \mathbf{0}$ and $\bar{\mathbf{r}}_0 = \mathbf{0}$.
Algorithm:
1: for $t = 1, 2, \ldots$ do
2:     Given a loss function $\ell_t$, compute subgradient $\nabla \ell_t(\mathbf{w}_t)$.
3:     Update the average of all previous subgradients $\bar{\mathbf{g}}_t$ as:
           $\bar{\mathbf{g}}_t = \frac{t-1}{t}\bar{\mathbf{g}}_{t-1} + \frac{1}{t}\nabla \ell_t(\mathbf{w}_t)$ .   (4.36)
4:     Calculate the regularized parameters $\mathbf{r}_t$:
           $u_t^{(i)} = u_{t-1}^{(i)} + |g_t^{(i)}|^p$ ,   $r_t^{(i)} = \sqrt[p]{u_t^{(i)}}$ .   (4.37)
5:     Update the regularized parameters $\bar{\mathbf{r}}_t$:
           $\bar{\mathbf{r}}_t = \frac{t-1}{t}\bar{\mathbf{r}}_{t-1} + \frac{1}{t}\mathbf{r}_t$ .   (4.38)
6:     Derive new weight vector $\mathbf{w}_{t+1}$ as:
           $w_{t+1}^{(i)} = \begin{cases} 0 & v_t^{(i)} \leq 0 \\ -\mathrm{sign}(\bar{g}_t^{(i)}) t \eta_t v_t^{(i)} & \text{otherwise} \end{cases}$ ,   (4.39)
       where we have defined $v_t^{(i)} = |\bar{g}_t^{(i)}| - \lambda \bar{r}_t^{(i)}$.
7: end for
```

Lazy Update

As the same as in the case of COMID, we can derive the lazy update form of FRDA with the feature-aware L_1-regularization. The evaluations of (4.38) and (4.39) can be delayed until the corresponding features occur in the current datum. Each time we receive one datum, the algorithm updates only the indices of $\mathbf{r}$ and $\mathbf{w}$ where the corresponding features occur. In this lazy update form, FRDA with the feature-aware L_1-regularization runs at a computational cost of O(the number of occurring features) in each round.

4.3.3 Follow-The-Proximally-Feature-aware-Regularized-Leader

In addition, we derive the FTPRL with the feature-aware L_1-regularization. The update formula of FTPRL with the feature-aware L_1-regularization is derived from the replacement of the simple L_1-regularization term by the feature-aware regularization defined in (4.17). The update formula is defined as

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{\tau=1}^{t} \left(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \Phi_\tau(\mathbf{w}) + \left(\frac{1}{\eta_\tau} - \frac{1}{\eta_{\tau-1}} \right) B_\psi(\mathbf{w}, \mathbf{w}_\tau) \right) \right\} . \quad (4.40)$$

Here, we set $\mathbf{w}_1$ as the minimizing points of argmin $B_\psi(\mathbf{w}, \mathbf{w}_1)$. We call this optimization method Follow-The-Proximally-Feature-aware-Regularized-Leader (FTPFRL).

We will give a closed-form update formula for FTPFRL, where $B_\psi(\mathbf{w}, \mathbf{w}_t) = \|\mathbf{w} - \mathbf{w}_t\|_2^2/2$, and $\Phi_\tau(\mathbf{w})$ is the feature-aware regularization (4.17). Let us assume that $\mathbf{w}_1$ is a zero vector and $1/\eta_0 = 0$. The update formula of FTPFRL is written as follows:

$$\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \left\{ \sum_{\tau=1}^{t} \left(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w} \rangle + \lambda \|\mathbf{R}_\tau \mathbf{w}\|_1 + \frac{1}{2} \left(\frac{1}{\eta_\tau} - \frac{1}{\eta_{\tau-1}} \right) \|\mathbf{w} - \mathbf{w}_\tau\|_2^2 \right) \right\} . \quad (4.41)$$

To simplify the following discussion, we define $1/\eta_t - 1/\eta_{t-1}$ as $\hat{\beta}_t$. First, we summarize the sum of the squared Euclidean norms as follows:

$$\begin{aligned} \sum_{\tau=1}^{t} \frac{\hat{\beta}_\tau}{2} \|\mathbf{w} - \mathbf{w}_\tau\|_2^2 &= \frac{1}{2} \left(\sum_{\tau=1}^{t} \hat{\beta}_\tau \|\mathbf{w}\|_2^2 - 2\hat{\beta}_\tau \mathbf{w}\mathbf{w}_\tau + \hat{\beta}_\tau \|\mathbf{w}_\tau\|_2^2 \right) \\ &= \frac{\sum \hat{\beta}_\tau}{2} \left\| \mathbf{w} - \frac{\sum \hat{\beta}_\tau \mathbf{w}_\tau}{\sum \hat{\beta}_\tau} \right\|_2^2 + \text{const.} \\ &= \frac{1}{2\eta_t} \left\| \mathbf{w} - \eta_t \mathbf{w}_t^\beta \right\|_2^2 + \text{const.} \, , \end{aligned} \quad (4.42)$$

where $\sum \hat{\beta}_\tau = 1/\eta_t$ from the definition of $\hat{\beta}_\tau$ and $\mathbf{w}_t^\beta = \sum \hat{\beta}_\tau \mathbf{w}_\tau$. The constant term does not affect the derivation of the next weight vector; therefore, we can ignore it.

We derive a new weight vector $\mathbf{w}_{t+1}$ in a closed-form update formula. The derivation of the closed-form solution is similar to the one for FRDA, therefore, we reuse the same definition of variables $\bar{\mathbf{g}}$, $\mathbf{u}$, and $\bar{\mathbf{r}}$. We derive $\bar{\mathbf{r}}_t$ by using eqs. (4.31), (4.32). The beta-weighted weight vector can be calculated as:

$$\mathbf{w}_t^\beta = \mathbf{w}_{t-1}^\beta + \hat{\beta}_t \mathbf{w}_t \, . \quad (4.43)$$

Since the optimization problem can be decomposed into a coordinate-wise update formula and is a convex function with respect to $\mathbf{w}$, we find

$$\bar{g}_t^{(i)} + \lambda \bar{r}_t^{(i)} \xi^{(i)} + \frac{1}{t\eta_t} \left(w_{t+1}^{(i)} - \eta_t \left(w_t^{\beta} \right)^{(i)} \right) = 0 \; . \tag{4.44}$$

As in the case of FRDA, the update formula is derived by properly assigning the value of ξ. In summary, the closed-form solution can be derived as follows:

$$w_{t+1}^{(i)} = \begin{cases} 0 & v_t^{(i)} \leq 0 \\ \operatorname{sgn}\left(\left(w_t^{\beta} \right)^{(i)} - t\bar{g}_t^{(i)} \right) \eta_t v_t^{(i)} & \text{otherwise} \end{cases} , \tag{4.45}$$

where we have defined $v_t^{(i)} = \left| \left(w_t^{\beta} \right)^{(i)} - t\bar{g}_t^{(i)} \right| - t\lambda \bar{r}_t^{(i)}$. The update procedure of the FTPFRL algorithm is summarized in Algorithm 6. Unfortunately, it is unclear whether the lazy update form of FTPFRL can speed up the update procedure.

4.3.4 How to eliminate the truncation bias

Here, we illustrate how the feature-aware L_1-regularization overcomes the rare feature truncation problem and the value range problem in this subsection.

Let us apply FRDA to the examples described in Sections 4.2.1 and 4.2.2. First, let us consider the dataset described in Section 4.2.1 under the condition that parameter settings are the same as RDA's. As described before, RDA truncates the feature α on a priority basis. On the other hand, FRDA does not truncate the feature α when $\hat{g}^{(\alpha)} \geq 100\lambda \bar{r}_t^{(\alpha)}$ is satisfied where $\hat{g}^{(i)}$ is defined as the average of the i-th absolute values of subgradients whose i-th index is non-zero. For the feature β, FRDA does not truncate the feature β when $\hat{g}^{(\beta)} \leq 2\lambda \bar{r}_t^{(\beta)}$ is satisfied. In FRDA, feature-aware parameters $\bar{r}_t^{(i)}$ are inserted. While the occurrence frequency of the feature β is fifty times greater than that of the feature α, it is expected that the value of $\bar{r}_t^{(\beta)}$ will be $\sqrt[q]{50}$ larger than the value of $\bar{r}_t^{(\alpha)}$ from the definition of $\bar{r}_t^{(i)}$. Thus, the intensity of truncation is adjusted so as to retain rare features compared with the plain RDA.

After that, let us consider the value range problem described in Section 4.2.2. We assume that the feature α is arbitrary defined and the value of the feature β is exactly 1000 times larger than that of the feature α. These two features have the same importance for prediction. However, the feature α would be truncated in an earlier round than feature β in RDA. On the other hand, in FRDA, $r_t^{(\alpha)}$ becomes exactly 1000 times larger than $r_t^{(\beta)}$ in all rounds; thus, the value of $\bar{r}_t^{(\alpha)}$ is also exactly 1000 times larger than the value of $\bar{r}_t^{(\beta)}$. Therefore, if $|\bar{g}_t^{(\alpha)}| < \lambda \bar{r}_t^{(\alpha)}$, i.e., $w_{t+1}^{(\alpha)} = 0$ is satisfied, $|\bar{g}_t^{(\beta)}| < \lambda \bar{r}_t^{(\beta)}$, i.e., $w_t^{(\beta)} = 0$ is

Algorithm 6 Follow-The-Proximally-Feature-aware-Regularized-Leader (FTPFRL)

```
Require:
1: $\{\eta_t\}_{t\geq 0}$ is a positive non-increasing sequence.
2: $\mathbf{w}_1 = \mathbf{0}$, $\mathbf{u}_0 = \mathbf{0}$, $\bar{\mathbf{g}}_0 = \mathbf{0}$, $\bar{\mathbf{r}}_0 = \mathbf{0}$, and $\mathbf{w}_0^{\beta} = \mathbf{0}$.
Algorithm:
1: for $t = 1, 2, \dots$ do
2:     Given a loss function $\ell_t$, compute the subgradient $\nabla \ell_t(\mathbf{w}_t)$.
3:     Update the average of all previous subgradients $\bar{\mathbf{g}}_t$ as:
           $$\bar{\mathbf{g}}_t = \frac{t-1}{t}\bar{\mathbf{g}}_{t-1} + \frac{1}{t}\nabla \ell_t(\mathbf{w}_t) \ . \qquad (4.46)$$
4:     Calculate the regularized parameters $\mathbf{r}_t$:
           $$u_t^{(i)} = u_{t-1}^{(i)} + |g_t^{(i)}|^p \ , \quad r_t^{(i)} = \sqrt[p]{u_t^{(i)}} \ . \qquad (4.47)$$
5:     Update the regularized parameters $\bar{\mathbf{r}}_t$:
           $$\bar{\mathbf{r}}_t = \frac{t-1}{t}\bar{\mathbf{r}}_{t-1} + \frac{1}{t}\mathbf{r}_t \ . \qquad (4.48)$$
6:     Update the weight vector weighted by the variables $\{\eta_t\}_{t\geq 0}$:
           $$\mathbf{w}_t^{\beta} = \mathbf{w}_{t-1}^{\beta} + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\mathbf{w}_t \ . \qquad (4.49)$$
7:     Derive new weight vector $\mathbf{w}_{t+1}$ as:
           $$w_{t+1}^{(i)} = \begin{cases} 0 & v_t^{(i)} \leq 0 \\ \mathrm{sgn}\left(\left(w_t^{\beta}\right)^{(i)} - t\bar{g}_t^{(i)}\right)\eta_t v_t^{(i)} & \text{otherwise} \end{cases} \ , \qquad (4.50)$$
       where we have defined $v_t^{(i)} = \left|\left(w_t^{\beta}\right)^{(i)} - t\bar{g}_t^{(i)}\right| - t\lambda\bar{r}_t^{(i)}$.
8: end for
```

also satisfied. The converse is also true. Accordingly, even if the range of feature values is skewed among features, FRDA automatically absorbs the disparity in features and adjusts the truncation intensity by using feature-aware parameters $\bar{\mathbf{r}}$.

4.4 Theoretical Analyses

The objective of sparse online learning algorithms is to achieve a low regret as in the definition of (4.3). No matter what a sequence of data or convex loss functions is received, the regret upper bound of sparse online learning algorithms is guaranteed to be $o(T)$; as such, the regret value per round converges to zero. Thus, sparse online learning algorithms obtain the minimal value of objective function per datum as the optimal weight vector is obtained in hindsight. As mentioned before, regret in the sparse online learning setting is defined as

$$\mathrm{Regret}_{\ell+\Phi}(T) = \sum_{t=1}^{T} \left(\ell_t(\mathbf{w}_t) + \Phi_t(\mathbf{w}_t)\right) - \min_{\mathbf{u}\in\mathcal{W}} \sum_{t=1}^{T} \left(\ell_t(\mathbf{u}) + \Phi_t(\mathbf{u})\right) \ . \tag{4.51}$$

Before proceeding the discussion of regret upper bounds, we set the upper bound of $r_{t,q}^{(i)}$ by using a scalar V so that we can ensure that the value of $\Phi_t(\mathbf{w}_t)$ does not approach infinity. This assumption is crucial to derive regret upper bounds of our derived feature-aware regularization methods. We redefine $r_{t,q}^{(i)}$ as

$$r_{t,q}^{(i)} = \min\left(V, \sqrt[q]{\sum_{\tau=1}^{t} \left|g_{\tau}^{(i)}\right|^{q}}\right) \ , \tag{4.52}$$

i.e., we set the upper bound for $r_{t,q}^{(i)}$ to V and derive the following inequality for any $t \geq 1$:

$$\frac{1}{t}\sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \leq \max_{i,\tau} r_{\tau,q}^{(i)} \|\mathbf{w}\|_1 \leq V\|\mathbf{w}\|_1 \ . \tag{4.53}$$

The left-hand side of the inequality has an upper bound.

After that, we prove the following lemma. This lemma is used to bound the difference of the feature-aware regularization.

Lemma 8. *When Φ_t is the feature-aware regularization defined in (4.17) and the initial weight vector becomes $\mathbf{w}_1 = \mathbf{0}$, the following inequality is satisfied.*

$$\sum_{\tau=1}^{T} \left(\Phi_\tau(\mathbf{w}_\tau) - \Phi_\tau(\mathbf{w}_{\tau+1})\right) \leq dV \max_{\mathbf{w}\in\mathcal{W}} \|\mathbf{w}\|_1 \ . \tag{4.54}$$

Proof. We reformulate the left-hand side of the formula (4.54) as:

$$\begin{aligned}\sum_{\tau=1}^{T}(\Phi_\tau(\mathbf{w}_\tau)-\Phi_\tau(\mathbf{w}_{\tau+1})) &= \sum_{\tau=1}^{T}(\|\mathbf{R}_{\tau,q}\mathbf{w}_\tau\|_1-\|\mathbf{R}_{\tau,q}\mathbf{w}_{\tau+1}\|_1)\\ &= \|\mathbf{R}_{1,q}\mathbf{w}_1\|_1+\sum_{\tau=2}^{T}\|(\mathbf{R}_{\tau,q}-\mathbf{R}_{\tau-1,q})\mathbf{w}_\tau\|_1-\|\mathbf{R}_{T,q}\mathbf{w}_{T+1}\|_1\\ &\leq 0+\sum_{\tau=2}^{T}\|(\mathbf{R}_{\tau,q}-\mathbf{R}_{\tau-1,q})\mathbf{w}_\tau\|_1-0\ , \end{aligned}\tag{4.55}$$

where the first inequality has used $\mathbf{w}_1=\mathbf{0}$ and $\Phi_\tau(\mathbf{w}_{\tau+1})\geq 0$. Let us focus on the second term of (4.55). We can reformulate it by applying Holder's inequality.

$$\begin{aligned}\sum_{\tau=2}^{T}\|(\mathbf{R}_{\tau,q}-\mathbf{R}_{\tau-1,q})\mathbf{w}_\tau\|_1 &\leq \sum_{\tau=2}^{T}\|\mathbf{r}_{\tau,q}-\mathbf{r}_{\tau-1,q}\|_1\|\mathbf{w}_\tau\|_1\\ &\leq \max_{\mathbf{w}\in\mathcal{W}}\|\mathbf{w}\|_1\sum_{\tau=2}^{T}\|\mathbf{r}_{\tau,q}-\mathbf{r}_{\tau-1,q}\|_1\ .\end{aligned}\tag{4.56}$$

From the definition of $\mathbf{r}_{t,q}$, all elements are non-decreasing with respect to γ. Therefore, the following inequality is satisfied through (4.52):

$$\sum_{\tau=2}^{T}\|(\mathbf{r}_{\tau,q}-\mathbf{r}_{\tau-1,q})\|_1=\|\mathbf{r}_{T,q}\|_1\leq dV\ .\tag{4.57}$$

By applying this inequality to (4.56), we can prove this lemma. □

4.4.1 Regret Upper Bound for FR-COMID

The regret upper bound of FR-COMID is derived as follows:

Theorem 12. *Let $\{\mathbf{w}_t\}_{t=1:T+1}$ be sequences generated by* Algorithm 4. *Furthermore, we assume that ψ is β-strongly convex with respect to a norm $\|\cdot\|$, there exist scalars $R,N,G\in\mathbb{R}$ such that $B_\psi(\mathbf{w},\tilde{\mathbf{w}})\leq R^2$ for all $\mathbf{w},\tilde{\mathbf{w}}\in\mathcal{W}$, $\|\mathbf{w}\|_1\leq N$ for all $\mathbf{w}\in\mathcal{W}$, and $\|\nabla\ell_t(\mathbf{w}_t)\|_*\leq G$ for all t. Let us set a positive non-increasing sequence of $\{\eta_t\}_{t\geq1}$ and $V\in\mathbb{R}$ such that (4.52) is satisfied, we obtain*

$$\text{Regret}_{\ell+\Phi}(T)\leq\frac{R^2}{\eta_T}+\frac{G^2}{2\beta}\sum_{t=1}^{T}\eta_t+dVN\ .\tag{4.58}$$

When we set $\eta_t=\gamma/\sqrt{t}$ where γ is a constant, for any $T\geq1$, we obtain $O(\sqrt{T})$ regret upper bound.

Proof. We prove the regret upper bound of FR-COMID. First, we introduce the following lemma. This lemma is an extension of Lemma 5 by adding the feature-aware L_1-regularization term.

Lemma 9. *We assume that the conditions defined in Theorem 12 are satisfied. When we define* $\mathbf{w}^*$ *as*

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{u} \in \mathcal{W}} \sum_{t=1}^{T} \left(\ell_t(\mathbf{u}) + \Phi_t(\mathbf{u}) \right) \ , \tag{4.59}$$

the following inequality is satisfied.

$$\begin{aligned} &\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) + \Phi_t(\mathbf{w}_{t+1}) - \Phi_t(\mathbf{w}^*) \\ &\quad \leq \frac{1}{\eta_t} \left(B_\psi(\mathbf{w}^*, \mathbf{w}_t) - B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \right) + \frac{\eta_t}{2\beta} \|\nabla \ell_t(\mathbf{w}_t)\|_*^2 \ . \end{aligned} \tag{4.60}$$

Proof. This lemma can be proved by following the discussion of COMID [61]. It is derived from the following reformulation:

$$\begin{aligned} &\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) + \Phi_t(\mathbf{w}_{t+1}) - \Phi_t(\mathbf{w}^*) \\ &\quad \leq \langle \mathbf{w}_t - \mathbf{w}^*, \nabla \ell_t(\mathbf{w}_t) \rangle + \langle \mathbf{w}_{t+1} - \mathbf{w}^*, \nabla \Phi_t(\mathbf{w}_{t+1}) \rangle \\ &\quad = \langle \mathbf{w}_{t+1} - \mathbf{w}^*, \nabla \ell_t(\mathbf{w}_t) \rangle + \langle \mathbf{w}_{t+1} - \mathbf{w}^*, \nabla \Phi_t(\mathbf{w}_{t+1}) \rangle + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t) \rangle \\ &\quad = \frac{1}{\eta_t} \langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla \psi(\mathbf{w}_t) - \nabla \psi(\mathbf{w}_{t+1}) - \eta_t \nabla \Phi_t(\mathbf{w}_{t+1}) - \eta_t \nabla \ell_t(\mathbf{w}_t) \rangle \\ &\qquad + \frac{1}{\eta_t} \langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla \psi(\mathbf{w}_{t+1}) - \nabla \psi(\mathbf{w}_t) \rangle + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t) \rangle \ . \end{aligned} \tag{4.61}$$

The first inequality is derived from the convexity of loss functions and the feature-aware L_1-regularization. Now, we obtain the first term in the last equation is non-positive from the optimality of $\mathbf{w}_{t+1}$ in the right-hand side of the formula (4.18). By the same reformulation of (2.18), the following inequality can be derived.

$$\begin{aligned} &\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) + \Phi_t(\mathbf{w}_{t+1}) - \Phi_t(\mathbf{w}^*) \\ &\quad \leq \frac{1}{\eta_t} \langle \mathbf{w}^* - \mathbf{w}_{t+1}, \nabla \psi(\mathbf{w}_{t+1}) - \nabla \psi(\mathbf{w}_t) \rangle + \langle \mathbf{w}_t - \mathbf{w}_{t+1}, \nabla \ell_t(\mathbf{w}_t) \rangle \\ &\quad \leq \frac{1}{\eta_t} B_\psi(\mathbf{w}^*, \mathbf{w}_t) - \frac{1}{\eta_t} B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) + \frac{\eta_t}{2\beta} \|\nabla \ell_t(\mathbf{w}_t)\|_*^2 \ . \end{aligned} \tag{4.62}$$

□

The regret bound is derived by using this lemma. From assumptions, $B_\psi(\mathbf{w}, \tilde{\mathbf{w}}) \leq R^2$ is satisfied for all $\mathbf{w}, \tilde{\mathbf{w}} \in \mathcal{W}$ and $\|\nabla \ell_t(\mathbf{w}_t)\|_* \leq G$ is satisfied for all t, and $\|\mathbf{w}\|_1 \leq N$ is satisfied for all $\mathbf{w} \in \mathcal{W}$. By using these conditions, we reformulate the following inequality by summing up the inequalities derived in Lemma 9 from $t = 1$ to $t = T$.

$$\begin{aligned}
\mathrm{Regret}_{\ell+\Phi}(T) &= \sum_{\tau=1}^{T} (\ell_t(\mathbf{w}_t) - \ell_t(\mathbf{w}^*) + \Phi_t(\mathbf{w}_{t+1}) - \Phi_t(\mathbf{w}^*)) + \sum_{\tau=1}^{T} (\Phi_\tau(\mathbf{w}_\tau) - \Phi_\tau(\mathbf{w}_{\tau+1})) \\
&\le \frac{1}{\eta_1} B_\psi(\mathbf{w}^*, \mathbf{w}_1) + \sum_{t=1}^{T-1} \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} \right) B_\psi(\mathbf{w}^*, \mathbf{w}_{t+1}) \\
&\quad - \frac{1}{\eta_T} B_\psi(\mathbf{w}^*, \mathbf{w}_{T+1}) + \sum_{t=1}^{T} \frac{\eta_t}{2\beta} \|\nabla \ell_t(\mathbf{w}_t)\|_*^2 + dVN \\
&\le \frac{R^2}{\eta_T} + \frac{G^2}{2\beta} \sum_{t=1}^{T} \eta_t + dVN \ . \qquad (4.63)
\end{aligned}$$

□

4.4.2 Regret Upper Bound for FRDA

The regret upper bound of FRDA can be derived as the following theorem.

Theorem 13. *Let sequences $\{\mathbf{w}_t\}_{t=1:T+1}$ be generated by* Algorithm 5*. We assume that ψ is β-strongly convex with respect to a norm $\|\cdot\|$. Furthermore, let us assume that there exist scalars $R, N, G \in \mathbb{R}$ such that $B_\psi(\mathbf{w}^*, \mathbf{w}_1) \le R^2$ for the optimal weight vector $\mathbf{w}^* \in \mathcal{W}$ and the initial weight vector $\mathbf{w}_1 \in \mathcal{W}$, $\|\mathbf{w}\|_1 \le N$ for all $\mathbf{w} \in \mathcal{W}$, and $\|\nabla \ell_t(\mathbf{w}_t)\|_* \le G$ for all t. In addition, we assume that* (4.52) *are satisfied. In this case, let us set the initial weight vector as*

$$\mathbf{w}_1 \in \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \Phi_1(\mathbf{w}) \ , \qquad (4.64)$$

$\{\eta_t\}_{t\ge1}$ be a positive non-increasing sequence, and we set V to upper bound the value of $r_{t,q}^{(i)}$ as (4.52), we have for any $T \ge 1$,

$$\mathrm{Regret}_{\ell+\Phi}(T) \le \frac{R^2}{\eta_T} + \frac{G^2}{2\beta} \sum_{\tau=1}^{t} \eta_{\tau-1} + dVN \ . \qquad (4.65)$$

In addition, when we set $\eta_t = \gamma/\sqrt{t}$ where $\gamma > 0$, we obtain $O(\sqrt{T})$ regret bound.

Proof. Our proof is written in line with the proof in [64]. The difference between out proof and the proof of [64] is in the definition of $\Phi_t(\mathbf{w})$ and the effect of this change. First, we define the summation of all previous subgradients as $\mathbf{s}_t$, i.e.,

$$\mathbf{s}_t = \sum_{\tau=1}^{t} \nabla \ell_\tau(\mathbf{w}_\tau) \ . \qquad (4.66)$$

Then, we define two conjugate-type functions for each $t \geq 0$:

$$U_t(\mathbf{s}) = \max_{\mathbf{w}\in F_D} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \right\} , \tag{4.67}$$

$$V_t(\mathbf{s}) = \max_{\mathbf{w}\in \mathcal{W}} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) - \frac{1}{\eta_t} B_\psi(\mathbf{w}, \mathbf{w}_1) \right\} , \tag{4.68}$$

where $F_D = \{\mathbf{w} \in \mathcal{W} | B_\psi(\mathbf{w}, \mathbf{w}_1) \leq R^2\}$ and R is a scalar with a positive value. We note that we set $\eta_0 = \eta_1$. Because $\eta_t > 0$, the function $\sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + B_\psi(\mathbf{w}, \mathbf{w}_1)/\eta_t$ is always strongly convex; therefore, the maximizer of equation (4.68) is unique. We assume that $\mathrm{dom}(V_t) = \mathrm{dom}(U_t)$.

We can derive Lemma 10 in this setting.

Lemma 10. *For any* $\mathbf{s} \in \mathrm{dom}(V_t)$ *and* $t \geq 0$*, we have*

$$U_t(\mathbf{s}) \leq V_t(\mathbf{s}) + \frac{R^2}{\eta_t} . \tag{4.69}$$

Proof. The derivation of this lemma is closely related to the proof of Lemma 6. We start with the definition of $U_t(\mathbf{s})$ and F_D,

$$\begin{aligned}
U_t(\mathbf{s}) &= \max_{\mathbf{w}\in F_D} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \right\} \\
&= \max_{\mathbf{w}\in \mathcal{W}} \min_{\beta \geq 0} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + \beta(R^2 - B_\psi(\mathbf{w}, \mathbf{w}_1)) \right\} \\
&\leq \min_{\beta \geq 0} \max_{\mathbf{w}\in \mathcal{W}} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + \beta(R^2 - B_\psi(\mathbf{w}, \mathbf{w}_1)) \right\} \\
&\leq \max_{\mathbf{w}\in \mathcal{W}} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + \frac{1}{\eta_t} \left(R^2 - B_\psi(\mathbf{w}, \mathbf{w}_1) \right) \right\} \\
&= V_t(\mathbf{s}) + \frac{R^2}{\eta_t} .
\end{aligned} \tag{4.70}$$

□

Let $\pi_t(\mathbf{s})$ be the unique maximizer of $V_t(\mathbf{s})$, and then we have the following equation for each $t \geq 0$.

$$\mathbf{w}_{t+1} = \pi_t(-\mathbf{s}_t) . \tag{4.71}$$

The subgradient of the function V_t is given as:

$$\nabla V_t(\mathbf{s}) = \pi_t(\mathbf{s}) - \mathbf{w}_1 . \tag{4.72}$$

The function V_t is differentiable and strongly convex with convexity parameter β/η_t. Therefore, its gradient is Lipschitz continuous with the constant η_t/β and we obtain the following inequality for any vector $\mathbf{a}, \mathbf{b}$ in $\mathrm{dom}(V_t)$.

$$V_t(\mathbf{a}+\mathbf{b}) \leq V_t(\mathbf{a}) + \langle \mathbf{b}, \nabla V_t(\mathbf{a})\rangle + \frac{\eta_t}{2\beta}\|\mathbf{b}\|_*^2 \ . \tag{4.73}$$

Lemma 11. *For each $t \geq 1$, we have*

$$V_t(-\mathbf{s}_t) + \Phi_t(\mathbf{w}_{t+1}) \leq V_{t-1}(-\mathbf{s}_t) + \left(\frac{1}{\eta_{t-1}} - \frac{1}{\eta_t}\right) B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_1) \ . \tag{4.74}$$

Proof.

$$\begin{aligned}
V_{t-1}(-\mathbf{s}_t) &= \max_{\mathbf{w}\in\mathcal{W}} \left\{ \langle -\mathbf{s}_t, \mathbf{w}-\mathbf{w}_1\rangle - \sum_{\tau=1}^{t-1}\Phi_\tau(\mathbf{w}) - \frac{1}{\eta_{t-1}} B_\psi(\mathbf{w},\mathbf{w}_1)\right\} \\
&\geq \langle -\mathbf{s}_t, \mathbf{w}_{t+1}-\mathbf{w}_1\rangle - \sum_{\tau=1}^{t-1}\Phi_\tau(\mathbf{w}_{t+1}) - \frac{1}{\eta_{t-1}} B_\psi(\mathbf{w}_{t+1},\mathbf{w}_1) \\
&= \langle -\mathbf{s}_t, \mathbf{w}_{t+1}-\mathbf{w}_1\rangle - \sum_{\tau=1}^{t}\Phi_\tau(\mathbf{w}_{t+1}) - \frac{1}{\eta_{t}} B_\psi(\mathbf{w}_{t+1},\mathbf{w}_1) \\
&\qquad + \Phi_t(\mathbf{w}_{t+1}) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) B_\psi(\mathbf{w}_{t+1},\mathbf{w}_1) \\
&= V_t(-\mathbf{s}_t) + \Phi_t(\mathbf{w}_{t+1}) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) B_\psi(\mathbf{w}_{t+1},\mathbf{w}_1) \ .
\end{aligned} \tag{4.75}$$

From equation (4.71), we use $\mathbf{w}_{t+1} = \pi_t(-\mathbf{s}_t)$ for the last equality. □

From the definitions, $B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_1) \geq 0$ and the sequence $\{\eta_t\}_{t\geq 0}$ is non-increasing. Thus, we can derive the following function.

$$\forall t \geq 1 \quad V_t(-\mathbf{s}_t) + \Phi_t(\mathbf{w}_{t+1}) \leq V_{t-1}(-\mathbf{s}_t) \ . \tag{4.76}$$

From these lemmas, we will prove Theorem 13. To measure the quality of the solutions, we define gap sequences $\{\delta_t\}_{t\geq 1}$ and use it for bounding the regret of FRDA. For each $t \geq 1$, the following inequality is satisfied as:

$$\begin{aligned}
\delta_t &= \max_{\mathbf{w}\in F_D}\left\{\sum_{\tau=1}^{t}(\langle\nabla\ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}\rangle + \Phi_\tau(\mathbf{w}_\tau)) - \sum_{\tau=1}^{t}\Phi_\tau(\mathbf{w})\right\} \\
&\geq \max_{\mathbf{w}\in F_D}\left\{\sum_{\tau=1}^{t}(\ell_\tau(\mathbf{w}_\tau) - \ell_\tau(\mathbf{w}) + \Phi_\tau(\mathbf{w}_\tau)) - \sum_{\tau=1}^{t}\Phi_\tau(\mathbf{w})\right\} \\
&= \sum_{\tau=1}^{t}(f_\tau(\mathbf{w}_\tau) + \Phi_\tau(\mathbf{w}_\tau)) - \min_{\mathbf{w}\in F_D}\left\{\sum_{\tau=1}^{t}(f_\tau(\mathbf{w}) + \Phi_\tau(\mathbf{w}))\right\} \\
&= \mathrm{Regret}_{\ell+\Phi}(t) \ .
\end{aligned} \tag{4.77}$$

To derive the first inequality, we use the convexity of loss functions.

In addition, we can derive the upper bound for δ_t. To achieve this, we reformalize gap sequences as:

$$\begin{aligned}\delta_t &= \sum_{\tau=1}^{t}(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \Phi_\tau(\mathbf{w}_\tau)) + \max_{\mathbf{w}\in F_D}\left\{\langle \mathbf{s}_t, \mathbf{w}_1 - \mathbf{w}\rangle - \sum_{\tau=1}^{t}\Phi_\tau(\mathbf{w})\right\}\\ &= \sum_{\tau=1}^{t}(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \Phi_\tau(\mathbf{w}_\tau)) + U_t(-\mathbf{s}_t)\\ &\le \sum_{\tau=1}^{t}(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \Phi_\tau(\mathbf{w}_\tau)) + V_t(-\mathbf{s}_t) + \frac{R^2}{\eta_t}\ . \end{aligned} \tag{4.78}$$

After that, we will discuss the upper bound for the right-hand side of inequality (4.78). For each $\tau \ge 1$,

$$\begin{aligned} V_\tau(-\mathbf{s}_\tau) + \Phi_\tau(\mathbf{w}_{\tau+1}) &\le V_{\tau-1}(-\mathbf{s}_\tau)\\ &= V_{\tau-1}(-\mathbf{s}_{\tau-1} - \nabla \ell_\tau(\mathbf{w}_\tau))\\ &\le V_{\tau-1}(-\mathbf{s}_{\tau-1}) + \langle -\nabla \ell_\tau(\mathbf{w}_\tau), \nabla V_{\tau-1}(-\mathbf{s}_{\tau-1})\rangle + \frac{\eta_{\tau-1}}{2\beta}\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_*^2\\ &= V_{\tau-1}(-\mathbf{s}_{\tau-1}) + \langle -\nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \frac{\eta_{\tau-1}}{2\beta}\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_*^2\ , \end{aligned} \tag{4.79}$$

where these four steps above used (4.76), (4.66), (4.73), and (4.72), respectively. In summary, we can derive

$$\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \Phi_\tau(\mathbf{w}_{\tau+1}) \le V_{\tau-1}(-\mathbf{s}_{\tau-1}) - V_\tau(-\mathbf{s}_\tau) + \frac{\eta_{\tau-1}}{2\beta}\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_*^2\ , \tag{4.80}$$

where $\tau \ge 1$.

Summing up the above inequalities from $\tau = 1$ to t, and noting that $V_0(-\mathbf{s}_0) = V_0(\mathbf{0}) = 0$, we can derive

$$\sum_{\tau=1}^{t}(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \Phi_\tau(\mathbf{w}_{\tau+1})) + V_t(-\mathbf{s}_t) \le \frac{1}{2\beta}\sum_{\tau=1}^{t}\eta_{\tau-1}\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_*^2\ . \tag{4.81}$$

From Lemma 8, we obtain

$$\sum_{\tau=1}^{t}(\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1\rangle + \Phi_\tau(\mathbf{w}_\tau)) + V_t(-\mathbf{s}_t) \le \frac{1}{2\beta}\sum_{\tau=1}^{t}\eta_{\tau-1}\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_*^2 + dVN\ . \tag{4.82}$$

Combining (4.78) with (4.82), we can derive

$$R_{\ell+\Phi}(t) \le \delta_t \le \frac{R^2}{\eta_t} + \frac{1}{2\beta}\sum_{\tau=1}^{t}\eta_{\tau-1}\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_*^2 + dVN\ . \tag{4.83}$$

From the condition of $\|\nabla \ell_\tau(\mathbf{w}_\tau)\|_* \leq G$ for all τ, we can reformalize (4.83) as:

$$\mathrm{Regret}_{\ell+\Phi}(t) \leq \delta_t \leq \frac{R^2}{\eta_t} + \frac{G^2}{2}\sum_{\tau=1}^{t} \eta_{\tau-1} + dVN \ , \tag{4.84}$$

When we set $\eta_t = \gamma/\sqrt{t}$ where $\gamma > 0$, we can derive

$$\mathrm{Regret}_{\ell+\Phi}(t) \leq \left(\frac{R^2}{\gamma} + G^2\gamma\eta_{\tau-1}\right)\sqrt{t} + dVN \ , \tag{4.85}$$

where we use inequality

$$\sum_{\tau=1}^{t-1} \frac{1}{\sqrt{\tau}} \leq 1 + \int_1^t \frac{1}{\sqrt{\tau}} d\tau = 2\sqrt{t} - 1 \ . \tag{4.86}$$

□

4.4.3 Regret Upper Bound for FTPFRL

The following theorem establishes the upper bound on the regret of FTPFRL. The theorem of upper bounding the regret of FTPFRL is as follows:

Theorem 14. *Let $\{\mathbf{w}_t\}_{t=1:T+1}$ be sequences generated by* Algorithm 6*. We assume that ψ is β-strongly convex with respect to a norm $\|\cdot\|$. Furthermore, let us assume that there exist scalars $R, N, G \in \mathbb{R}$ such that $B_\psi(\mathbf{w}^*, \mathbf{w}_1) \leq R^2$ for the optimal weight vector $\mathbf{w}^* \in \mathcal{W}$ and the initial weight vector $\mathbf{w}_1 \in \mathcal{W}$, $\|\mathbf{w}\|_1 \leq N$ for all $\mathbf{w} \in \mathcal{W}$, and $\|\nabla \ell_t(\mathbf{w}_t)\|_* \leq G$ for all t. In this case, let us set the initial weight vector as*

$$\mathbf{w}_1 \in \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}} \Phi_1(\mathbf{w}) \ , \tag{4.87}$$

Let us set a positive non-increasing sequence of $\{\eta_t\}_{t\geq 1}$, $1/\eta_0 = 0$, and $V \in \mathbb{R}$ such that (4.52) is satisfied, we obtain

$$\mathrm{Regret}_{\ell+\Phi}(T) \leq \frac{R^2}{\eta_T} + \frac{G^2}{2\beta}\sum_{\tau=1}^{t-1} \eta_\tau + dVN \ . \tag{4.88}$$

When we set $\eta_t = \gamma/\sqrt{t}$ where γ is a constant, for any $T \geq 1$, we obtain $O(\sqrt{T})$ regret upper bound.

We will prove this theorem to upper bounds on regret for FTPFRL with the feature-aware L_1-regularization below. The proof of this bound basically follows the FRDA analysis. We need some modifications from the FRDA derivation due to the effect of the Bregman divergence changes.

Proof. As the same as the analysis of FRDA, we define the summation of all previous subgradients as $\mathbf{s}_t$ and two conjugate-type functions for each $t \geq 0$:

$$\mathbf{s}_t = \sum_{\tau=1}^{t} \nabla \ell_\tau(\mathbf{w}_\tau) \ . \tag{4.89}$$

$$U_t(\mathbf{s}) = \max_{\mathbf{w} \in F_t} \{\langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w})\} \ , \tag{4.90}$$

$$V_t(\mathbf{s}) = \max_{\mathbf{w} \in \mathcal{W}} \{\langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) - \bar{B}_t(\mathbf{w})\} \ , \tag{4.91}$$

where

$$\bar{B}_t(\mathbf{w}) = \sum_{\tau=1}^{t} \left(\frac{1}{\eta_\tau} - \frac{1}{\eta_{\tau-1}} \right) B_\psi(\mathbf{w}, \mathbf{w}_\tau) \tag{4.92}$$

where $F_t = \{\mathbf{w} \in \mathcal{W} | \bar{B}_t(\mathbf{w}) \leq R^2/\eta_t\}$ and R is a scalar with a positive value. When a sequence of step sizes $\{\eta_t\}_{t \geq 1}$ is positive and non-increasing, the function $\bar{B}_t(\mathbf{w})$ is always strongly convex because each $B_\psi(\mathbf{w}, \mathbf{w}_\tau)$ is strongly convex and its summation also becomes strongly convex. The strong convexity parameter is β/η_t. We assume that $\mathrm{dom}(V_t) = \mathrm{dom}(U_t)$.

We derive Lemma 12 in this setting.

Lemma 12. *For any* $\mathbf{s} \in \mathrm{dom}(V_t)$ *and* $t \geq 0$*, we have*

$$U_t(\mathbf{s}) \leq V_t(\mathbf{s}) + \frac{R^2}{\eta_t} \ . \tag{4.93}$$

Proof. We start with the definition of $U_t(\mathbf{s})$ and F_t,

$$\begin{aligned}
U_t(\mathbf{s}) &= \max_{\mathbf{w} \in F_t} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \right\} \\
&= \max_{\mathbf{w} \in \mathcal{W}} \min_{\beta \geq 0} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + \beta \left(\frac{R^2}{\eta_t} - \bar{B}_t(\mathbf{w}) \right) \right\} \\
&\leq \min_{\beta \geq 0} \max_{\mathbf{w} \in \mathcal{W}} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + \beta \left(\frac{R^2}{\eta_t} - \bar{B}_t(\mathbf{w}) \right) \right\} \\
&\leq \max_{\mathbf{w} \in \mathcal{W}} \left\{ \langle \mathbf{s}, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) + \left(\frac{R^2}{\eta_t} - \bar{B}_t(\mathbf{w}) \right) \right\} \\
&= V_t(\mathbf{s}) + \frac{R^2}{\eta_t} \ .
\end{aligned} \tag{4.94}$$

In the second inequality, we set $\beta = 1$. □

Let $\pi_t(\mathbf{s})$ be the unique maximizer of $V_t(\mathbf{s})$, and then we have the following equation for each $t \geq 1$.

$$\mathbf{w}_{t+1} = \pi_t(-\mathbf{s}_t) \ . \tag{4.95}$$

The subgradient of the function V_t is given as:

$$\nabla V_t(\mathbf{s}) = \pi_t(\mathbf{s}) - \mathbf{w}_1 \ . \tag{4.96}$$

The function V_t is differentiable and strongly convex with convexity parameter β/η_t. Therefore, its gradient is Lipschitz continuous with the constant η_t/β and we obtain the following inequality for any vector $\mathbf{a}, \mathbf{b}$ in $\mathrm{dom}(V_t)$.

$$V_t(\mathbf{a}+\mathbf{b}) \leq V_t(\mathbf{a}) + \langle \mathbf{b}, \nabla V_t(\mathbf{a}) \rangle + \frac{\eta_t}{2\beta} \|\mathbf{b}\|_*^2 \ . \tag{4.97}$$

Lemma 13. *For each $t \geq 1$, we have*

$$V_t(-\mathbf{s}_t) + \Phi_t(\mathbf{w}_{t+1}) \leq V_{t-1}(-\mathbf{s}_t) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) \ . \tag{4.98}$$

Proof.

$$\begin{aligned} V_{t-1}(-\mathbf{s}_t) &= \max_{\mathbf{w} \in \mathcal{W}} \left\{ \langle -\mathbf{s}_t, \mathbf{w} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t-1} \Phi_\tau(\mathbf{w}) - \bar{B}_{t-1}(\mathbf{w}) \right\} \\ &\geq \langle -\mathbf{s}_t, \mathbf{w}_{t+1} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t-1} \Phi_\tau(\mathbf{w}_{t+1}) - \bar{B}_{t-1}(\mathbf{w}_{t+1}) \\ &= \langle -\mathbf{s}_t, \mathbf{w}_{t+1} - \mathbf{w}_1 \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}_{t+1}) - \bar{B}_t(\mathbf{w}_{t+1}) \\ &\qquad + \Phi_t(\mathbf{w}_{t+1}) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) \\ &= V_t(-\mathbf{s}_t) + \Phi_t(\mathbf{w}_{t+1}) + \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) \ . \end{aligned} \tag{4.99}$$

□

From the definitions, $B_\psi(\mathbf{w}_{t+1}, \mathbf{w}_t) \geq 0$ and sequence $\{\eta_t\}_{t\geq 1}$ is non-increasing. Thus, we can derive the following function.

$$\forall t \geq 1 \quad V_t(-\mathbf{s}_t) + \Phi_t(\mathbf{w}_{t+1}) \leq V_{t-1}(-\mathbf{s}_t) \ . \tag{4.100}$$

From these lemmas, we will prove Theorem 14. To measure the quality of the solutions, we define gap sequences $\{\delta_t\}_{t\geq 1}$ and use it for bounding the regret of FTPFRL. For each

$t \geq 1$, the following inequality is satisfied as:

$$\begin{aligned}
\delta_t &= \max_{\mathbf{w} \in F_D} \left\{ \sum_{\tau=1}^{t} (\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w} \rangle + \Phi_\tau(\mathbf{w}_\tau)) - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \right\} \\
&\geq \max_{\mathbf{w} \in F_D} \left\{ \sum_{\tau=1}^{t} (\ell_\tau(\mathbf{w}_\tau) - \ell_\tau(\mathbf{w}) + \Phi_\tau(\mathbf{w}_\tau)) - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \right\} \\
&= \sum_{\tau=1}^{t} (f_\tau(\mathbf{w}_\tau) + \Phi_\tau(\mathbf{w}_\tau)) - \min_{\mathbf{w} \in F_D} \left\{ \sum_{\tau=1}^{t} (f_\tau(\mathbf{w}) + \Phi_\tau(\mathbf{w})) \right\} \\
&= \mathrm{Regret}_{\ell+\Phi}(t) \; . \qquad (4.101)
\end{aligned}$$

In addition, we derive the upper bound for δ_t. To achieve this, we reformalize gap sequences as:

$$\begin{aligned}
\delta_t &= \sum_{\tau=1}^{t} (\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1 \rangle + \Phi_\tau(\mathbf{w}_\tau)) + \max_{\mathbf{w} \in F_t} \left\{ \langle \mathbf{s}_t, \mathbf{w}_1 - \mathbf{w} \rangle - \sum_{\tau=1}^{t} \Phi_\tau(\mathbf{w}) \right\} \\
&= \sum_{\tau=1}^{t} (\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1 \rangle + \Phi_\tau(\mathbf{w}_\tau)) + U_t(-\mathbf{s}_t) \\
&\leq \sum_{\tau=1}^{t} (\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1 \rangle + \Phi_\tau(\mathbf{w}_\tau)) + V_t(-\mathbf{s}_t) + \frac{R^2}{\eta_t} \; . \qquad (4.102)
\end{aligned}$$

After that, we derive the upper bound for the right-hand side of inequality. By the similar discussion of FRDA, we can derive

$$\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1 \rangle + \Phi_\tau(\mathbf{w}_{\tau+1}) \leq V_{\tau-1}(-\mathbf{s}_{\tau-1}) - V_\tau(-\mathbf{s}_\tau) + \frac{\eta_{\tau-1}}{2\beta} \| \nabla \ell_\tau(\mathbf{w}_\tau) \|_*^2 \; , \qquad (4.103)$$

where $\tau \geq 2$ and for $\tau = 1$,

$$\langle \nabla \ell_\tau(\mathbf{w}_\tau), \mathbf{w}_\tau - \mathbf{w}_1 \rangle + \Phi_\tau(\mathbf{w}_{\tau+1}) \leq V_{\tau-1}(-\mathbf{s}_{\tau-1}) - V_\tau(-\mathbf{s}_\tau) \; , \qquad (4.104)$$

We cut the discussion of this derivation because this procedure is almost the same as FRDA. As a result, we obtain the following formula.

$$\mathrm{Regret}_{\ell+\Phi}(t) \leq \delta_t \leq \frac{R^2}{\eta_t} + \frac{G^2}{2} \sum_{\tau=1}^{t-1} \eta_\tau + dVN \; , \qquad (4.105)$$

When we set $\eta_t = \gamma / \sqrt{t}$ where $\gamma > 0$, we derive

$$\mathrm{Regret}_{\ell+\Phi}(t) \leq \left(\frac{R^2}{\gamma} + G^2 \gamma \right) \sqrt{t} + dVN \; . \qquad (4.106)$$

□

4.4.4 Discussion

In the algorithm part, we select the L_2 norm as the *Legendre* functions. We note in this case that our derived regret bounds are optimal even up to the multiplicative constant of the dominant term $O(\sqrt{t})$ when the input space is restricted to a L_2-ball and we can select an appropriate step size [127]. However, when the input space has other restriction (such as the feasible region is D-dimensional simplex), the upper regret bounds of our proposed algorithms become non-optimal with respect to the constant factor of the dominant term $O(\sqrt{t})$ like COMID and RDA [73].

4.5 Experiments

We assessed the performance of our framework in several binary classification tasks. We used the Amazon.com review sentiment classification datasets [12] and the news category classification datasets for 20NewsGroups [13]. Table 5.2 lists the specifications of each dataset, including the number of features, data, classes, and data types. The first one is a sentiment classification task for reviews of Amazon.com goods. In this task, each sparse online learning algorithm tries to determine whether each review text states a positive or negative opinion for a specific category of products. We used four categories of review documents: books, dvd, electronics, and kitchen. Feature vectors consisted of the occurrence counts of unigram and bigram words in each review text. The second task is a news categorization task in which sparse online learning algorithms predict the category which each news article is assigned to. We used the 20Newsgroups datasets (news20). This dataset consists of about $20,000$ news articles and each article is assigned to one of 20 categories. We used two subsets of the news20: ob-2-1 and sb-2-1 [128]. The name of these datasets has specific meaning. The first letter indicates whether there is an overlap between two categories. The 'o' indicates 'overlap', 's' indicates 'separated' for the first letter of each subset name. Predictions over the 'overlap' dataset is more difficult than the one over 'separated'. The second letter indicates the heterogeneity between categories. We used the balanced dataset for each experiment. The first number shows the number of categories. We used the binary classification datasets. Feature vectors consist of the occurrence counts of unigram words in each document.

We conducted several types of experiments. First, we validated the effect of q toward the predictive performance by setting $1, 2$, or ∞. We compare the predictive performance of our proposed algorithms, FR-COMID, FRDA, and FTPFRL, by changing the setting

Table. 4.2. Dataset specifications

	# of data	# of features	# of classes	data type
books	4,465	332,440	2	review
dvd	3,586	282,900	2	review
electronics	5,681	235,796	2	review
kitchen	5,945	205,665	2	review
ob-2-1	1,000	5,942	2	news
sb-2-1	1,000	6,276	2	news

of q. After that, we validated the effect of our proposed feature-aware regularization methods by comparing the predictive performance with the original COMID, RDA and FTPRL. We also compared our methods with the AdaGrad versions of COMID and RDA [73]. The results of AdaGrad versions were very similar to the original COMID and RDA. We assume that it is natural because the optimal points are the same as the ones derived from the original algorithms.

The detail experimental settings are introduced as below: We used the hinge loss function as loss functions, and we used a squared Euclidean distance of the form $B_\phi(\mathbf{a}, \mathbf{b}) = \frac{1}{2}\|\mathbf{a} - \mathbf{b}\|_2^2$ as Bregman divergence for all algorithms we used. A sequence of step sizes is set so as to achieve the $O(\sqrt{T})$ regret bound with respect to the number of rounds T. Therefore, we set $\eta_t = 1/\sqrt{t}$ in these algorithms*1. Moreover, we set $V = 10^6$ in order to satisfy the upper bound of the value of $r_{t,q}^{(i)}$ for our framework. However, in this experiment, the value of $r_{t,q}^{(i)}$ did not exceed 10^6; thus, the value of V did not influence the result. Upon evaluating the predictive performances of sparse online learning algorithms, 10-fold cross-validation was used to tune the hyper-parameter λ to achieve high precision through a sparse weight vector. After tuning λ, sparse online learning algorithms were iterated 20 times to optimize weight vectors; that is, we ran these algorithms through all training data 20 times. Because the definition of the regret changed between the feature-aware version of sparse online learning algorithms and the original ones, we compare the result by the prediction accuracy and sparseness rate of the finally derived weight vector over the test dataset. This score matches how the derived weight vector can precisely predict the label of the next coming data or not. This criterion is introduced in Section 2.2.

*1 The value of η_t has an insignificant effect on the result as long as these variables satisfy $\eta_t \propto 1/\sqrt{t}$.

Table. 4.3. Precision (sparseness) of FR-COMID versus parameter q (Iterations : 20)

	FR-COMID		
	$(q=1)$	$(q=2)$	$(q=\infty)$
books	83.91	**85.24**	84.46
	(47.04)	(72.69)	(**87.05**)
dvd	81.26	82.10	**83.52**
	(49.43)	(73.69)	(**85.57**)
electronics	86.24	87.68	**88.05**
	(53.26)	(71.76)	(**85.62**)
kitchen	88.63	88.95	**90.11**
	(53.04)	(78.29)	(**83.84**)
ob-2-1	**95.00**	93.90	94.90
	(52.89)	(**76.00**)	(70.14)
sb-2-1	96.50	95.80	**97.20**
	(66.33)	(**78.32**)	(78.31)

Table. 4.4. Change in the truncation parameter $r_{t,q}^{(i)}$ versus parameter q

	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$q=1$	1	2	3	4	5	6
$q=2$	1	$\sqrt{2}$	$\sqrt{3}$	2	$\sqrt{5}$	$\sqrt{6}$
$q=\infty$	1	1	1	1	1	1

4.5.1 Experimental Results of the Feature-aware Regularization Methods

We first check the performance of sparse online learning algorithms with the feature-aware L_1-regularization by changing the variable q. Tables 4.3, 4.5, and 4.6 show the experimental results for FR-COMID, FRDA, and FTPFRL against the change in parameter q. The figures written in parentheses represent sparseness rates of the finally derived weight vectors. The sparseness rates is calculated by the formula: "(the number of non-zero features) is divided by (the number of features) * 100.0". With respect to each dataset, the figures representing the lowest error rate or the highest sparseness rate are written in a **bold** form.

We note that when $q = 1$, it is very difficult to obtain a sparse weight vector except for the 100% sparseness rate, in other words the zero vector. When we set $q = 1$, we cannot derive the helpful sparse weight vector with sufficient predictive ability by these sparse online learning algorithms. This phenomenon occurs due to the over-truncation originated from the incremental value of $r_{t,q}^{(i)}$ added in round t, which is equal to the i-th element of the t-th subgradient vector $(\nabla \ell_t(\mathbf{w}_t))^{(i)}$ as described in Figure 4.1. Therefore, once the feature-aware parameters, that is, $r_{t,q}^{(i)}$ in FR-COMID and $\bar{r}_{t,q}^{(i)}$ in FRDA and FTPFRL, become large, it is hard to recover the parameter of the i-th feature because these parameters increase linearly. These phenomena make it very hard to control hyper-parameters λ and as a result, to obtain sparse weight vectors that are helpful for prediction. When $q > 1$, this problem is unlikely to occur because $r_{t,q}^{(i)}$ added in round t decreases as the round number t increases. Table 4.4 show these characteristics in the table form, which indicates the changes in the truncation parameter $r_{t,q}^{(i)}$ when the subgradient value $(\nabla \ell_t(\mathbf{w}_t))^{(i)} = 1$ for all t.

From these results shown in Table 4.3, we can say that FR-COMID $q = \infty$ outperforms the other FR-COMID algorithms in most experiments. In particular, $q = \infty$ settings obtain very good solutions in terms of sparsity. These experimental results show that it is relatively difficult for COMID-type algorithms to obtain a sparse solution. Table 4.5 indicates that there are no significant differences between FRDA $q = 2$ and FRDA $q = \infty$. In addition, FRDA algorithms obtain the most sophisticated results in terms of precision compared with FR-COMID and FTPFRL. The results shown in Table 4.6 indicate that FTPFRL $q = \infty$ outperforms the other q settings of FTPFRL in most experiments. These results show that FTPFRL can obtain more sparse solution than other two types of algorithms while preserving its predictive ability.

4.5.2 Comparisons with Previous Work

Table 4.7 shows the experimental results of the comparisons between feature-aware regularization frameworks and the original sparse online learning algorithms. The figures of the lower error rate or higher sparsity rate between the original sparse online learning algorithm and the one with the feature-aware regularization methods are written in **bold**.

First, we compare the results of FR-COMID $q = \infty$, the best-performer in Table 4.3, with the original COMID. It is clear from this table that FR-COMID outperforms COMID in terms of the precision rate in all datasets; however, from the viewpoint of the sparsity rate, FR-COMID is inferior to COMID in two out of six datasets. These experimental results reveal that feature-aware COMID algorithms had better precisions than the orig-

Table. 4.5. Precision (sparseness) of FRDA versus parameter q (Iterations : 20)

	FRDA		
	($q = 1$)	($q = 2$)	($q = \infty$)
books	86.35	**86.88**	86.50
	(30.34)	(87.73)	(**91.13**)
dvd	**87.34**	87.23	86.31
	(31.31)	(82.97)	(**94.74**)
electronics	**90.14**	89.49	89.30
	(31.98)	(**95.89**)	(88.93)
kitchen	**92.11**	91.02	90.95
	(35.08)	(95.90)	(**97.36**)
ob-2-1	**98.50**	97.20	96.20
	(54.39)	(80.74)	(**82.30**)
sb-2-1	**99.20**	98.60	98.80
	(60.25)	(86.99)	(**93.69**)

Table. 4.6. Precision (sparseness) of FTPFRL versus parameter q (Iterations : 20)

	FTPFRL		
	($q = 1$)	($q = 2$)	($q = \infty$)
books	84.10	**85.15**	84.71
	(47.90)	(94.37)	(**97.27**)
dvd	81.06	83.29	**83.32**
	(49.48)	(90.94)	(**98.46**)
electronics	85.83	86.80	**88.29**
	(53.93)	(96.93)	(**99.11**)
kitchen	88.41	89.03	**90.28**
	(54.15)	(97.12)	(**97.50**)
ob-2-1	**95.10**	93.90	94.90
	(52.47)	(87.56)	(**90.50**)
sb-2-1	96.50	96.00	**97.10**
	(68.10)	(74.56)	(**96.98**)

inal method, but did not always derive sparse solutions due to the unstable behavior of the COMID-based algorithms.

After that, we show that FRDA outperformed in terms of both the precision rate and sparseness rate in four out of six datasets compared with the original RDA. We used $q = \infty$ as the best competitor of the FRDA. FRDA is inferior to RDA on the electronics and ob-2-1 dataset; however, FRDA $q = 2$ outperformed RDA on them. The results mean that the feature-aware truncation methods improve the prediction accuracy by retaining rare but informative features. At the same time, FRDA could truncate unimportant features to attain the same sparseness. Comparing FRDA with FR-COMID, we see that FRDA significantly outperforms FR-COMID in terms of precision on all datasets and had better sparsity as well. This result indicates that the RDA-based algorithms have significant advantages over the COMID-based ones we described in Chapter 2 and 4. These results back up the experimental results presented in some other papers [64, 72, 31, 32] showing that RDA is superior to FOBOS.

Finally, we compared FTPFRL $q = \infty$ with FTPRL. Experimental results show that FTPFRL outperform FTPRL in terms of precision on all datasets. On the other hand, there is no significant difference between FTPFRL and FTPRL with respect to the sparsity rate. From these experimental results, we conclude that the feature-aware truncation methods improve the prediction performance of weight vectors with at least similar than and the same sparseness as the state-of-the-art methods.

Furthermore, to check how our proposed feature-aware regularization methods work, we conduct experiments that check what features FRDA and RDA obtain as important discriminative ones (Table 4.8). We used the books dataset in this experiment. The listed features are for where one algorithm has determined that these features are important while the other algorithm has determined that these are unimportant. Important features were chosen from a set of features in the Top-100 relative to the size of the values. The result of Figure 4.8 indicates that FRDA could retain rare features, e.g., "smearing" and "some interesting", while these features were overly truncated by RDA. On the other hand, we can see that FRDA truncated frequently occurring but non predictive terms, such as "his" and "more", while RDA retained.

4.5.3 Comparison with Feature-based Feature-aware Regularization Methods

Furthermore, we show the validity of the subgradient-based definition of the feature-aware L_1-regularization compared with the feature-based definition. For feature-based definition

Table. 4.7. Comparison with the original sparse online learning algorithms with respects to precision (sparseness) rate (Iterations : 20)

	FR-COMID ($q=\infty$)	COMID	FRDA ($q=\infty$)	RDA	FTPFRL ($q=\infty$)	FTPRL
books	**84.46** (**87.05**)	84.28 (80.39)	**86.50** (**91.13**)	86.00 (88.69)	**84.71** (**97.27**)	83.83 (93.08)
dvd	**83.52** (85.57)	81.87 (**91.60**)	**86.31** (**94.74**)	85.58 (93.23)	**83.32** (98.46)	81.68 (**99.38**)
electronics	**88.05** (85.62)	87.41 (**90.54**)	**89.30** (88.93)	88.94 (**91.93**)	**88.29** (99.11)	87.59 (**99.59**)
kitchen	**90.11** (83.84)	89.27 (**90.36**)	**90.95** (**97.49**)	90.23 (91.23)	**90.28** (97.50)	89.39 (**99.06**)
ob-2-1	**94.90** (70.14)	94.00 (**83.05**)	96.20 (**82.30**)	**96.90** (76.96)	**94.90** (**90.50**)	93.80 (80.16)
sb-2-1	**97.20** (**78.31**)	96.00 (70.91)	**98.80** (**93.69**)	97.70 (89.74)	**97.10** (**96.98**)	95.70 (86.45)

Table. 4.8. Samples of important features.

FRDA ($p=\infty$)	RDA
"some interesting" (117)	"his" (1491)
"was blatantly" (101)	"more" (877)
"be successful" (64)	"time" (1161)
"a constructive" (29)	"almost" (376)
"smearing" (30)	"say" (2407)

of the feature-aware regularization methods, we redefine $r_{t,q}^{(i)}$ by using the information of a sequence of input vectors as follows:

$$r_{t,q}^{(i)} = \sqrt[q]{\sum_{\tau=1}^{t} \left| x_{\tau}^{(i)} \right|^{q}} . \tag{4.107}$$

We compare the original subgradient-based regularization methods with the feature-based regularization methods by using FRDA. Table 4.9 shows the experimental results.

From the results, we can see that FRDA $q=\infty$ based on the subgradient information definition outperform other algorithms in terms of both sparseness and precision in four

Table. 4.9. Experimental results of FRDA with the subgradient-based feature-aware regularization methods and feature-based feature-aware regularization methods : the rates of error and sparseness (Iterations : 20). The sparseness rate are in parentheses. The best figures among all algorithms except FRDA ($q = 1$) for each dataset are written in **bold**.

	FRDA ($q = \infty$) Subgradient-based	FRDA ($q = 1$) Feature-based	FRDA ($q = 2$) Feature-based	FRDA ($q = \infty$) Feature-based
books	86.50	87.14	**86.56**	86.36
	(**91.13**)	(32.28)	(86.77)	(91.09)
dvd	**86.31**	87.12	85.36	86.12
	(**94.74**)	(32.36)	(89.23)	(89.12)
electronics	89.30	89.92	89.07	**89.49**
	(**88.93**)	(39.66)	(87.44)	(88.56)
kitchen	**90.95**	92.04	90.46	90.88
	(**97.36**)	(40.51)	(89.40)	(88.34)
ob-2-1	**96.20**	95.10	94.60	93.80
	(**82.30**)	(78.28)	(78.44)	(75.01)
sb-2-1	**98.80**	90.80	97.40	97.30
	(**93.69**)	(78.92)	(87.80)	(87.87)

out of six tasks. In the other two tasks, the subgradient-based feature-aware regularization methods derive the most sparsest predictive model with a competitive prediction ability. These results support the discussion in Section 4.3.

We note that the results are very unstable in FRDA $q = 1$ even if we define the feature-aware regularization through the feature-based information.

4.6 Chapter Summary

In this chapter, we establish the notion of the truncation bias happened in the sparse online learning settings. We first point out the effect of the truncation bias as the crucial component to alleviate the predictive performance of sparse online learning. To heal the truncation bias in an online manner, we propose the feature-aware L_1-regularization methods for sparse online learning. Our proposed feature-aware truncation methods integrate the information on all previous subgradients into the L_1-regularization term to adjust the truncation effect of each feature in an online manner without pre-processing. By introducing the feature-aware regularization methods into the state-of-the-art subgradient-based sparse online learning algorithms, we solve the difficulties of the simple L_1-regularization in an online setting, where rare or low value range features are truncated on a priority basis even if these features were important for prediction. We mathematically formalize the feature-aware L_1-regularization methods and apply them to several novel sparse online learning algorithms. In addition, we prove theoretical guarantees of our framework by deriving upper bounds on regret and show several computational efficient form for fast computations. We conduct several experiments by using six binary classification datasets. Experimental results reveal that the feature-aware L_1-regularization methods outperform the plain sparse online learning algorithms in terms of predictive ability, while preserving similar sparsity rates. Moreover, we show that our methods could retain rare but informative features in these experiments.

There are several remaining issues about the feature-aware regularization methods. The first issue is whether we can modify the algorithm to choose the optimal parameter q in an online setting. In this thesis, we only evaluate the performances of two-types of classification tasks and in these tasks, FRDA outperformed the others. We would like to investigate what algorithms are the best to be applied to other tasks, such as regression and learning-to-rank. In addition, there are some other novel subgradient-based sparse online learning algorithms (e.g. [129]). We need to explore the effect of the feature-aware L_1-regularization methods for these other algorithms. The applicability for stochastic learning setting and the combination with AdaGrad are other directions of the future research. The feature-aware regularization methods do not have the guarantee of the online to batch conversion due to the existence of the time-varying regularization yet. We will investigate these questions and further extend our proposed methods.

Chapter 5

Online and Stochastic Learning with a Human Cognitive Bias

Incremental learning framework, such as online learning and stochastic learning, is an effective for supervised learning tasks in the machine learning community. In particular, online learning and stochastic learning are advantageous for massive data analysis owing to the sequential processing nature of the data compared with batch learning framework. Among a large variety of machine learning tasks, binary classification problems are one of the most important tasks. In binary classification tasks, algorithms learn how to classify each datum into $\{-1, 1\}$. For binary classification tasks, these incremental learning algorithms process a bunch of data one by one, make predictions over the received datum, and update its classification rule at every round. We focus on these binary classification problems in this chapter.

In this chapter, we focus on another bias originating from the human's perception. Due to its incremental learning procedure, incremental learning algorithms sometimes make wrong predictions on the data that were correctly classified in the past. The standard performance evaluation measures for incremental learning, such as the regret or the expected loss, do not reflect the history of the past prediction results. Therefore, state-of-the-art online and stochastic algorithms have not considered this inconsistent behavior as a crucial factor. The key statement in this thesis is that such an inconsistent phenomenon has a crucial impact on the evaluation of algorithms when humans are evaluators. We find that the performance evaluation conducted by humans does not match the standard objective functions of incremental learning algorithms, such as the normal regret and expected loss, due to a human cognitive bias. Humans have a cognitive bias that they attach a higher value to the data that were correctly classified in the past than the other data. This effect originates from the endowment effect that has been widely analyzed in the field of behavior economics. We show two motivating examples in which this cognitive bias has important roles:

User utility maximization
Incremental learning algorithms has been used in many services such as email filtering [9] and advertisement placement optimization [10]. Many users continuously utilize services whose prediction rules have been changed over time. Furthermore, users sometimes receive the same data with prediction results as previously seen. Negative flips for these data, in other words, the misclassification over the data which were correctly classified in the past, may drastically decrease the utilities of these users.

Interactive annotation
There are many human-computer interaction systems based on incremental learning framework such as annotation systems based on active learning framework [130]. These interactive annotation systems is important to improve generalization ability with a small amount of data and reduce the annotation cost. To generate less noisy labeled data, encouraging people to make annotations is crucial for better performance. Some annotators may feel frustrated annotating the data correctly classified in the past as wrong ones.

We find that this phenomenon should be treated appropriately to maximize the human's utility. A human cognitive bias becomes the key to appropriately deal with the effect of such an inconsistent prediction behavior. We first explain how such an inconsistent prediction behavior affects the human utility through the endowment effect. Then, we conduct an experiment to verify whether such a prediction behavior negatively affects human's evaluations and show its effect for user utilities as a human cognitive bias. From this result, we see that we need to internalize this cost as the divestiture loss and integrate this bias into objective functions to formalize new evaluation measures for online and stochastic learning. Therefore, we formulate the new objective functions with the human cognitive bias for online and stochastic learning framework. We note that we easily solve these problems if algorithms could store all previous examples and their prediction results in the memory and scan all data in each round; however, the memorization and scan of all data are unpractical for massive data analysis due to the memory size and computational resource constraints. In addition, memorization and scan violates the advantages of incremental learning framework. To maximize the availability of incremental learning algorithms, these algorithms which interact with humans need some methods to heal the human cognitive bias derived from the past prediction history. We develop new algorithms without the memorization and scan of the past prediction history. This algorithm is based on OGD/SGD algorithms and our proposed variants achieve the theoretical guarantees with respect to the new objectives. We lastly conduct experiments to evaluate the performance of our proposed algorithms and the results show our algorithms outperform the conventional ones with respect to the new objectives.

Our framework is similar to the cost-sensitive learning framework wherein the importance of false positives is different from the one of false negatives. For example, Langford and Beygelzimer [131] provides a reduction technique that works for classification ranging from cost-sensitive to simple binary problems. Wang, Zhao, and Hoi [132] proposes a cost-sensitive online classification framework. In our framework, the cost of each example dynamically changes depending on the history of past prediction results. Therefore, the problem becomes more complicated than these cost-sensitive frameworks.

The constitution of this chapter is as follows: In Section 5.1, we introduce the notion of the human cognitive bias and in particular the endowment effect to be used for further analysis. Furthermore, we verify the effect of the endowment effect in the incremental learning framework through the subjective experiment. In Section 5.2, we formulate new objective functions with a human cognitive bias in the online and stochastic learning settings. In Section 5.3, we develop new subgradient-based incremental algorithms based on OGD/SGD to solve the newly defined problems with a human cognitive bias. In Section 5.4, we show the theoretical properties of our developed algorithms from the perspectives of both online and stochastic learning frameworks. In Section 5.5, we conduct several online learning experiments to verify the practical performance of our developed algorithms by using massive data. In Section 5.6, we summarize the discussion of this chapter and discuss several future work to be solved.

5.1 Background: Human Cognitive Bias

As a background, we show the human cognitive bias in this section. Human cognitive biases and their effects toward the utility of human have been actively investigated in several research fields. Theories and subjective experiments about human cognitive biases largely affect the economics, social science, and many other fields including the human involvement. Among several human cognitive biases such as the anchoring effect [133] and the loss aversion effect [134], the endowment effect is a key component to analyze the evaluation measure of incremental learning algorithms. In the incremental learning setting, the endowment effect [135, 136] becomes an essential factor to newly formalize the incremental learning framework, algorithms, and objectives. We first introduce what the endowment effect is in the next subsection. Then, we show how the endowment effect affects objectives in the incremental learning setting. In the last, we conduct the experiment to verify the existence of the endowment effect in the incremental learning setting.

5.1.1 Endowment Effect

We introduce the endowment effect as one of the major human cognitive bias in this section. The endowment effect [135] induces in humans a cognitive bias to prevent rational decision-making. The endowment effect states that people tend to put a higher value on preventing the loss of an item they already possess than on buying the same item they do not possess. This human psychological bias has an important role for utility maximization and human engagements. There are many work on theoretical explanations and experimental tests of the endowment effect [136]. The endowment effect suggests that the cost of compensation is larger than the cost of paying.

From the nature of incremental update procedure, incremental learning algorithms sometimes make mistakes on the data that were correctly classified in the past. Here, the notion of the endowment effect suggests that people would pay more in order to sustain the correct prediction result for past data than to pay for a correct prediction on new data. When the predictive model misclassifies the data where they are correctly predicted by the past predictive model, the utility of humans using these incremental machine learning systems may be decreased even if the incremental learning algorithms pursue to minimize the predictive error. As a result, incremental learning algorithms must take the processed data in the past into consideration when updating the predictive model. To maximize the utility of humans, this notion should be incorporated into objective functions as an additional cost.

5.1.2 Experiment on the Endowment Effect

In this section, we show that the human cognitive bias induced by the endowment effect largely affects user utilities neglected in the context of the standard incremental learning setting. After that, we propose a new objective taking in this human cognitive bias.

Here, we describe an experiment to verify how the endowment effect has an impact on

Table. 5.1. Experimental result on the endowment effect. The table shows the number of people who evaluate each session's predictability.

	1	2	3	4	5	Average
type-I	2	3	15	73	7	3.80
type-II (duplicate)	2	11	26	54	7	3.53

the user utility of the machine learning-based services. This result demonstrates that the endowment effect is prominent in the user utilities. We conduct a subjective experiment using a crowdsourcing market place to assign tasks to humans. We set up a synthetic scene recognition task as a binary classification problem using the indoor recognition dataset[*1]. We use pictures of bookstores and restaurants from this large dataset.

We assign a certain amount of tasks to each worker. Each session consists of two phases, training phase and evaluation phase. In the training phase, workers receive eight pairs of pictures and their predicted labels. After seeing the given pairs, workers check whether each label is correct and then send their answers to the system as user feedback. In the evaluation phase, the system show eight different pairs of pictures and their prediction results to workers. Workers are told that the previous user feedbacks are used to classify samples in the evaluation phase. After checking the prediction results derived from the predictive models, workers evaluate the learnability of this system on a five-star scale.

Each worker deals with two types of sessions. In the type-I, the pictures displayed in the training phase do not appear in the evaluation phase. Therefore, the endowment effect is not activated in this session. In the type-II, two pictures are redisplayed in the evaluation phase. These pictures are correctly classified in the training phase but misclassified in the evaluation phase. The number of correctly classified pictures in both phases is fixed; there were four correctly classified pictures in both phases. Therefore, if worker evaluations are largely different between two sessions, we can see that the endowment effect influence workers' evaluations. In each session, 100 workers evaluate its learnability and verifies whether there is any difference of workers' cognition between these two types or not.

Table 5.1 shows an experimental result. The result in the table indicates that the type-II sessions have a lower evaluation in comparison with the type-I. The p-value calculated by Mann-Whitney test is less than a 1% level of significance ($p = 0.0093$). This result shows that the endowment effect largely affects workers' evaluations.

[*1] http://web.mit.edu/torralba/www/indoor.html

5.2 Online and Stochastic Learning with the Endowment Effect

We define this negative side-effect as a divestiture loss. The divestiture loss is actualized when the classifier makes an inaccurate prediction but it correctly classifies the same data in the past. To internalize this loss explicitly, we integrate this loss into the optimization problem. To simplify the following discussion, we denote a pair of the input vector and the corresponding label as $z = (\mathbf{x}, y)$. For each datum, algorithms predict the label of the datum as $\hat{y} = \mathrm{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$ by using the weight vector. When incremental learning algorithms already processed t data ($\{z_\tau\}_{\tau=1:t}$) and predicted labels for them ($\{\hat{y}_\tau\}_{\tau=1:t}$), the divestiture loss is defines as:

$$C(\mathbf{w}; z, \{\hat{y}_\tau\}_{\tau=1:t}, \{z_\tau\}_{\tau=1:t}) = \gamma 1_{\mathrm{prev}}(z, \{\hat{y}_\tau\}_{\tau=1:t}, \{z_\tau\}_{\tau=1:t}) \ell(\mathbf{w}; z) \ , \tag{5.1}$$

where

$$1_{\mathrm{prev}}(z, \{\hat{y}_\tau\}_{\tau=1:t}, \{z_\tau\}_{\tau=1:t}) = \min \left(1, \sum_{\tau=1}^{t} \mathbb{1}[\![z = z_\tau]\!] \mathbb{1}[\![y_\tau = \hat{y}_\tau]\!] \right) \ . \tag{5.2}$$

γ is a non-negative trade-off parameter between the original objectives and the divestiture loss. γ is chosen according to the stakeholder's preference. If $\gamma = 0$, the divestiture loss disappears and the objective function becomes the conventional one. If $\gamma \to \infty$, the divestiture loss becomes the dominant factor. When the problem is non-separable, that is, there is no hypothesis to correctly classify all received data, the value of the objective function is always ∞. The function 1_{prev} indicates whether the algorithm correctly classified z in the past. When z is correctly classified in the past, 1_{prev} outputs 1 and the algorithm incurs an additional loss $\gamma \ell(\mathbf{w}; z)$ from the definition of the divestiture loss. Otherwise, 1_{prev} becomes 0 and this loss will not be activated. We assume that if we correctly classify the same datum more than once, the divestiture loss does not change. New objective functions consist of the sum of the original losses and the divestiture loss.

First, we introduce the new regret with the human cognitive bias. The new regret is defined as follows:

$$\begin{aligned} &\mathrm{Regret}(T) = \sum_{t=1}^{T} F_t(\mathbf{w}_t) - \min_{\mathbf{u} \in \mathcal{W}} \sum_{t=1}^{T} F_t(\mathbf{u}) \ , \\ &\text{where} \ \ F_t(\mathbf{w}) = \ell_t(\mathbf{w}) + C(\mathbf{w}; z_t, \{\hat{y}_\tau\}_{\tau=1:t-1}, \{z_\tau\}_{\tau=1:t-1}) \ , \end{aligned} \tag{5.3}$$

where $\ell_t(\cdot) = \ell(\cdot; z_t)$. The difference between the new regret and the original regret (2.2) is the divestiture loss. When we set $\gamma = 0$, (5.3) becomes exactly the same as (2.2). After that, we define the new objective in the stochastic learning framework by inducing the notion of the human cognitive bias. The new objective is defined as follows:

$$\mathbb{E}_{\mathcal{D}^T}\left[\mathbb{E}_{z\sim\mathcal{D}}\left[\ell(\mathbf{w};z)+\frac{1}{T}\sum_{t=1}^{T}C(\mathbf{w};z,\{\hat{y}_\tau\}_{\tau=1:t-1},\{z_\tau\}_{\tau=1:t-1})\right]\right]. \tag{5.4}$$

The difference between the new objective function and the original objective in the stochastic learning setting (3.1) is in the part of the divestiture loss. The effect of the divestiture loss is averaged over the all rounds processed to optimize the weight vector. The first expectation is calculated over the sequence of the training examples, and this sequence affects a sequence of $\{z_\tau\}_{\tau=1:t-1}$ and $\{\hat{y}_\tau\}_{\tau=1:t-1}$. The optimal weight vector is defined as:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{u}\in\mathcal{W}}\mathbb{E}_{\mathcal{D}^T}\left[\mathbb{E}_{z\sim\mathcal{D}}\left[\ell(\mathbf{w};z)+\frac{1}{T}\sum_{t=1}^{T}C(\mathbf{w};z,\{\hat{y}_\tau\}_{\tau=1:t-1},\{z_\tau\}_{\tau=1:t-1})\right]\right]. \tag{5.5}$$

We focus on the minimization of these two objectives defined in the online learning setting and the stochastic learning setting.

5.3 Endowment-induced Algorithms

In the incremental learning setting with a human cognitive bias, the original OGD/SGD and other subgradient-based incremental algorithms do not achieve a good experimental result because of the existence of divestiture loss. Although the divestiture loss appears only when the corresponding examples were correctly classified, these algorithms treat all examples the same without referring to on-the-fly prediction results. These algorithms cannot capture this skewness.

We devise the Endowment-induced Online Gradient Descent (E-OGD) to incorporate the notion of the endowment effect into the original OGD/SGD to solve these difficulties. To simplify the following discussion, we use the name ”E-OGD” when we apply this algorithm in the stochastic learning setting. The key idea is to heavily weigh the importance of correctly classified examples in order to absorb the skewness.

E-OGD divides all examples into two categories:

1. correctly classified examples, i.e., $\hat{y}_t = y_t$
2. wrongly classified examples, i.e., $\hat{y}_t \neq y_t$

We see that the loss corresponding to the former examples is bigger than that of the latter examples due to the divestiture loss. Therefore, correct examples should be treated as

Algorithm 7 Endowment-induced Online Gradient Descent

```
Require: scaling constant $c$, trade-off parameter $\gamma$
  Initialize $\mathbf{w}_1 = \mathbf{0}$
  for $t = 1, \dots, T$ do
    Receive $\mathbf{x}_t$
    Predict corresponding output $\hat{y} = \mathrm{sgn}(\langle \mathbf{w}_\mathrm{t}, \mathbf{x}_\mathrm{t} \rangle)$
    Unveil true output $y_t$
    if $y_t = \hat{y}_t$ then
      $\mathbf{v}_{t+1} = \mathbf{w}_t - c(1+\gamma)\nabla\ell(\mathbf{w}_t; z_t)/\sqrt{t}$
    else
      $\mathbf{v}_{t+1} = \mathbf{w}_t - c\nabla\ell(\mathbf{w}_t; z_t)/\sqrt{t}$
    end if
    $\mathbf{w}_{t+1} = \operatorname*{argmin}_{\mathbf{w} \in \mathcal{W}} \|\mathbf{w} - \mathbf{v}_{t+1}\|_2$
  end for
```

being more important than wrong ones. E-OGD first classifies each example into one of two types it should belong to. After the type identification, E-OGD updates parameters heavily with the trade-off parameter γ with respect to correctly classified examples. In summary, weight vectors are updated as follows:

$$\mathbf{w}_{t+1} = \Pi_{\mathcal{W}}\left(\mathbf{w}_t - \eta_t \nabla\ell(\mathbf{w}_t; z_t)\right)$$
$$\text{where} \quad \eta_t = \begin{cases} c(1+\gamma)/\sqrt{t} & \text{if } \hat{y}_t = y_t \\ c/\sqrt{t} & \text{if } \hat{y}_t \neq y_t \end{cases}. \tag{5.6}$$

Algorithm 7 is the pseudo-code of E-OGD. We note that this E-OGD can update parameter by using only the currently received datum. We show that an appropriate step size setting makes the algorithm adaptive to the endowment effect in the following theoretical analysis.

5.3.1 Importance-aware Update for E-OGD

When E-OGD receives a correctly classified example, weight vectors are updated by $1+\gamma$ scaling. This update can be viewed as an approximate update of the original OGD at $1+\gamma$ times. An importance-aware update can be established in order to make an exact $1+\gamma$ times update through an one-time update [76] written in Section 2.4.3.

The original OGD and E-OGD do not hold some important properties such as the invariance and safety. In particular, when the endowment effect is strong (γ is large), the plain E-OGD would be in danger of overshooting when the prediction is correct. The safety property guarantees to prevent this type of overshooting. We derive the importance-aware version of E-OGD.

5.4 Theoretical Analyses

Let us analyze the theoretical aspects of E-OGD. For simplifying the following discussions, we introduce a new term:

$$r_t(z, \{\hat{y}_\tau\}_{\tau=1:t-1}, \{z_\tau\}_{\tau=1:t-1}) = 1 + \gamma \min\left(1, \sum_{\tau=1}^{t-1} \mathbb{1}[\![z = z_\tau]\!] \mathbb{1}[\![y_\tau = \hat{y}_\tau]\!]\right) . \tag{5.7}$$

Furthermore, to simplify the discussion, we denote $r_t(z_t, \{\hat{y}_\tau\}_{\tau=1:t-1}, \{z_\tau\}_{\tau=1:t-1})$ as r_t and $\ell(\cdot; z_t)$ as $\ell_t(\cdot)$. We analyze the upper regret bound and the upper bound of the expected loss of E-OGD in this section. Furthermore, we analyze another option of step sizes and its theoretical analysis in the Appendix A.

First, we show the relationship between a sequence of step sizes in E-OGD and the endowment effect. We set a sequence of step sizes as:

$$\eta_t = \begin{cases} c(1+\gamma)/\sqrt{t} & \text{if } \hat{y}_t = y_t \\ c/\sqrt{t} & \text{if } \hat{y}_t \neq y_t \end{cases} , \tag{5.8}$$

where c is some positive constant. Regret (5.3) is rewritten as:

$$\text{Regret}(T) = \sum_{t=1}^{T} r_t \ell_t(\mathbf{w}_t) - \min_{\mathbf{u} \in \mathcal{W}} \sum_{t=1}^{T} r_t \ell_t(\mathbf{u}) . \tag{5.9}$$

The next theorem gives the regret upper bound of E-OGD.

Theorem 15. *Let $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to E-OGD's update rule according to (5.8). Assume that for all $t \geq 1$, $\|\mathbf{w}_t\|_2 \leq R$ and $\|\nabla \ell_t(\mathbf{w}_t)\|_2 \leq G$. In addition, we assume that counts of consecutive classifications are less than $O(\sqrt{T})$. When we set $c = \sqrt{2}R/G(1+\gamma)$, the upper bound of regret is obtained as follows:*

$$\text{Regret}(T) \leq 2\sqrt{2}RG(1+\gamma)\sqrt{T} . \tag{5.10}$$

From this theorem, E-OGD is guaranteed to converge to obtain the optimal average loss with respect to the online learning setting with a human cognitive bias. To prove Theorem 15, we first introduce two lemmas.

Lemma 14. *When we set $\{\eta_t\}_{t=1:T}$ as following the rule of E-OGD, values of r_t and η_t become one of the following two types: (1) $r_t = 1, \eta_t = c/\sqrt{t}$, (2) $r_t = 1+\gamma, \eta_t = c(1+\gamma)/\sqrt{t}$, or (3) $r_t = 1+\gamma$, $\eta_t = c/\sqrt{t}$.*

Proof. This lemma is proved directly from the definition of r_t and η_t. When $y_t = \hat{y}_t$ is satisfied, $r_t = 1+\gamma$ and $\eta_t = c(1+\gamma)/\sqrt{t}$. When $y_t \neq \hat{y}_t$ and it has never been correctly classified in the past, $r_t = 1$ and $\eta_t = c/\sqrt{t}$. If the datum was correctly classified in the past, $r_t = 1+\gamma$. In summary, r_t and η_t becomes one of three pairs of values. □

Lemma 15. *For all $t \geq 3$, when $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) < 0$, there is $t > s \geq 2$ such that*

$$\frac{r_s}{\eta_s} = \frac{\sqrt{s}}{c} \ . \tag{5.11}$$

In addition, let us define s_t as the largest $s < t$ such that $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) < 0$ and $r_s/\eta_s = \sqrt{s}/c$ are satisfied. Then, the following inequalities are satisfied.

$$\sum_{\tau=s_t+1}^{t} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) \geq 0 \ , \tag{5.12}$$

$$\left(\frac{r_m}{\eta_m} - \frac{r_{m-1}}{\eta_{m-1}} \right) \geq 0 \ . \tag{5.13}$$

where m is arbitrary data ID such that $s_t < m < t$.

Proof. This result can be obtain directly from the results of Lemma 14. We see that $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) < 0$ is satisfied only when (3) occurs at round $t-1$ and (1) or (2) occurs at round t. (3) occurs only when the same datum is correctly classified in the past. Therefore, there is the round s such that the same datum is correctly classified and (1) happens at round s. When (1) occurs, $r_s/\eta_s = \sqrt{s}/c$. Furthermore, from the definition of s_t and m, (3) occurs for all m. Therefore, $r_m/\eta_m - r_{m-1}/\eta_{m-1} > 0$ is always satisfied regardless of the round $m-1$ result. In addition, the following inequality is satisfied because at round t and s_t, (1) or (2) occurs.

$$\sum_{\tau=s_t+1}^{t} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) = \frac{r_t}{\eta_t} - \frac{r_{s_t}}{\eta_{s_t}} \geq 0 \ . \tag{5.14}$$

□

Lemma 16. *Let us define s_t as the largest $s < t$ such that $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) < 0$ and $r_s/\eta_s = \sqrt{s}/c$ are satisfied. In addition, the conditions defined in Theorem 16 are* *satisfied. Then, for all $t \geq 3$ such that $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) < 0$, the following inequality is satisfied for all $\mathbf{u} \in \mathcal{W}$.*

$$\sum_{\tau=s_t+1}^{t} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) \|\mathbf{w}_\tau - \mathbf{u}\|_2^2 \leq \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) R^2 + o(\sqrt{T}) \ . \tag{5.15}$$

Proof. From the update formula (5.6), we see $\|\mathbf{w}_t - \mathbf{w}_{t-1}\| \leq c^2(1+\gamma)^2 \|\nabla \ell_t(\mathbf{w}_t)\|^2/t$ for any $t \geq 2$. Therefore, when $r_t/\eta_t - r_{t-1}/\eta_{s_{t-1}} \geq 0$ is satisfied,

$$\begin{aligned}\left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{s_{\tau-1}}} \right) \|\mathbf{w}_{\tau+1} - \mathbf{w}_\tau\|_2^2 &\leq c(1+\gamma)^2 \|\nabla \ell_\tau(\mathbf{w}_\tau)\|^2 \frac{(1+\gamma)\sqrt{t} - \sqrt{t-1}}{t} \\ &\leq c(1+\gamma)^2 G^2 \frac{(1+\gamma)\sqrt{t} - \sqrt{t-1}}{t} \ . \end{aligned} \tag{5.16}$$

From the property of norms, we see that $\|\mathbf{w}_t - \mathbf{u}\| \geq \|\mathbf{w}_{t-1} - \mathbf{u}\| + \|\mathbf{w}_t - \mathbf{w}_{t-1}\|$.

$$\begin{aligned} &\sum_{\tau=s_t+1}^{t} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) \|\mathbf{w}_\tau - \mathbf{u}\|^2 \\ &\leq \sum_{\tau=s_t+1}^{t} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) \|\mathbf{w}_t - \mathbf{u}\|^2 + \sum_{\tilde{\tau}=s_t+1}^{t-1} \sum_{\tau=\tilde{\tau}}^{t-1} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{s_{\tau-1}}} \right) \|\mathbf{w}_{\tau+1} - \mathbf{w}_\tau\|^2 \\ &\leq \sum_{\tau=s_t+1}^{t} \left(\frac{r_\tau}{\eta_\tau} - \frac{r_{\tau-1}}{\eta_{\tau-1}} \right) R^2 + \sum_{\tilde{\tau}=\tau}^{t-1} \sum_{\tau=s_t+1}^{t-1} c(1+\gamma)^2 G^2 \frac{(1+\gamma)\sqrt{t} - \sqrt{t-1}}{t} \ . \end{aligned} \tag{5.17}$$

The second term of the last formula becomes less than $O(\sqrt{T})$ from the assumption that the consecutive classification counts is less than $O(\sqrt{T})$. $\square$

From these lemmas, we can analyze the upper bound of regret of E-OGD in the online learning setting with a human cognitive bias. Below is the proof of Theorem 15.

Proof of Theorem 15. For simplicity, let us denote $\nabla \ell(\mathbf{w}_t; z_t)$ as $\mathbf{g}_t$. The convexity of loss functions guarantees that the first-order approximation inequality is satisfied for all $\mathbf{u}$, i.e.,

$$\ell_t(\mathbf{u}) \geq \langle \mathbf{g}_t, \mathbf{u} - \mathbf{w}_t \rangle + \ell_t(\mathbf{w}_t) \ . \tag{5.18}$$

This convexity property reformulates the regret bound.

$$\begin{aligned} \mathrm{Regret}(T) &= \sum_{t=1}^{T} r_t (\ell_t(\mathbf{w}_t) - \min_{\mathbf{u} \in \mathcal{W}} \ell_t(\mathbf{u})) \\ &\leq \max_{\mathbf{u} \in \mathcal{W}} \sum_{t=1}^{T} r_t (\ell_t(\mathbf{w}_t) + \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u} \rangle - \ell_t(\mathbf{w}_t)) \\ &= \max_{\mathbf{u} \in \mathcal{W}} \sum_{t=1}^{T} r_t \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u} \rangle \ . \end{aligned} \tag{5.19}$$

We define for all t, $\mathbf{v}_{t+1} = \mathbf{w_t} - \eta_t \nabla \ell_t(\mathbf{w}_t)$. From this definition, we obtain the following equations.

$$\begin{aligned}
\mathbf{v}_{t+1} - \mathbf{u} &= (\mathbf{w}_t - \mathbf{u}) - \eta_t \mathbf{g}_t \\
\|\mathbf{v}_{t+1} - \mathbf{u}\|_2^2 &= \|\mathbf{w}_t - \mathbf{u}\|_2^2 - 2\eta_t \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u}\rangle + \eta_t^2 \|\mathbf{g}_t\|_2^2 \\
\|\mathbf{w}_{t+1} - \mathbf{u}\|_2^2 &\le \|\mathbf{w}_t - \mathbf{u}\|_2^2 - 2\eta_t \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u}\rangle + \eta_t^2 \|\mathbf{g}_t\|_2^2 \\
\langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u}\rangle &\le \frac{1}{2\eta_t}(\|\mathbf{w}_t - \mathbf{u}\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|_2^2) + \frac{\eta_t}{2}\|\mathbf{g}_t\|_2^2
\end{aligned}$$

Through these inequalities and some assumptions on the norms of weight vectors and gradients, the following reformulation can be applied for any $\mathbf{u}$,

$$\begin{aligned}
\sum_{t=1}^{T} r_t \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u}\rangle &\le \sum_{t=1}^{T} \frac{r_t}{2\eta_t}(\|\mathbf{w}_t - \mathbf{u}\|_2^2 - \|\mathbf{w}_{t+1} - \mathbf{u}\|_2^2) + \frac{\eta_t r_t}{2}\|\mathbf{g}_t\|_2^2 \\
&\le \frac{r_1}{2\eta_1}\|\mathbf{w}_1 - \mathbf{u}\|_2^2 - \frac{r_T}{2\eta_T}\|\mathbf{w}_{T+1} - \mathbf{u}\|_2^2 \\
&\quad + \frac{1}{2}\sum_{t=2}^{T}\left(\frac{r_t}{\eta_t} - \frac{r_{t-1}}{\eta_{t-1}}\right)\|\mathbf{w}_t - \mathbf{u}\|_2^2 + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t r_t \qquad (5.20)
\end{aligned}$$

From Lemma 16, we can reformulate as follows:

$$\begin{aligned}
\sum_{t=1}^{T} r_t \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u}\rangle &\le 2R^2\left(\frac{r_1}{\eta_1} + \sum_{t=2}^{T}\left(\frac{r_t}{\eta_t} - \frac{r_{t-1}}{\eta_{t-1}}\right)\right) + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t r_t \\
&= \frac{2R^2 r_T}{\eta_T} + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t r_t\ . \qquad (5.21)
\end{aligned}$$

The assumption $\|\mathbf{g}\|_2 \le G$ is used in the second inequality and $\|\mathbf{w}\|_2 \le R$ is used in the third inequality.

The following inequality is derived by setting $r_t = 1 + \gamma$ and $\eta_t = c(1+\gamma)/\sqrt{t}$ for all t in the second term.

$$\begin{aligned}
\mathrm{Regret}(T) &\le \left(\frac{2R^2\sqrt{T}}{c} + \frac{G^2(1+\gamma)^2}{2}\sum_{t=1}^{T}\frac{c}{\sqrt{t}}\right) \\
&\le \left(\frac{2R^2}{c} + cG^2(1+\gamma)^2\right)\sqrt{T}\ . \qquad (5.22)
\end{aligned}$$

by using $\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \le 2\sqrt{T}$. When $c = \sqrt{2}R/G(1+\gamma)$, we have

$$\mathrm{Regret}(T) \le 2\sqrt{2}RG(1+\gamma)\sqrt{T}\ .$$

□

For stochastic learning setting, we assume that the data are i.i.d. sampled from a distribution $\mathcal{D}$. The final goal is to minimize the sum of the expected loss and divestiture loss, as described by formula (5.4). Lemma 17 reformulates the optimization problem into an easily analyzable form.

Lemma 17. *The optimization problem in the stochastic learning setting can be reformulated through $r_t(z)$.*

$$\mathbb{E}_{\mathcal{D}^T}\left[\mathbb{E}_{z\sim\mathcal{D}}\left[\frac{1}{T}\sum_{t=1}^{T} r_t(z)\ell(\mathbf{w};z)\right]\right] . \tag{5.23}$$

Furthermore, it can be reformulated through a new distribution $\mathcal{D}_P$ and an appropriate constant value $H_{\mathcal{D}^T}$ conditioned on $\{z_t\}_{t=1:T}$ as

$$\mathbb{E}_{\mathcal{D}^T}\left[H_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}_P}\left[\ell(\mathbf{w};z)\right]\right] . \tag{5.24}$$

We prove Lemma 17.

Proof. When E-OGD runs in the stochastic setting, algorithms implicitly construct a set of correctly classified examples P. These examples are sampled from $\mathcal{D}$ and chosen with an on-the-fly prediction. For simplifying the discussion, we define

$$\bar{r}(z) = \frac{1}{T}\sum_{t=1}^{T} r_t(z) . \tag{5.25}$$

All examples in P are distinct for the sake of simplicity. It is easy to extend the following proposition to a general case.

Let us define $p(\mathbf{x}, y)$ as a probability density function according to a distribution $\mathcal{D}$. The set P is constructed from examples sampled from $\mathcal{D}$. For any datum in P, the datum's occurrence probability in D becomes necessarily greater than 0. The optimization formula can be reformulated according to basic probabilistic properties:

$$\begin{aligned}
&\mathbb{E}_{z\sim\mathcal{D}}\left[\ell(\mathbf{w};z) + \bar{C}(\mathbf{w};z)\right] \\
&= \mathbb{E}_{z\sim\mathcal{D}}\left[\ell(\mathbf{w};z) + \frac{\gamma}{T}\sum_{t=1}^{T}\min\left(1, \sum_{s=1}^{t-1}\mathbb{1}[\![z = z_s]\!]\mathbb{1}[\![y_s = \hat{y}_s]\!]\right)\ell(\mathbf{w};z)\right] \\
&= \mathbb{E}_{z\sim\mathcal{D}}\left[\bar{r}(z)\ell(\mathbf{w};z)\right] .
\end{aligned} \tag{5.26}$$

Furthermore, we can construct a new distribution $\mathcal{D}_P$ as follows: The probability function q of $\mathcal{D}_P$ is defined as

$$q(z) = \frac{\bar{r}(z)p(z)}{\int \bar{r}(z)p(z)dz} . \tag{5.27}$$

We denote the denominator of formula (5.27) as $H_{\mathcal{D}^T}$. In this case, the optimization problem can be reformulated according to basic probabilistic properties:

$$\mathbb{E}_{z\sim\mathcal{D}}\left[\bar{r}(z)\ell(\mathbf{w};z)\right] = H_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}_P}\left[\ell(\mathbf{w};z)\right] \ . \tag{5.28}$$

□

The following theorem is derived from Theorem 15 and Lemma 17 in order to upper bound the expected loss with a human cognitive bias (5.4).

Theorem 16. *Assume that the conditions in Theorem 15 are satisfied and there is an integer t_p such that $r_t(z) = r_{t_p}(z)$ for any $t \geq t_p$. In this setting, the following formula is satisfied for any $\mathbf{u} \in \mathcal{W}$.*

$$\begin{aligned} &\mathbb{E}_{\mathcal{D}^T}\left[\mathbb{E}_{z\sim\mathcal{D}_P}\left[\ell(\bar{\mathbf{w}};z)\right]\right] - \mathbb{E}_{\mathcal{D}^T}\left[\mathbb{E}_{z\sim\mathcal{D}_P}\left[\ell(\mathbf{u};z)\right]\right] \\ &\leq \frac{\sqrt{2}RG(1+\gamma)}{(\sqrt{T}-(t_p+1)/\sqrt{T})/(2-\sqrt{t_p-1}/\sqrt{T})} \ , \end{aligned} \tag{5.29}$$

where $\bar{\mathbf{w}} = \sum_{t=t_p}^{T}\mathbf{w}_t/(T-t_p+1)$.

Lemma 17 derives that the left-hand side of the formula (5.29) equals the original objective function (5.4). From this theoretical result, the average weight vector converges to the optimal one that minimizes the sum of the expected loss and the divestiture loss. If $t_p \ll T$, the convergence speed is $O(1/\sqrt{T})$. And, when the number of data is finite, there is some constant t_p such that $r_t(z) = r_{t_p}(z)$ for any $t \geq t_p$.

We prove Theorem 16 below.

Proof. First, we analyze the regret bound from round t_p to T.

$$\begin{aligned} \mathrm{Regret}(t_p:T) &\leq \left(\frac{2R^2\sqrt{T}}{c} + \frac{G^2(1+\gamma)^2}{2}\sum_{t=t_p}^{T}\frac{c}{\sqrt{t}}\right) \\ &\leq \left(\frac{2R^2}{c} + cG^2(1+\gamma)^2\right)\sqrt{T} - cG^2(1+\gamma)^2\sqrt{t_p-1} \ . \end{aligned} \tag{5.30}$$

When we set $c = \sqrt{2}R/G(1+\gamma)$,

$$\mathrm{Regret}(t_p:T) \leq \sqrt{2}RG(1+\gamma)\left(2\sqrt{T} - \sqrt{t_p-1}\right) \ . \tag{5.31}$$

For simplicity, we will describe $H_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}_P}[\ell(\cdot;z)]$ as $\mathcal{D}_P(\cdot)$. Let us take the expectation of the regret in the online learning framework with a human cognitive bias. Now let us analyze the first term and the second term of the expectation of the regret. Note that this reformulation uses Lemma 4.3. The expectation of the second term is reformulated as follows:

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}^T}\left[\sum_{t=t_p}^{T} r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{u})\right] &= \sum_{t=t_p}^{T} \mathbb{E}_{\mathcal{D}^T}\left[r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{u})\right] \\
&= \sum_{t=t_p}^{T} \mathbb{E}_{\mathcal{D}^{t-1}}\mathbb{E}_{\mathcal{D}^t}\left[r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{u})|\mathcal{D}^{t-1}\right] \\
&= \sum_{t=t_p}^{T} \mathbb{E}_{\mathcal{D}^{t-1}}\mathbb{E}_{z\sim\mathcal{D}}\left[r_t(z)\ell(\mathbf{u}; z)\right] \\
&= \mathbb{E}_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}}\left[\sum_{t=t_p}^{T} r_t(z)\ell(\mathbf{u}; z)\right] \\
&= (T - t_p + 1)\mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\mathbf{u})\right] \ .
\end{aligned}
\tag{5.32}
$$

The expectation of the first term can be reformulated with a similar procedure.

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}^T}\left[\sum_{t=t_p}^{T} r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{w}_t)\right] &= \sum_{t=t_p}^{T} \mathbb{E}_{\mathcal{D}^T}\left[r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{w}_t))\right] \\
&= \sum_{t=t_p}^{T} \mathbb{E}_{\mathcal{D}^{t-1}}\mathbb{E}_{z\sim\mathcal{D}}\left[r_t(z)\ell(\mathbf{w}_t; z)\right] \\
&= \mathbb{E}_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}}\left[\sum_{t=t_p}^{T} r_t(z)\ell(\mathbf{w}_t; z)\right] \\
&= \mathbb{E}_{\mathcal{D}^T}\left[H_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}_P}\left[\sum_{t=t_p}^{T} \ell(\mathbf{w}_t; z)\right]\right] \\
&\geq (T - t_p + 1)\mathbb{E}_{\mathcal{D}^T}\left[H_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}_P}\left[\ell(\bar{\mathbf{w}}_t; z)\right]\right] \\
&= (T - t_p + 1)\mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\bar{\mathbf{w}})\right] \ .
\end{aligned}
\tag{5.33}
$$

The inequality is satisfied from the convexity of loss functions. The following formula is obtained by combining these two formula with Lemma 17.

$$
\begin{aligned}
\mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\bar{\mathbf{w}})\right] - \mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\mathbf{u})\right] &\leq \frac{1}{T - t_p + 1}\mathbb{E}_{\mathcal{D}^T}\left[\mathrm{Regret}(t_p : T)\right] \\
&\leq \frac{\sqrt{2}RG(1+\gamma)}{(\sqrt{T} - (t_p + 1)/\sqrt{T})/(2 - \sqrt{t_p - 1}/\sqrt{T})} \ .
\end{aligned}
$$

□

Table. 5.2. Dataset Specifications. T is the number of training data. S is test data size. D is the number of features.

	T	S	D
news20	15,000	4,996	1,335,191
rcv1	20,242	677,399	47,236
algebra	8,407,752	510,302	20,216,830
BtA	19,264,097	748,401	29,890,095
webspam-t	315,000	35,000	16,609,143

5.5 Experiments

We conduct experiments to test the performance of the conventional OGD and E-OGD as stochastic learning framework with a human cognitive bias. We use five large-scale datasets from the LIBSVM binary data collections[*2]. The specifications of these dataset are listed in Table 5.2. news20 and rcv1 are news category classification tasks. algebra and BtA (Bridge to Algebra) are KDD Cup 2010 datasets to predict whether students correctly answer algebra problems. webspam-t is a tri-gram webspam classification dataset used in the Pascal Large Scale Learning Challenge. The original webspam-t dataset is not splited to two sets, therefore, we randomly sample 90% data from the dataset and used them as a training set and remaining data as a test set.

We use these datasets to compare the performances of OGD and E-OGD in a new stochastic learning setting. We incur both expected loss and divestiture loss. To evaluate the divestiture loss, we replace some examples in the test data with some training examples at a specific rate. The training examples are randomly extracted from the training set. If the algorithm correctly classified in the training phase but it misclassifies the same example in the test phase, they incur a divestiture loss. We conduct experiments by setting the replacement rate of the test examples by training examples as 5, 10, and 30%. We quantified the performance as

$$\frac{1}{S}\sum_{s=1}^{S}\ell(\mathbf{w};z_s)+\frac{\gamma}{S}\sum_{z_p\in P}\ell(\mathbf{w};z_p)\ . \tag{5.34}$$

The first term corresponds to the expected loss, and each datum z_s corresponds to one datum in the test set or a replaced training example. S is the number of test data. The

[*2] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

Table. 5.3. Experimental results compared to the conventional OGD: the expected loss, divestiture loss, and cumulative loss (Iteration: 1). The lowest values in each replace rate r, loss type, and dataset are written in **bold**.

		$r = 0.05$		$r = 0.1$		$r = 0.3$	
	Loss Type	E-OGD	OGD	E-OGD	OGD	E-OGD	OGD
news20	Expected	$\mathbf{5.10 \times 10^{-2}}$	5.31×10^{-2}	$\mathbf{5.67 \times 10^{-2}}$	5.84×10^{-2}	$\mathbf{9.18 \times 10^{-2}}$	9.32×10^{-2}
	Divestiture	$\mathbf{8.71 \times 10^{-3}}$	1.39×10^{-2}	$\mathbf{7.73 \times 10^{-3}}$	1.27×10^{-2}	$\mathbf{6.71 \times 10^{-3}}$	1.07×10^{-2}
	Cumulative	$\mathbf{5.97 \times 10^{-2}}$	6.70×10^{-2}	$\mathbf{6.44 \times 10^{-2}}$	7.11×10^{-2}	$\mathbf{9.85 \times 10^{-2}}$	1.04×10^{-1}
rcv1	Expected	7.39×10^{-2}	$\mathbf{7.37 \times 10^{-2}}$	7.83×10^{-2}	$\mathbf{7.80 \times 10^{-2}}$	9.71×10^{-2}	$\mathbf{9.65 \times 10^{-2}}$
	Divestiture	$\mathbf{1.10 \times 10^{-2}}$	1.84×10^{-2}	$\mathbf{1.04 \times 10^{-2}}$	1.74×10^{-2}	$\mathbf{8.04 \times 10^{-3}}$	1.35×10^{-2}
	Cumulative	$\mathbf{8.49 \times 10^{-2}}$	9.21×10^{-2}	$\mathbf{8.87 \times 10^{-2}}$	9.54×10^{-2}	$\mathbf{1.05 \times 10^{-1}}$	1.10×10^{-1}
algebra	Expected	3.31×10^{-1}	$\mathbf{3.06 \times 10^{-1}}$	3.30×10^{-1}	$\mathbf{3.06 \times 10^{-1}}$	3.26×10^{-1}	$\mathbf{3.03 \times 10^{-1}}$
	Divestiture	$\mathbf{6.00 \times 10^{-2}}$	1.01×10^{-1}	$\mathbf{5.68 \times 10^{-2}}$	9.60×10^{-2}	$\mathbf{4.41 \times 10^{-2}}$	7.46×10^{-2}
	Cumulative	$\mathbf{3.91 \times 10^{-1}}$	4.08×10^{-1}	$\mathbf{3.87 \times 10^{-1}}$	4.02×10^{-1}	$\mathbf{3.70 \times 10^{-1}}$	3.78×10^{-1}
BtA	Expected	3.29×10^{-1}	$\mathbf{3.11 \times 10^{-1}}$	3.27×10^{-1}	$\mathbf{3.10 \times 10^{-1}}$	3.18×10^{-1}	$\mathbf{3.03 \times 10^{-1}}$
	Divestiture	$\mathbf{7.64 \times 10^{-2}}$	1.17×10^{-1}	$\mathbf{7.24 \times 10^{-2}}$	1.11×10^{-1}	$\mathbf{5.63 \times 10^{-2}}$	8.62×10^{-2}
	Cumulative	$\mathbf{4.05 \times 10^{-1}}$	4.28×10^{-1}	$\mathbf{3.99 \times 10^{-1}}$	4.21×10^{-1}	$\mathbf{3.74 \times 10^{-1}}$	3.90×10^{-1}
webspam-t	Expected	$\mathbf{3.45 \times 10^{-2}}$	3.51×10^{-2}	$\mathbf{3.49 \times 10^{-2}}$	3.53×10^{-2}	$\mathbf{3.75 \times 10^{-2}}$	3.76×10^{-2}
	Divestiture	$\mathbf{8.11 \times 10^{-3}}$	1.09×10^{-2}	$\mathbf{7.72 \times 10^{-3}}$	1.04×10^{-2}	$\mathbf{5.63 \times 10^{-3}}$	7.73×10^{-3}
	Cumulative	$\mathbf{4.29 \times 10^{-2}}$	4.60×10^{-2}	$\mathbf{4.26 \times 10^{-2}}$	4.57×10^{-2}	$\mathbf{4.32 \times 10^{-2}}$	4.54×10^{-2}

second term corresponds to the divestiture loss, and each datum z_p corresponds to the example regarding the divestiture loss. P is an example set that satisfies two conditions: (1) the example was extracted from the training dataset in exchange for test examples; (2) the example was correctly classified when the example appeared in the training phase. The cumulative loss is defined as the sum of these two losses.

Let the weight vector spaces $\mathcal{W}$ be a D-dimensional Euclidean space where D is the number of features. We use the logistic loss as a loss function. Each algorithm learns the weight vector from training set through 1 iteration. Learning rates are $\eta_t = \eta/\sqrt{t}$. We vary η from 10^3 to 1.91×10^{-3} with common ratio 1/2 to minimize cumulative loss.

5.5.1 Experimental Results

Table 5.3 shows the experimental results when we apply OGD and E-OGD to five datasets. These results indicate E-OGD has a crucial advantage to make divestiture losses lower in all settings, and this effect contributes to low cumulative losses. As a result, E-OGD outperforms OGD on all datasets. Figure 5.1 plots loss values in each 10,000 rounds when we used BtA dataset to evaluate the performance. These results show that E-OGD has obtained significantly lower divestiture losses than OGD during most rounds. Low

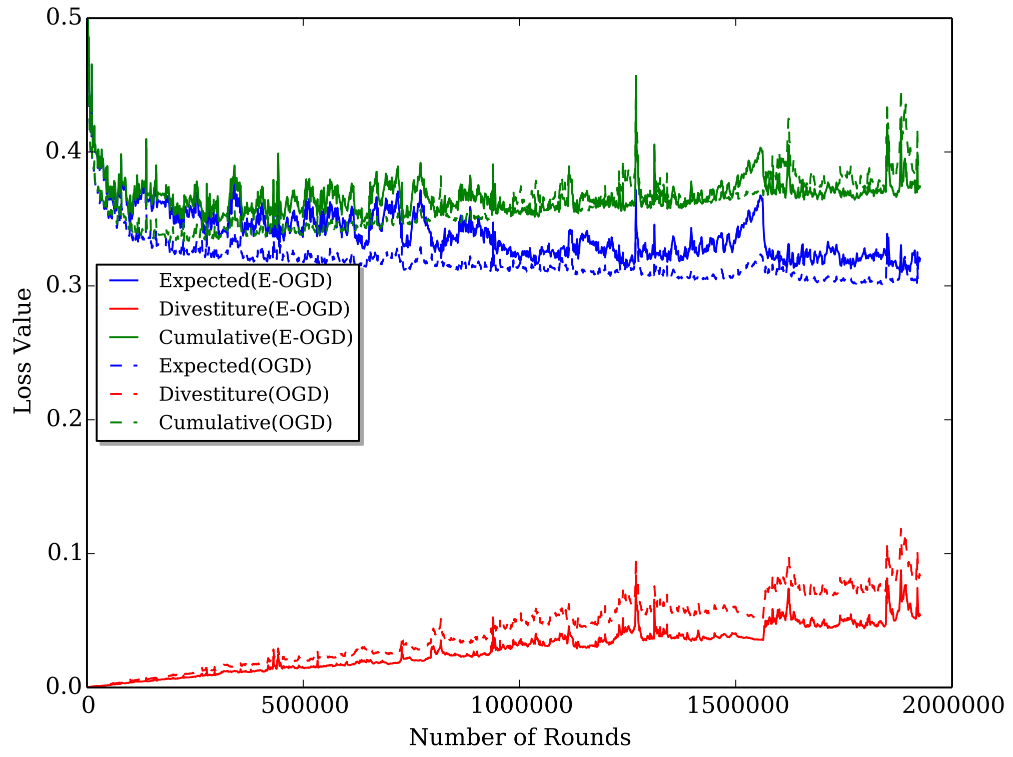

Fig. 5.1. Experimental results on BtA dataset in each 1,000 rounds: the expected loss, divestiture loss, and cumulative loss. The x-axis is the number of rounds. The y-axis describes the value of each loss. The solid curves are the results obtained by E-OGD. The dotted curves are the results by OGD.

divestiture loss leads to low cumulative loss, and E-OGD has constantly outperformed OGD with respect to cumulative loss. The difference of expected losses between two algorithms becomes smaller while the number of received data increases. On the other hand, the difference of divestiture losses between two algorithms becomes bigger. This result means that E-OGD becomes superior to the normal OGD with respect to the cumulative loss while the data increases.

Table 5.4 shows the results of importance-aware update versions. These results indicate that the importance-aware update improves the performance of E-OGD in most experimental settings. Moreover, E-OGD largely outperforms OGD in terms of cumulative losses.

In addition to the normal setting, we perform several experiments. We show a brief result here. First, we verify that E-OGD outperformed OGD in most datasets when we set the hinge loss as a loss function. After that, we make the value of γ bigger and

Table. 5.4. Experimental results among importance-aware update family: the expected loss, divestiture loss, and cumulative loss (Iteration: 1). The lowest value in each replace rate r, loss type, and dataset are written in **bold**.

		$r = 0.05$		$r = 0.1$		$r = 0.3$	
	Loss Type	E-OGD	OGD	E-OGD	OGD	E-OGD	OGD
news20	Expected	$\mathbf{3.11 \times 10^{-2}}$	3.85×10^{-2}	$\mathbf{3.40 \times 10^{-2}}$	4.14×10^{-2}	$\mathbf{5.20 \times 10^{-2}}$	5.86×10^{-2}
	Divestiture	$\mathbf{1.37 \times 10^{-2}}$	2.20×10^{-2}	$\mathbf{1.31 \times 10^{-2}}$	2.08×10^{-2}	$\mathbf{9.84 \times 10^{-3}}$	1.59×10^{-2}
	Cumulative	$\mathbf{4.48 \times 10^{-2}}$	6.05×10^{-2}	$\mathbf{4.70 \times 10^{-2}}$	6.22×10^{-2}	$\mathbf{6.18 \times 10^{-2}}$	7.46×10^{-2}
rcv1	Expected	$\mathbf{3.53 \times 10^{-2}}$	3.80×10^{-2}	$\mathbf{3.95 \times 10^{-2}}$	4.18×10^{-2}	$\mathbf{5.69 \times 10^{-2}}$	5.79×10^{-2}
	Divestiture	$\mathbf{1.25 \times 10^{-2}}$	1.78×10^{-2}	$\mathbf{1.18 \times 10^{-2}}$	1.68×10^{-2}	$\mathbf{9.10 \times 10^{-3}}$	1.30×10^{-2}
	Cumulative	$\mathbf{4.79 \times 10^{-2}}$	5.58×10^{-2}	$\mathbf{5.13 \times 10^{-2}}$	5.87×10^{-2}	$\mathbf{6.60 \times 10^{-2}}$	7.09×10^{-2}
algebra	Expected	3.35×10^{-1}	$\mathbf{3.13 \times 10^{-1}}$	3.34×10^{-1}	$\mathbf{3.13 \times 10^{-1}}$	3.30×10^{-1}	$\mathbf{3.09 \times 10^{-1}}$
	Divestiture	$\mathbf{5.89 \times 10^{-2}}$	1.02×10^{-1}	$\mathbf{5.58 \times 10^{-2}}$	9.68×10^{-2}	$\mathbf{4.33 \times 10^{-2}}$	7.52×10^{-2}
	Cumulative	$\mathbf{3.94 \times 10^{-1}}$	4.16×10^{-1}	$\mathbf{3.90 \times 10^{-1}}$	4.09×10^{-1}	$\mathbf{3.73 \times 10^{-1}}$	3.85×10^{-1}
BtA	Expected	3.29×10^{-1}	$\mathbf{3.11 \times 10^{-1}}$	3.27×10^{-1}	$\mathbf{3.10 \times 10^{-1}}$	3.18×10^{-1}	$\mathbf{3.03 \times 10^{-1}}$
	Divestiture	$\mathbf{7.64 \times 10^{-2}}$	1.17×10^{-1}	$\mathbf{7.24 \times 10^{-2}}$	1.11×10^{-1}	$\mathbf{5.62 \times 10^{-2}}$	8.61×10^{-2}
	Cumulative	$\mathbf{4.05 \times 10^{-1}}$	4.28×10^{-1}	$\mathbf{3.99 \times 10^{-1}}$	4.21×10^{-1}	$\mathbf{3.74 \times 10^{-1}}$	3.89×10^{-1}
webspam-t	Expected	$\mathbf{2.62 \times 10^{-2}}$	2.75×10^{-2}	$\mathbf{2.61 \times 10^{-2}}$	2.74×10^{-2}	$\mathbf{2.79 \times 10^{-2}}$	2.90×10^{-2}
	Divestiture	$\mathbf{8.29 \times 10^{-3}}$	1.16×10^{-2}	$\mathbf{7.94 \times 10^{-3}}$	1.11×10^{-2}	$\mathbf{5.87 \times 10^{-3}}$	8.30×10^{-3}
	Cumulative	$\mathbf{3.45 \times 10^{-2}}$	3.91×10^{-2}	$\mathbf{3.40 \times 10^{-2}}$	3.85×10^{-2}	$\mathbf{3.38 \times 10^{-2}}$	3.73×10^{-2}

verify that E-OGD has maintained an advantage over OGD. These results indicate that the advantage of E-OGD becomes more crucial as the importance of the divestiture loss becomes larger.

5.6 Chapter Summary

We establish an online and stochastic learning framework with a human cognitive bias by incorporating the notion of the endowment effect. We establish an online and stochastic learning framework with a human cognitive bias by incorporating the notion of the endowment effect. In this framework, algorithms need to focus on minimizing not only the original loss but also the divestiture loss. We developed new algorithms applicable to this framework; Endowment-induced Online Gradient Descent (E-OGD). We theoretically show that E-OGD has strong guarantees to have some desirable properties for both online and stochastic learning frameworks with a human cognitive bias. Finally, we experimentally show that our derived algorithms are effective at a large number of tasks involving human engagements in this framework.

The incremental learning framework with a human cognitive bias has several open issues. The first challenge is a more sophisticated choice of η_t. To obtain the lowest convergence rate, it is the best to set η_t as proportional to $r_t(z_t)$. However, exact matching is almost impossible because the parameter $r_t(z_t)$ is conditioned on a sequence of previous predictions. In the standard incremental learning setting, algorithms cannot preserve the history of observations and its prediction results. Online density estimation is one possible way to bridge this gap for precisely reproducing the data distribution $\mathcal{D}$ and dynamically refining step size η_t.

Another difficulty is how to deal with the similarity of data. When we use the machine learning algorithm, sometimes we encounter situations in which several data are similar to but not exactly the same as previously seen data. We will verify whether the endowment effect is activated even when data are quite similar to each other. Furthermore, we will incorporate this effect into the optimization problem as a modified version of this framework.

The third issue is an extension to other frameworks of large-scale learning, such as the mini-batch and distribution frameworks. To incorporate the endowment effect into these settings, the divestiture loss should be internalized into the optimization problems and the appropriate definition of the divestiture loss should be analyzed. We will integrate a human cognitive bias into other frameworks to achieve a user-friendly machine learning architecture.

The last issue is the integration of the regularization term, especially for sparsity-inducing regularization methods. The above analyses are focused on the condition where the regularization terms does not exist. It is not easy how to integrate these regularization terms into the online and stochastic learning framework with a human cognitive bias.

Chapter 6

Conclusion

This thesis focuses on the analysis and design of subgradient-based online and stochastic learning frameworks applicable to real-world problems. In applying these learning algorithms to real tasks, the characteristics of the data and the humans that evaluate prediction results affect the performance of the algorithms, and we have seen that several biases occur. Here, we focus on the truncation bias and cognitive bias in an attempt to improve the applicability of learning algorithms.

In Chapter 1, we described how machine learning is being used for massive data analysis. In particular, we introduced the supervised learning framework and described the basics of linear learners, loss functions, regularization methods, and the subgradient-based learning framework. Furthermore, we described some of the basic mathematical tools, such as sparse vectors, convex analysis, Bregman divergence, and duality. These tools have been frequently used in analyses of online and stochastic learning algorithms.

In Chapter 2, we introduced the online learning framework, in particular, the subgradient-based online learning framework. Online learning processes a bunch of data on datum at a time. This sequential process can be applied to numerous kinds of data, such as streaming data and adversarial data. We summarized the subgradient-based online learning algorithms, including the parameter update procedure, the performance measures (i.e., regret), and several state-of-the-art subgradient-based online learning algorithms. In addition, we showed several extensions and mistake-bound-based online learning algorithms as alternatives to regret-based algorithms. In Chapter 3, we introduced the stochastic learning framework. Stochastic learning is based on stochastic optimization, and it works by iteratively updating a hypothesis function through an unbiased estimator of the subgradient of the objective function. Similar to Chapter 2, we described the parameter update procedure of the subgradient-based stochastic learning algorithms and their performance measure (i.e., expected loss). Next, we described several approaches to giving an upper bound to the objective function: the generalization bound and online-to-batch conversion. After that, we described several state-of-the-art stochastic learning algorithms. In particular, we described Pegasos, a novel algorithm which utilizes online-to-batch conversion. Moreover, as a novel way of efficiently utilizing the generalization bound, we showed a family of stochastic gradient descents with a linear convergence property.

These frameworks and algorithms are very efficient; however, we find that several unique biases occur when subgradient-based incremental learning algorithms are used in practical applications. These biases critically affect the performance of incremental learning algorithms. We analyzed the characteristics of these biases and showed that they make the conventional objective definitions in online and stochastic learning frameworks inappropriate as performance measures.

In Chapter 4, we dealt with the truncation bias. The truncation bias appears in the sparse online learning framework. Sparse online learning has recently gained a lot of attention because it enables us to obtain a compact predictive model with an online update scheme, which saves memory space and speeds up computations. First, we described sparsity-inducing regularization methods and sparse online learning as a combination of sparsity-inducing regularization and the online learning framework. Then, we introduced the truncation bias by showing how the heterogeneity of each feature characteristic affects the algorithm's predictive performance through toy examples. Through the effect of this bias, many features tend to be removed regardless of their importance to the prediction. To solve this problem, we mathematically formulated the truncation bias, integrated it into the objective functions, and developed new algorithms to solve this problem in an online manner. We proposed feature-aware L_1-regularization methods to dynamically integrate information on all previous subgradients for alleviating the effect of the truncation bias without pre-processing. We theoretically proved that our algorithms maintain the advantages of computational efficiency and a convergence guarantee and empirically demonstrated their superiority over the state-of-the-art algorithms in terms of prediction performance and sparsity of the predictive models.

In Chapter 5, we solved the human cognitive bias problem. Incremental learning algorithms iteratively update predictive models, therefore, they sometimes make incorrect predictions on the data that were correctly classified in the past. We find that such inconsistent predictions have a big effect on the humans who conduct the evaluations, and that the effect originates from cognitive biases, especially the endowment effect theory. For example, when developing services based on the incremental learning algorithms, maximization of the user utility becomes the objective. In these cases, cognitive bias is a very important consideration. We experimentally showed how such inconsistent prediction behavior affects the user utility through subjective experiments. The results verified that this effect on the user utility is non-negligible. After that, we formalized the divestiture loss, which is a mathematical model to represent this cognitive bias and its effect on user utility. We then developed new objective functions and new algorithms by integrating this new loss as an objective measure we would like to minimize. We showed theoretical and empirical results verifying the effectiveness of our algorithms.

Acknowledgements

My heartfelt appreciation goes to my supervisor Prof. Hiroshi Nakagawa. He provided helpful comments and suggestions for my thesis and other several previous papers when I was on my master's and doctor's course. My deepest appreciation goes to Dr. Shin Matsushima and Dr. Issei Sato. They are co-authors of the achievements written in this thesis. Without insightful discussions with them or their enormous supports, I would not have been able to write these papers. The achievements written in this thesis had been done without them. I would like to thank Prof. Kenji Yamanishi, Hisashi Kashima, Akiko Takeda, and Taiji Suzuki for providing useful comments on the earlier version of this thesis.

I would like to thank all members of Nakagawa Laboratory. They have given the useful suggestion in the laboratory meeting and daily conversation. In addition, I am deeply grateful to Miki Yoshitomi. She has been a secretary of my laboratory and provide a suitable environment for my research. Also, I would like to thank the Japan Society for the Promotion of Science for a grant. Without this support, I could not complete my Ph.D. degree. Finally, I would like to express my gratitude to my family for their several types of supports and warm encouragements.

Bibliography

[1] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. MIT Press, 2011.

[2] Trevor J. Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer-Verlag, 2nd edition, 2009.

[3] Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[4] Michael C. Schat and Ben Langmead. The dna data deluge. *IEEE Spectrum*, 50(7):28–33, 2013.

[5] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[6] National Research Council. *Frontiers in Massive Data Analysis*. The National Academies Press, 2013.

[7] Dimitri P. Bertsekas. Incremental proximal methods for large scale convex optimization. *Mathematical Programming*, 129(2):163–195, 2011.

[8] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.

[9] Douglas Aberdeen, Ondrej Pacovsky, and Andrew Slater. The learning behind gmail priority inbox. In *LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*, 2010.

[10] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad click prediction: A view from the trenches. In *KDD*, pages 1222–1230, 2013.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[12] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *ACL*, pages 440–447, 2007.

[13] Ken Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[14] Riki Eto, Ryohei Fujimaki, Satoshi Morinaga, and Hiroshi Tamano. Fully-automatic

bayesian piecewise sparse linear models. In *AISTATS*, pages 238–246, 2014.

[15] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[16] Noah A. Smith. *Linguistic Structure Prediction.* Morgan & Claypool Publishers, 2011.

[17] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications.* Springer-Verlag, 2006.

[18] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[19] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Regularization techniques for learning with matrices. *Journal of Machine Learning Research*, 13:1865–1890, 2012.

[20] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2nd edition, 1999.

[21] Naum Z. Shor. *Minimization Methods for Non-differentiable Functions.* Springer-Verlag, 1985.

[22] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *Lecture notes*, 2003.

[23] Yurii Nesterov. *Introductory lectures on convex optimization : a basic course.* Kluwer Academic Publishers, 2004.

[24] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[25] Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[26] Arkadiĭ S. Nemirovsky and David B. Yudin. *Problem complexity and method efficiency in optimization.* John Wiley and Sons, 1983.

[27] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[28] Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[29] Hsiang-Fu Yu, Cho-Jui Hsieh, Kai-Wei Chang, and Chih-Jen Lin. Large linear classification when data cannot fit in memory. *ACM Transactions on Knowledge Discovery from Data*, 5(4):1–23, 2012.

[30] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

[31] Hidekazu Oiwa, Shin Matsushima, and Hiroshi Nakagawa. Frequency-aware truncated methods for sparse online learning. In *ECML/PKDD*, pages 533–548, 2011.

[32] Hidekazu Oiwa, Shin Matsushima, and Hiroshi Nakagawa. Healing truncation bias: Self-weighted truncation framework for dual averaging. In *ICDM*, pages 575–584, 2012.

[33] Hidekazu Oiwa, Shin Matsushima, and Hiroshi Nakagawa. Feature-aware regularization for sparse online learning. *SCIENCE CHINA Information Sciences*, 57(5):1–21, 2014.

[34] Hidekazu Oiwa and Hiroshi Nakagawa. Online and stochastic learning with a human cog-

nitive bias. In *AAAI*, pages 2020–2026, 2014.

[35] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[36] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[37] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[38] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[39] Lev M. Bregman. The Relaxation Method of Finding the Common Point of Convex Sets and Its Application to the Solution of Problems in Convex Programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[40] Jonathan M. Borwein and Adrian S. Lewis. *Convex Analysis And Nonlinear Optimization: Theory And Examples.* Springer-Verlag, 2006.

[41] Shai Shalev-Shwartz. *Online learning: Theory, algorithms, and applications.* PhD thesis, The Hebrew University, 2007.

[42] Constantin Zalinescu. *Convex analysis in general vector spaces.* World Scientific Publishing, 2002.

[43] Kai-Wei Chang and Dan Roth. Selective block minimization for faster convergence of limited memory large-scale linear models. In *KDD*, pages 699–707, 2011.

[44] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[45] Paul Ruvolo and Eric Eaton. ELLA: an efficient lifelong learning algorithm. In *ICML*, pages 507–515, 2013.

[46] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.

[47] Thomas M. Cover. Behavior of sequential predictors of binary sequences. In *Conference on Information Theory, Statistical Decision Functions, Random Processes*, pages 263–272, 1965.

[48] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[49] Elad Hazan and Comandur Seshadhr. Efficient learning algorithms for changing environments. In *ICML*, page 50, 2009.

[50] Eric C. Hall and Rebecca Willett. Dynamical models and tracking regret in online convex programming. In *ICML*, pages 579–587, 2013.

[51] Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *Journal of Computer and System Sciences*, 74(1):97–114, 2008.

[52] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval*

Research Logistics, 3:95–110, 1956.

[53] Elad Hazan and Satyen Kale. Projection-free online learning. In *ICML*, 2012.

[54] Jun Liu and Jieping Ye. Efficient euclidean projections in linear time. In *ICML*, pages 657–664, 2009.

[55] John C. Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML*, pages 272–279, 2008.

[56] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

[57] Shai Shalev-Shwartz and Sham M. Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, pages 1457–1464, 2008.

[58] Peter L. Bartlett, Elad Hazan, and Alexander Rakhlin. Adaptive online gradient descent. In *NIPS*, 2007.

[59] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

[60] Shai Shalev-shwartz and Yoram Singer. Convex repeated games and fenchel duality. In *NIPS*, 2006.

[61] John C. Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, pages 14–26, 2010.

[62] Shai Shalev-shwartz and Yoram Singer. Logarithmic regret algorithms for strongly convex repeated games. Technical report, The Hebrew University, 2007.

[63] Nati Srebro, Karthik Sridharan, and Ambuj Tewari. On the universality of online mirror descent. In *NIPS*, pages 2645–2653, 2011.

[64] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

[65] John C. Duchi, Alekh Agarwal, and Martin J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2012.

[66] Saghar Hosseini, Airlie Chapman, and Mehran Mesbahi. Online distributed optimization via dual averaging. In *CDC*, pages 1484–1489, 2013.

[67] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71:291–307, 2005.

[68] Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69:115–142, 2007.

[69] James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

[70] Elad Hazan. The convex optimization approach to regret minimization. In *Optimization for Machine Learning*, pages 287–303. MIT Press, 2011.

[71] H. Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and implicit updates. *CoRR*, abs/1009.3240, 2010.

[72] H. Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization. In *AISTATS*, pages 525–533, 2011.

[73] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[74] Stéphane Ross, Paul Mineiro, and John Langford. Normalized online learning. In *UAI*, 2013.

[75] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[76] Nikos Karampatziakis and John Langford. Online importance weight aware updates. In *UAI*, pages 392–399, 2011.

[77] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[78] Albert B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.

[79] Nick Littlestone. *Mistake Bounds and Logarithmic Linear-threshold Learning Algorithms*. PhD thesis, University of California at Santa Cruz, 1989.

[80] Richard O. Duda and Peter E. Hart. *Pattern Classification and scene analysis*. John Wiley and Sons, 1973.

[81] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.

[82] Junpei Komiyama, Hidekazu Oiwa, and Hiroshi Nakagawa. Robust distributed training of linear classifiers based on divergence minimization principle. In *ECML/PKDD*, pages 1–17, 2014.

[83] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[84] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *ICML*, pages 264–271, 2008.

[85] Koby Crammer, Mark Dredze Fern, and O Pereira. Exact convex confidence-weighted learning. In *NIPS*, 2008.

[86] Jialei Wang, Peilin Zhao, and Steven C.H. Hoi. Exact soft confidence-weighted learning. In *ICML*, pages 121–128, 2012.

[87] Tianbing Xu, Jianfeng Gao, Lin Xiao, and Amelia C. Regan. Online classification using a voted RDA method. In *AAAI*, pages 2170–2176, 2014.

[88] Balas K. Natarajan. On learning sets and functions. *Machine Learning*, 4(1):67–97, 1989.

[89] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk

bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2003.

[90] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[91] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer-Verlag, 2nd edition, 2012.

[92] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[93] Anatoli Juditsky, Guanghui Lan, Arkadi Nemirovski, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[94] Harold J. Kushner and G. George Yin. *Stochastic approximation and recursive algorithms and applications*. Springer-Verlag, 2nd edition, 2003.

[95] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, pages 71–79, 2013.

[96] Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[97] Alekh Agarwal, Peter L. Bartlett, Pradeep D. Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, 2012.

[98] Elad Hazan and Satyen Kale. Beyond the regret minimization barrier: an optimal algorithm for stochastic strongly-convex optimization. In *COLT*, pages 421–436, 2011.

[99] Mehrdad Mahdavi, Tianbao Yang, Rong Jin, Shenghuo Zhu, and Jinfeng Yi. Stochastic gradient descent with only one projection. In *NIPS*, pages 503–511, 2012.

[100] Elad Hazan and Satyen Kale. Projection-free online learning. In *ICML*, 2012.

[101] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *ICML*, pages 343–351, 2013.

[102] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

[103] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[104] Thorsten Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, 2006.

[105] Thorsten Joachims. Making large-scale support vector machine learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, pages 169–184. MIT Press, 1999.

[106] Nicolas Le Roux, Mark W. Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pages 2672–2680, 2012.

[107] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive

variance reduction. In *NIPS*, pages 315–323, 2013.

[108] Julien Mairal. Optimization with first-order surrogate functions. In *ICML*, pages 783–791, 2013.

[109] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, 2013.

[110] Gerard Salton and Christopher Buckley. *Term-weighting approaches in automatic text retrieval*, pages 323–328. Morgan Kaufmann Publishers Inc., 1997.

[111] John C. Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

[112] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B*, 67(2):301–320, 2005.

[113] H. D. Bondell and B. J. Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123, 2008.

[114] Francis R. Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends in Machine Learning*, 4(1):1–106, 2012.

[115] Hiroshi Nakagawa Hidekazu Oiwa, Issei Sato. Novel sparse modeling by l2 + l0 regularization. In *DISCML: NIPS Workshop on Discrete and Combinatorial Problems in Machine Learning*, 2013.

[116] Dijun Luo, Chris H. Q. Ding, and Heng Huang. Toward structural sparsity: an explicit ℓ_2/ℓ_0 approach. *Knowledge and Information Systems*, 36(2):411–438, 2013.

[117] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1178–1192, 2013.

[118] Haixian Wang and Wenming Zheng. Robust sparsity-preserved learning with application to image visualization. *Knowledge and Information Systems*, 39(2):287–304, 2014.

[119] Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical report, CORE, Catholic University of Louvain, 2007.

[120] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

[121] Paul Tseng. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Mathematical Programming*, 125:263–295, 2010.

[122] Bob Carpenter. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, Alias-i, 2008.

[123] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[124] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. Stochastic gradient descent training for l1-regularized log-linear. In *ACL*, pages 477–485, 2009.

[125] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202, 2012.

[126] Sangkyun Lee and Stephen J. Wright. Manifold identification in dual averaging for regularized stochastic online learning. *Journal of Machine Learning Research*, 13:1705–1744, 2012.

[127] Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal strategies and minimax lower bounds for online convex games. In *COLT*, pages 414–424. Omnipress, 2008.

[128] Shin Matsushima, Nobuyuki Shimizu, Kazuhiro Yoshida, Takashi Ninomiya, and Hiroshi Nakagawa. Exact passive-aggressive algorithm for multiclass classification using support class. In *SDM*, pages 303–314, 2010.

[129] Julien Mairal. Stochastic majorization-minimization algorithms for large-scale optimization. In *NIPS*, pages 2283–2291, 2013.

[130] Burr Settles. Closing the loop: fast, interactive semi-supervised annotation with queries on features and instances. In *EMNLP*, pages 1467–1478, 2011.

[131] John Langford and Alina Beygelzimer. Sensitive error correcting output codes. In *COLT*, pages 158–172, 2005.

[132] Jialei Wang, Peilin Zhao, and Steven C. H. Hoi. Cost-sensitive online classification. In *ICDM*, pages 1140–1145, 2012.

[133] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 185:1124–1131, 1974.

[134] Daniel Kahneman and Amos Tversky. Choices, values and frames. *American Psychologist*, 39:341–350, 1984.

[135] Richard H. Thaler. Toward a positive theory of consumer choice. *Journal of Economic Behavior & Organization*, 1(1):39–60, 1980.

[136] Daniel Kahneman, Jack Knetsch, and Richard H. Thaler. Experimental tests of the endowment effect and the coase theorem. *Journal of Political Economy*, 98(6):1325–1348, 1990.

Appendix A

Another Choice of Step Sizes for E-OGD

As an alternative to the setting of (5.6), we utilize another sequence of step sizes in the appendix. We set the step size as follows:

$$\eta_t = \frac{c}{\sqrt{t}} \frac{1}{(1+\gamma)^{N_t}} \tag{A.1}$$

where N_t is the number of wrong prediction from round 1 to round t.

Lemma 18. *When we set step sizes as (A.1) , $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) \geq 0$ for all $t \geq 2$.*

Proof. It is straightforward to show that for any $t \geq 2$, $\eta_t < \eta_{t-1}$ is satisfied from the definition of η_t. From this property, we see that $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) \geq 0$ is always satisfied when $r_t = 1 + \gamma$ (regardless of whether $r_{t-1} = 1$ or $1 + \gamma$). When $r_t = 1$, the algorithm misclassified the example at t-th round. Therefore, N_t is incremented and $r_t/\eta_t > (1 + \gamma)/\eta_{t-1}$ is satisfied. In summary, $(r_t/\eta_t) - (r_{t-1}/\eta_{t-1}) \geq 0$ is always satisfied. □

The next theorem gives the regret upper bound of E-OGD.

Theorem 17. *Let $\{\mathbf{w}_t\}_{t=1:T+1}$ be derived according to E-OGD's update rule. Assume that for all $t \geq 1$, $\|\mathbf{w}_t\|_2 \leq R$, $\|\nabla \ell_t(\mathbf{w}_t)\|_2 \leq G$. When we set $\{\eta_t\}_{t=1:T}$ as in (A.1) and $c = \sqrt{2}R/G$, the upper bound of regret becomes*

$$\text{Regret}(T) \leq \sqrt{2}RG\left((1+\gamma)^{N_T+1} + (1+\gamma)\right)\sqrt{T}\ .$$

If $N_T \ll T$, the regret bound becomes sublinear. In general, the probability of mistakes gets lower when the algorithm converges and N_T would not get larger.

Proof. The main procedure of the proof follow the one of Theorem 15, we obtain

$$\sum_{t=1}^{T} r_t \langle \mathbf{g}_t, \mathbf{w}_t - \mathbf{u} \rangle \leq \frac{2R^2 r_T}{\eta_T} + \frac{G^2}{2} \sum_{t=1}^{T} \eta_t r_t \ . \tag{A.2}$$

In addition,

$$\sum_{t=1}^{T} \eta_t \leq \sum_{t=1}^{T} \frac{c}{\sqrt{t}} \leq 2c\sqrt{T} \tag{A.3}$$

The following inequality is derived by setting $r_t = 1 + \gamma$ for all t in the second term.

$$\begin{aligned} \mathrm{Regret}(T) &\leq (1+\gamma) \left(\frac{2R^2}{\eta_T} + \frac{G^2}{2} \sum_{t=1}^{T} \eta_t \right) \\ &\leq \left(\frac{2R^2(1+\gamma)^{N_T+1}}{c} + cG^2(1+\gamma) \right) \sqrt{T} \ . \end{aligned} \tag{A.4}$$

When $c = \sqrt{2}R/G$, we have

$$\mathrm{Regret}(T) \leq \sqrt{2}RG \left((1+\gamma)^{N_T+1} + (1+\gamma) \right) \sqrt{T} \ .$$

□

Theorem 18. *Assume that the conditions set in Theorem 17 are satisfied and there is an integer t_p such that $r_t(z) = r_{t_p}(z)$ for any $t \geq t_p$. The following formula is satisfied for any $\mathbf{u} \in \mathcal{W}$,*

$$\mathbb{E}_{\mathcal{D}^T} \left[\mathbb{E}_{z \sim \mathcal{D}_P} \left[\ell(\bar{\mathbf{w}}; z) \right] \right] - \mathbb{E}_{z \sim \mathcal{D}_P} \left[\ell(\mathbf{u}; z) \right] \leq \frac{\sqrt{2}RG \left((1+\gamma)^{N_T+1} + (1+\gamma) \right)}{\sqrt{T} - (t_p - 1)/\sqrt{T}} \ . \tag{A.5}$$

If $N_T \ll T$ and $t_p \ll T$, the convergence speed is $O(1/\sqrt{T})$. When the number of data is finite, there exists a constant t_p such that $r_t(z) = r_{t_p}(z)$ for any $t \geq t_p$.

Proof. First, we analyze the regret bound from round t_p to T.

$$\begin{aligned} \mathrm{Regret}(t_p : T) &\leq (1+\gamma) \left(\frac{2R^2}{\eta_T} + \frac{G^2}{2} \sum_{t=t_p}^{T} \eta_t \right) \\ &\leq (1+\gamma) \left(\frac{2R^2(1+\gamma)^{N_T}}{c} + cG^2 \right) \sqrt{T} \end{aligned} \tag{A.6}$$

When $c = \sqrt{2}R/G$,

$$\mathrm{Regret}(T) \leq \sqrt{2}RG \left((1+\gamma)^{N_T+1} + (1+\gamma) \right) \sqrt{T} \ .$$

For simplicity, we will describe $H_{\mathcal{D}^T}\mathbb{E}_{z\sim\mathcal{D}_P}[\ell(\cdot;z)]$ as $\mathcal{D}_P(\cdot)$. By the reformulation as in the same case of Theorem 16, the expectation of the second term is reformulated as follows:

$$\mathbb{E}_{\mathcal{D}^T}\left[\sum_{t=t_p}^{T} r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{u})\right] = (T - t_p + 1)\mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\mathbf{u})\right] .$$

The expectation of the first term can be reformulated as:

$$\mathbb{E}_{\mathcal{D}^T}\left[\sum_{t=t_p}^{T} r_t((\mathbf{x}_t, y_t))\ell_t(\mathbf{w}_t)\right] = (T - t_p + 1)\mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\bar{\mathbf{w}})\right] .$$

The following formula is obtained by combining these formulas with Lemma 17

$$\begin{aligned}\mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\bar{\mathbf{w}})\right] - \mathbb{E}_{\mathcal{D}^T}\left[\mathcal{D}_P(\mathbf{u})\right] &\leq \frac{1}{T - t_p + 1}\mathbb{E}_{\mathcal{D}^T}\left[\mathrm{Regret}(t_p : T)\right] \\ &\leq \frac{\sqrt{2}RG\left((1+\gamma)^{N_T+1} + (1+\gamma)\right)}{\sqrt{T} - (t_p - 1)/\sqrt{T}} . \qquad \text{(A.7)}\end{aligned}$$

□