Kernel based online learning

Zhao, Peilin

2013

Zhao, P. (2013). Kernel based online learning. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54684>

<https://doi.org/10.32657/10356/54684>

Kernel Based Online Learning

Zhao Peilin

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2013

Acknowledgments

I would express my deep gratitude to my supervisor, Dr. Steven HOI, who has brought me into the beautiful world of machine learning. Over the past four years, he has taught me a lot about every area of machine learning. He not only inspires me by sharing his sharp insights for various problems, but also encourages me by his passion and enthusiasm in pursuing great research. He is not only a perfect supervisor for my research, but also a good friend in my life.

Secondly, I would thank my collaborators: Rong Jin, Ivor W. Tsang, Liang-Tien Chia, Vivekanand Gopalkrishnan, Zhi-Yong Liu, Ying He, Shenghua Gao, Pengcheng Wu, Bin Li, Jiale Wang, Tianbao Yang, Jinfeng Zhuang and Hao Xia. I feel deeply honored to cooperate with them in my past research.

Thirdly, I would also appreciate the other members of our research group. It is my honor to talk, study and work with them together. Thank Jinfeng Zhuang, Bin Li, Pengcheng Wu, Hao Xia, Dayong Wang, Guangxia Li, Ning Chen, Boshen Zhang, Jing Lv, and Doyen Sahoo.

Besides, during the past few years, I have made quite a few friends here. I also acknowledge the help from them: Min Wu, Shuguo Han, Yu Zhang, Juyong Zhang, Lei Wu, Linhong Zhu, Yu Nie, Xiangang Cheng, Zhihua Fan, Shengqing Zhu, Maodong Li, Danqi Wang, Jie Zeng, Bin Zhao, Zhengxiang Wang, Shengbo Yang, Da Zhang, Qi Duan, Feng Zhong, Xuejie Li, Yiqun Hu, Lixin Duan, Xiao Teng, Mingkui Tan, Li Wang, Hui Li, Changjiu Jin, Dingxiong Deng, Yinan Yu, Chengliang Li, Rajesh Sharma, Wenting Liu, Shaohua Li, Yong Liu, Haifeng Lu, Wen Li, Shumo Chu, Zhen Hai, Peng Cheng, Liang Feng, Qi Mao, Yuming Fang, Jiang Qian, Yushen Li, Wei Shou, Yifan Li, Xinxing Xu, Wenjie Zhang, Diwen Zhu, Zhiqiang Xu, Youze Tang, Jun Zhang, Xin Cao, Lisi Chen, Wenqing Lin, Huanhuan Zhang, Zongyang Ma and so on.

As a Ph.D. student, I am also very grateful to Nanyang Technological University, where I have attended interesting courses, obtained plenty research materials and enjoyed strong academic atmosphere. In addition, I would also thank all the staffs from the school of computer engineering, graduate studies office and other related offices for their assistances.

Finally, I must express my heartfelt thanks to my parents and brother. It is their support and love that encourage me to pursue my dreams.

Abstract

Kernel Based Online Learning (KBOL) is an important branch of online learning in machine learning, in which the objective is to optimize the online predictive performance, typically measured by classification accuracy. It enjoys many advantages when solving real-world large-scale applications, such as classification, regression, ranking, and clustering, etc. Although being extensively studied, KBOL can be more practical if several critical problems can be solved.

Firstly, most existing KBOL algorithms keep the weights of existing Support Vectors (SVs) unchanged during the whole online learning process, which is too conservative to significantly improve the online accuracy. To overcome this limitation, based on the primal-dual framework, we extend some existing single updating online learning algorithms to “Double Updating Online Learning” (DUOL), which not only updates the weight of the newly added SV, but also simultaneously updates the weight of one existing SV, referred to “auxiliary example”. In this thesis, we investigate effective DUOL algorithms for both binary and multi-class online classification tasks.

Secondly, most existing KBOL algorithms are designed with unbounded number of support vectors, making them unsuitable for handling large-scale datasets. To overcome this challenge, we investigate the problem of budget KBOL by designing various budget maintenance strategies for the binary DUOL algorithm that aims to constrain the number of support vectors by a predefined budget when learning the kernel-based prediction function in the online learning process.

Thirdly, traditional KBOL algorithms mainly aim to maximize the online accuracy, which is not suitable for some real world tasks, where the datasets may be highly imbalanced so that the cost-sensitive metrics are more appropriate. To solve this issue, we propose a family of cost-sensitive online classification algorithms by exploiting the DUOL techniques, which are designed to directly optimize two well-known cost-sensitive measures: (i) maximization of weighted sum of *sensitivity* and *specificity*, and (ii) minimization of weighted *misclassification cost*.

Finally, most of traditional KBOL algorithms only learn from one single target domain, which ignore any existing knowledge learnt from other related source domain. This obviously is a waste of resource. To tackle this problem, we investigate an online learning framework called “Online Transfer Learning” (OTL) that aims to transfer knowledge from some source domain to an online learning task on a target domain.

Contents

Acknowledgments	i
Abstract	iii
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Background and Motivations	1
1.2 Problems and Research Scope	3
1.3 Approach and Methodology	4
1.4 Summary of Contributions	5
1.5 Thesis Organization	6
1.6 Notation Table	8
2 Literature Survey	9
2.1 Linear Support Vector Machines	9
2.1.1 The Separable Case	9
2.1.2 The Non-Separable Case	10
2.2 Kernel Function	11
2.2.1 Positive Definite Kernels	12
2.2.2 Reproducing Kernel Hilbert Space	13
2.2.3 Some Examples	14
2.3 Convex Optimization	14
2.3.1 Basics of Convex Optimization	14
2.3.2 Solution of Linear Separable SVM	16
2.4 Online Learning	17

2.4.1	Formal Formulation	17
2.4.2	Update Type	19
2.4.3	Problem Type	20
2.4.4	Aggressive Level	24
2.5	Summary	25
3	Binary Double Updating Online Learning	27
3.1	Related Work	28
3.2	Binary Double Updating Online Learning	29
3.2.1	Problem Setting	29
3.2.2	Motivation	30
3.2.3	Double Updating Online Learning	32
3.3	Experimental Results	37
3.3.1	Testbeds and Experimental Setup	37
3.3.2	Performance Evaluation	39
3.3.3	Evaluation of Different Auxiliary Example Selection Strategies and the Sensitivity to Parameter C for DUOL	41
3.3.4	Empirical Evaluation of Mistake Bounds	41
3.4	Summary	44
4	Multiclass Double Updating Online Learning	45
4.1	Related Work	45
4.2	Multiclass Double Updating Online Learning	46
4.2.1	Problem Setting	46
4.2.2	Label Ranking for Multiclass Learning	46
4.2.3	A Perceptron Algorithm for Online Multiclass Learning	47
4.2.4	Multiclass Double Updating Online Learning	47
4.3	Experimental Results	52
4.3.1	Testbeds and Experimental Setup	52
4.3.2	Performance Evaluation	53
4.4	Summary	55

5	Budget Double Updating Online Learning	57
5.1	Related Work	57
5.2	Double Updating Online Learning on A Fixed Budget	59
5.2.1	Problem Setting	59
5.2.2	Framework of Budget Double Updating Online Learning	61
5.2.3	Theoretical Analysis	61
5.3	Budget Maintenance Strategies	63
5.3.1	BDUOL Algorithm by Removal Strategy	63
5.3.2	BDUOL Algorithm by Projection Strategy	64
5.3.3	BDUOL Algorithm by Nearest Neighbor Strategy	65
5.4	Experimental Results	66
5.4.1	Algorithms for Comparison	66
5.4.2	Experimental Testbed and Setup	67
5.4.3	Performance Evaluation of Non-budget Algorithms	68
5.4.4	Performance Evaluation of Budget Algorithms	68
5.5	Summary	70
6	Cost-Sensitive Double Updating Online Learning	75
6.1	Related Work	76
6.2	Cost-Sensitive Online Classification	77
6.2.1	Problem Formulation	77
6.2.2	Algorithms	78
6.3	Experiments of Cost-Sensitive Online Classification	86
6.3.1	Experimental Testbed and Setup	86
6.3.2	Evaluation of Weighted Sum Performance	87
6.3.3	Evaluation of Weighted Cost Performance	90
6.4	Application to Online Anomaly Detection	90
6.4.1	Application Domains and Testbeds.	90
6.4.2	Empirical Evaluation Results.	93
6.5	Summary	94

7	Online Transfer Learning	97
7.1	Related Work	97
7.2	Problem Formulation	99
7.3	Single-view Online Transfer Learning	100
7.4	Multi-view Online Transfer Learning	102
7.5	Experimental Results	105
7.5.1	Experimental Testbed and Setup for Single-view OTL	105
7.5.2	Evaluation of Single-view OTL Tasks	106
7.5.3	Experimental Testbed and Setup for Multi-view OTL	107
7.5.4	Evaluation of Multi-view OTL Tasks	108
7.6	Summary	109
8	Conclusions and Future Work	111
8.1	Conclusions	111
8.2	Future Directions	112
8.2.1	Online Learning with Concept Drifting	112
8.2.2	Active Online Learning	113
8.2.3	Application of Kernel Based Online Learning	113
8.3	Summary	114
A	Supplementary Proofs	115
A.1	A Pure Online Version of LASVM	115
A.2	Proofs for the Chapter 3	116
A.2.1	The Proof for Proposition 3.2	116
A.3	Proofs for the Chapter 4	119
A.3.1	The Proof for Proposition 4.3	119
A.3.2	The Proof for Proposition 4.4	120
A.4	Proofs for the Chapter 7	124
A.4.1	The Proof for Proposition 7.8	124
A.4.2	The Proof for Theorem 19	125
A.4.3	The Proof for Proposition 7.9	125
A.4.4	The Proof for Lemma 7.3	126

B Publication List	129
References	132

List of Figures

1.1	The dependence structure of the topics of this thesis.	6
3.1	Comparison between DUOL and DUOL _{rand} with varied C values.	42
3.2	Empirical comparison of M , $M_d^w(\rho)$ and $M_d^s(\rho)$ w.r.t. varied $\rho \in [0, 1]$ values.	43
5.1	Evaluation of online mistake rates against the number of samples on three datasets. The plotted curves are averaged over 20 random permutations.	71
5.2	Evaluation of online mistake rates against the number of samples on three datasets. The plotted curves are averaged over 20 random permutations.	72
6.1	Evaluation of online “sum” performance of the CSDUOL _{sum} algorithm.	89
6.2	Evaluation of online “cost” performance of the CSDUOL _{cost} algorithm.	92

List of Tables

1.1	Some common notations used in the thesis.	8
3.1	Binary-class data sets used in the experiments.	38
3.2	Evaluation of online learning algorithms on the binary-class data sets. . .	40
4.1	Multiclass data sets used in the experiments.	53
4.2	Evaluation of multiclass online learning algorithms on the multiclass data sets.	54
5.1	Details of the datasets in our experiments.	67
5.2	Evaluation of non-budget algorithms on the the data sets.	68
5.3	Evaluation of several budgeted algorithms with varied budget sizes. . . .	69
6.1	List of binary datasets in our experiments.	87
6.2	Evaluation of the performance of CSDUOL _{sum} and other existing algorithms.	88
6.3	Evaluation of the performance of CSDUOL _{cost} and other existing algorithms. .	91
6.4	Data Sets for Online Anomaly Detection.	93
6.5	Evaluation of balanced accuracy performance for online anomaly detection. . .	94
7.1	Datasets used in the Single-view OTL tasks.	105
7.2	Evaluation of Single-view OTL Tasks.	106
7.3	Summary of data sets used for multi-view OTL tasks.	108
7.4	Evaluation of Multi-view OTL Tasks.	108

Chapter 1

Introduction

This introduction will present a brief review to online learning, formal statements of our problems and the main contribution of this thesis. The concepts introduced here will be explained with more details in the later chapters.

1.1 Background and Motivations

Online learning takes place in a sequence of rounds. On each round the learner is asked a question, and then the learner should predict the answer yes or no. Taking spam detection for example, when receiving an email, we hope the learner can predict the answer to the question of whether the email is a spam or not. After the learner provides its prediction, the learner will get the true answer as a feedback. As in the spam detection, the learner will be told whether the email is a spam or not. And after the feedback, the learner will suffer a loss depending on the difference between the prediction and the answer. Finally, the learner could update its prediction rule in order to improve its performance on the future examples. As in the previous example, the learner can update its classification method to improve its prediction accuracy on emails received in the future. Furthermore, a pair of question and answer is usually termed as an “example”, while question and answer are termed as “instance” and “label”, respectively.

The problem setting in the above paragraph is formally referred as binary classification problem, since there are only two possible answers. Besides, as a generalization of the binary case, multi-class classification [CS03] is also a typical problem, where the number of possible answers is more than two. While the number of possible answers is finite in the last two settings, there are also several popular settings where the number of possible answers is infinite including: online regression [CDK⁺06], online metric learning [SSSN04, JKDG08, DKJ⁺07] and online convex optimization [HKKA06, FKM05, SSK08], in which the learner should predict an answer from a set of infinite elements. Specifically, online regression is to fit a curve according to a sequence of online examples, while online metric

learning is to learn a dynamic metric for a sequence of example pairs and their distances (or triplet). Besides, online convex optimization is a general framework, which needs to predict the minimum of a sequence of convex functions, which will be disclosed after the prediction. Finally, the problem of online ranking [CS02, CS05, SSS07c] is to predict the relevance of several labels with respect to a sequence of instances.

The reason online learning is applied to so many settings is that it enjoys many advantages for real-world large-scale applications. For example, in some real applications, data often arrives sequentially while predictions must be made immediately. Examples include spam detection and portfolio selection [LZHG12]. Moreover, online learning is very attractive for large-scale learning tasks, e.g., training Support Vector Machine from billions of data, since when the training data is abundant and computing resources are comparatively scarce, suitably designed online learning algorithm asymptotically outperforms any batch learning algorithm [BL03], although the solution of online learning converges slower to the solution of SVM. Third, online learning can adapt to online changing data distributions, for which concept drift is a typical problem. Formally, concept drift means that the target variable to be predicted changes over time in the learning process [HW01].

Online learning owns so many advantages that it has been applied to various specific problems. For example, online ranking could be used for information retrieval (IR), which is the science of retrieving useful information from huge amount of documents. One of the best applications for IR techniques is the web search engine. Another application of online learning is stock market trading, which is to maximize the final return from the stock market. Specifically, suppose there are n stocks and 1 dollar fund initially. Then at the beginning, the learner will allocate the fund on the n stocks according to the current strategy, while at the end the learner will update its allocation strategy based on the return value of every stock provided by the market.

Although online learning is successfully applied on many real world applications, the existing work on online learning is far from perfect. Here, we point out several critical limitations in online learning. The first key drawback is that most existing Kernel Based Online Learning (KBOL) algorithms keep the weights of existing Support Vectors (SVs) unchanged during the whole online learning process, which is too conservative to significantly improve the online accuracy. The second critical shortcoming of most existing KBOL algorithms is the unbounded number of support vectors, making them unsuitable for handling large-scale datasets. The third crucial disadvantage is that traditional KBOL algorithms mainly aim to maximize the online accuracy, which is not suitable for some real world tasks, where the datasets may be highly imbalanced. Finally, most of traditional KBOL algorithms only learn from one single target domain and ignore any existing knowledge learnt from other related source domain, which obviously is a waste of resource.

1.2 Problems and Research Scope

My research mainly aims to investigate new techniques to address the four limitations mentioned in the last section.

To overcome the first limitation, based on the primal-dual framework [SSS06b], we extend some existing single updating online learning approach to “double updating” approach to online learning named as “DUOL”, which not only updates the weight of the newly added support vector but also adjusts the weight of one existing support vector that the most seriously conflicts with the new support vector. We show that the mistake bounds for online binary and multiclass classification tasks can be significantly reduced by the proposed DUOL algorithms. We have conducted an extensive set of experiments by comparing with a number of competing algorithms on both binary and multiclass online classification tasks. Promising empirical results showed that the proposed double updating online learning algorithms always consistently surpass the regular online learning algorithms without double updates.

The second problem is to save the memory cost of storing SVs for online learning. Specifically, we would keep the number of stored SVs less than a pre-fixed budget number, and at the same time minimize the number of mistakes made by the algorithm. So, the goal of this problem is to find an effective budget maintenance strategy so that the modified classifier keeps as more useful information as possible. For this problem, we investigate the problem of budget kernel-based online learning that aims to constrain the number of support vectors by a predefined budget when learning the kernel-based prediction function in the online learning process. We present a new framework of budget kernel-based online learning based on DUOL. We analyze the theoretical underpinning of the proposed Budget Double Updating Online Learning (BDUOL) framework, and then propose several BDUOL algorithms by designing different budget maintenance strategies. We evaluate the empirical performance of the proposed BDUOL algorithms by comparing them with several well-known budget kernel-based online learning algorithms, in which encouraging results validate the efficacy of the proposed technique.

While all the above problems adopt the learning accuracy as metric, the third problem is to online optimize cost-sensitive metrics, which are more suitable for real world applications. For this issue, we formally study cost-sensitive online classification, and propose a new framework for cost-sensitive online classification by exploiting the DUOL techniques. Based on the framework, we propose a family of cost-sensitive online classification algorithms, which are designed to directly optimize two well-known cost-sensitive measures: (i) maximization of weighted sum of *sensitivity* and *specificity*, and (ii) minimization of weighted *misclassification cost*. We analyze the theoretical bounds of the cost-sensitive measures made by the proposed algorithms, and extensively examine their

empirical performance on a variety of cost-sensitive online classification tasks. Finally, we demonstrate the application of the proposed technique for solving several online anomaly detection tasks, showing that the proposed technique could be a highly efficient and effective tool to tackle cost-sensitive online classification tasks in various application domains.

The final problem is to improve the learning efficiency of kernel based online learning by transferring the learnt knowledge from other domains. Concretely, we could utilize a learnt classifier from other domains to improve the online learning accuracy. For this problem, we investigate a new machine learning framework called Online Transfer Learning (OTL) that aims to transfer knowledge from some source domain to improve online learning performance on a target domain. Specifically, we do not assume the target data follows the same class or generative distribution as the source data, and our key motivation is to improve a supervised online learning task in a target domain by exploiting the knowledge that had been learnt from large amount of training data in source domains. OTL is in general challenging since data in both domains not only can be different in their class distributions but also can be different in their feature representations. As a first attempt to this problem, we propose techniques to address two kinds of OTL tasks: one is to perform OTL for single-view task, and the other is to perform OTL for multi-view task. We show the mistake bounds of the proposed OTL algorithms, and empirically examine their performance on several challenging OTL tasks. Encouraging results validate that our technique do effectively reduce the number of mistakes suffered.

1.3 Approach and Methodology

To solve the first problem, we extend existing single updating online learning to double updating online learning, which will update the weights for the newly added SV and one existing SV simultaneously to improve the learning efficiency. The criteria for selecting from the set of existing support vectors are two-fold: 1) the selected example should share similar input pattern with the newly misclassified one; 2) the selected example belongs to a different class compared with the newly introduced SV. For the performance analysis of our algorithm, we adopt the Primal-Dual framework, which is essentially based on convex optimization to minimize the primal objective through maximizing the dual objective. Through the Primal-Dual framework, we proved that the dual ascent for our updating strategy is greatly increased, so that the mistake bound is significantly decreased.

For the second problem, we propose the framework of Budget Double Updating Online Learning. Specifically, we would modify the DUOL algorithm by introduce a budget maintenance to keep the number of stored SVs less than a pre-fixed budget number. To achieve the goal, we proposed three budget maintenance strategies: removal, projection

and nearest neighbor. In order to guarantee the proposed strategies are reasonable, we adopt the primal-dual analysis to decide how the budget maintenance strategies should be applied so that the final BDUOL algorithms are theoretically sound.

For the third problem, to online optimize the cost-sensitive metrics, we investigate a new framework of Cost-Sensitive Online Classification, which will directly optimize an regularized surrogate (or convex upper bound) of the cost-sensitive metrics. We proposed two effective algorithms based on double updating online learning techniques, which are designed to directly optimize two well-known cost-sensitive measures: (i) maximization of weighted sum of *sensitivity* and *specificity*, and (ii) minimization of weighted *misclassification cost*. For the proposed algorithm, we theoretically analyzed their cost-sensitive bounds by adopting primal-dual framework.

For the final problem, we proposed two kinds of Online Transfer Learning problem: single-view task and multi-view task. For the single-view task, we adopt ensemble learning in order to avoid potential covariance shift, the weights for the old model and the new model depend on the loss made by each algorithm. Through the intuitive weight updating, we can guarantee the performance of our algorithm is as better as the best between the source model and the target model. For the multi-view task, we assume the feature space of source domain is a subset of the one of target domain, so we can split the feature space of the target domain into two parts: the old features set and the new features set. In order to effectively combine the old features and the new features and transfer the knowledge from the source domain to the target domain, we adopt the co-training approach to solve the multi-view task.

1.4 Summary of Contributions

Firstly, to significantly improve the online accuracy, we presented a novel “double updating online learning” algorithm by extending some single updating online learning algorithm, based on the primal-dual framework.

Secondly, to keep the number of stored SVs less than a pre-fixed number, we presented a new framework of Budget Double Updating Online Learning (BDUOL) for kernel-based online learning on a fixed budget.

Thirdly, to online optimize the cost-sensitive metrics, we investigated a new framework of Cost-Sensitive Double Updating Online Learning, which aims to directly optimize cost-sensitive measures for online classification tasks.

Finally, to transfer well-learned knowledge from source domain to improve online learning performance on target domain, we studied the new problem of Online Transfer Learning (OTL), where both single-view and multi-view OTL are investigated.

1.5 Thesis Organization

In chapter 2, we will conduct a survey on the topic kernel based online learning. Firstly, we will briefly discuss about linear Support Vector Machines (SVM) [Bur98], which is important for us to understand online learning algorithm although it is an off-line method. After that, we will describe kernel map and its application. In addition, we will present some elementary background on convex optimization. Finally, we will give a more detailed summary of the online learning.

In chapter 3, we will present our double updating online learning (DUOL) method for binary classification.

We extend our DUOL methods from binary classification to multi-class classification in the chapter 4.

Figure 1.1: The dependence structure of the topics of this thesis.

In the chapter 5, we present a new framework of budget double updating online learning (BDUOL) for kernel-based online learning on a fixed budget.

In the chapter 6, we formally study cost-sensitive online classification, and propose a new framework for cost-sensitive online classification by exploiting the DUOL techniques.

In the chapter 7, we investigate a new machine learning framework called “*Online Transfer Learning*” (OTL) that aims to improve online learning on a target domain by transferring knowledge from some source domain.

In the last chapter 8, I conclude the thesis and propose some future promising directions on the topic of kernel based online learning.

Finally, Figure 1.1 describes the dependence structure of the topics of this thesis. Specifically, chapter 2, 3, 4, 5, 6 and 7 belong to “Online Learning”, “Binary Classification”, “Multi-class Classification”, “Budget Online Learning”, “Optimizing Cost-sensitive Metrics”, and “Online Transfer Learning”, respectively.

1.6 Notation Table

In order to make the presentation more clear, we show some common notations in the following table.

Table 1.1: Some common notations used in the thesis.

Notation	Explanation
$\mathbb{R}$	domain of real number
$\mathcal{X}$	domain from which the samples come from
$\mathbf{x}$	a single point, typically belonging to $\mathbb{R}^d$
y	the label of $\mathbf{x}$, taking a value from $\{-1, +1\}$ in binary classification
$\mathbf{x}^\top$	transpose of $\mathbf{x}$
κ	kernel function
Φ	the implicit mapping function of some kernel κ
$\mathbf{K}$	kernel matrix, the value of (i, j) entry is the evaluation $\kappa(\mathbf{x}_i, \mathbf{x}_j)$
$f(\cdot)$	a decision function defined on $\mathcal{X}$
$\ell(\cdot)$	a loss function defined on $\mathbb{R}$, typically the input is $yf(\mathbf{x})$
$\mathcal{F}$	the space in which the solution f exists
$\mathcal{H}$	hypothesis space
$\mathcal{H}_\kappa$	Hilbert space induced by some kernel κ

Chapter 2

Literature Survey

Online learning is closely connected with off-line learning, so before review online learning, we will introduce some off-line learning basics. Among existing off-line learning algorithms, support vector machines (SVM) appeared in the early nineties as optimal margin classifiers in the context of Vapnik’s statistical learning theory [Vap98]. It has shown strengths in a large variety of applications and become a standard method. Its *regularization+loss* framework has revolutionized the field of machine learning. Therefore, we first introduce SVM before discussing online algorithms.

2.1 Linear Support Vector Machines

To make the introduction easily understood, we firstly start with the simplest case: linear SVM trained on separable data.

2.1.1 The Separable Case

For a set of training data $\{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$, if there exists a linear hyperplane, which can separate all the positive examples (i.e. the examples with positive labels) from all the negative ones (i.e. the examples with negative labels), then that hyperplane is called separating hyperplane. Usually, a linear separating hyperplane is formulated as $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal vector for the separating hyperplane, $\frac{b}{\|\mathbf{w}\|}$ is the distance from the hyperplane to the origin, and $\|\mathbf{w}\|$ is the Euclidean norm of $\mathbf{w}$. Let’s denote the minimal distances from the separating hyperplane to the nearest positive example and the nearest negative example as d_p and d_n , respectively, then the “margin” of a separating hyperplane is denoted as $d_p + d_n$. For linearly separable case, the support vector machine simply finds the separating hyperplane with the maximal margin. This can be formulated as follows:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1, i = 1, \dots, n, \quad (2.1)$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1, i = 1, \dots, n, \quad (2.2)$$

which can be combined into one set of inequalities:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1, \quad i = 1, \dots, n. \quad (2.3)$$

Now consider the points for which the equality in 2.1 holds, they lie on the hyperplane $H_1 : \mathbf{x}_i \cdot \mathbf{w} + b = 1$ with normal $\mathbf{w}$ and perpendicular distance from the origin $|1 - b|/\|\mathbf{w}\|$; similarly, the points for which the equality in 2.2 holds lie on the hyperplane $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$, with normal again $\mathbf{w}$, and perpendicular distance from the origin $|-1 - b|/\|\mathbf{w}\|$. Hence $d_p = d_n = 1/\|\mathbf{w}\|$ and the margin is simply $2/\|\mathbf{w}\|$. Note that H_1 and H_2 are parallel and that no training points fall between them, we can find the pair of hyperplanes which gives the maximum margin by minimizing $\|\mathbf{w}\|^2$, subject to constraints 2.3. In summary, the optimization to be solved is as follows:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \geq 1, \quad i = 1, \dots, n. \end{aligned} \quad (2.4)$$

The solution of 2.4, is in the form of $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, where $\alpha_i \geq 0$ (explained in the later sections). $\alpha_i \neq 0$ only for those training points for which the equality in 2.3 holds, these points are termed as **support vectors**.

2.1.2 The Non-Separable Case

The above algorithm for separable data, when applied to non-separable data, can not find a feasible solution. To extend these ideas to handle non-separable data, we would like to relax the constraints 2.1 and 2.2, but only when necessary. Specifically, this can be done by introducing positive slack variables ξ_i , $i = 1, \dots, n$ in the constraints, which then become:

$$\begin{aligned} \mathbf{x}_i \cdot \mathbf{w} + b &\geq +1 - \xi_i \quad \text{for } y_i = +1, \\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq -1 + \xi_i \quad \text{for } y_i = -1, \\ \xi_i &\geq 0, \quad i = 1, \dots, n. \end{aligned}$$

Thus, for an error to occur, the corresponding ξ_i must exceed unity, so $\sum_i \xi_i$ is an upper bound on the number of training errors. Hence a natural way to assign an extra cost for errors is to change the objective function to be minimized from $\frac{\|\mathbf{w}\|^2}{2}$ to

$$\begin{aligned} & \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^n \xi_i, \\ & \xi_i \geq 0 \quad \text{and} \quad \xi_i \geq 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b), \quad i = 1, \dots, n, \end{aligned}$$

where $C > 0$ is a parameter to be chosen by the user, a larger C corresponding to assigning a higher penalty to errors. This is a convex optimization problem, which will be solved in the later sections.

Another way to formulate the above objective function is through introducing a hinge loss function $\ell(\mathbf{w}, b; (\mathbf{x}, y)) = \max\{0, 1 - y(\mathbf{w} \cdot \mathbf{x} + b)\}$. Then, we would minimize the cumulative loss $\sum_i \ell(\mathbf{w}, b; (\mathbf{x}_i, y_i))$ of $(\mathbf{w}, b)$ on the set of training data and at the same time maximize the margin $\frac{2}{\|\mathbf{w}\|}$, this can be formulated as follows:

$$\min_{\mathbf{w}, b} \frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^n \ell(\mathbf{w}, b; (\mathbf{x}_i, y_i)).$$

It is easy to verify the equivalence between the two above formulations.

2.2 Kernel Function

In the previous section, we reword linear SVMs, however in some cases the training data is highly non-separable, for example, positive and negative examples lie on two concentric circles, respectively. To generalize the above methods to the case where the decision function is not a linear function of the data, [BGV92] showed that a rather old trick can be used to accomplish this in an astonishingly straightforward way. Suppose we first mapped the data to some other (possibly infinite dimensional) Euclidean space $\mathcal{H}$ (i.e., a vector space on which a real-valued inner product is defined), using a mapping which we will call Φ :

$$\Phi : \mathbb{R}^d \rightarrow \mathcal{H},$$

so that the new training set $\{(\Phi(\mathbf{x}_i), y_i) | i = 1, \dots, n\}$ is roughly linearly separable in $\mathcal{H}$. Then, we will train the previous SVMs on this new set instead of the old $\{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$.

Because SVMs will only depend on the data through dot products $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ in $\mathcal{H}$ (which will be explained later), if there were a “kernel” function κ , such that $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, we would only need to use κ in SVMs, and would never need to explicitly even know what Φ is. One example of kernels is the Gaussian kernel function

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right),$$

where, $\mathcal{H}$ is infinite dimensional, so it is even impossible to work with Φ explicitly. However, if we replace $\mathbf{x}_i \cdot \mathbf{x}_j$ by $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ everywhere in SVMs, it will produce a classifier in the following nonlinear form:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}),$$

which can nonlinearly separate the set $\{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$.

After the above simple illustration, we will discuss about kernel function in detail, since it plays an important role in machine learning research.

2.2.1 Positive Definite Kernels

A kernel function [SS01] is a similarity measure defined on $\mathcal{X}$:

$$\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, \quad (\mathbf{x}_1, \mathbf{x}_2) \rightarrow \kappa(\mathbf{x}_1, \mathbf{x}_2),$$

satisfying, $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$,

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \Phi(\mathbf{x}_1) \cdot \Phi(\mathbf{x}_2),$$

where Φ is called as feature map and the similarity measure κ is referred as kernel.

The advantage of kernel is that it allows us to construct algorithms in implicit high dimensional dot product spaces so that the data has a higher chance to be separated by a linear classifier in the feature space than the input space. Well-known examples of such algorithms include support vector machine [Vap98], kernel logistic regression [ZH01], kernel principle component analysis [SSM97], etc.

After a brief introduction, we will introduce some basic definitions for kernels below.

Definition 2.1 *Given the kernel function κ and inputs $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathcal{X}$, the $n \times n$ matrix*

$$K := [\kappa(\mathbf{x}_i, \mathbf{x}_j)]_{ij}$$

is called the Gram matrix (or kernel matrix) of κ w.r.t. $\mathbf{x}_1, \dots, \mathbf{x}_n$.

Definition 2.2 *A real $n \times n$ symmetric matrix K satisfying*

$$\sum_{i,j} c_i c_j K_{ij} \geq 0$$

for all $c_i \in \mathbb{R}$ is called positive semi-definite. If equality in the above inequality only occurs when $c_1 = \dots = c_n = 0$, then we shall call the matrix positive definite.

Definition 2.3 *Let $\mathcal{X}$ be a nonempty set. A function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ which for all $n \in \mathbb{N}, \mathbf{x}_i \in \mathcal{X}, i \in [n]$ gives rise to a positive semi-definite Gram matrix is called a positive semi-definite kernel. A function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ which for all $n \in \mathbb{N}$ and distinct $\mathbf{x}_i \in \mathcal{X}$ gives rise to a positive definite Gram matrix is called a positive definite kernel.*

We simply use kernels for positive definite kernels when no confusion is expected in the context.

2.2.2 Reproducing Kernel Hilbert Space

In this subsection, following the previous introduction to kernels, we introduce the concept of “Reproducing Kernel Hilbert Space” (RKHS), which can be constructed from a kernel κ . In such a space, the solutions of many algorithms have been proven to be existent and unique, for example, the solution of SVMs is existent and unique according to representer theorem.

Given a kernel κ , define the “reproducing kernel feature map” $\Phi : \mathcal{X} \rightarrow \mathbb{R}^{\mathcal{X}}$ as:

$$\Phi(\mathbf{x}) = \kappa(\mathbf{x}, \cdot).$$

Consider the vector space:

$$\text{span}(\{\Phi(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}) = \{f(\cdot) = \sum_{i=1}^n \alpha_i \kappa(\mathbf{x}_i, \cdot) : n \in \mathbb{N}, \mathbf{x}_i \in \mathcal{X}, \alpha_i \in \mathbb{R}\}.$$

For $f = \sum_i \alpha_i \kappa(\mathbf{u}_i, \cdot)$ and $g = \sum_j \beta_j \kappa(\mathbf{v}_j, \cdot)$, define inner product $\langle f, g \rangle = \sum_{i,j} \alpha_i \beta_j \kappa(\mathbf{u}_i, \mathbf{v}_j)$.

Note that: $\langle f, \kappa(\cdot, \mathbf{x}) \rangle = \sum_i \alpha_i \kappa(\mathbf{x}, \mathbf{u}_i) = f(\mathbf{x})$, i.e., κ has the reproducing property. We show that $\langle f, g \rangle$ is an inner product by checking the following conditions:

- (i) Symmetry: $\langle f, g \rangle = \sum_{i,j} \alpha_i \beta_j \kappa(\mathbf{u}_i, \mathbf{v}_j) = \sum_{i,j} \beta_j \alpha_i \kappa(\mathbf{v}_j, \mathbf{u}_i) = \langle g, f \rangle$;
- (ii) Bilinearity: $\langle f, g \rangle = \sum_i \alpha_i g(\mathbf{u}_i) = \sum_j \beta_j f(\mathbf{v}_j)$;
- (iii) Positive definiteness: $\langle f, f \rangle = \alpha K \alpha \geq 0$ with equality iff $f = 0$.

Now, we have defined an inner product space $\langle \cdot, \cdot \rangle$. Given this inner product space, we can further define the norm as $\|f\| = \sqrt{\langle f, f \rangle}$. If the inner product space is complete with respect to the norm, i.e., every Cauchy sequence in this space will converge to a point in it, it is termed as a Hilbert space. However, we will complete it to give the Hilbert space constructed from κ , by adding the limit point of every Cauchy sequence into the space.

Definition 2.4 For an arbitrary set $\mathcal{X}$, and a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \rightarrow \mathbb{R}$, we say $\mathcal{H}$ is a Reproducing Kernel Hilbert Space if $\exists \kappa : \mathcal{X} \rightarrow \mathbb{R}$, s.t.

- (i) κ has the reproducing property, i.e., $f(\mathbf{x}) = \langle f(\cdot), \kappa(\cdot, \mathbf{x}) \rangle$;
- (ii) $\mathcal{H} = \overline{\text{span}\{\kappa(\cdot, \mathbf{x}) : \mathbf{x} \in \mathcal{X}\}}$.

2.2.3 Some Examples

After briefly introducing the basic theories, we would list some kernels, which are widely applied in the real world applications. For example:

- (i) Linear Kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$. This kernel is the inner product in the input space, which is the most trivial one;
- (ii) Polynomial Kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$;
- (iii) Gaussian Kernel: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\}$.

The linear kernel is very effective in text categorization because text data is usually high-dimensional already, while Gaussian kernel is widely used in image data.

2.3 Convex Optimization

To solve the SVMs optimization problem with or without kernels, we need some well-studied tools of convex optimization, since the formulation of SVMs is a special convex optimization problem.

2.3.1 Basics of Convex Optimization

Convex optimization is not only very important for off-line learning problem, but also quite crucial for online learning algorithms. So, we will briefly introduce convex optimization in this section. For more details, please refer to [BV04].

Firstly, consider the following optimization problem (primal problem):

$$p^* = \min_{\mathbf{x} \in \mathbb{R}^d} f_0(\mathbf{x}), \text{ s.t. } f_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m,$$

where $f_0, \dots, f_m : \mathbb{R}^d \rightarrow \mathbb{R}$ are primal convex functions and $p^* = f_0(\mathbf{x}^*)$ is the optimal solution. This formulation actually includes the SVM optimization problem as a special case.

Convex optimization problems, like the above one, are widely used in machine learning algorithms, since the global optimum can be efficiently solved. To solve a convex optimization problem, we can utilize the well-studied optimization tools, some basics of which will be introduced below.

A significant tool for convex optimization is the Lagrangian, which is defined for the above problem as follows:

$$\mathcal{L}(\mathbf{x}, \lambda) = f_0(\mathbf{x}) + \sum_{i=1}^m \lambda_i f_i(\mathbf{x})$$

where $\lambda_i \geq 0$, $i = 1, \dots, m$, are called dual variables or Lagrange multipliers. Two main properties of the Lagrangian are as follows:

- 1 $d^* = \sup_{\lambda \geq 0} \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) \leq \inf_{\mathbf{x}} \sup_{\lambda \geq 0} \mathcal{L}(\mathbf{x}, \lambda) = p^*$, since if the min player gets to play second, a minimax problem can achieve a lower value;
- 2 suppose there is a saddle point $(\mathbf{x}^*, \lambda^*)$, i.e., $\mathcal{L}(\mathbf{x}^*, \lambda) \leq \mathcal{L}(\mathbf{x}^*, \lambda^*) \leq \mathcal{L}(\mathbf{x}, \lambda^*)$ for all feasible $\mathbf{x}$ and $\lambda \geq 0$, then

$$\inf_{\mathbf{x}} \sup_{\lambda \geq 0} \mathcal{L}(\mathbf{x}, \lambda) \leq \sup_{\lambda \geq 0} \mathcal{L}(\mathbf{x}^*, \lambda) = \mathcal{L}(\mathbf{x}^*, \lambda^*) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda^*) \leq \sup_{\lambda \geq 0} \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) \quad (2.5)$$

and the inequalities and equalities in the 2.5 hold because of the definitions of supremum and infimum. As a result, when a saddle point exists, we can guarantee that $\sup_{\lambda \geq 0} \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \inf_{\mathbf{x}} \sup_{\lambda \geq 0} \mathcal{L}(\mathbf{x}, \lambda)$.

After the presentation of Lagrangian, now we can introduce the closely related Lagrange dual function, or dual function in short, compared with the primal functions f_i , $i = 1, \dots, m$, as follows:

$$g(\lambda) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \inf_{\mathbf{x}} (f_0(\mathbf{x}) + \sum_{i=1}^m \lambda_i f_i(\mathbf{x})),$$

which is concave with respect to λ , since it is a point-wise minima of affine functions of λ . In addition, if $\lambda_i \geq 0$ and $\mathbf{x}$ is primal feasible, i.e., $f_i(\mathbf{x}) \leq 0$, then $g(\lambda) \leq f_0(\mathbf{x})$, according to the definition of the Lagrange dual function (in particular, $\forall \lambda \geq 0$, $g(\lambda) \leq p^*$ is a special case).

Given the dual function, the dual (optimization) problem compared with the primal (optimization) problem is

$$d^* = \max_{\lambda} g(\lambda) \quad \text{s.t. } \lambda \geq 0,$$

which is to maximize a concave function with convex constraints.

Between the dual problem and primal problem, there are two basic relations: weak duality and strong duality. Weak duality implies the optimal duality gap $p^* - d^*$ is always nonnegative (i.e., the above first property of Lagrangian), even if the primal problem is not convex. While strong duality states $d^* = p^*$, which holds if f_0 and f_i are convex and there is a suitable qualification on the constraints, for example, Slater's condition requires that the primal is strictly feasible:

$$\exists \mathbf{x}, \quad f_i(\mathbf{x}) < 0, \quad i = 1, \dots, m.$$

After introducing the primal and dual problem, we would present the key components of convex optimization - Karush-Kuhn-Tucker (KKT) conditions, which are usually used

to obtain the closed-form solution. Specifically, if f_0 and f_i are differentiable, $\exists \mathbf{x}^*, \lambda^*$, which are optimal for the primal and dual problems respectively, and the duality gap is zero

$$\Rightarrow KKT(\mathbf{x}^*, \lambda^*) = \begin{cases} f_i(\mathbf{x}^*) \leq 0 & i = 1, \dots, m; \\ \lambda_i^* \geq 0 & i = 1, \dots, m; \\ \lambda_i^* f_i(\mathbf{x}^*) = 0 & i = 1, \dots, m; \\ \nabla f_0(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(\mathbf{x}^*) = 0, \end{cases}$$

while $KKT(\mathbf{x}, \lambda)$ and f_0, f_i are convex imply $(\mathbf{x}, \lambda)$ is optimal, and the duality gap is zero. Furthermore f_0, f_i are convex, differentiable and the duality gap is zero then $KKT(\mathbf{x}, \lambda) \Leftrightarrow (\mathbf{x}, \lambda)$ is optimal.

Finally, we will briefly explain one of the conditions - Complementary Slackness: $\lambda_i^* f_i(\mathbf{x}^*) = 0, \forall i$, since it will facilitate us to present the concepts in the later sections. If there is a zero duality gap, then it is easy to verify that

$$f_0(\mathbf{x}^*) = g(\lambda^*) = \inf_{\mathbf{x}} \left(f_0(\mathbf{x}) + \sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}) \right) \leq f_0(\mathbf{x}^*) + \sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}^*),$$

hence $\sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}^*) \geq 0$. Furthermore, because $f_i(\mathbf{x}^*) \leq 0$ and $\lambda_i^* \geq 0$, as a result $\sum_{i=1}^m \lambda_i^* f_i(\mathbf{x}^*) = 0$ and hence $\lambda_i^* f_i(\mathbf{x}^*) = 0 \quad \forall i$, which implies if constraint i is inactive at $\mathbf{x}^*$ (i.e. $f_i(\mathbf{x}^*) < 0$) then $\lambda_i^* = 0$.

2.3.2 Solution of Linear Separable SVM

After a brief review of convex optimization, let us see how to use it to solve the linear separable SVM as an example, for which the primal problem is (here, we omit the bias b for simplicity)

$$\min_{\mathbf{w}} \frac{\|\mathbf{w}\|^2}{2} \quad \text{s.t. } y_i \mathbf{w} \cdot \mathbf{x}_i \geq 1, \quad i = 1, \dots, m.$$

Given the above primal problem, the Lagrangian is as follows:

$$\mathcal{L}(\mathbf{w}, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^m \alpha_i (1 - y_i \mathbf{w} \cdot \mathbf{x}_i), \quad \alpha_i \geq 0, \quad i = 1, \dots, m.$$

Minimizing the above Lagrangian with respect to $\mathbf{w}$, we could derive the dual function. Because the minimum of the Lagrangian with respect to $\mathbf{w}$ can be achieved through setting the derivative to zero: i.e., $\frac{\partial \mathcal{L}(\mathbf{w}, \alpha)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i = 0$, we can find the optimal $\mathbf{w}$ should be in the form

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, \quad \alpha_i \geq 0, \quad i = 1, \dots, m.$$

The above equation implies the optimal solution $\mathbf{w}^*$ is a linear combination of the training vectors, among which the vector $\mathbf{x}_i$ s corresponding to $\alpha_i \neq 0$ are referred to as support vectors. Further, substituting the above equation into the Lagrangian, we can derive that the dual problem of linear separable SVM is as follows:

$$\max_{\alpha_1, \dots, \alpha_m} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad \alpha_i \geq 0, \quad i = 1, \dots, m,$$

which is a quadratic programming problem and can be solved using the existing tools.

Finally, we would briefly discuss the solution using the previously introduced complementary slackness property:

$$\alpha_i^* (1 - y_i \mathbf{w}^* \cdot \mathbf{x}_i) = 0 \quad i = 1, \dots, m,$$

where, α_i^* , $i = 1, \dots, m$ and $\mathbf{w}^*$ are the optimal solutions for the dual and primal problems, respectively. On one hand, if $\alpha_i^* > 0$, then $1 - y_i \mathbf{w}^* \cdot \mathbf{x}_i = 0$, which implies if a training vector is a support vector, the confidence of the optimal classifier for it, i.e., $|y_i \mathbf{w}^* \cdot \mathbf{x}_i|$, will be 1. On the other hand if $1 - y_i \mathbf{w}^* \cdot \mathbf{x}_i < 0$, then the value of α_i^* must be zero, which means if the training vector $\mathbf{x}_i$ can be correctly classified with confidence larger than 1, it will not be a support vector; In summary, the optimal solution is a linear combination of the vectors $\mathbf{x}_i$ s satisfying $y_i \mathbf{w}^* \cdot \mathbf{x}_i = 1$.

2.4 Online Learning

In this section, we will firstly present the formulation of binary online classification, which is a classical online learning formulation. Then we will introduce the Perceptron [Ros58] Algorithm, which is the most well-known and simplest algorithm for binary online classification. Finally, a detailed summary of the existing work on online learning will be provided.

2.4.1 Formal Formulation

Because supervised binary classification is one of the fundamental online learning problems, so we will begin with introducing the binary classification problem. For instance, the previous spam detection problem in the Section 1.1 of Chapter 1 is a typical binary online classification problem, where the answer to every question is binary: yes or no. To formulate a binary online classification, we need to introduce several definitions. Firstly, we will refer the object provided to be classified, such as the email in the spam detection problem, as instance (sometimes as pattern, case, input, point or sample), and denote the t -th object by a vector $\mathbf{x}_t \in \mathbb{R}^d$. In addition, we will refer the true class for the object,

such as the class for a received email, as label (also as target, output or observation), and denote the two labels as $+1$ and -1 .

After the above preparations, we now formally state the online classification procedure. Formally speaking, the online binary classification problem will takes place in a sequence of rounds with length $T \in \mathbb{N}$, which may not be available in advance. At the t -th step, the instance $\mathbf{x}_t$ is provided to be classified. The online learning procedure will utilize a classifier f_t , which is learnt using the previous examples $(\mathbf{x}_i, y_i)$, $i < t$, to predict the label of $\mathbf{x}_t$ as $f_t(\mathbf{x}_t)$. After the prediction, the true label y_t of $\mathbf{x}_t$ will be offered to the procedure and the learner will suffer a loss ℓ_t , which depends on the difference between the prediction and the true label. Since observing the true label of $\mathbf{x}_t$, the learner could update the prediction function from f_t to f_{t+1} using $(\mathbf{x}_t, y_t)$. Finally, the Online Binary Classification problem is formally summarized in Procedure 1.

Procedure 1 The Online Binary Classification Procedure.

```

Initialize the prediction function as  $f_1$ ;
for  $t = 1, 2, \dots, T$  do
    Receive instance:  $\mathbf{x}_t \in \mathbb{R}^d$ ;
    Predict  $\hat{y}_t = f_t(\mathbf{x}_t)$  as the label of  $\mathbf{x}_t$ ;
    Receive correct label:  $y_t \in \{-1, +1\}$ ;
    Suffer loss:  $\ell_t$ , which depends on the difference between  $f_t(\mathbf{x}_t)$  and  $y_t$ ;
    Update the prediction function  $f_t$  to  $f_{t+1}$ ;
end for

```

On this procedure, the performance measure of online algorithm is cumulative loss $\sum_{t=1}^T \ell_t$ suffered. Many algorithms are proposed to minimize the cumulative loss, among which the Perceptron [Agm54, Ros58, Nov62] algorithm should be the most well-known and simplest one. The perceptron algorithm uses a weight vector $\mathbf{w} \in \mathbb{R}^d$ initialized as 0 for classification, by which the label predicted is the sign of $\mathbf{w} \cdot \mathbf{x}$. After a prediction, if the predicted label is correct, the weight vector will remain unchanged, otherwise the weight vector will be updated by adding the product $y_t \mathbf{x}_t$ to it. As a result, the t -th weight vector is in the form of $\mathbf{w}_t = \sum_{i=1}^{t-1} \mathbb{I}_{(y_i \neq \hat{y}_i)} y_i \mathbf{x}_i$, where $\mathbb{I}_x$ is the indicator function, which equals to 1 if x is valid, otherwise 0. Similar with SVM, the vector $\mathbf{x}_i$ s corresponding to none-zero $\mathbb{I}_{(y_i \neq \hat{y}_i)}$ s are called support vectors (SV). Finally, Perceptron algorithm is summarized as Algorithm 2.

Although the perceptron algorithm is extremely simple, it can guarantee a finite number of mistakes for a linearly separable dataset. Besides this perceptron algorithm, many algorithms have been proposed these years, which can roughly be classified by update type, problem type and aggressive level.

Algorithm 2 The Perceptron Algorithm.

```

Initialize  $\mathbf{w}_1 = 0$ ;
for  $t = 1, 2, \dots, T$  do
    Receive instance:  $\mathbf{x}_t \in \mathbb{R}^d$ ;
    Predict:  $\hat{y}_t = \text{sign}(\mathbf{w}_t \cdot \mathbf{x}_t)$ ;
    Receive correct label:  $y_t \in \{-1, +1\}$ ;
    Suffer loss:  $\ell_t = \mathbb{I}_{(y_t \neq \hat{y}_t)}$ ;
    if  $\hat{y}_t \neq y_t$  then
         $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$ ;
    end if
end for

```

2.4.2 Update Type

Firstly, additive type update is the most popular update, which is adopted by the Perceptron algorithm. Besides the Perceptron algorithm, the Passive Aggressive (PA) algorithm [CDK⁺06] is another well-known additive update type algorithm, which use more aggressive criterion ($y_t \mathbf{w}_t \cdot \mathbf{x}_t \leq 1$) and more adaptive weight ($\frac{\max(0, 1 - y_t \mathbf{w}_t \cdot \mathbf{x}_t)}{\|\mathbf{x}_t\|^2}$) for the instance added to weight vector. Furthermore, Yoav Freund and Robert E. Schapire proposed large margin classification using the perceptron algorithm [FS99], which uses a combination of the weight vectors learnt during the whole process. In addition, ROMMA [LL99] and MIRA [CS03] are another two additive type update algorithms, which use maximal margin as the update criterion. Finally, Ballseptron [SSS05], a generalization of the Perceptron algorithm, is proposed by Shai Shalev-Shwartz and Yoram Singer, which will update the weight vector for margin error ($y_t(\mathbf{w}_t \cdot \mathbf{x}_t)/\|\mathbf{w}_t\| \leq 1$).

Besides additive type, there exist various multiplicative update methods these years, among which Weighted Majority [LW86], Winnow [Lit87] and Exponentiated Gradient [KW97] are several well-known multiplicative algorithms. Firstly, Weighted Majority Algorithm (WMA) is a meta-learning algorithm used to learn a prediction method from the combinations of several expert algorithms, which can be any learning algorithms, or even human experts. Secondly, the winnow algorithm learns linear threshold hypotheses, which is considered as a counterpart of perceptron. Finally, Exponentiated Gradient (EG) is inspired by the gradient descent algorithm, which is more effective than additive update type for the situation that the irrelevant attributes in the instance vector abound.

In addition, Claudio Gentile and Nick Littlestone proposed p -norm algorithms [Gen03], which are very robust and interpolate between the additive and multiplicative update type. Besides, the algorithm ALMA _{p} [Gen01], also proposed for p -norm, is motivated by maximizing margin.

Although many algorithms are proposed with different update types, at the same time, many researchers try to provide some frameworks to unify those seemingly different

algorithms, among which, one of best frameworks is the quasi-additive family [GLS97] proposed by Adam J. Grove, Nick Littlestone and Dale Schuurmans, which includes the winnow and perceptron as two special cases. Besides, a similar unified view for regression was derived by Kivinen and Warmuth [KW01]. Cesa-Bianchi and Lugosi proposed a potential based gradient descent framework [CBL06], which includes the gradient descent and EG algorithms as special cases. Recently the state-of-the-art Primal-Dual framework [SSS06b, SSS07b] for online learning is proposed by Shai Shalev Shwartz and Yoram Singer. In their proposed framework, the primal mistake rate descent problem can be converted to a corresponding dual value ascent problem, under which it is easy to interpret many well-known existing algorithms, such as, the Perceptron algorithm, the online Passive Aggressive (PA) algorithms, Winnow, p -norm algorithm, etc.

2.4.3 Problem Type

Binary classification is the most popular problem type, for which the Perceptron algorithm is originally designed; while multi-class classification is another important problem, for which there are many algorithms. Among the existing algorithms, Koby Crammer and Yoram Singer proposed ultraconservative online algorithms for multiclass problems [CS03], which maintains one weight vector for each class. In addition, the Passive-Aggressive algorithm [CDK⁺06] mentioned before is also generalized to the multi-class problem. Furthermore, Michael Fink, Shai Shalev-Shwartz, Yoram Singer and Shimon Ullman proposed a framework for multi-class learning through hypothesis sharing [FSSSU06], which can deal with the setting where the number of labels is not known in advance but revealed during the learning goes.

Besides online multi-class classification, the online category ranking problem is another important research topic. Category ranking is the task of ordering labels with respect to their relevances to an input instance. For online ranking, Koby Crammer and Yoram Singer proposed a family of algorithms called MMP [CS02], which are descendants of the Perceptron algorithm. Later, Koby Crammer and Yoram Singer proposed an additive and a multiplicative algorithm [CS05], for online category ranking, in which the updates are derived by casting a constrained optimization for each new instance. Furthermore, Shai Shalev-Shwartz and Yoram Singer proposed a unified algorithmic approach for efficient online label ranking [SSS07c], which is motivated by the Primal-Dual framework [SS07].

In addition, online metric learning is also a well-studied research topic, which tries to learn a metric matrix which can predict the distance between two instances. On this topic, the first algorithm POLA is by Shai Shalev-Shwartz, Yoram Singer and Andrew Y. Ng [SSSN04], which is motivated by the Passive-Aggressive algorithm, however one drawback of POLA is the need of projection, since it is quite slow. Later, Jason V. Davis et al. proposed a new formulation for online metric learning [DKJ⁺07], which is

motivated by one paper [KW97] of Kivinen and Warmuth. Finally, Pratteek Jain et al. present a new online metric learning algorithm that updates a learned Mahalanobis metric based on LogDet regularization and gradient descent [JKDG08], which enjoys the advantage of closed-form and semi-definite matrix solution.

Another topic for online learning is learning with a budget or budget online learning, which would pre-fix a budget for the number of stored SVs. The reason for budget online learning is that when non-linear kernel is adopted for online learning, the computation time and memory cost are proportional to the number of support vectors, since the value of $f(\mathbf{x})$ is determined by the sum of $\alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x})$, $\forall i$, so learning with a budget tries to use as less SVs as possible and at the same time maximize the performance. This topic is firstly proposed by Koby Crammer, Jaz Kandola and Yoram Singer [CKS03], which will remove the SV achieving maximal confidence (i.e., $y_i f(\mathbf{x}_i)$) under a modified classifier, the current classifier minus the SV to be removed. Soon, [WB05] proposed to remove a SV, whose removal will minimize the accumulative loss on all the previous data by a modified classifier, the current classifier minus the SV to be removed. In addition, Forgetron algorithm [DSSS05] and Randomized Budget Perceptron algorithm [CCBG07] are another two well-done work for the learning with budget. The Forgetron [DSSS05] is perhaps the first approach for budget online learning that offers a theoretical bound of the total number of mistakes. In particular, at each iteration, if the online classifier makes a mistake, it conducts a three-step updating: (i) it first runs the standard Perceptron update [Ros58] for updating the prediction function; (ii) it then shrinks the weights of support vectors by a carefully chosen scaling factor; and (iii) it finally removes the support vector with the least weight. The Randomized Budget Perceptron algorithm just removes one SV randomly. Another recent work for learning with budget is the Projectron algorithm [OKC08], which will project the kernel feature of the removed SV into the space spanned by the kernel features of the left SV.

All the settings above will request the label of every instance, which limit the applications of those algorithms designed there. Since in reality, the cost of labeling the instance may be very high, while the labels of some instances may be uninformative. To solve this issue, many active online learning algorithms have been proposed to online actively request labels of those most informative instances, so that the cost of labeling is reduced. According to the criteria for requesting labels, these algorithms can be roughly assigned into two groups: confidence based algorithms [CBCG03, CBGZ04, CBGZ06, VCB08, CCBG11] and distribution based algorithms [CBGO09, DGS10, OCB11]. Confidence based algorithms will use the absolute value of the prediction to decide whether to request the label, where the probability of requesting label is always inversely proportional to the absolute value. While distribution based algorithms will use the correlation between the instance and the distribution stored by the SVs to determine whether to ask

for the labels, where the probability of requesting label is always inversely proportional to the correlation.

In all the previous settings, the information of the label is fully offered, however it could only be partially obtained in some real applications. Formally, if only partial information of the true label is offered for the multi-class classification, this problem is called bandit online multi-class classification. The first algorithm tries to solve this problem is the Banditron [KSST08], which extends the regular multi-class Perceptron to tackle the partial feedback issue. While this Banditron has many good properties, such as, high efficiency, it only guarantees a expected mistake $\mathbb{E}[\sum_{t=1}^T \mathbb{I}_{y_t \neq \hat{y}_t}]$ bound of order $O(T^{2/3}) + \min_{\mathbf{w}} \sum_{t=1}^T \max(0, 1 - y_t \mathbf{w} \cdot \mathbf{x}_t)$. Later, a framework of bandit online multi-class classification is proposed in [WJV10], which is adapted from the potential based gradient framework, includes the Banditron as a special case and derives the Exponential Gradient Algorithm for online bandit multi-class learning. Further, a new algorithm [CG11] is proposed based on the 2nd-order Perceptron and upper-confidence bounds to trade off exploration and exploitation, which can guarantee a better regret bound of order $O(\sqrt{T} \log(T))$. More recently, the NEWTRON algorithm [HK11] for log-loss is proposed based on online Newton method, which can guarantees a even better regret bound of order $O(\log(T))$.

Besides the above specific problems, there exists a more general framework called as Online Convex Optimization (OCO), which is formally formulated in Procedure 3.

Procedure 3 The Online Convex Optimization (OCO) Game.

```

for  $t = 1, 2, \dots, T$  do
    Player choose  $\mathbf{w}_t \in \mathbf{S} \subseteq \mathbb{R}^d$ ;
    Adversary chooses  $\ell_t \in \mathfrak{F}$ , which is a set of loss functions;
    Player suffers loss  $\ell_t(\mathbf{w}_t)$ ;
end for

```

This OCO is firstly proposed and studied by Martin Zinkevich [Zin03], who uses the projection of the gradient descent solution and guarantees a square root regret, i.e., $R_T \leq O(\sqrt{T})$, where $R_T = \sum_{t=1}^T \ell_t(\mathbf{w}_t) - \min_{\mathbf{w} \in \mathbf{S}} \sum_{t=1}^T \ell_t(\mathbf{w})$. Later, Elad Hazan et al. [HKKA06] present an algorithm that achieves log regret, i.e., $R_T \leq O(\log(T))$ for an arbitrary sequence of strictly convex functions, but the gain is from using the second order differential information which will cost more computation. Further, Adam Kalai and Santosh Vempala proposed two geometric algorithms for online convex optimization: follow the expected leader and follow the lazy leader [KV05], which can guarantee log regret. Furthermore, Peter L. Bartlett, Elad Hazan and Alexander Rakhlin [BHR07] proposed adaptive online gradient descent achieving intermediate rates between $\sqrt{T}$ and $\log(T)$. In addition, Jacob Abernethy et al. [ABRT08] analyze online convex optimization

from a minimax perspective, proving minimax strategies and lower bounds, where the existing algorithms are proven essentially optimal. Shai Shalev-Shwartz and Yoram Singer proposed a framework [SSS06a, SSK08] for online convex optimization, which is essentially a primal-dual framework and can guarantee a $O(\sqrt{T})$ regret. Besides the last framework, they also proposed a family of prediction algorithms using Bregman divergence as progress measure for online strongly convex optimization [SsS07a], which can provide logarithmic regret.

For all the previous online learning settings, there is no any distribution assumption on the datasets, which is different from the traditional batch learning setting, where it is assumed that the examples are independently identically distributed (i.i.d.) from some potential distribution. If we further assume the examples are i.i.d from some potential distribution, then the generalization ability of a classifier is the probability of making a correct prediction on a random example from the real distribution by it. The generalization ability of online algorithm is also a very interesting research topic. Existing work for generalization ability includes [BL03, CBCG04, BB07, CBG08, KT08]. One set of work [CBCG04, CBG08, KT08] is to use the hypotheses generated by online learning algorithm to construct some hypothesis with high generalization ability, which can be formally considered as online to batch conversation, where it is nontrivial to determine the termination condition for online methods in order to achieve the optimal batch-mode prediction performance. The other set of work [BL03, BB07] focuses more on the generalization ability of the online hypothesis, given the training data is abundant, and excitingly concludes that suitably designed online learning algorithms asymptotically outperform any batch learning algorithm.

Although being extensively studied, one drawback of online learning is that the parameters in many algorithms need be tuned manually, so how to tune the parameters automatically is an interesting problem. As a trial, Peter Auer et al. proposed adaptive and self-confident online learning algorithms [ACBG02], which can tune the parameters by themselves.

Besides the above problems for online setting, online learning is also used for solving large scale offline (or batch) problem, such as support vector machine (SVM). One of well-known online algorithms for solving SVM is Pegasos [SSSS07] by Shai Shalev-Shwartz, Yoram Singer and Nathan Srebro.

All the problems described above only involve one task for every step, while online multiple tasks learning is also proposed, under the assumption that learning multiple related tasks simultaneously will be advantageous compared with learning every single task independently. As we know, the first work on this topic is done by Ofer Dekel, Philip M. Long, and Yoram Singer [DLS06], where they combine multiple tasks together using a global loss which is a function of all the single losses. Later, Jacob Abernethy,

Peter Bartlett, and Alexander Rakhlin proposed the online multitask learning with expert advices [ABR07], where several tasks are presented to the forecaster simultaneously. Furthermore, Giovanni Cavallanti et al. formalized task relatedness in different ways [CCBG08], derived formal guarantees on the performance advantage and proposed the multitask perceptron algorithm. Finally, Gábor Lugosi et al. discuss online multitask learning with hard constraints [LPS09], where the main difficulty is the requirement that the decision maker should be chosen from a restricted subset of actions.

2.4.4 Aggressive Level

Perceptron adopts the simplest updating strategy. Specifically, it updates the weight vector only when a mistake occurs, and the weights for the support vectors are always set to 1. In addition, only the newest instance is used to update the weight vector.

Passive-Aggressive (PA) algorithms [CDK⁺06] are more aggressive, since they not only update the weight vector when mistakes occur, but also when the confidence is not large enough. In addition, PA will use different weights for different support vectors depending on the loss of the prediction. Besides PA, many other works exist for learning adaptive weights and updates criterion [Her01, Gen01, LL99, KSW04, SSS05, CS03].

All of the algorithms mentioned above in this subsection only use the first order information of the previous data. More aggressive algorithms will use the second order information, which have shown that parameter confidence information can be explored to guide and improve online learning performance [CBCG05]. As we know, V. Vovk proposed the first online linear regression algorithm [Vov97] utilizing the second order information of previous data. Later, the second order Perceptron [CBCG05] algorithm is proposed by Nicolò Cesa-Bianchi et al. for online classification. More recently, we have seen a surge of studies on the second-order online learning algorithms [DCP08, CKD09, OC10, DHS11]. Confidence-weighted (CW) learning [DCP08, CDK09] maintains a Gaussian distribution over some linear classifier hypotheses and applies it to control the direction and scale of parameter updates [DCP08]. Although CW learning has formal guarantees in the mistake-bound model [CDP08], it can overfit in certain situations due to its aggressive update rules based upon a separable data assumption. Recently, an improved online algorithm, i.e., Adaptive Regularization of Weights (AROW) [CKD09, OC10], relaxes such separable assumption by employing an adaptive regularization for each training example based upon its current confidence. This regularization comes in the form of minimizing a combination of the Kullback-Leibler divergence between Gaussian distributed weight vectors and a confidence penalty of vectors. Although AROW is able to improve the original CW learning by handling noisy and non-separable cases, it is not the exact corresponding soft extending part of CW (Like PA with PA-I and PA-II). In particular, the directly added loss and confidence regularization make AROW lose an important

property of Confidence-weighted learning, i.e., the Adaptive Margin property. Following the similar idea of soft margin support vector machines, the adaptive margin assigns different margins for different instances via a probability formulation, which enables CW to gain extra efficiency and effectiveness. To tackle this issue, some researchers have further proposed the soft confidence weighted learning algorithms [WZH12].

2.5 Summary

In this chapter, we firstly introduced support vector machines, which is an effective classification technique originally introduced by Vapnik and Lerner [VL63] (under a different name). SVM finds the maximum margin hyperplane at once by solving an optimization problem defined over the entire sequence of examples. In their modern form, SVMs were introduced by Boser, Guyon, and Vapnik [BGV92] and Cortes and Vapnik [CV95]. See the monographs Cristianini and Shawe-Taylor [CST00], Schölkopf and Smola [SS01], Vapnik [Vap98], Steinwart [SC08] for extensive accounts on the theory of SVM.

After the review of support vector machine, a brief introduction to kernel function was provided. Kernel function is a kind of generalized similarity measure, which is used to attack the problem where the dataset is highly non-separable. With kernel, the concept of Reproducing Kernel Hilbert Space was introduced, which is a function space spanned by the set $\{\kappa(\mathbf{x}, \cdot), \forall \mathbf{x}\}$. The study of reproducing kernel Hilbert spaces was developed by Aronszajn [Aro50] in the 1940's. The use of kernels has been introduced in learning since 1964 with the influential work of Aizerman, Braverman and Rozonoer [ABR64] and Bashkirov, Braverman, and Muchnik [OBM64].

Because many machine learning problems are formulated as convex optimization problems, the tools and theory of convex optimization play an important role in machine learning. So, an extremely brief review was made to cover the convex optimization. Where the basic theories about primal problem, dual problem, and KKT condition are all introduced. For more detailed explanation, please refer to the book by Stephen Boyd [BV04].

Finally, a brief summary of online learning was presented. Online learning is more suitable than batch method in some situation, where the data arrives sequentially while predictions are required immediately, examples include spam filtering and stock market prediction. According to the problem of spam filtering, we presented the framework of online learning and one of the best-known algorithms: Perceptron. Finally, a detailed summary of online learning was summarized from three perspectives: update type, problem type and aggressiveness level.

Chapter 3

Binary Double Updating Online Learning

Most kernel based online binary classification algorithms [Ros58, FS99, KSW04, CDK⁺06] work by assigning a fixed weight to a new example when it is misclassified, and the assigned weight remains unchanged for the rest of the learning process. This is clearly insufficient because when a new example is added to the pool of support vectors, we expect that it is better to update the weights assigned to the existing support vectors.

In this chapter, we extend some well-known algorithms to explicitly address this problem. It is designed to dynamically update the example weights in order to improve the classification accuracy as well as the mistake bound. In any trial of online learning, besides assigning a weight to the misclassified example that is received in current trial, the proposed online learning algorithm also updates the weight for one of the existing support vectors, leading to the example weights that are dynamically updated as learning goes. We refer to the proposed approach as **Double Updating Online Learning**, or **DUOL** for short.

We emphasize that although several online algorithms are proposed to update the example weights as the learning process goes, most of them are designed for different problem settings. For instance, in [OKC08, CKS03, DSSS08], online learning algorithms are proposed to adjust the example weights in order to fit in the constraint of fixed number of support vectors; in [LW94], example weights are adjusted to track the drifting concepts.

Besides, we notice that [BEWB05] proposed an algorithm LASVM, which in principle can be viewed as the first double updating online learning algorithm, although they do not explicitly claim this. But its motivation of conducting double updating is to guarantee the feasibility of the online solution, where the specific strategy is quite different with ours. Furthermore, this algorithm has to conduct double updating twice one iteration, so that it is computationally extremely intensive. More importantly, this algorithm has no theoretical online mistake bound.

The key challenge in the proposed online learning approach is to decide which existing support vector should be selected for weight updating besides the new misclassified example. An intuitive choice is to select the existing support vector that “conflict” with the new example, i.e., the existing support vector that on one hand shares similar input pattern as the new example and on the other hand belongs to a class different from that of the new example. In order to quantitatively analyze the impact of updating the weight for such an existing support vector, we employ an analysis that is based on the recent work of online convex programming by incremental dual ascent [SSS06b, SSS07b]. Our analysis shows that under certain conditions, the proposed online learning algorithm can significantly reduce the mistake bound of the existing online algorithms.

The rest of this chapter is organized as follows. Section 3.1 briefly discusses related work. Section 3.2 first introduces the problem setting, motivation, theoretical analysis and then the details of the double updating online learning algorithm for binary classification. Section 3.3 extensively evaluates the proposed algorithm on several benchmark datasets. Section 3.4 summarizes this chapter.

3.1 Related Work

Most existing online learning algorithms [Ros58, CS03, CBCG04, CDK⁺06, FSSSU06] only update the weight of the newly added support vector, and keep the weights of the existing support vectors unchanged. This constraint could significantly limit the effectiveness of online learning.

Besides regular online learning, several algorithms are proposed for budget online learning [WB05, CKS03, CCBG07, DSSS08]. The key idea of these algorithms is to dynamically update the weights of the existing support vectors as a new support vector is added, and the support vector with the least weight will be discarded when the number of support vectors exceeds the budget. The idea of discarding support vectors is also used in studies [KSW04] and [CVS⁺06]. In a very recently proposed method [OKC08], a new “projection” approach is proposed for online learning that ensures the number of support vectors is bounded. Besides, in [LW94], an online learning algorithm is proposed to handle the drifting concept, in which the weights of the existing support vectors are reduced whenever a new support vector is added. Although these online learning algorithms are capable of dynamically adjusting the weights of support vectors, they are designed to either fit in the budget of the number of support vectors or to handle drifting concepts, whose problem settings are quite different from ours.

In addition, the paper [STY07] proposed an online classification algorithm, which could update the weights of multiple instances at every step. However, their goal is make the online solution converge to the optimal solution under the condition the number of

noise examples is finite, while our goal is to minimize the number of cumulative online mistakes, without any requirement on the number of noises.

Besides, we notice that [BEWB05, BBGW07] proposed some online SVM solver, named as “LASVM”, that updates two weights at the same time, which is somewhat similar to our idea. However, there are three key differences between this algorithm and ours: 1) the two kinds of example pairs are selected by different motivations. In LASVM, they have to select two instances to guarantee the feasibility of the online solution. For our algorithm, we select two instances to extend single update algorithms and achieve more greater dual ascent than single update. 2) LASVM will give up some updates when there are no sufficient violating pairs, while our approach at least makes single updates when there is no auxiliary example for double updates. 3) LASVM will firstly add every example into the SVs set and then perform Sequential Minimal Optimization (SMO) [?] twice each iteration, while our algorithm only adds the example which suffers non-zero loss into the SVs set and then performs one double update or single update. This difference will make the LASVM more aggressive and effective but computationally extremely intensive as will be revealed in our study. In summary, LASVM can be viewed as a reduction of the SMO algorithm by reducing from full update (batch SVM) to double update, while DUOL is an enhancement of Passive-Aggressive [CDK⁺06] (single update) by improving from single update to double update. Finally, a pure online version of the detailed algorithm of LASVM is provided in Appendix A, such that the non-expert readers can easily find the differences between DUOL and LASVM.

The proposed online learning algorithm is closely related to some recent work of online convex programming by incremental dual ascent [SSS06b, SSS07b]. Although the idea of simultaneously updating the weights of multiple support vectors was mentioned in [SSS06b, SSS07b], neither efficient updating algorithm nor theoretical result was explicitly given in their work. Our work comprehensively solves this problem by presenting a novel effective algorithm and deriving the mistake bounds explicitly. Finally, we note that there were also some existing studies that either suggesting incrementally updating the weights of all existing support vectors [CP00] or propose to update all the support vector weights by using second-order information. However, since these methods often have to perform a full update for every example, their computational costs are often highly intensive, which limits their applications to online learning tasks.

3.2 Binary Double Updating Online Learning

3.2.1 Problem Setting

We consider the problem of online binary classification. For online binary classification, we need to learn a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ based on a sequence of training examples

$\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)\}$, where $\mathbf{x}_t \in \mathbb{R}^d$ is called *instance* and $y_t \in \mathcal{Y} = \{-1, +1\}$ is called *label*, $\forall t \in \{1, \dots, T\}$, where T may be not available in advance. We will use $\text{sign}(f(\mathbf{x}))$ as the prediction of the algorithm, and treat the value $|f(\mathbf{x})|$ as the confidence of the prediction. Moreover we assume that there exists a loss function $\ell(f(\mathbf{x}), y) : \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$, which penalises the deviation of estimates $f(\mathbf{x})$ from observed labels y . We call the output f of the learning algorithm a *hypothesis* and denote the set of all possible hypotheses by $\mathcal{H}$.

In this chapter, we will always assume $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (**RKHS**) with a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ [SS01]. For the **RKHS** $\mathcal{H}$ depends on a specific kernel κ , we will use $\mathcal{H}_\kappa$ instead of $\mathcal{H}$ below. Throughout the analysis, we assume $\kappa(\mathbf{x}, \mathbf{x}) \leq 1$ for any example $\mathbf{x}$.

Given these assumptions, the goal of online binary classification is to minimize the number of online mistakes $\sum_{t=1}^T \mathbb{I}_{(y_t \neq \hat{y}_t)}$.

3.2.2 Motivation

We consider trial t in an online learning task where the training example $(\mathbf{x}_a, y_a)$ is misclassified (i.e., $y_a f(\mathbf{x}_a) \leq 0$). Let $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, \dots, n\}$ be the collection of n misclassified examples received before the trial t . We also refer to these misclassified training examples as “support vectors”. We denote by $\alpha = (\alpha_1, \dots, \alpha_n) \in (0, C]^n$ the weights assigned to the support vectors in $\mathcal{D}$, where C is a predefined constant. The resulting classifier, denoted by $f(\mathbf{x})$, is given by

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i).$$

In the conventional approach to online learning [Ros58, CDK⁺06], we simply assign a constant weight, denoted by $\beta \in (0, C]$, to $(\mathbf{x}_a, y_a)$, and the resulting classifier becomes

$$f'(\mathbf{x}) = \beta y_a \kappa(\mathbf{x}, \mathbf{x}_a) + \sum_{i=1}^n \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i) = \beta y_a \kappa(\mathbf{x}, \mathbf{x}_a) + f(\mathbf{x}).$$

The shortcoming of the conventional online learning approach is that the introduction of the new support vector $(\mathbf{x}_a, y_a)$ may harm the classification of existing support vectors in $\mathcal{D}$, which is revealed by the following proposition.

Proposition 3.1: *Let $(\mathbf{x}_a, y_a)$ be an example misclassified by the current classifier $f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i)$ with $\alpha_i \geq 0, i = 1, \dots, n$, that is, $y_a f(\mathbf{x}_a) < 0$. Let $f'(\mathbf{x}) = \beta y_a \kappa(\mathbf{x}, \mathbf{x}_a) + f(\mathbf{x})$ be the updated classifier with $\beta > 0$. There exists at least one support vector $\mathbf{x}_i \in \mathcal{D}$ such that $y_i f(\mathbf{x}_i) > y_i f'(\mathbf{x}_i)$.*

Proof: The claim follows from the fact that: $\exists \mathbf{x}_i \in \mathcal{D}, y_i y_a \kappa(\mathbf{x}_i, \mathbf{x}_a) < 0$ when $y_a f(\mathbf{x}_a) < 0$. $\square$

As indicated by Proposition 3.1, when a misclassified example $(\mathbf{x}_a, y_a)$ is added to the classifier, the classification confidence of at least one existing support vector will be reduced. When $y_a f(\mathbf{x}_a) \leq -\gamma$, there exists one support vector $(\mathbf{x}_b, y_b) \in \mathcal{D}$ that satisfies $\beta y_a y_b k(\mathbf{x}_a, \mathbf{x}_b) \leq -\beta\gamma/n$. This support vector will be misclassified by the updated classifier $f'(\mathbf{x})$ if $y_b f(\mathbf{x}_b) \leq \beta\gamma/n$. In order to alleviate this problem, we propose to update the weight for the existing support vector whose classification confidence is significantly affected by the new misclassified example. In particular, we consider a support vector $(\mathbf{x}_b, y_b) \in \mathcal{D}$ for weight updating if it satisfies the following two conditions

- $y_b f(\mathbf{x}_b) \leq 0$, that is, support vector $(\mathbf{x}_b, y_b)$ is misclassified by the current classifier $f(\mathbf{x})$,
- $k(\mathbf{x}_b, \mathbf{x}_a) y_a y_b \leq -\rho$ where $\rho \in (0, 1)$ is a predetermined threshold, that is, support vector $(\mathbf{x}_b, y_b)$ “**conflicts**” with the new misclassified example $(\mathbf{x}_a, y_a)$.

We refer to the support vector satisfying the above conditions as an **auxiliary example**. It is clear that by adding the misclassified example $(\mathbf{x}_a, y_a)$ to classifier $f(\mathbf{x})$ with weight β , the classification score of $(\mathbf{x}_b, y_b)$ will be reduced by at least $\beta\rho$, which could lead to a significant misclassification of the auxiliary example $(\mathbf{x}_b, y_b)$. To avoid such a mistake, we propose to update the weights for both $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$ simultaneously. In the next section, we show the details of the double updating algorithm for online learning, and the analysis for mistake bound.

Following the previous work [SSS06b, SSS07b], the online learning algorithms, such as Perceptron, are interpreted as efficient updating rules for maximizing the dual objective function of SVM defined on the examples observed so far [SSS06b, SSS07b].

More specifically, we introduce Theorem 1 in [SSS06b], as follows:

Theorem 1 Assume for an online learning algorithm $\mathcal{A}$, the dual ascent Δ_t is

$$\Delta_t = \mathcal{D}_t(\gamma_1^t, \dots, \gamma_{t-1}^t, \gamma_t^t) - \mathcal{D}_{t-1}(\gamma_1^{t-1}, \dots, \gamma_{t-1}^{t-1}),$$

where $\mathcal{D}_t(\gamma_1, \dots, \gamma_t)$ is the dual objective of $\mathcal{P}_t(f) = \frac{1}{2}\|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^t \ell(f(\mathbf{x}_i), y_i)$. Then the number of mistakes M made by $\mathcal{A}$ when trained over a sequence of examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ (where T may be not available in advance), is upper bounded by

$$M \leq \frac{1}{\Delta} \min_{f \in \mathcal{H}_\kappa} \left(\frac{1}{2}\|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^T \ell(f(\mathbf{x}_i), y_i) \right),$$

where $\Delta = \frac{1}{M} \sum_{t=1}^T \Delta_t$.

Proof: According to weak duality,

$$M * \Delta \leq \sum_{t=1}^T \Delta_t = \mathcal{D}_T(\gamma_1^T, \dots, \gamma_T^T) \leq \min_{f \in \mathcal{H}_\kappa} \left(\frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^T \ell(f(\mathbf{x}_i), y_i) \right).$$

Rearranging the above inequality concludes the theorem.

According to [SSS06b, SSS07b], the bounding constant $\Delta = 1/2$ when we only update the classifier with the newly misclassified example. In our analysis, we will show that Δ can be significantly improved when updating the weights for both the misclassified example and the auxiliary example.

For the remaining part of this section, we denote by $(\mathbf{x}_b, y_b)$ an auxiliary example that satisfies the two conditions specified before. We define

$$k_a = \kappa(\mathbf{x}_a, \mathbf{x}_a), \quad k_b = \kappa(\mathbf{x}_b, \mathbf{x}_b), \quad k_{ab} = \kappa(\mathbf{x}_a, \mathbf{x}_b), \quad w_{ab} = y_a y_b k_{ab}.$$

According to the assumption of auxiliary example, we have $w_{ab} = k_{ab} y_a y_b \leq -\rho$. Finally, we denote by $\hat{\gamma}_b$ the weight for the auxiliary example $(\mathbf{x}_b, y_b)$ that is used in the current classifier $f(\mathbf{x})$, by γ_a and γ_b the updated weights for $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$, respectively, and by d_{γ_b} the difference $\gamma_b - \hat{\gamma}_b$. In addition, we will adopt the hinge loss function $\ell(f(\mathbf{x}), y) = \max(0, 1 - yf(\mathbf{x}))$ for binary classification in this chapter.

3.2.3 Double Updating Online Learning

Recall an auxiliary example $(\mathbf{x}_b, y_b)$ should satisfy two conditions (I) $y_b f(\mathbf{x}_b) \leq 0$, and (II) $w_{ab} \leq -\rho$. In addition, the example $(\mathbf{x}_a, y_a)$ received in the current iteration t is misclassified, that is, $y_a f(\mathbf{x}_a) \leq 0$. Following the framework of dual formulation for online learning, the following lemma shows how to compute Δ_t , that is, the improvement in the objective function of dual SVM by adjusting weights for $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$.

Lemma 3.1 *The maximal improvement in the objective function of dual SVM by adjusting weights for $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$, denoted by Δ_t , is computed by solving the following optimization problem (which is a special case of the optimization problem (28) in [SSS06b]):*

$$\Delta_t = \max_{\gamma_a, d_{\gamma_b}} \{h(\gamma_a, d_{\gamma_b}) : 0 \leq \gamma_a \leq C, -\hat{\gamma}_b \leq d_{\gamma_b} \leq C - \hat{\gamma}_b\}, \quad (3.1)$$

where

$$h(\gamma_a, d_{\gamma_b}) = \gamma_a(1 - y_a f(\mathbf{x}_a)) + d_{\gamma_b}(1 - y_b f(\mathbf{x}_b)) - \frac{k_a}{2} \gamma_a^2 - \frac{k_b}{2} d_{\gamma_b}^2 - w_{ab} \gamma_a d_{\gamma_b}. \quad (3.2)$$

Proof: The dual objective of $\mathcal{P}_t(f) = \frac{1}{2}\|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^t \ell(f(\mathbf{x}_i), y_i)$, $C > 0$ is

$$\mathcal{D}_t(\gamma_1, \dots, \gamma_t) = \sum_{i=1}^t \gamma_i - \frac{1}{2} \left\| \sum_{i=1}^t \gamma_i y_i \kappa(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}_\kappa}^2, \quad \gamma_i \in [0, C],$$

where the relation between f and γ_i , $i = 1, \dots, t$ is $f(\cdot) = \sum_{i=1}^t \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)$.

The value of $h(\gamma_a, d_{\gamma_b})$ can be derived as follows:

$$\begin{aligned} h(\gamma_a, d_{\gamma_b}) &= \mathcal{D}_t(\gamma_1, \dots, \gamma_{b-1}, \hat{\gamma}_b + d_{\gamma_b}, \gamma_{b+1}, \dots, \gamma_{t-1}, \gamma_a) - \mathcal{D}_{t-1}(\gamma_1, \dots, \gamma_{b-1}, \hat{\gamma}_b, \gamma_{b+1}, \dots, \gamma_{t-1}) \\ &= d_{\gamma_b} + \gamma_a - \frac{1}{2} \|f(\cdot) + y_b d_{\gamma_b} \kappa(\mathbf{x}_b, \cdot) + y_a \gamma_a \kappa(\mathbf{x}_a, \cdot)\|_{\mathcal{H}_\kappa}^2 + \frac{1}{2} \|f(\cdot)\|_{\mathcal{H}_\kappa}^2 \\ &= \gamma_a(1 - y_a f(\mathbf{x}_a)) + d_{\gamma_b}(1 - y_b f(\mathbf{x}_b)) - \frac{k_a}{2} \gamma_a^2 - \frac{k_b}{2} d_{\gamma_b}^2 - w_{ab} \gamma_a d_{\gamma_b}. \end{aligned}$$

The theorem below bounds the bounding constant Δ when C is sufficiently large.

Theorem 2 Assume $C \geq \hat{\gamma}_b + 1/(1 - \rho)$ with $\rho \in [0, 1)$ for the selected auxiliary example $(\mathbf{x}_b, y_b)$. Then we have the following bound for the bounding constant Δ :

$$\Delta \geq \frac{1}{1 - \rho}.$$

Proof: First, we show that $d_{\gamma_b} \geq 0$. The reason is that for given $\gamma_a \geq 0$, the optimal solution for d_{γ_b} , given by

$$d_{\gamma_b} = \frac{1 - y_b f(\mathbf{x}_b) - w_{ab} \gamma_a}{k_b},$$

is positive because $y_b f(\mathbf{x}_b) \leq 0$ and $w_{ab} \leq -\rho$. Using the fact $k_a, k_b \leq 1$, $\gamma_a, d_{\gamma_b} \geq 0$, $y_a f(\mathbf{x}_a) \leq 0$, $y_b f(\mathbf{x}_b) \leq 0$, and $w_{a,b} \leq -\rho$, we have

$$h(\gamma_a, d_{\gamma_b}) \geq \gamma_a + d_{\gamma_b} - \frac{1}{2} \gamma_a^2 - \frac{1}{2} d_{\gamma_b}^2 + \rho \gamma_a d_{\gamma_b}.$$

Thus, Δ is bounded as

$$\Delta \geq \max_{\gamma_a \in [0, C], d_{\gamma_b} \in [0, C - \hat{\gamma}_b]} \gamma_a + d_{\gamma_b} - \frac{1}{2} (\gamma_a^2 + d_{\gamma_b}^2) + \rho \gamma_a d_{\gamma_b}.$$

Under the condition that $C \geq \hat{\gamma}_b + 1/(1 - \rho)$, it is easy to verify that the optimal solution for the above problem is $\gamma_a = d_{\gamma_b} = 1/(1 - \rho)$, which leads to the result in the theorem.

We refer to the case as a **strong double update** when the condition of Theorem 2 is satisfied. We have the following theorem for the general case when we only have $C \geq 1$.

Theorem 3 Assume $C \geq 1$. We have the following bound for Δ when updating the weight for the misclassified example $(\mathbf{x}_a, y_a)$ and the auxiliary example $(\mathbf{x}_b, y_b)$:

$$\Delta \geq \frac{1}{2} + \frac{1}{2} \min((1 + \rho)^2, (C - \hat{\gamma})^2).$$

Proof: By setting $\gamma_a = 1$, we have $h(\gamma_a, d_{\gamma_b})$ computed as

$$h(\gamma_a = 1, d_{\gamma_b}) \geq \frac{1}{2} + (1 + \rho)d_{\gamma_b} - \frac{1}{2}d_{\gamma_b}^2.$$

Hence, Δ is lower bounded by

$$\Delta \geq \frac{1}{2} + \max_{d_{\gamma_b} \in [0, C - \hat{\gamma}_b]} \left((1 + \rho)d_{\gamma_b} - \frac{1}{2}d_{\gamma_b}^2 \right) \geq \frac{1}{2} + \frac{1}{2} \min((1 + \rho)^2, (C - \hat{\gamma})^2).$$

Although Theorem 2 and 3 show that the double update strategy could significantly improve the bounding constant Δ over $1/2$ and consequentially reduce the mistake bound, it is applicable only when there exists an auxiliary example. Below, we extend the double update strategy to the cases when there is no auxiliary example. Specifically, we relax the condition for performing double update as follows: there exists $(\mathbf{x}_b, y_b) \in \mathcal{D}$ that (i) $w_{ab} \leq -\rho$, (ii) $y_b f_{t-1}(\mathbf{x}_b) \leq 1$, and (iii) $C \geq \hat{\gamma}_b + \rho$. We refer to these cases as **weak double update**.

Theorem 4 Assume $w_{ab} \leq -\rho$, $y_b f_{t-1}(\mathbf{x}_b) \leq 1$ and $C \geq \hat{\gamma}_b + \rho$, we have the following bound for the bounding constant

$$\Delta \geq \frac{1 + \rho^2}{2}.$$

Proof: Following the definitions and assumptions, we have

$$\Delta = \max_{\gamma_a, d_{\gamma_b}} h(\gamma_a, d_{\gamma_b}) \geq h(1, \rho) \geq 1 - \frac{1}{2} + 0 - \frac{\rho^2}{2} + \rho^2 = \frac{1 + \rho^2}{2}.$$

Solving the optimization problem (3.1) is the key to the double update. The following proposition provides the optimal solution to the problem (3.1).

Proposition 3.2: Denote $\ell_a := 1 - y_a f(\mathbf{x}_a)$ and $\ell_b := 1 - y_b f(\mathbf{x}_b)$. Assume $\ell_a, \ell_b \geq 0$, $k_a, k_b > 0$ and $w_{ab} \leq 0$, then the solution of optimization problem (3.1) is as follows:

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \text{if } (k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) < 0 \text{ and} \\ & (k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \text{ and} \\ & \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \text{ and} \\ & \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab} \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2}) & \text{if } (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2}) \in [0, C] \times [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}.$$

□

The detailed proof for Proposition 3.2 can be found in Appendix A. Algorithm 4 summarizes the proposed Double Updating Online Learning (DUOL) algorithm. In this algorithm, to efficiently find the auxiliary example $(\mathbf{x}_b, y_b)$, we introduce a variable f_t^i for each support vector to keep track of its classification score, i.e., $y_i f_t(\mathbf{x}_i)$. Parameter ρ is used to trade off between efficiency and efficacy for DUOL: the smaller ρ the more double updates will be performed.

Finally, we give the mistake bound for the DUOL algorithm. We denote by $\mathcal{M}$ the set of indexes that correspond to the trials of misclassification, that is,

$$\mathcal{M} = \{t \mid y_t \neq \text{sign}(f_{t-1}(\mathbf{x}_t)), \forall t \in [T]\}.$$

In addition, we denote by $\mathcal{M}_d^s(\rho)$ and $\mathcal{M}_d^w(\rho)$ the sets of indexes for the cases of *strong* and *weak* double updating, respectively, that is,

$$\begin{aligned} \mathcal{M}_d^s(\rho) &= \{t \mid \exists \text{ auxiliary example } (\mathbf{x}_b, y_b) \text{ s.t. } C \geq \hat{\gamma}_b + \frac{1}{1-\rho} \text{ for } (\mathbf{x}_t, y_t), t \in \mathcal{M}\}, \\ \mathcal{M}_d^w(\rho) &= \{t \mid \exists (\mathbf{x}_b, y_b) \text{ s.t. } w_{ab} \leq -\rho, y_b f_{t-1}(\mathbf{x}_b) \leq 1 \text{ and } C \geq \hat{\gamma}_b + \rho, t \in \mathcal{M}/\mathcal{M}_d^s(\rho)\}. \end{aligned}$$

Note that in set $\mathcal{M}_d^s(\rho)$, for the convenience of analysis, we only consider the subset of strong updates when the condition $C \geq \hat{\gamma}_b + 1/(1-\rho)$ is satisfied. Finally, we denote the cardinalities of sets $\mathcal{M}$, $\mathcal{M}_d^s$, and $\mathcal{M}_d^w$ by $M = |\mathcal{M}|$, $M_d^s(\rho) = |\mathcal{M}_d^s(\rho)|$, $M_d^w(\rho) = |\mathcal{M}_d^w(\rho)|$, and $M_s = M - M_d^s(\rho) - M_d^w(\rho)$, respectively.

Theorem 5 *Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$, $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t , and T may not be known in advance. Assume $C \geq 1$. Then for any function f in $\mathcal{H}_\kappa$, the number of prediction mistakes M made by DUOL on this sequence of examples is bounded by:*

$$2 \min_{f \in \mathcal{H}_\kappa} \left(\frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^T \ell(f(\mathbf{x}_i), y_i) \right) - \rho^2 M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

where $\rho \in [0, 1)$.

Proof: According to Theorem 2 and 4, we have

$$\min_{t \in \mathcal{M}_d^s(\rho)} \Delta_t \geq \frac{1}{1-\rho}, \quad \min_{t \in \mathcal{M}_d^w(\rho)} \Delta_t \geq \frac{1+\rho^2}{2}.$$

Moreover, according to Theorem 3, we have $\Delta_t \geq 1/2, \forall t \in \mathcal{M}$. Combining these results with Theorem 1 together, we have

$$\frac{1}{2} M_s + \frac{1+\rho^2}{2} M_d^w(\rho) + \frac{1}{1-\rho} M_d^s(\rho) \leq \min_{f \in \mathcal{H}_\kappa} \left(\frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^T \ell(f(\mathbf{x}_i), y_i) \right). \quad (3.3)$$

Algorithm 4 The Algorithm of Double Updating Online Learning (DUOL).

```

Initialize  $S_0 = \emptyset$ ,  $f_0 = 0$ ;
for  $t = 1, 2, \dots, T$  do
    Receive a new instance  $\mathbf{x}_t$ ;
    Predict  $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$ ;
    Receive its label  $y_t$ ;
     $\ell_t = \max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$ ;
    if  $\ell_t > 0$  then
         $w_{\min} = \infty$ ;
        for  $\forall i \in S_{t-1}$  do
            if  $(f_{t-1}^i \leq 1)$  then
                if  $(y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{\min})$  then
                     $w_{\min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$ ;
                     $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$ ;
                end if
            end if
        end for
         $f_{t-1}^t = y_t f_{t-1}(\mathbf{x}_t)$ ;
         $S_t = S_{t-1} \cup \{t\}$ ;
        if  $(w_{\min} \leq -\rho)$  then
            Compute  $\gamma_t$  and  $d_{\gamma_b}$  by solving the optimization (3.1);
            for  $\forall i \in S_t$  do
                 $f_t^i \leftarrow f_{t-1}^i + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) + y_i d_{\gamma_b} y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$ ;
            end for
             $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} y_b \kappa(\mathbf{x}_b, \cdot)$ ;
        else
             $\gamma_t = \min(C, \ell_t / \kappa(\mathbf{x}_t, \mathbf{x}_t))$ ;
            for  $\forall i \in S_t$  do
                 $f_t^i \leftarrow f_{t-1}^i + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$ ;
            end for
             $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$ ;
        end if
    else
         $f_t = f_{t-1}$ ;  $S_t = S_{t-1}$ ;
        for  $\forall i \in S_t$  do
             $f_t^i \leftarrow f_{t-1}^i$ ;
        end for
    end if
end for
Return  $f_T, S_T$ 
    
```

We complete the proof using $M = M_s + M_d^w(\rho) + M_d^s(\rho)$.

As revealed by the above theorem, the number of mistakes made by the proposed double updating online learning algorithm will be smaller than the online learning algorithm that only performs a single update in each trial. The difference in the mistake bound is essentially due to the double updating, that is, the more the number of double updates, the more advantageous the proposed algorithm will be. Besides, the above bound also indicates that a strong double update is more powerful than a weak double update given that the associated weight of a strong double update $(1 + \rho)/(1 - \rho)$ is always much larger than that of a weak double update ρ^2 . It is worthwhile pointing out that although according to Theorem 5, it seems that the larger the value of ρ the smaller the mistake bound will be. This however may not be true because $M_d^s(\rho)$ in general decreases as ρ increases. Finally, we note that Theorem 5 bounds the number of mistakes made by the proposed DUOL algorithm for $C \geq 1$. When $C < 1$, the mistake bound for the proposed algorithm follows Theorem 2, 3 and Corollary 2 in [SSS07b].

3.3 Experimental Results

In this section, we evaluate the empirical performance of the proposed double updating online learning algorithms for online learning tasks.

3.3.1 Testbeds and Experimental Setup

We compare our technique with a number of state-of-the-art techniques, including:

- “Perceptron”: the kernel Perceptron algorithm [Ros58],
- “ROMMA”: the **R**elaxed **O**nline **M**aximum **M**argin **A**lgorithm [LL99],
- “agg-ROMMA”: the aggressive version of ROMMA [LL99],
- “ALMA_p(α)”: the **A**pproximate **L**arge **M**argin **A**lgorithm [Gen01],
- “PA”: the **P**assive-**A**ggressive algorithms [CDK⁺06]. For PA, two versions of algorithms (PA-I and PA-II) are implemented as described in [CDK⁺06],
- “LASVM”: the LASVM algorithm [BEWB05], which is compared as an ideal yardstick but computationally extremely intensive as will be revealed in our study.

Note that one may also compare with the online SVM algorithm [SSS06b], which updates the weights for all support vectors in each trial. Although online SVM will probably achieve the minimal number of mistakes, we do not include this baseline for comparison because it is too computationally intensive to run on some large data sets [ZHJ09].

For the proposed DUOL algorithms, we implement three variants based on different solvers to the problem in (3.1):

- “DUOL_{appr}” that employs an approximate solution to (3.1), that is, $\gamma_t = \frac{1}{1-\rho}$ and $\gamma_b = \hat{\gamma}_b + \frac{1}{1-\rho}$,
- “DUOL” that uses the exact solution to (3.1) given in Proposition 3.2, and
- “DUOL_{iter}” that first updates the weight for the misclassified example and then the weight for auxiliary example, as suggested in [SSS07b].

We test all the algorithms on eight benchmark data sets listed in table 3.1 from web machine learning repositories, which are randomly selected to cover different data sizes and dimensions. All of the data sets can be downloaded from LIBSVM website,¹ UCI machine learning repository,² and MIT CBCL face data sets.³

Data Set	# examples	# features
sonar	208	60
splice	1,000	60
german	1,000	24
mushrooms	8,124	112
dorothea	1,150	100,000
spambase	4,601	57
MITFace	6,977	361
w7a	24,692	300

Table 3.1: Binary-class data sets used in the experiments.

To make a fair comparison, for all algorithms in comparison, we set $C = 5$ and use the same Gaussian kernel with $\sigma = 8$. For the ALMA_p(α) algorithm, parameter p and α are set to 2 and 0.9, respectively, based on our experience. For LASVM, $\tau = 0.001$, according to [BEWB05]. For the proposed DUOL algorithm, we fix ρ to be 0 for all cases. All the experiments are repeated 20 times, each with an independent random permutation of the data points (the online sequences of the examples are determined by the random permutations). All the results are reported by averaging over the 20 runs. We evaluate the online learning performance by measuring the *mistake rate*, that is, the percentage of examples that are misclassified by the online learning algorithm. We measure the sparsity of the learned classifiers by the number of support vectors. We evaluate computational efficiencies of all the algorithms in terms of their CPU running time (in seconds). All the experiments are run in Matlab on a windows machine of 2.3GHz CPU.

¹LIBSVM website is <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>.

²UCI ML repository is at <http://www.ics.uci.edu/~mllearn/MLRepository.html>.

³MIT CBCL face data sets can be found at <http://cbcl.mit.edu/software-datasets>.

3.3.2 Performance Evaluation

Table 3.2 summarizes the performance of all the compared online learning algorithms over the binary data sets. We can draw several observations from the results.

First, we observe that LASVM yields considerably better performance than the other online learning algorithms for all dataset except "german", however, at the price of extremely high computational cost. For most cases, the running time of LASVM is one order, sometimes two order, higher than the other online learning algorithms, making it unsuitable for online learning. For the remaining part of this section, we restrict our discussion to the other nine online learning algorithms.

Second, among the six baseline algorithms in comparison, we observe that the agg-ROMMA and two PA algorithms (PA-I and PA-II) perform considerably better than the other three algorithms (i.e., Perceptron, ROMMA, and ALMA) in most cases. We also notice that the agg-ROMMA and the two PA algorithms consume considerably larger numbers of support vectors than the other three algorithms. We believe this is because the agg-ROMMA and the two PA algorithms adopt more aggressive strategies than the other three algorithms, resulting in more updates and better classification performance. For the convenience of discussion, we refer to agg-ROMMA and two PA algorithms as *aggressive* algorithms, and the other three online learning algorithms as *non-aggressive* ones.

Third, we observe that among the three variants of double updating online learning, the DUOL approach, which solves the optimization problem exactly, yields the least *mistake rate* with the smallest number of support vectors for most of the cases. Comparing with the baseline algorithms, we observe that DUOL achieves significantly smaller *mistake rates* than the other single-updating algorithms in all cases. This shows that the proposed double updating approach is effective in improving the performance of online prediction. By examining the number of support vectors, we observed that DUOL results in sparser classifiers than the three aggressive online learning algorithms, and denser classifiers than the three non-aggressive algorithms.

Finally, according to the results of running time, we observe that DUOL is overall efficient as compared with the state-of-the-art online learning algorithms. Among all the algorithms in comparison, Perceptron, due to its simplicity nature, is clearly the most efficient algorithm. Since DUOL requires double updates, it is less efficient than PA, ROMMA and ALMA algorithms, but is comparable to the agg-ROMMA algorithm. Note that the comparisons of running time costs are slightly different compared with the results in our previous conference paper [ZHJ09] because we did some improvements of efficiency for the implementations of some existing algorithms in this thesis, i.e., we introduce a variable f_t^i for each support vector to keep track of its classification score i.e., $y_i f_t(\mathbf{x}_i)$.

CHAPTER 3. BINARY DOUBLE UPDATING ONLINE LEARNING

Algorithm	sonar				splice		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	38.125 $\pm$ 3.815	79.30 $\pm$ 7.93	0.004		27.120 $\pm$ 0.975	271.20 $\pm$ 9.75	0.017
ROMMA	36.587 $\pm$ 2.976	76.10 $\pm$ 6.19	0.006		25.560 $\pm$ 0.814	255.60 $\pm$ 8.14	0.032
agg-ROMMA	34.928 $\pm$ 2.860	130.05 $\pm$ 7.51	0.009		22.980 $\pm$ 0.780	602.90 $\pm$ 7.42	0.044
ALMA ₂ (0.9)	36.370 $\pm$ 3.572	86.25 $\pm$ 6.43	0.006		26.040 $\pm$ 0.965	314.95 $\pm$ 9.41	0.032
PA-I	40.986 $\pm$ 2.837	154.15 $\pm$ 6.95	0.004		23.815 $\pm$ 1.042	665.60 $\pm$ 5.60	0.029
PA-II	40.481 $\pm$ 3.023	162.40 $\pm$ 6.26	0.004		23.515 $\pm$ 1.005	689.00 $\pm$ 7.85	0.029
DUOL _{iter}	39.495 $\pm$ 3.299	149.85 $\pm$ 3.42	0.014		23.205 $\pm$ 0.932	566.85 $\pm$ 13.08	0.097
DUOL _{appr}	41.010 $\pm$ 2.335	162.25 $\pm$ 5.01	0.013		21.945 $\pm$ 1.134	721.85 $\pm$ 9.10	0.095
DUOL	34.255 $\pm$ 2.811	137.60 $\pm$ 6.99	0.017		20.875 $\pm$ 0.868	577.15 $\pm$ 10.81	0.087
LASVM	27.957 $\pm$ 1.801	133.75 $\pm$ 3.78	0.044		17.405 $\pm$ 0.789	649.75 $\pm$ 5.50	0.689
Algorithm	german				mushrooms		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	34.760 $\pm$ 0.947	347.60 $\pm$ 9.47	0.019		2.083 $\pm$ 0.278	169.25 $\pm$ 22.58	0.148
ROMMA	34.725 $\pm$ 1.009	347.25 $\pm$ 10.09	0.037		2.429 $\pm$ 0.101	197.35 $\pm$ 8.24	0.264
agg-ROMMA	32.925 $\pm$ 1.184	633.40 $\pm$ 14.02	0.049		1.568 $\pm$ 0.096	1307.90 $\pm$ 39.59	0.576
ALMA ₂ (0.9)	33.480 $\pm$ 0.681	394.75 $\pm$ 9.24	0.036		2.538 $\pm$ 0.297	304.80 $\pm$ 38.02	0.267
PA-I	33.010 $\pm$ 1.025	721.10 $\pm$ 12.99	0.031		1.661 $\pm$ 0.089	1221.55 $\pm$ 22.80	0.454
PA-II	32.630 $\pm$ 1.016	749.50 $\pm$ 11.84	0.032		1.657 $\pm$ 0.088	1326.20 $\pm$ 22.85	0.483
DUOL _{iter}	35.985 $\pm$ 1.077	714.35 $\pm$ 12.75	0.125		1.537 $\pm$ 0.101	860.05 $\pm$ 23.00	0.521
DUOL _{appr}	30.275 $\pm$ 0.937	716.10 $\pm$ 10.44	0.096		1.459 $\pm$ 0.101	1291.35 $\pm$ 32.03	0.658
DUOL	31.810 $\pm$ 1.090	656.30 $\pm$ 14.36	0.108		0.596 $\pm$ 0.053	453.70 $\pm$ 19.40	0.341
LASVM	30.045 $\pm$ 0.964	729.90 $\pm$ 12.62	0.780		0.395 $\pm$ 0.033	360.15 $\pm$ 15.14	4.297
Algorithm	dorothea				spambase		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	13.257 $\pm$ 0.973	152.45 $\pm$ 11.18	0.016		24.987 $\pm$ 0.525	1149.65 $\pm$ 24.16	0.215
ROMMA	17.461 $\pm$ 0.537	200.80 $\pm$ 6.18	0.035		23.953 $\pm$ 0.510	1102.10 $\pm$ 23.44	0.275
agg-ROMMA	17.435 $\pm$ 0.500	438.30 $\pm$ 13.83	0.044		21.242 $\pm$ 0.384	2550.70 $\pm$ 27.28	0.515
ALMA ₂ (0.9)	14.478 $\pm$ 0.378	210.25 $\pm$ 5.68	0.035		23.579 $\pm$ 0.411	1550.15 $\pm$ 15.65	0.348
PA-I	17.500 $\pm$ 0.491	461.30 $\pm$ 15.80	0.026		22.112 $\pm$ 0.374	2861.50 $\pm$ 24.36	0.479
PA-II	17.500 $\pm$ 0.491	461.30 $\pm$ 15.80	0.027		21.907 $\pm$ 0.340	3029.10 $\pm$ 24.69	0.504
DUOL _{iter}	21.109 $\pm$ 0.796	559.20 $\pm$ 19.44	0.080		21.907 $\pm$ 0.432	2511.20 $\pm$ 34.14	1.215
DUOL _{appr}	17.500 $\pm$ 0.491	461.30 $\pm$ 15.80	0.054		20.185 $\pm$ 0.351	2981.00 $\pm$ 26.95	1.091
DUOL	11.757 $\pm$ 0.237	407.50 $\pm$ 12.80	0.080		19.438 $\pm$ 0.282	2494.95 $\pm$ 26.19	1.069
LASVM	10.487 $\pm$ 1.067	1017.50 $\pm$ 7.40	1.192		18.033 $\pm$ 0.371	2774.35 $\pm$ 19.74	14.960
Algorithm	MITFace				w7a		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Perceptron	4.665 $\pm$ 0.192	325.50 $\pm$ 13.37	0.207		4.027 $\pm$ 0.095	994.40 $\pm$ 23.57	3.392
ROMMA	4.114 $\pm$ 0.155	287.05 $\pm$ 10.84	0.285		4.158 $\pm$ 0.087	1026.75 $\pm$ 21.51	1.875
agg-ROMMA	3.137 $\pm$ 0.093	1121.15 $\pm$ 24.18	0.555		3.500 $\pm$ 0.061	2318.65 $\pm$ 60.49	3.257
ALMA ₂ (0.9)	4.467 $\pm$ 0.169	400.10 $\pm$ 10.53	0.297		3.518 $\pm$ 0.071	1031.05 $\pm$ 15.33	1.314
PA-I	3.190 $\pm$ 0.128	1155.45 $\pm$ 14.53	0.439		3.701 $\pm$ 0.057	2839.60 $\pm$ 41.57	2.691
PA-II	3.108 $\pm$ 0.112	1222.05 $\pm$ 13.73	0.463		3.571 $\pm$ 0.053	3391.50 $\pm$ 51.94	3.311
DUOL _{iter}	2.551 $\pm$ 0.128	963.45 $\pm$ 23.80	0.572		4.456 $\pm$ 0.073	3048.85 $\pm$ 54.53	4.566
DUOL _{appr}	2.687 $\pm$ 0.140	1262.50 $\pm$ 20.68	0.656		3.116 $\pm$ 0.104	2908.95 $\pm$ 28.65	3.679
DUOL	2.151 $\pm$ 0.106	697.95 $\pm$ 13.17	0.445		2.914 $\pm$ 0.045	2402.55 $\pm$ 39.88	6.470
LASVM	1.140 $\pm$ 0.079	468.50 $\pm$ 7.21	4.071		2.116 $\pm$ 0.111	2679.55 $\pm$ 130.78	74.015

Table 3.2: Evaluation of online learning algorithms on the binary-class data sets.

3.3.3 Evaluation of Different Auxiliary Example Selection Strategies and the Sensitivity to Parameter C for DUOL

As the performance of DUOL quite relies on the choice of auxiliary examples, in this section, we evaluate different auxiliary example selection strategies. Specifically, we compare the proposed strategy to a random selection approach, referred to as “DUOL_{rand}”, which randomly chooses an auxiliary example from the existing support vectors. The exact solution to the problem in (3.1), given by Proposition 3.2, is used for updating the weights of both examples. We set $\rho = 0$ and $\sigma = 8$ for all the data sets, same as the previous experiments.

Figure 3.1 compares the online prediction performance between DUOL and DUOL_{rand} as well as the other competing algorithms with varied C values across eight different data sets. Several observations can be drawn from the results.

First, it is clear to see that the proposed strategy for selecting auxiliary examples is more effective than the random selection strategy for most cases. Second, we observe that the proposed DUOL algorithm is significantly more accurate than the other two variants of double updating online learning algorithms (DUOL_{iter} and DUOL_{appr}) for varied C values, as we expected. We observe that DUOL_{iter}, the iterative updating approach, performs unstably, which might be due to local optimum suffered from its heuristic update. This observation validates the importance of performing the optimal double updates by the proposed DUOL algorithm.

3.3.4 Empirical Evaluation of Mistake Bounds

To examine how the double updating strategy affects the mistake bound, we empirically compare M , the total number of mistakes made by the DUOL algorithm, $M_d^w(\rho)$, the number of mistake cases where the weak double updates are applied, and $M_d^s(\rho)$, the number of mistake cases where the strong double updates are applied. Figure 3.2 shows the comparison between M , $M_d^w(\rho)$, and $M_d^s(\rho)$ by varying ρ from 0 to 1.

First, we observe that double updates are frequently applied when ρ is small. This is because it is easier to find an auxiliary example for double updating when ρ is small. Further, we find that setting ρ close to 0 by default often leads to the best or close to the best results. Second, we observe that the number of weak updates is significantly larger than that of strong updates. This is because the condition of conducting a strong double update is significantly more difficult to be satisfied than that for a weak double update. Third, we observe that both $M_d^w(\rho)$ and $M_d^s(\rho)$ monotonically decrease when increasing the value of ρ . In the extreme case, when ρ is close to 1, their value often drops to zero, indicating that no double update was applied. In the meantime, we find that the total number of mistakes often reaches the maximum, as ρ approaches 1. These results again validate the importance and effectiveness of the proposed double updating algorithm.

 Figure 3.1: Comparison between $DUOL$ and $DUOL_{rand}$ with varied C values.

 Figure 3.2: Empirical comparison of M , $M_d^w(\rho)$ and $M_d^s(\rho)$ w.r.t. varied $\rho \in [0, 1]$ values.

3.4 Summary

This chapter presented a novel “double updating” approach to online learning named as “DUOL”, which not only updates the weight of the newly added support vector, but also adjusts the weight of one existing support vector that the most seriously conflicts with the new support vector. We show that the mistake bound for an online classification task can be significantly reduced by the proposed DUOL algorithm. We have conducted an extensive set of experiments by comparing with a number of competing algorithms on binary online classification tasks. Promising empirical results validate the effectiveness of our technique.

It is an interesting problem how to extend the DUOL method for binary case to the situation of multiclass classification. In the next chapter, we will provide a detailed derivation.

Chapter 4

Multiclass Double Updating Online Learning

The limitation of binary classification observed in the previous chapter also exists in the multi-class classification case. To solve this problem, in this chapter, we will extend the proposed double updating techniques, which is originally designed for binary classification, to the online learning tasks of *multiclass* classification. In particular, we will address the *multiclass multilabel* classification problem, in which each instance can be assigned to multiple labels.

The key challenge for the multiclass multilabel classification is also to decide which existing support vector should be selected for weight updating besides the new misclassified example. To tackle this issue, we defined a new kind of “conflict” for the multiclass multilabel classification, which is not as intuitive as the binary case, however we provide a proposition that can offer a bit of intuitive explanation. In order to quantitatively analyze the impact of updating the weight for such an existing support vector, we also employ an analysis that is based on the recent primal-dual framework [SSS06b, SSS07b]. Our analysis shows that under certain conditions, the proposed online multiclass multilabel learning algorithm can significantly reduce the mistake bound of the existing online algorithms.

The rest of this chapter is organized as follows. Section 4.1 briefs the related work. Section 4.2 first introduces the problem setting and then presents both theoretical analysis and details of the proposed multiclass DUOL algorithm. Section 4.3 empirically evaluate the proposed algorithm. Section 4.4 summarizes this chapter.

4.1 Related Work

For multiclass multilabel classification problem, many algorithms are proposed [CS03, CDK⁺06, FSSSU06, SSS07c]. However, in these studies, all the algorithms will only update the weight of the newly introduced support vector, while never update the weights of

the existing support vectors, which is obviously inefficient to reduce the online cumulative number of mistakes.

So in this chapter, we would adapt the idea in the previous chapter to solve multiclass multilabel classification problem.

4.2 Multiclass Double Updating Online Learning

4.2.1 Problem Setting

Similar to online binary classification, online *multiclass multilabel* learning is performed over a sequence of training examples $(\mathbf{x}_1, Y_1), \dots, (\mathbf{x}_T, Y_T)$. Unlike binary classification where $y_t \in \{-1, +1\}$, in multiclass multilabel learning, each class assignment $Y_t \subseteq \mathcal{Y} = \{1, \dots, k\}$ could contain multiple labels, making it a more challenging problem. We use $\hat{Y}_t$ to represent the label set predicted by the online learning algorithm. Before presenting our algorithm, we first review online multiclass multilabel learning based on the framework of label ranking [CS05, CDK⁺06], which encompasses the single label setting as a special case. In addition, we will refer to *multiclass multilabel* as *multiclass* to simplify the discussion below.

4.2.2 Label Ranking for Multiclass Learning

Given an instance $\mathbf{x}$, the label ranking approach first computes a score for every class label in $\mathcal{Y}$, and ranks the labels in the descending order of their scores. The predicted label set $\hat{Y}_t$ is formed by the labels with the highest scores. The objective of label ranking is to ensure that the score of label r is significantly larger than that of class s if $r \in Y_t$ is a true label assignment while $s \in \mathcal{Y} \setminus Y_t$ is not. If this condition is violated, the instance $\mathbf{x}$ is misclassified.

We follow the protocol of *multi-prototype* classification [Vap98, CS01, CDK⁺06] for the design of multiclass multilabel learning algorithm. It learns multiple hypotheses/classifiers, one classifier for each class in $\mathcal{Y}$, leading to a total of k classifiers that are trained for the classification task. Specifically, for trial t , upon receiving an instance $\mathbf{x}_t$, the scores of k classes output by the set of k hypotheses are given by

$$\bar{f}_{t-1}(\mathbf{x}_t) = (f_{t-1,1}(\mathbf{x}_t), \dots, f_{t-1,k}(\mathbf{x}_t))^T,$$

where $f_{t-1,i} \in \mathcal{H}_\kappa, i = 1, \dots, k$. We introduce two variables r_t and s_t that are defined as follows:

$$r_t = \arg \min_{r \in Y_t} f_{t-1,r}(\mathbf{x}_t) \quad \text{and} \quad s_t = \arg \max_{s \notin Y_t} f_{t-1,s}(\mathbf{x}_t). \quad (4.1)$$

Here, r_t and s_t represent the class of the smallest score among all relevant classes and the class of the largest score among the irrelevant classes, respectively. Using the notation of r_t and s_t , the *margin* with respect to the hypothesis vector $\bar{f}_{t-1}$ at trial t is defined as follows:

$$\Gamma(\bar{f}_{t-1}; (\mathbf{x}_t, Y_t)) = f_{t-1, r_t}(\mathbf{x}_t) - f_{t-1, s_t}(\mathbf{x}_t).$$

Based on the notation of classification margin, we define the loss function of hypotheses $\bar{f}_{t-1}(\mathbf{x})$ for training example $(\mathbf{x}_t, Y_t)$ as follows:

$$\ell(\bar{f}_{t-1}(\mathbf{x}_t), Y_t) = \max_{r \in Y_t, s \notin Y_t} [1 - (f_{t-1, r}(\mathbf{x}_t) - f_{t-1, s}(\mathbf{x}_t))]_+,$$

where $[x]_+ = \max(0, x)$.

4.2.3 A Perceptron Algorithm for Online Multiclass Learning

According to [CS03], when an example is misclassified at trial t , we update each component of the classifier $\bar{f}_{t-1}$ as follows:

$$f_{t,i}(\mathbf{x}) = f_{t-1,i}(\mathbf{x}) + \sigma_{Y_t}(i, t) \gamma_t \kappa(\mathbf{x}_t, \mathbf{x}), \quad \forall i \in \mathcal{Y}, \quad (4.2)$$

where $\gamma_t \in (0, C]$, and function $\sigma_{Y_t}(i, t)$, which is simplified as $\sigma(i, t)$, is defined below:

$$\sigma(i, t) = \begin{cases} 1 & \text{if } i = r_t \\ -1 & \text{if } i = s_t \\ 0 & \text{otherwise} \end{cases}.$$

Using notation $H(Y_t) = (\sigma(1, t), \dots, \sigma(k, t))^T$, we rewrite equation (4.2) as $\bar{f}_t(\mathbf{x}) = \bar{f}_{t-1}(\mathbf{x}) + \gamma_t H(Y_t) \kappa(\mathbf{x}_t, \mathbf{x})$, or equivalently

$$\bar{f}(\mathbf{x}) = \sum_{i=1}^n \gamma_i H(Y_i) \kappa(\mathbf{x}_i, \mathbf{x}),$$

where n is the number of examples received so far.

4.2.4 Multiclass Double Updating Online Learning

We extend the DUOL algorithm to multiclass learning. We denote by $(\mathbf{x}_a, Y_a)$ the misclassified example received at the current trial, that is, $(\bar{f}(\mathbf{x}_a))_{r_a} - (\bar{f}(\mathbf{x}_a))_{s_a} \leq 0$. Similar to DUOL for binary classification, we introduce an auxiliary example $(\mathbf{x}_b, Y_b)$ from the existing support vectors that obey the following conditions:

- (i) $(\bar{f}(\mathbf{x}_b))_{r_b} - (\bar{f}(\mathbf{x}_b))_{s_b} \leq 0$, that is, $(\mathbf{x}_b, Y_b)$ is misclassified by current classifier $\bar{f}$,

- (ii) $(H(Y_a) \cdot H(Y_b))\kappa(\mathbf{x}_a, \mathbf{x}_b) \leq -2\rho$ where $\rho \in (0, 1)$ is a threshold. This property indicates that example $(\mathbf{x}_a, Y_a)$ **conflicts** with example $(\mathbf{x}_b, Y_b)$.

Compared to auxiliary example defined for binary classification, we introduce $H(Y_a) \cdot H(Y_b)$ in above when defining two conflicting instances. Given $\kappa(\mathbf{x}_a, \mathbf{x}_b) \geq 0$, the second condition of auxiliary example implies $H(Y_a) \cdot H(Y_b) \leq 0$, which further indicates that two examples $(\mathbf{x}_a, Y_a)$ and $(\mathbf{x}_b, Y_b)$ have the opposite prediction, that is, $(r_a = s_b)$ or $(s_a = r_b)$. This result is revealed by the following proposition.

Proposition 4.3: *The inequality $H(Y_a) \cdot H(Y_b) < 0$ holds if and only if $(r_a = s_b)$ or $(s_a = r_b)$.* $\square$

The proof of Proposition 4.3 is given in the Appendix A.

Similar to the DUOL algorithm for binary classification, our analysis aims to show that by updating weights for both misclassified example and the auxiliary example, we may be able to significantly improve the bounding constant Δ , which is a lower bound of the dual ascent Δ_t .

To facilitate the presentation, we summarize the section 2 of [SSS07c] as the following theorem,

Theorem 6 *If an online learning algorithm $\mathcal{A}$ for multiclass classification is designed to ensure that for all t , Δ_t , is bounded from below by a **bounding constant** Δ , then the number of mistakes M made by $\mathcal{A}$ when trained over a sequence of examples $(\mathbf{x}_1, Y_1), \dots, (\mathbf{x}_T, Y_T)$ (where T may be not available in advance), is upper bounded by*

$$M \times \Delta \leq \min_{\bar{f} \in \bar{\mathcal{H}}_\kappa} \left(F(\bar{f}) + C \sum_{i=1}^T \ell(\bar{f}(\mathbf{x}_i), Y_i) \right), \quad (4.3)$$

where $\bar{\mathcal{H}}_\kappa = \prod_{i=1}^k \mathcal{H}_\kappa$ and $F(\bar{f}) = \sum_{i=1}^k \frac{1}{2} \|\bar{f}_i\|_{\mathcal{H}_\kappa}^2$.

To facilitate our further discussions, we define $k_a = \kappa(\mathbf{x}_a, \mathbf{x}_a)$, $k_b = \kappa(\mathbf{x}_b, \mathbf{x}_b)$, $w_{ab} = (H(Y_a) \cdot H(Y_b))\kappa(\mathbf{x}_a, \mathbf{x}_b)$.

The following proposition shows the optimization problem related to the multiclass double updating online learning algorithm, which forms the basis for deriving the bounding constant Δ .

Proposition 4.4: *With the double updating, that is, adjusting the weight of some auxiliary support vector $(\mathbf{x}_b, Y_b)$ from $\hat{\gamma}_b$ to γ_b (denoted by $d_{\gamma_b} = \gamma_b - \hat{\gamma}_b$) and assigning weight γ_a to the current misclassified example $(\mathbf{x}_a, Y_a)$, the improvement in the objective function of dual SVM, denoted by Δ_t , is computed by the following optimization problem:*

$$\begin{aligned} \max_{\gamma_a, d_{\gamma_b}} \quad & \gamma_a \left(1 - (f_{t-1, r_a}(\mathbf{x}_a) - f_{t-1, s_a}(\mathbf{x}_a)) \right) + d_{\gamma_b} \left(1 - (f_{t-1, r_b}(\mathbf{x}_b) - f_{t-1, s_b}(\mathbf{x}_b)) \right) \\ & - k_a \gamma_a^2 - k_b d_{\gamma_b}^2 - w_{ab} \gamma_a d_{\gamma_b} \\ \text{s.t.} \quad & 0 \leq \gamma_a \leq C, -\hat{\gamma}_b \leq d_{\gamma_b} \leq C - \hat{\gamma}_b. \end{aligned} \quad (4.4)$$

□

The proof for this proposition can be found in the Appendix A.

Theorem 7 Assume $\kappa(\mathbf{x}, \mathbf{x}) \leq 1$ for any $\mathbf{x}$ and $C \geq \hat{\gamma}_b + \frac{1}{2(1-\rho)}$ for the selected auxiliary example $(\mathbf{x}_b, Y_b)$, we have the following bound for Δ :

$$\Delta \geq \frac{1}{2(1-\rho)}.$$

We refer to the case as a strong double update when there exists a auxiliary example $(\mathbf{x}_b, Y_b)$ s.t. $C \geq \hat{\gamma}_b + \frac{1}{2(1-\rho)}$. Similar to double updating for binary classification, we introduce *weak* double update when there exists $(\mathbf{x}_b, Y_b)$ s.t. $w_{ab} \leq -2\rho$, $f_{t-1, r_b}(\mathbf{x}_b) - f_{t-1, s_b}(\mathbf{x}_b) \leq 1$, and $C \geq \hat{\gamma}_b + \frac{\rho}{2}$.

Theorem 8 Assume there exists $(\mathbf{x}_b, Y_b)$ s.t. $w_{ab} \leq -2\rho$, $f_{t-1, r_b}(\mathbf{x}_b) - f_{t-1, s_b}(\mathbf{x}_b) \leq 1$, $C \geq \hat{\gamma}_b + \frac{\rho}{2}$ and the current instance is misclassified. Then we have the following bounding constant

$$\Delta \geq \frac{1 + \rho^2}{4}. \quad (4.5)$$

However, if no auxiliary example can be found, we will perform single update. And for single update, we can prove the follow theorem.

Theorem 9 Assume $C \geq 1/2$. We have the following bound for Δ when updating the weight for the misclassified example $(\mathbf{x}_a, Y_a)$:

$$\Delta \geq 1/4$$

Proof: Since $\ell(\bar{f}(\mathbf{x}_a), Y_a) \geq 1$ and $k_a \leq 1$, we have

$$\gamma_a \ell(\bar{f}(\mathbf{x}_a), Y_a) - k_a \gamma_a^2 \geq \gamma_a - \gamma_a^2 \geq 1/2 - (1/2)^2 = 1/4.$$

The exact solution to the Quadratic Programming (QP) problem in (4.4) is given by the following proposition.

Proposition 4.5: Denote $\ell_a := 1 - (f_{t-1, r_a}(\mathbf{x}_a) - f_{t-1, s_a}(\mathbf{x}_a))$ and $\ell_b := 1 - (f_{t-1, r_b}(\mathbf{x}_b) - f_{t-1, s_b}(\mathbf{x}_b))$. Assume $\ell_a, \ell_b \geq 0$, $k_a, k_b > 0$ and $w_{ab} \leq 0$, then the solution of optimization (4.4) is as follows:

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \text{if } (2k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) < 0 \text{ and} \\ & (2k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{2k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - 4k_a k_b C + 2k_b \ell_a}{2k_b} > 0 \text{ and} \\ & \frac{\ell_b - w_{ab}C}{2k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{2k_a}, C - \hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{2k_a} \in [0, C] \text{ and} \\ & \ell_b - 2k_b(C - \hat{\gamma}_b) - w_{ab} \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{2k_a} > 0 \\ (\frac{2k_b \ell_a - w_{ab} \ell_b}{4k_a k_b - w_{ab}^2}, \frac{2k_a \ell_b - w_{ab} \ell_a}{4k_a k_b - w_{ab}^2}) & \text{if } (\frac{2k_b \ell_a - w_{ab} \ell_b}{4k_a k_b - w_{ab}^2}, \frac{2k_a \ell_b - w_{ab} \ell_a}{4k_a k_b - w_{ab}^2}) \in [0, C] \times [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}.$$

□

We skip the proof due to its similarity to that of Proposition 3.2.

Similar to the binary case, the next issue is to identify the auxiliary example (x_b, Y_b) efficiently, which requires updating the classification score $f_{t,r_i}(x_i) - f_{t,s_i}(x_i)$ for all the support vectors. We also introduce a variable for each support vector, denoted by f_t^i , to keep track of the classification score. When a new support vector (x_a, Y_a) with weight γ_a is added to the classifier, we update the classification score f_t^i by

$$f_t^i \leftarrow f_{t-1}^i + [\gamma_a * H(Y_a) * \kappa(x_a, x_i)] \cdot H(Y_i)$$

and when the weight of an auxiliary example (x_b, y_b) is updated from $\hat{\gamma}_b$ to γ_b , we update the classification score f_t^i by

$$f_t^i \leftarrow f_{t-1}^i + [(\gamma_b - \hat{\gamma}_b) * H(Y_b) * \kappa(x_b, x_i)] \cdot H(Y_i)$$

This updating procedure ensures that the computational cost of double updating online learning is $O(n)$ where n is the number of support vectors, similar to that of the kernel online learning algorithm. Furthermore, to make the proposed algorithm more aggressive, we replace the condition of making a mistake with a new condition of suffering margin error to allow more updates. Algorithm 5 summarizes the steps of the multiclass DUOL (M-DUOL) algorithm. Note that we replace the conditions for auxiliary example with the margin error in order to make more double updates.

Finally, we show a bound on the number of mistakes. To ease our discussion, let us denote by $\mathcal{M}$ a set of indexes where mistakes occurred, i.e.,

$$\mathcal{M} = \{t \mid |f_{t-1,r_t}(x_t) - f_{t-1,s_t}(x_t)| < 0, \forall t \in [T]\}$$

$\mathcal{M}_d^s(\rho)$ a set of indexes where *strong* double updates occurred, i.e.,

$$\mathcal{M}_d^s(\rho) = \{t \mid \exists \text{ auxiliary example } (x_b, Y_b) \text{ s.t. } C \geq \hat{\gamma}_b + \frac{1}{2(1-\rho)} \text{ for } (x_t, Y_t), t \in \mathcal{M}\}$$

and $\mathcal{M}_d^w(\rho)$ a set of indexes where *weak* double updates occurred, i.e.,

$$\mathcal{M}_d^w(\rho) = \{t \in \mathcal{M} / \mathcal{M}_d^s(\rho) \mid \exists (x_b, Y_b) \text{ s.t. } w_{ab} \leq -2\rho, \Gamma(\bar{f}_{t-1}; (\mathbf{x}_b, Y_b)) \leq 1, C \geq \hat{\gamma}_b + \frac{\rho}{2} \text{ for } (x_t, Y_t)\}.$$

We also introduce some notations to denote the various numbers of mistakes, i.e., $M = |\mathcal{M}|$, $M_d^s(\rho) = |\mathcal{M}_d^s(\rho)|$, $M_d^w(\rho) = |\mathcal{M}_d^w(\rho)|$ and $M_s = M - M_d^s(\rho) - M_d^w(\rho)$.

A mistake bound for the M-DUOL algorithm, similar to Theorem 5, is given by the following theorem.

Algorithm 5 Algorithm of multiclass double-updating online learning (M-DUOL).

```

Initialize  $H_0 = \emptyset$ ,  $S_0 = \emptyset$ ,  $\bar{f}_0 = 0$ ;
for  $t = 1, 2, \dots, T$  do
    Receive a new instance  $\mathbf{x}_t$ ;
    Predict  $W_{t-1} = \bar{f}_{t-1}(\mathbf{x}_t)$ ;
    Receive its label set  $Y_t$ ;
     $\ell_t = [1 - W_{t-1} \cdot H(Y_t)]_+$ ;
    if  $\ell_t > 0$  then
         $w_{min} = \infty$ ;
        for  $\forall i \in S_{t-1}$  do
            if  $f_{t-1}^i \leq 1$  then
                if  $Hk_{ti} < w_{min}$  then
                     $w_{min} = Hk_{ti}$ ;
                     $(\mathbf{x}_b, Y_b) = (\mathbf{x}_i, Y_i)$ ;
                end if
            end if
        end for
         $f_{t-1}^t = W_{t-1} \cdot H(Y_t)$ ;
         $S_t = S_{t-1} \cup \{t\}$ ;  $H_t = H_{t-1} \cup \{H(Y_t)\}$ ;
        if  $(w_{min} \leq -2\rho)$  then
            Compute  $\gamma_t$  and  $d_{\gamma_b}$  by solving the optimization (4.4);
            for  $\forall i \in S_t$  do
                 $f_t^i \leftarrow f_{t-1}^i + [\gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \mathbf{x}_i)] \cdot H(Y_i) + [d_{\gamma_b} * H(Y_b) * \kappa(\mathbf{x}_b, \mathbf{x}_i)] \cdot H(Y_i)$ ;
            end for
             $\bar{f}_t = \bar{f}_{t-1} + \gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} * H(Y_b) * \kappa(\mathbf{x}_b, \cdot)$ ;
        else
             $\gamma_t = \min(C, \frac{\ell_t}{2\kappa(\mathbf{x}_t, \mathbf{x}_t)})$ ;
            for  $\forall i \in S_t$  do
                 $f_t^i \leftarrow f_{t-1}^i + [\gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \mathbf{x}_i)] \cdot H(Y_i)$ ;
            end for
             $\bar{f}_t = \bar{f}_{t-1} + \gamma_t * H(Y_t) * \kappa(\mathbf{x}_t, \cdot)$ ;
        end if
    else
         $\bar{f}_t = \bar{f}_{t-1}$ ,  $S_t = S_{t-1}$ ,  $H_t = H_{t-1}$ ;
        for  $\forall i \in S_t$  do
             $f_t^i \leftarrow f_{t-1}^i$ ;
        end for
    end if
end for
Return  $\bar{f}_T$ ,  $S_T$ ,  $H_T$ 
    
```

Theorem 10 *Let $(\mathbf{x}_1, Y_1), \dots, (\mathbf{x}_T, Y_T)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^n$, $Y_t \subseteq \mathcal{Y}$ and $\kappa(\mathbf{x}_i, \mathbf{x}_j) \in [0, 1]$ for all i, j and T may not be available in advance. And assume $C \geq 1/2$. Then for any function $\bar{f} \in \prod_{i=1}^k \mathcal{H}_\kappa$, the number of prediction mistakes M made by M-DUOL on this sequence of examples is bounded by:*

$$4 \min_{\bar{f} \in \mathcal{H}_\kappa} \left(F(\bar{f}) + C \sum_{i=1}^T \ell(\bar{f}(\mathbf{x}_i), Y_i) \right) - \rho^2 M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho). \quad (4.6)$$

Proof: According to Theorem 7 and 8, we have

$$\min_{t \in \mathcal{M}_d^s(\rho)} \Delta_t \geq \frac{1}{2(1-\rho)}, \quad \min_{t \in \mathcal{M}_d^w(\rho)} \Delta_t \geq \frac{1+\rho^2}{4}.$$

Moreover, according to Theorem 9, we have $\Delta_t \geq 1/4, \forall t \in \mathcal{M}$. Combining these results with Theorem 6 together, we have

$$\frac{1}{4} M_s + \frac{1+\rho^2}{4} M_d^w(\rho) + \frac{1}{2(1-\rho)} M_d^s(\rho) \leq \min_{\bar{f} \in \mathcal{H}_\kappa} \left(F(\bar{f}) + C \sum_{i=1}^T \ell(\bar{f}(\mathbf{x}_i), Y_i) \right).$$

We complete the proof using $M = M_s + M_d^w(\rho) + M_d^s(\rho)$.

4.3 Experimental Results

In this section, we evaluate the empirical performance of the proposed double updating online learning algorithms for online learning tasks.

4.3.1 Testbeds and Experimental Setup

Table 4.1 shows the multiclass data sets from Web machine learning repository used in our experiments. We compare the proposed M-DUOL algorithm with several state-of-the-art online learning algorithms, including:

- “Max”: the variant of Perceptron based on the **Max**-score multiclass update method studied in [CS03],
- “Uniform”: the variant of Perceptron based on the **Uniform** multiclass update method studied in [CS03],
- “Prop”: the variant of Perceptron based on the **Proportion** multiclass update method studied in [CS03],
- “MIRA”: the **M**argin **I**nfused **R**elaxed **A**lgorithm proposed by [CS03],

- “PA-I”: the **P**assive-**A**ggressive-I algorithm , proposed in [CDK⁺06],
- “PA-II”: the **P**assive-**A**ggressive-II algorithm , proposed in [CDK⁺06].

Similar to the experiments of binary classification, we implement three variants of the proposed M-DUOL algorithm based on different solvers to the problem in (4.4), that is, “M-DUOL_{appr}”, “M-DUOL”, and “M-DUOL_{iter}”. For all experiments, we use the Gaussian kernel with $\sigma = 8$ and set $C = 10$. The threshold ρ in the proposed algorithms is set to 0 for all experiments. All the experiments were repeated 20 times on 20 random permutations of the datasets and the final results are averaged over these 20 runs.

data set	# training examples	# classes	# features
vehicle	846	4	18
dna	2,000	3	180
segment	2,310	7	19
satimage	4,435	6	36
usps	7,291	10	256
mnist	10,000	10	780
letter	15,000	26	16
protein	17,766	3	357

Table 4.1: Multiclass data sets used in the experiments.

4.3.2 Performance Evaluation

Table 4.2 summarizes the empirical performance for multi-class online learning. Several observations can be drawn from the experimental results.

First, by comparing all the baseline algorithms, we find that the two PA algorithms yield considerably lower mistake rates than the other single-updating online learning algorithms. On the other hand, the classifiers learned by the three Perceptron-based algorithms (Max, Uniform, and Prop) and MIRA are considerably sparser than those learned by the two PA algorithms. We believe that this can be attributed to the aggressive updating strategies used by the PA algorithms. Second, among the three variants of double updating for multi-label learning, it is not surprising to observe that M-DUOL yields the lowest mistake rates for all data sets. Further, among all the algorithms, we observe that the M-DUOL algorithm makes the least number of mistakes for all data sets, and significantly outperforms all the baseline algorithms.

Second, by examining the sparsity of classifiers learned by the proposed algorithms, we observe that the number of support vectors identified by M-DUOL is usually smaller

CHAPTER 4. MULTICLASS DOUBLE UPDATING ONLINE LEARNING

Algorithm	vehicle				dna		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	64.882 $\pm$ 1.643	548.90 $\pm$ 13.90	0.079		20.460 $\pm$ 0.770	409.20 $\pm$ 15.41	0.192
Uniform	65.934 $\pm$ 1.554	557.80 $\pm$ 13.15	0.109		19.875 $\pm$ 0.427	397.50 $\pm$ 8.54	0.264
Prop	66.678 $\pm$ 1.757	564.10 $\pm$ 14.86	0.116		20.268 $\pm$ 0.555	405.35 $\pm$ 11.10	0.267
MIRA	62.252 $\pm$ 2.114	526.65 $\pm$ 17.89	1.821		26.920 $\pm$ 0.880	538.40 $\pm$ 17.61	5.304
PA-I	67.086 $\pm$ 1.479	781.70 $\pm$ 12.42	0.091		15.503 $\pm$ 0.474	1224.35 $\pm$ 13.48	0.326
PA-II	66.909 $\pm$ 1.475	789.30 $\pm$ 10.73	0.089		15.398 $\pm$ 0.467	1237.50 $\pm$ 13.12	0.325
M-DUOL _{iter}	70.674 $\pm$ 1.194	758.05 $\pm$ 8.65	0.162		11.668 $\pm$ 0.599	1086.00 $\pm$ 16.39	0.502
M-DUOL _{appr}	69.634 $\pm$ 1.463	828.05 $\pm$ 4.48	0.158		14.105 $\pm$ 0.611	1281.75 $\pm$ 14.44	0.495
M-DUOL	51.950 $\pm$ 1.948	719.25 $\pm$ 10.95	0.172		10.340 $\pm$ 0.513	869.80 $\pm$ 12.61	0.438
Algorithm	segment				satimage		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	41.342 $\pm$ 1.013	955.00 $\pm$ 23.40	0.414		29.628 $\pm$ 0.561	1314.00 $\pm$ 24.89	0.826
Uniform	41.468 $\pm$ 0.550	957.90 $\pm$ 12.71	0.566		28.440 $\pm$ 0.398	1261.30 $\pm$ 17.64	1.071
Prop	41.589 $\pm$ 0.714	960.70 $\pm$ 16.48	0.565		28.878 $\pm$ 0.467	1280.75 $\pm$ 20.72	1.087
MIRA	35.784 $\pm$ 3.770	826.55 $\pm$ 87.08	9.193		27.536 $\pm$ 2.228	1221.20 $\pm$ 98.80	15.229
PA-I	39.775 $\pm$ 0.665	1852.75 $\pm$ 19.90	0.573		27.377 $\pm$ 0.361	2676.40 $\pm$ 24.88	1.296
PA-II	39.842 $\pm$ 0.655	1870.70 $\pm$ 18.97	0.577		27.258 $\pm$ 0.429	2709.50 $\pm$ 23.77	1.307
M-DUOL _{iter}	41.416 $\pm$ 1.084	1787.90 $\pm$ 31.00	0.903		33.894 $\pm$ 0.567	2787.45 $\pm$ 43.18	2.024
M-DUOL _{appr}	39.314 $\pm$ 0.791	1923.60 $\pm$ 14.31	0.871		26.222 $\pm$ 0.464	3052.50 $\pm$ 31.39	2.024
M-DUOL	20.580 $\pm$ 0.705	1265.15 $\pm$ 28.39	0.693		22.524 $\pm$ 0.482	2066.85 $\pm$ 32.99	1.505
Algorithm	usps				mnist		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	10.025 $\pm$ 0.195	730.90 $\pm$ 14.21	1.459		15.318 $\pm$ 0.168	1531.80 $\pm$ 16.80	2.744
Uniform	9.445 $\pm$ 0.150	688.60 $\pm$ 10.91	1.858		14.603 $\pm$ 0.201	1460.25 $\pm$ 20.15	3.631
Prop	9.614 $\pm$ 0.176	700.95 $\pm$ 12.86	1.868		14.763 $\pm$ 0.228	1476.30 $\pm$ 22.78	3.635
MIRA	11.572 $\pm$ 0.403	843.75 $\pm$ 29.39	44.663		18.037 $\pm$ 0.539	1803.70 $\pm$ 53.93	67.168
PA-I	6.641 $\pm$ 0.158	2528.45 $\pm$ 23.48	2.669		11.026 $\pm$ 0.208	4773.70 $\pm$ 32.84	5.771
PA-II	6.568 $\pm$ 0.116	2561.95 $\pm$ 27.94	2.606		10.959 $\pm$ 0.238	4830.40 $\pm$ 27.06	5.824
M-DUOL _{iter}	5.743 $\pm$ 0.158	2284.15 $\pm$ 40.06	3.160		8.947 $\pm$ 0.182	4398.95 $\pm$ 46.46	9.031
M-DUOL _{appr}	6.002 $\pm$ 0.132	2725.40 $\pm$ 23.55	3.541		9.640 $\pm$ 0.164	5163.05 $\pm$ 37.34	10.386
M-DUOL	5.162 $\pm$ 0.149	1759.30 $\pm$ 23.44	2.408		8.282 $\pm$ 0.183	3557.15 $\pm$ 25.17	7.050
Algorithm	letter				protein		
	Mistake (%)	Support Vectors (#)	Time (s)		Mistakes (%)	Support Vectors (#)	Time (s)
Max	71.562 $\pm$ 0.538	10734.35 $\pm$ 80.63	18.749		47.657 $\pm$ 0.221	8466.75 $\pm$ 39.21	12.842
Uniform	71.973 $\pm$ 0.280	10795.90 $\pm$ 41.99	47.031		46.828 $\pm$ 0.272	8319.45 $\pm$ 48.36	14.342
Prop	72.033 $\pm$ 0.273	10804.95 $\pm$ 40.89	43.683		47.260 $\pm$ 0.260	8396.15 $\pm$ 46.13	14.620
MIRA	67.709 $\pm$ 1.196	10156.35 $\pm$ 179.54	467.019		47.905 $\pm$ 0.922	8510.80 $\pm$ 163.74	42.174
PA-I	72.283 $\pm$ 0.338	14708.55 $\pm$ 15.27	24.848		47.657 $\pm$ 0.230	14153.25 $\pm$ 49.06	23.409
PA-II	72.339 $\pm$ 0.380	14735.55 $\pm$ 15.86	24.131		47.550 $\pm$ 0.285	14285.85 $\pm$ 44.94	23.602
M-DUOL _{iter}	73.066 $\pm$ 0.326	14614.65 $\pm$ 22.26	210.684		50.070 $\pm$ 0.392	14191.85 $\pm$ 64.80	55.622
M-DUOL _{appr}	69.992 $\pm$ 0.331	14892.70 $\pm$ 11.77	215.587		51.459 $\pm$ 0.582	16000.55 $\pm$ 72.07	63.065
M-DUOL	54.068 $\pm$ 0.351	13140.40 $\pm$ 37.33	186.452		46.281 $\pm$ 0.418	12550.10 $\pm$ 87.27	43.774

Table 4.2: Evaluation of multiclass online learning algorithms on the multiclass data sets.

than that of the PA algorithms (except for data set “vehicle”), but is significantly larger than those of the four non-aggressive algorithms (i.e., Max, Uniform, Prop, and MIRA).

Finally, comparing the running time cost, we observe that the Max algorithm is the most efficient one, while MIRA is the least efficient approach for all the data sets. More importantly, we found that the running time of the proposed M-DUOL algorithm is comparable to those of the two PA algorithms (except for the “letter” data set where the time costs of the M-DUOL algorithms are considerably greater than those of the PA algorithms). This demonstrates that despite the additional time needed for double updates, the proposed M-DUOL overall enjoys the same time complexity with regular kernel based online learning, as discussed in the subsection 4.2.4.

4.4 Summary

This chapter provided the double updating online learning method for the multiclass multilabel classification problem. To find the auxiliary example, we defined a new kind of conflict, which is different from the previous binary case. After the determination of auxiliary example, we adopted the primal-dual framework to decide how to update the weights of the current misclassified and the auxiliary examples. Furthermore, theoretical analysis was provided and extensive experiments demonstrated the effectiveness of the proposed algorithm.

Although DUOL algorithms achieved significant accuracy, it still suffer from a critical shortcoming, that is the number of stored SVs is unbounded, which will result in high computation and storage costs. To tackle this issue, in the next chapter, we extend the proposed double updating framework to the budgeted online learning, which is to minimize the number of online mistakes and at the same time keep the number of stored SVs less than a pre-defined number.

Chapter 5

Budget Double Updating Online Learning

Kernel-based online learning [KSW04, CVS⁺06, CDK⁺06] often exhibits promising empirical performance for various applications according to previous studies. However, it often suffers a main shortcoming, that is, the unbounded number of support vectors, making it unsuitable for handling large-scale datasets. In this chapter, we investigate the problem of budget kernel-based online learning that aims to constrain the number of support vectors by a predefined budget when learning the kernel-based prediction function in the online learning process. Unlike the existing studies, we present a new framework of budget kernel-based online learning based on the proposed “Double Updating Online Learning” (DUOL) in Chapter 3, which has shown state-of-the-art performance as compared with the other traditional kernel-based online learning algorithms. We analyze the theoretical underpinnings of the proposed Budget Double Updating Online Learning (BDUOL) framework, and then propose several BDUOL algorithms by designing different budget maintenance strategies. We evaluate the empirical performance of the proposed BDUOL algorithms by comparing them with several well-known budget kernel-based online learning algorithms, in which encouraging results validate the efficacy of the proposed technique.

The rest of this chapter is organized as follows. Section 5.1 concisely presents some related work. Section 5.2 first introduces the problem setting and then presents both theoretical and algorithmic framework of the proposed budget online learning technique. Section 5.3 presents several different budget maintenance strategies for BDUOL. Section 5.4 discusses our empirical studies. Section 5.5 summarizes this chapter.

5.1 Related Work

Several algorithms have been proposed for budget online learning in literature.

Yaakov Engel et al. [EMM02] presented the first kernel based sparse online regression algorithm. This algorithm only adds the new instance into the support vectors set, if the feature space image of the new instance is linearly independent of the images of those existing support vectors. Otherwise, the image of the new instance will be projected into the space spanned by the images of those existing support vectors. However, this method can not guarantee that the number of stored support vectors is bounded by a pre-fixed number.

Crammer et al. [CKS03] proposed a heuristic approach for budget online learning by extending the classical kernel-based perceptron method [Ros58], which was further improved in [WB05]. The basic idea of these two algorithms is to remove the support vector that has the least impact on the classification performance whenever the budget, i.e., the maximal number of support vectors, is reached. The main shortcoming of these two algorithms is that they are heuristic without solid theoretic supports (e.g., no any mistake/regret bound was given).

Forgetron [DSSS05] is perhaps the first approach for budget online learning that offers a theoretical bound of the total number of mistakes. In particular, at each iteration, if the online classifier makes a mistake, it conducts a three-step updating: (i) it first runs the standard Perceptron update [Ros58] for updating the prediction function; (ii) it then shrinks the weights of support vectors by a carefully chosen scaling factor; and (iii) it finally removes the support vector with the least weight. Another similar approach is the Randomized Budget Perceptron (RBP) [CCBG07], which randomly removes one of existing support vectors when the number of support vectors exceeds the predefined budget. In general, RBP achieves similar mistake bound and empirical performance as Forgetron.

Unlike the above strategy that discards one support vector to maintain the budget, Projectron [OKC08] adopts a projection strategy to bound the number of support vectors. Specifically, at each iteration where a training example is misclassified, it first updates the kernel classifier by applying a standard Perceptron update; it then projects the new classifier into the space spanned by all the support vectors except the new example received at the current iteration. The classifier would remain unchanged if the difference between the new classifier and its projection is smaller than a given threshold. Empirical studies show that Projectron usually outperforms Forgetron in classification but with significantly longer running time. In addition to its high computational cost, another shortcoming of Projectron is that although the number of support vectors is bounded, it is unclear the exact number of support vectors achieved by Projectron in theory.

All the above algorithms for classification are based on Perceptron, while Budgeted PA [WV10] is a budget algorithm based on the Passive-Aggressive algorithm. By requiring the number of stored support vectors less than a pre-defined budget on the original

optimization problem, a closed-form solution can be derived. Besides, several more special constraints are introduced into the objective, so that the solutions are similar with the previous research results: removal, projection and merging.

Budget Multi-Class Pegasos algorithm [WCV10], is based on the multi-class version of the popular Pegasos algorithm [SSSS07]. This algorithm adopts budget maintenance to keep the number of stored support vectors bounded. It is further proved that the gap between the budgeted Pegasos and the optimal solution directly depends on model degradation caused by the budget maintenance. To minimize the model degradation, several effective budget maintenance methods are proposed based on removal, projection, and merging of support vectors. However, this algorithm mainly aims to online learn a budget solution of SVM, while our algorithm aims to minimize the number of online mistakes.

Most of the above budget online learning approaches were designed based on the Perceptron learning framework [Ros58]. In this chapter, we propose a new framework of Budget Double Updating Online Learning (BDUOL) based on the proposed Double Updating Online Learning (DUOL) technique in the Chapter 3, which has shown state-of-the-art performance for online learning. The key challenge is to develop an appropriate strategy for maintaining the budget whenever the size of support vectors overflows. In this chapter, following the theory of double updating online learning, we analyze the theoretical underpinning of the BDUOL framework, and prove the upper bound of the number of mistakes depends directly on the dual ascent caused by the budget maintenance, which is different from the previous analyses techniques. To maximize the dual ascent, we propose a principled approach to developing three different budget maintenance strategies: removal, projection and nearest neighbor, which are similar with the previous budget maintenance strategies. We also analyze the mistake bounds of the proposed BDUOL algorithms and evaluate their empirical performance extensively.

5.2 Double Updating Online Learning on A Fixed Budget

In this section, we first introduce the problem setting for online learning, and then present the details of the proposed Budget Double Updating Online Learning framework.

5.2.1 Problem Setting

The problem setting here is exactly the same with the one in the chapter 3.

Algorithm 6 The Algorithms of Budget Double Updating Online Learning (BDUOL).

```
Initialize  $S_0 = \emptyset, f_0 = 0$ ;  
for  $t = 1, 2, \dots, T$  do  
  Receive a new instance  $\mathbf{x}_t$ ;  
  Predict  $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$ ;  
  Receive its label  $y_t$ ;  
   $\ell_t = \max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$ ;  
  if  $\ell_t > 0$  then  
    if  $(|S_t| == B)$  then  
       $f_{t-1} = f_{t-1} - \Delta f_{t-1}$ ; (Budget Maintenance)  
    end if  
     $\ell_t = \max\{0, 1 - y_t f_{t-1}(\mathbf{x}_t)\}$ ;  
    if  $\ell_t > 0$  then  
       $w_{\min} = \infty$ ;  
      for  $\forall i \in S_{t-1}$  do  
        if  $(f_{t-1}^i \leq 1)$  then  
          if  $(y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{\min})$  then  
             $w_{\min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$ ;  
             $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$ ;  
          end if  
        end if  
      end for  
       $f_{t-1}^t = y_t f_{t-1}(\mathbf{x}_t)$ ;  
       $S_t = S_{t-1} \cup \{t\}$ ;  
      if  $(w_{\min} \leq -\rho)$  then  
        Compute  $\gamma_t$  and  $d_{\gamma_b}$  by solving the optimization (3.1);  
        for  $\forall i \in S_t$  do  
           $f_t^i \leftarrow f_{t-1}^i + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) + y_i d_{\gamma_b} y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$ ;  
        end for  
         $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} y_b \kappa(\mathbf{x}_b, \cdot)$ ;  
      else  
         $\gamma_t = \min(C, \ell_t / \kappa(\mathbf{x}_t, \mathbf{x}_t))$ ;  
        for  $\forall i \in S_t$  do  
           $f_t^i \leftarrow f_{t-1}^i + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$ ;  
        end for  
         $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$ ;  
      end if  
    else  
       $f_t = f_{t-1}; S_t = S_{t-1}$ ;  
      for  $\forall i \in S_t$  do  
         $f_t^i \leftarrow f_{t-1}^i$ ;  
      end for  
    end if  
  end if  
end for  
Return  $f_T, S_T$ 
```

5.2.2 Framework of Budget Double Updating Online Learning

Let us denote by B a predefined budget size for the maximal number of support vectors associated with the prediction function. The key difference of Budget DUOL over regular DUOL is to develop an appropriate budget maintenance step to ensure that the number of support vectors with the classifier f_t is less than the budget B at the beginning of each online updating step. In particular, let us denote by f_{t-1} the classifier produced by a regular DUOL at the $t-1$ -th step, when the number of support vectors of f_{t-1} is equal to the B , BDUOL performs the classifier update towards budget maintenance: $f_{t-1} \leftarrow f_{t-1} - \Delta f_{t-1}$ such that the number of support vectors of the updated f_{t-1} is smaller than B . The details of the proposed Budget DUOL (BDUOL) algorithm are summarized in Algorithm 6.

The key challenge of BDUOL is to choose an appropriate reduction term Δf_{t-1} by a proper budget maintenance strategy, which can only meet the budget requirement but also minimize the impact of the reduction on the prediction performance. Unlike some existing heuristic budget maintenance approaches, in this chapter, we propose a principled approach for developing several different budget maintenance strategies. Before presenting the detailed strategies, in the following, we analyze the theoretical underpinning of the proposed budget double updating online learning scheme, which is the theoretical foundation for the proposed budget maintenance strategies in Section 5.3.

5.2.3 Theoretical Analysis

In this subsection, we analyze the mistake bound of the proposed BDUOL algorithm. To simplify the analysis, the primal-dual framework is used to derive the mistake bound following the strategy of DUOL algorithm. Through this framework, we will show that the gap between the mistake bound of DUOL and BDUOL is bounded by the cumulative dual ascent induced by the function reduction, i.e., $\Delta f = \sum \Delta \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)$.

At the t -th budget maintenance, the resultant dual ascent will be computed as follows:

Theorem 11 *The **Dual Ascent** $DA_t = \mathcal{D}_t(\gamma_1 - \Delta \gamma_1, \dots, \gamma_t - \Delta \gamma_t) - \mathcal{D}_t(\gamma_1, \dots, \gamma_t)$ for the t -th budget maintenance, i.e., $f_t = f_t - \Delta f_t$, is given as follows:*

$$DA_t = - \sum_{i=1}^t \Delta \gamma_i + \sum_{i=1}^t \Delta \gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2} \|\Delta f_t\|_{\mathcal{H}_\kappa}^2.$$

Proof: The dual objective of $\mathcal{P}_t(f) = \frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^t \ell(f(\mathbf{x}_i), y_i)$, $C > 0$ is

$$\mathcal{D}_t(\gamma_1, \dots, \gamma_t) = \sum_{i=1}^t \gamma_i - \frac{1}{2} \left\| \sum_{i=1}^t \gamma_i y_i \kappa(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}_\kappa}^2, \quad \gamma_i \in [0, C],$$

where the relation between f and γ_i , $i = 1, \dots, t$ is $f(\cdot) = \sum_{i=1}^t \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)$.

$$\begin{aligned}
 & \mathcal{D}_t(\gamma_1 - \Delta\gamma_1, \dots, \gamma_t - \Delta\gamma_t) - \mathcal{D}_t(\gamma_1, \dots, \gamma_t) \\
 &= \sum_{i=1}^t (\gamma_i - \Delta\gamma_i) - \frac{1}{2} \left\| \sum_{i=1}^t (\gamma_i - \Delta\gamma_i) y_i \kappa(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}_\kappa}^2 - \left[\sum_{i=1}^t \gamma_i - \frac{1}{2} \left\| \sum_{i=1}^t \gamma_i y_i \kappa(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}_\kappa}^2 \right] \\
 &= - \sum_{i=1}^t \Delta\gamma_i + \frac{1}{2} [\|f_t\|_{\mathcal{H}_\kappa}^2 - \|f_t - \Delta f_t\|_{\mathcal{H}_\kappa}^2] \\
 &= - \sum_{i=1}^t \Delta\gamma_i + \frac{1}{2} [\|f_t\|_{\mathcal{H}_\kappa}^2 - \|f_t\|_{\mathcal{H}_\kappa}^2 + 2\langle f_t, \Delta f_t \rangle - \|\Delta f_t\|_{\mathcal{H}_\kappa}^2] \\
 &= - \sum_{i=1}^t \Delta\gamma_i + \langle f_t, \Delta f_t \rangle - \frac{1}{2} \|\Delta f_t\|_{\mathcal{H}_\kappa}^2 \\
 &= - \sum_{i=1}^t \Delta\gamma_i + \langle f_t, \sum_{i=1}^t \Delta\gamma_i y_i \kappa(\mathbf{x}_i, \cdot) \rangle - \frac{1}{2} \|\Delta f_t\|_{\mathcal{H}_\kappa}^2 \\
 &= - \sum_{i=1}^t \Delta\gamma_i + \sum_{i=1}^t \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2} \|\Delta f_t\|_{\mathcal{H}_\kappa}^2.
 \end{aligned}$$

Based on the above dual ascent for budget maintenance, we can now analyze the mistake bound of the proposed BDUOL algorithm.

Theorem 12 *Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ (T may not be available in advance) be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t , and assume $C \geq 1$. Then the number of prediction mistakes M made by BDUOL on this sequence of examples is bounded by:*

$$2 \min_{f \in \mathcal{H}_\kappa} \left(\frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^T \ell(f(\mathbf{x}_i), y_i) \right) - 2 \sum_{i=1}^T DA_i - \rho^2 M_d^w(\rho) - \frac{1+\rho}{1-\rho} M_d^s(\rho),$$

where $\rho \in [0, 1)$, $M_d^w(\rho)$, $M_d^s(\rho)$ is defined as in the Chapter 3.

Proof: According to Theorem 2 and 4, we have

$$\min_{t \in \mathcal{M}_d^s(\rho)} \Delta_t \geq \frac{1}{1-\rho}, \quad \min_{t \in \mathcal{M}_d^w(\rho)} \Delta_t \geq \frac{1+\rho^2}{2}.$$

Moreover, according to Theorem 3, we have $\Delta_t \geq 1/2, \forall t \in \mathcal{M}$. Taking the dual ascents of budget maintenance into consideration and combining these results with Theorem 1 together, we have

$$\sum_{i=1}^T DA_i + \frac{M_s}{2} + \frac{1+\rho^2}{2} M_d^w(\rho) + \frac{M_d^s(\rho)}{1-\rho} \leq \min_{f \in \mathcal{H}_\kappa} \left(\frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^T \ell(f(\mathbf{x}_i), y_i) \right).$$

We complete the proof using $M = M_s + M_d^w(\rho) + M_d^s(\rho)$.

According to the above theorem, we can see that, if there is no budget maintenance step, the mistake bound of BDUOL is reduced to the previous mistake bound for the regular DUOL algorithm. This theorem indicates that in order to minimize the mistake bound of BDUOL, one should try to maximize the cumulative dual ascent, i.e., $\sum_{i=1}^T DA_i$, when designing an appropriate budget maintenance strategy.

5.3 Budget Maintenance Strategies

In this section, we follow the above theoretical results to develop several different budget maintenance strategies in a principled approach. In particular, as revealed by Theorem 12, a key to improving the mistake bound of BDUOL is to maximize the cumulative dual ascent $\sum_{t=1}^T DA_t$ caused by the budget maintenance. To achieve this purpose, we propose to maximize the dual ascent caused by budget maintenance at each online learning step, i.e.,

$$\max_{\Delta\gamma_1, \dots, \Delta\gamma_t} DA_t = - \sum_{i=1}^t \Delta\gamma_i + \sum_{i=1}^t \Delta\gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2} \|\Delta f_t\|_{\mathcal{H}_\kappa}^2.$$

Below, we propose three different budget maintenance strategies, and analyze the principled approach of achieving the best dual ascent as well as the time complexity and memory cost for each strategy.

5.3.1 BDUOL Algorithm by Removal Strategy

The first strategy for budget maintenance is the removal strategy that discards one of existing support vectors, which is similar to the strategies used by Forgetron [DSSS08] and RBP [CCBG07]. Unlike the previous heuristic removal strategy, the key idea of our removal strategy is to discard the support vector which can maximize the dual ascent by following our previous analysis.

Specifically, let us assume the j -th SV is selected for removal. We then have the following function reduction term:

$$\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot).$$

As a result, the optimal removal solution is to discard the SV which can maximize the following dual ascent term:

$$DA_{t,j} = -\gamma_j(1 - y_j f_t(\mathbf{x}_j)) - \frac{1}{2}(\gamma_j)^2 \kappa(\mathbf{x}_j, \mathbf{x}_j).$$

We note that the above removal strategy is similar with the one in [CKS03], when the Gaussian kernel is adopted where $\kappa(\mathbf{x}_j, \mathbf{x}_j) = 1$. However, our strategy is strongly theoretically motivated.

Complexity Analysis. Since the BDUOL algorithm will cache the value $y_j f_t(\mathbf{x}_j)$ for every SV, the computational complexity of $DA_{t,j}$ is $O(1)$ in practice. Thus, this BDUOL algorithm requires $O(B)$ time complexity of computing all the $DA_{t,j}$'s. After removing the SV with the largest $DA_{t,j}$, the complexity of updating all the values of $y_j f_t(\mathbf{x}_j)$ is $O(B)$. Furthermore, combining the above discussion with the fact that the original DUOL's time complexity is $O(B)$, we can conclude that the overall time complexity of this BDUOL algorithm is also $O(B)$. As for the memory cost, since only B SVs, their weight parameters and the $y_j f_t(\mathbf{x}_j)$ s have to be cached, the space complexity is thus also $O(B)$.

5.3.2 BDUOL Algorithm by Projection Strategy

Although the above removal strategy is optimized to find the best support vector that maximizes the dual ascent (i.e., minimizes the loss of dual ascent caused by budget maintenance), it is still unavoidable to result in the loss of the dual ascent due to the removal of one existing support vector. To minimize such loss, we propose a projection strategy for budget maintenance.

Specifically, in the projection strategy, the selected j -th SV for removal will be projected to the space spanned by the rest SVs, which is inspired by the Projectron algorithm [OKC08]. The objective is to find the function closest to $\gamma_j y_j \kappa(\mathbf{x}_j, \cdot)$ in the space spanned by the remaining SVs, or formally:

$$\min_{\beta_i \in [-\gamma_i, C - \gamma_i], i \neq j} \left\| \gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \sum_{i \neq j} \beta_i y_i \kappa(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}_\kappa}^2.$$

which is essentially a Quadratic Programming (QP) problem. So, we can exploit the existing efficient QP solvers to find the optimal solution.

However, solving the above QP problem directly may not be efficient enough for online learning purpose. To further improve the efficiency, we also proposed an approximate solution by firstly solving the unconstrained optimization problem and then projecting the solution into the feasible region of the constraints. Specifically, setting the gradient of the above equation with respect to $\bar{\beta} = [\beta_i]^\top, i \neq j$ as zero, one can obtain the optimal solution as

$$\bar{\beta} = \gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j / \bar{y},$$

where $\mathbf{K}$ is the kernel matrix for $\mathbf{x}_i, i \neq j$, $\mathbf{k}_j = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]^\top$, $i \neq j$, $\cdot/\cdot$ is element-wise division and $\bar{y} = [y_i]^\top$, $i \neq j$. In the above, inverting $\mathbf{K}$ can be efficiently realized by using Woodbury formula [CP00]. As a result, we should set

$$\Delta f_t(\cdot) = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \sum_{i \neq j} \beta_i y_i \kappa(\mathbf{x}_i, \cdot).$$

However, the resultant $f_t - \Delta f_t$'s SV weights may not fall into the range $[0, C]$. To fix this problem, we will project each β_i as follows:

$$\bar{\beta} = \Pi_{[-\bar{\gamma}, C-\bar{\gamma}]}(\gamma_j y_j \mathbf{K}^{-1} \mathbf{k}_j / \bar{y}), \quad (5.1)$$

where $\Pi_{[a,b]}(x) = \max(a, \min(b, x))$ and $\bar{\gamma} = [\gamma_i]^\top$, $i \neq j$.

Now the key problem is to find the best SV among $B+1$ candidates for projection. Since after the projection of $\gamma_j y_j \kappa(\mathbf{x}_j, \cdot)$, the resultant dual ascent is

$$DA_{t,j} = - \sum_i \Delta \gamma_i + \sum_i \Delta \gamma_i y_i f_t(\mathbf{x}_i) - \frac{1}{2} \|\Delta f_t\|_{\mathcal{H}_\kappa}^2.$$

So the best SV is the one which can achieve the largest value $DA_{t,j}$.

Complexity Analysis. The time complexity for BDUOL is dominated by the computation K^{-1} . Computing K^{-1} using K will cost $O(B^3)$ time, however using Woodbury formula, we can efficiently compute it using only $O(B^2)$ time. In addition, we need to compute B times projections for every SV, so the total time complexity for one step of updating is $O(B^3)$. Finally, the memory burden for the BDUOL algorithm is $O(B^2)$, since the storage of kernel matrix K and inverse kernel matrix K^{-1} dominated the main memory cost.

5.3.3 BDUOL Algorithm by Nearest Neighbor Strategy

The above projection strategy is able to achieve a better improvement of dual ascent than the removal strategy, it is however much more computationally expensive. To balance the tradeoff between efficiency and effectiveness, we propose an efficient nearest neighbor strategy which approximates the projection strategy by projecting the removed SV to its nearest neighbor SV, based on the distance in the mapped feature space. This strategy is motivated by the fact that the nearest neighbor SV usually could be a good representative of the removed SV. Using this strategy, we can significantly improve the time efficiency. In particular, as we use only the nearest neighbor for projection, the corresponding solution according to equation 5.1 can be expressed:

$$\beta_{N_j} = \Pi_{[-\gamma_{N_j}, C-\gamma_{N_j}]}(\gamma_j y_j \kappa(\mathbf{x}_{N_j}, \mathbf{x}_{N_j})^{-1} \kappa(\mathbf{x}_{N_j}, \mathbf{x}_j) / y_{N_j}),$$

where $\kappa(\mathbf{x}_{N_j}, \cdot)$ is the nearest neighbor of $\kappa(\mathbf{x}_j, \cdot)$. As a result, the corresponding $\Delta f_t = \gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \beta_{N_j} y_{N_j} \kappa(\mathbf{x}_{N_j}, \cdot)$. Since after the projection of $\gamma_j y_j \kappa(\mathbf{x}_j, \cdot)$, the resultant dual ascent is

$$DA_{t,j} = -(\gamma_j - \beta_{N_j}) + (\gamma_j y_j f_t(\mathbf{x}_j) - \beta_{N_j} y_{N_j} f_t(\mathbf{x}_{N_j})) - \frac{1}{2} \|\gamma_j y_j \kappa(\mathbf{x}_j, \cdot) - \beta_{N_j} y_{N_j} \kappa(\mathbf{x}_{N_j}, \cdot)\|_{\mathcal{H}_\kappa}^2.$$

Thus, the best SV is the one that has the largest value $DA_{t,j}$.

Complexity Analysis. All the computation steps except looking for the nearest neighbor have constant time complexity of $O(1)$. For improving nearest neighbor searching, we can cache the indexes and the distances about the nearest neighbors of the SVs, making finding the nearest neighbor in $O(1)$. After remove the best SV, we should also update the caches. The time complexity for this updating is at most $O(B)$. In summary, the time complexity for the overall strategy is $O(B)$, and the overall memory cost is also $O(B)$.

Remark: To improve the efficiencies of the projection and nearest neighbor strategies, we could use the projection or the nearest neighbor strategy to keep the information from the SV selected by the removal strategy and then remove it, since the search cost of the removal strategy is quite lower than the other two strategies.

5.4 Experimental Results

In this section, we evaluate the empirical performance of the proposed algorithms for Budget Double Updating Online Learning (BDUOL) by comparing them with the state-of-the-art algorithms for budget online learning.

5.4.1 Algorithms for Comparison

In our experiments, we implement the proposed BDUOL algorithms as follows:

- “BDUOL_{remo}”: the BDUOL algorithm by the **removal** strategy for budget maintenance described in section 5.3.1,
- “BDUOL_{proj}”: the BDUOL algorithm by the exact **projection** strategy by a standard QP solver for budget maintenance described in section 5.3.2,
- “BDUOL_{appr}”: the BDUOL algorithm by the **approximate** projection strategy for budget maintenance described in section 5.3.2,
- “BDUOL_{near}”: the BDUOL algorithm by the **nearest** neighbor strategy for budget maintenance described in section 5.3.3,

For comparison, we include the following state-of-the-art algorithms for budget online learning:

- “RBP”: the Random Budget Perceptron algorithm [CCBG07],
- “Forgetron”: the Forgetron algorithm [DSSS08],
- “Projectron”: the Projectron algorithm [OKC09], and
- “Projectron++”: the aggressive version of Projectron algorithm [OKC09].

Besides, we also include two non-budget online learning algorithms as yardstick:

- “Perceptron”: the classical Perceptron algorithm [Ros58], and
- “DUOL”: the Double Updating Online Learning algorithm [ZHJ11].

5.4.2 Experimental Testbed and Setup

Table 5.1: Details of the datasets in our experiments.

Dataset	# instances	# features
german	1000	24
MITface	6977	361
mushrooms	8124	112
spambase	4601	57
splice	3175	60
w7a	24692	300

We test all the algorithms on six benchmark data sets from web machine learning repositories listed in Table 5.1. These data sets can be downloaded from LIBSVM website¹, UCI machine learning repository², and MIT CBCL face data sets³. These datasets were chosen fairly randomly to cover various sizes of datasets.

To make a fair comparison, all the algorithms in our comparison adopt the same experimental setup. A gaussian kernel is adopted in our study, for which the kernel width is set to 8 for all the algorithms and datasets. To make the number of support vectors fixed for Projectron and Projectron++ algorithms, we simply store the received SVs before the budget overflows, and then project the subsequent ones into the space spanned by the stored SVs afterward. The penalty parameter C in the DUOL algorithm was selected by 5-fold cross validation for all the datasets from range $2^{[-10:10]}$. Due to the cross-validation, we randomly divide every dataset into two equal subsets: cross

¹<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>.

²<http://www.ics.uci.edu/~mllearn/MLRepository.html>.

³<http://cbcl.mit.edu/software-datasets>.

validation (CV) dataset and online learning dataset. And C is set as the same value with DUOL. Furthermore, the value ρ for the DUOL and its budget variants is set as 0, according to the previous study on the effect of ρ .

The budget sizes B for different datasets are set as proper fractions of the number of support vectors of Perceptron, which are shown in Table 5.3. All the experiments were conducted 20 times, each with a different random permutation of data points. All the results were reported by averaging over the 20 runs. For performance metrics, we evaluate the online classification performance by evaluating online cumulative mistake rates and running time cost.

5.4.3 Performance Evaluation of Non-budget Algorithms

Table 5.2 summarizes the average performance of the two non-budget algorithms for kernel-based online learning. First of all, similar to the previous study [ZHJ11], we found that DUOL outperforms Perceptron significantly for all the datasets according to t-test results, which validates our motivation of choosing DUOL as the basic online learning algorithm for budget online learning. Second, we noticed that the number of support vectors of DUOL is in general much larger than that of Perceptron. Finally, the time cost of DUOL is much higher than that of Perceptron, mostly due to the larger number of support vectors. Both the large number of support vectors and high computational time motivate the need of studying budget DUOL algorithms in this work.

Algorithm	Perceptron			DUOL		
Datasets	Mistake (%)	Support Vectors (#)	Time (s)	Mistakes (%)	Support Vectors (#)	Time (s)
german	35.760 % $\pm$ 1.149	178.800 $\pm$ 5.745	0.007	29.820 % $\pm$ 1.243	381.750 $\pm$ 5.866	0.044
MITface	6.246 % $\pm$ 0.252	217.85 $\pm$ 8.774	0.04	2.418 % $\pm$ 0.156	408.5 $\pm$ 9.339	0.114
mushrooms	3.175 % $\pm$ 0.463	128.950 $\pm$ 18.805	0.040	0.591 % $\pm$ 0.086	247.050 $\pm$ 14.084	0.099
spambase	27.354 % $\pm$ 0.561	629.150 $\pm$ 12.906	0.050	22.680 % $\pm$ 0.557	1385.800 $\pm$ 17.519	0.317
splice	21.808 % $\pm$ 0.709	346.100 $\pm$ 11.257	0.026	15.633 % $\pm$ 0.461	777.450 $\pm$ 11.551	0.130
w7a	4.366 % $\pm$ 0.093	539.000 $\pm$ 11.530	0.272	3.068 % $\pm$ 0.114	1171.600 $\pm$ 30.396	0.728

Table 5.2: Evaluation of non-budget algorithms on the the data sets.

5.4.4 Performance Evaluation of Budget Algorithms

Table 5.3 summarizes the results of different budget online learning algorithms.

We can draw several observations.

First of all, we observe that RBP and Forgetron achieve very similar performance for most cases. In addition, we also find that Projectron++ achieves a lower mistake rate

CHAPTER 5. BUDGET DOUBLE UPDATING ONLINE LEARNING

Budget Size		B=50		B=100		B=150	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
german	RBP	39.140 % $\pm$ 1.338	0.010	36.940 % $\pm$ 1.837	0.009	36.080 % $\pm$ 1.392	0.008
	Fogetron	38.840 % $\pm$ 1.551	0.014	37.250 % $\pm$ 1.399	0.013	36.570 % $\pm$ 1.641	0.013
	Projectron	36.990 % $\pm$ 1.652	0.020	36.310 % $\pm$ 1.441	0.027	35.870 % $\pm$ 1.149	0.063
	Projectron++	35.370 % $\pm$ 1.413	0.036	35.620 % $\pm$ 1.251	0.046	35.680 % $\pm$ 1.380	0.103
	BDUOL _{remo}	39.970 % $\pm$ 3.150	0.068	37.180 % $\pm$ 2.297	0.077	33.330 % $\pm$ 2.264	0.088
	BDUOL _{near}	37.090 % $\pm$ 1.763	0.098	33.330 % $\pm$ 1.697	0.112	31.360 % $\pm$ 1.511	0.129
	BDUOL _{appr}	35.540 % $\pm$ 2.010	0.160	32.340 % $\pm$ 1.570	0.381	30.450 % $\pm$ 1.338	0.941
	BDUOL _{proj}	34.030 % $\pm$ 1.104	0.214	30.710 % $\pm$ 1.261	0.624	30.330 % $\pm$ 1.221	1.341
Budget Size		B=50		B=100		B=150	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
MITface	RBP	18.964 % $\pm$ 1.330	0.063	9.557 % $\pm$ 0.615	0.053	7.463 % $\pm$ 0.609	0.050
	Fogetron	18.038 % $\pm$ 1.470	0.081	10.707 % $\pm$ 0.883	0.077	8.373 % $\pm$ 0.546	0.076
	Projectron	7.137 % $\pm$ 0.384	0.086	6.461 % $\pm$ 0.288	0.098	6.316 % $\pm$ 0.304	0.142
	Projectron++	6.135 % $\pm$ 0.312	0.149	5.973 % $\pm$ 0.215	0.197	5.978 % $\pm$ 0.219	0.373
	BDUOL _{remo}	17.516 % $\pm$ 2.264	0.225	8.096 % $\pm$ 0.811	0.208	4.700 % $\pm$ 0.449	0.194
	BDUOL _{near}	4.435 % $\pm$ 0.396	0.179	3.078 % $\pm$ 0.304	0.187	2.701 % $\pm$ 0.277	0.205
	BDUOL _{appr}	3.632 % $\pm$ 0.277	0.249	2.618 % $\pm$ 0.221	0.429	2.501 % $\pm$ 0.258	0.836
	BDUOL _{proj}	3.611 % $\pm$ 0.319	0.273	2.645 % $\pm$ 0.228	0.492	2.481 % $\pm$ 0.168	0.755
Budget Size		B=50		B=75		B=100	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
mushrooms	RBP	6.551 % $\pm$ 1.056	0.054	17.841 % $\pm$ 1.539	0.073	3.488 % $\pm$ 0.713	0.049
	Fogetron	11.273 % $\pm$ 1.750	0.085	14.154 % $\pm$ 2.748	0.094	3.895 % $\pm$ 1.141	0.079
	Projectron	4.264 % $\pm$ 0.613	0.095	3.703 % $\pm$ 0.731	0.099	3.207 % $\pm$ 0.473	0.107
	Projectron++	3.986 % $\pm$ 0.297	0.161	3.557 % $\pm$ 0.174	0.178	3.484 % $\pm$ 0.117	0.197
	BDUOL _{remo}	8.754 % $\pm$ 2.438	0.210	2.547 % $\pm$ 0.865	0.181	0.891 % $\pm$ 0.152	0.162
	BDUOL _{near}	1.329 % $\pm$ 0.198	0.141	0.710 % $\pm$ 0.095	0.132	0.618 % $\pm$ 0.081	0.131
	BDUOL _{appr}	1.065 % $\pm$ 0.218	0.193	0.667 % $\pm$ 0.092	0.194	0.623 % $\pm$ 0.080	0.248
	BDUOL _{proj}	0.729 % $\pm$ 0.060	0.230	0.604 % $\pm$ 0.080	0.266	0.577 % $\pm$ 0.071	0.290
Budget Size		B=200		B=400		B=600	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
spambase	RBP	31.826 % $\pm$ 0.924	0.065	29.220 % $\pm$ 0.550	0.069	27.417 % $\pm$ 0.598	0.059
	Fogetron	32.461 % $\pm$ 0.971	0.077	29.641 % $\pm$ 0.742	0.084	27.424 % $\pm$ 0.644	0.082
	Projectron	29.237 % $\pm$ 0.750	0.238	27.480 % $\pm$ 0.484	0.929	27.750 % $\pm$ 2.134	2.776
	Projectron++	28.822 % $\pm$ 0.725	0.819	28.693 % $\pm$ 6.781	3.874	27.559 % $\pm$ 1.572	7.277
	BDUOL _{remo}	34.559 % $\pm$ 1.308	0.522	28.989 % $\pm$ 0.927	2.013	25.661 % $\pm$ 0.709	4.541
	BDUOL _{near}	28.180 % $\pm$ 1.084	0.756	25.607 % $\pm$ 0.748	3.874	24.187 % $\pm$ 0.584	8.570
	BDUOL _{appr}	28.950 % $\pm$ 2.001	5.555	26.843 % $\pm$ 0.948	16.396	24.846 % $\pm$ 0.695	30.186
	BDUOL _{proj}	27.448 % $\pm$ 0.961	8.837	25.307 % $\pm$ 0.793	53.987	23.780 % $\pm$ 0.640	197.199
Budget Size		B=100		B=200		B=300	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
splice	RBP	30.003 % $\pm$ 1.318	0.035	25.126 % $\pm$ 0.900	0.033	22.530 % $\pm$ 0.967	0.028
	Fogetron	29.912 % $\pm$ 1.483	0.044	25.545 % $\pm$ 0.582	0.043	22.457 % $\pm$ 0.734	0.041
	Projectron	22.851 % $\pm$ 0.756	0.061	22.007 % $\pm$ 1.023	0.146	21.830 % $\pm$ 0.811	0.356
	Projectron++	22.628 % $\pm$ 0.649	0.163	21.843 % $\pm$ 1.002	0.445	21.919 % $\pm$ 1.188	0.922
	BDUOL _{remo}	26.150 % $\pm$ 1.279	0.238	22.117 % $\pm$ 1.321	0.409	18.299 % $\pm$ 0.797	0.641
	BDUOL _{near}	23.125 % $\pm$ 1.161	0.435	18.847 % $\pm$ 0.866	0.458	16.991 % $\pm$ 0.600	1.201
	BDUOL _{appr}	19.779 % $\pm$ 0.754	1.110	18.062 % $\pm$ 2.000	2.596	17.927 % $\pm$ 2.065	3.905
	BDUOL _{proj}	20.243 % $\pm$ 0.667	1.041	17.190 % $\pm$ 0.734	2.685	16.191 % $\pm$ 0.517	5.475
Budget Size		B=300		B=400		B=500	
Dataset	Algorithm	Mistake (%)	Time (s)	Mistakes (%)	Time (s)	Mistakes (%)	Time (s)
w7a	RBP	4.681 % $\pm$ 0.257	0.291	4.597 % $\pm$ 0.227	0.316	4.434 % $\pm$ 0.108	0.332
	Fogetron	4.697 % $\pm$ 0.102	0.367	4.584 % $\pm$ 0.180	0.390	4.466 % $\pm$ 0.137	0.398
	Projectron	4.680 % $\pm$ 0.290	0.727	4.625 % $\pm$ 0.456	1.174	4.406 % $\pm$ 0.130	1.929
	Projectron++	3.880 % $\pm$ 0.526	4.366	3.672 % $\pm$ 0.214	7.763	3.666 % $\pm$ 0.152	8.796
	BDUOL _{remo}	3.832 % $\pm$ 0.170	0.995	3.361 % $\pm$ 0.145	1.824	3.175 % $\pm$ 0.133	2.401
	BDUOL _{near}	3.572 % $\pm$ 0.114	1.241	3.269 % $\pm$ 0.133	3.324	3.183 % $\pm$ 0.087	4.859
	BDUOL _{appr}	4.126 % $\pm$ 0.502	4.553	4.001 % $\pm$ 0.153	6.794	3.775 % $\pm$ 0.121	9.646
	BDUOL _{proj}	3.367 % $\pm$ 0.096	11.306	3.227 % $\pm$ 0.126	18.965	3.254 % $\pm$ 0.332	29.807

Table 5.3: Evaluation of several budgeted algorithms with varied budget sizes.

than Projectron for almost all the datasets and for varied budget sizes, which is similar to the previous results reported in [OKC08]. Moreover, compared with the baseline algorithms RBP and Forgetron, the proposed BDUOL_{remo} algorithm using a simple removal strategy achieves comparable or better mistake rate when the budget size is large, but fails to improve when the budget size is very small, which indicates a simple removal strategy may not be always effective and a better budget maintenance strategy is needed.

Second, among all the algorithms in comparison for budget online learning, we find that BDUOL_{proj} always achieves the lowest mistake rates for most cases. These promising results indicate the projection strategy can effectively reduce the information loss. However, we also notice that the time cost of the BDUOL_{proj} is among the highest ones, which indicates it is important to find some more efficient strategy.

Third, by comparing two approximate strategies, we find that BDUOL_{appr} achieves better mistake rates than BDUOL_{near} only on the german and mushrooms datasets, while it consumes more time than the proposed BDUOL_{near} algorithms, which indicates BDUOL_{near} achieves better trade off between mistake rates and time complexity than BDUOL_{appr} . In addition, when the size of budget is large, BDUOL_{near} always achieves similar performance with the BDUOL_{proj} , while consumes significantly less time, which indicates the proposed nearest neighbor strategy is a good alternative of the projection strategy.

Finally, Figure 5.1 and Figure 5.2 show the detailed online evaluation processes of the several budget online learning algorithms. Similar observations from these figures further verified the efficacy of the proposed BDUOL technique. Furthermore, the curves of BDUOL_{remo} on several datasets go up at the ends of the learning processes, which indicates simply dropping one support vector is not effective. In addition, the curves of BDUOL_{appr} on w7a goes up, which implies the approximate projection may generate solutions very different from the exact projection.

5.5 Summary

This chapter presented a new framework of budget double updating online learning for kernel-based online learning on a fixed budget, which requires the number of support vectors stored is always bounded by a predefined number. We theoretically analyzed its performance, which reveals that its effectiveness is tightly connected with the dual ascent achieved by the model reduction for budget maintenance. Based on the theoretical analysis, we proposed three budget maintenance strategies: removal, projection, and nearest neighbor. We evaluate the proposed algorithms extensively on benchmark datasets. The promising empirical results show that the proposed algorithms outperform the state-of-the-art budget online learning algorithms in terms of mistake rates.

Figure 5.1: Evaluation of online mistake rates against the number of samples on three datasets. The plotted curves are averaged over 20 random permutations.

Figure 5.2: Evaluation of online mistake rates against the number of samples on three datasets. The plotted curves are averaged over 20 random permutations.

All the previous learning algorithms adopt the learning accuracy as performance measure, while other more complex cost-sensitive metrics are usually used as performance measure in the cost-sensitive learning. However, optimizing the accuracy can not guarantee the cost-sensitive metrics are optimized. To directly optimize the cost-sensitive metrics, we will formally study the cost-sensitive online learning in the next chapter.

Chapter 6

Cost-Sensitive Double Updating Online Learning

Despite being studied extensively in machine learning, most existing online learning techniques are unsuitable and potentially would not be effective enough to solve a cost-sensitive classification task, an important problem for data mining which takes the misclassification costs into consideration [Elk01, Dom99]. This is because most existing online learning studies often concern the performance of an online classification algorithm in terms of prediction *mistake rate* or *accuracy*, which is obviously *cost-insensitive* and thus *inappropriate* for many real applications in data mining, especially for cost-sensitive classification tasks where datasets are often class-imbalanced and the misclassification costs of instances from different classes can be very different [VCC99, CBHK02, DH03].

To address the above challenge of cost-sensitive classification, researchers especially in data mining literature have proposed more meaningful metrics, such as the weighted sum of *sensitivity* and *specificity* [HK00, BOSB10] and the weighted *misclassification cost* [Elk01, AKJ04]. Over the past decades, substantial research efforts have been devoted to developing batch classification algorithms to improve the cost-sensitive measures, including the weighted sum of sensitivity and specificity and the weighted misclassification cost metrics [Elk01, AKJ04]. However, these batch classification algorithms often suffer poor efficiency and scalability when solving large-scale problems, which thus are unsuitable for online classification applications.

Although both *cost-sensitive classification* and *online learning* have been studied extensively in data mining and machine learning communities, respectively, there was very few comprehensive study of cost-sensitive online classification in both data mining and machine learning literature. In this chapter, we formally investigate this problem by directly optimizing cost-sensitive measures for an online classification task. As the first comprehensive study, we proposed Cost-Sensitive Double Updating Online Learning (CS-DUOL) algorithms based on double updating technique to tackle the online optimization

task of maximizing the weighted sum or minimizing the weighted misclassification cost. We theoretically analyze the cost-sensitive measure bounds of the proposed algorithms, extensively examine their empirical performance of cost-sensitive online classification tasks, and finally demonstrate the application of our technique for solving online anomaly detection tasks.

Finally, the rest of the chapter is organized as follows. Section 6.1 briefly presents the related work. Section 6.2 formulates the problem, presents the proposed algorithms and theoretically analyzes the bounds of the proposed algorithms. Section 6.3 discusses our experimental results. Section 6.4 shows an application to online anomaly detection tasks, and finally Section 6.5 concludes this chapter.

6.1 Related Work

Our work is mainly related to two groups of research in data mining and machine learning communities, that is, cost-sensitive classification in data mining literature, and online learning in machine learning literature.

Cost-sensitive classification has been extensively studied in data mining and machine learning. To address this problem, researchers have proposed a variety of cost-sensitive metrics. The well-known examples include the weighted sum of *sensitivity* and *specificity* [HK00, BOSB10], and the weighted *misclassification cost* that takes cost into consideration when measuring classification performance [Elk01, AKJ04]. As a special case, when the weights are both equal to 0.5, the weighted sum of sensitivity and specificity is reduced to the well-known *balanced accuracy* [BOSB10]. Over the past decades, various batch learning algorithms have been proposed for cost-sensitive classification in literature [Tan93, Tur95, Dom99, Elk01, LA08, LKZ10].

Most online learning algorithms [Ros58, LL99, FS99, Gen01, CS03, KSW04, CDK⁺06, DCP08] are cost-insensitive, except the prediction-based PA algorithm ('CPA_{ML}') [CDK⁺06] and the perceptron algorithm with uneven margin ('PAUM') [LZH⁺02]. However, to the best of our knowledge, no existing work in this area had attempted to directly optimize the two cost-sensitive metrics in an online learning setting. Finally, we note that our work is very different from another recent online learning study [ZHJY11], which aims to optimize AUC, but cannot be guaranteed to optimize the cost-sensitive measures in our study. Take one extreme case for example, if the weights for false positive and false negative are the same, then the objective of minimizing the weighted cost is to maximize accuracy, which is not equivalent to maximizing AUC [CM03].

6.2 Cost-Sensitive Online Classification

6.2.1 Problem Formulation

Without loss of generality, let us consider an online binary classification problem. Formally, let us denote by $\mathbf{x}_t \in \mathbb{R}^d$ the instance received at the t -th learning step, and $f_{t-1} \in \mathcal{H}_\kappa$ a linear prediction model learned from the previous $t - 1$ training examples. We also denote the prediction for the t -th instance as $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$, while the value $|f_{t-1}(\mathbf{x}_t)|$, known as the “margin”, is used as the confidence of the learner on the prediction. The true label for instance $\mathbf{x}_t$ is denoted as $y_t \in \{-1, +1\}$. If $\hat{y}_t \neq y_t$, the learner made a mistake; otherwise it made a correct prediction.

For binary classification, the result of each prediction for an instance can be classified into four cases: (1) *True Positive* (TP) if $\hat{y}_t = y_t = +1$; (2) *False Positive* (FP) if $\hat{y}_t = +1$ and $y_t = -1$; (3) *True Negative* (TN) if $\hat{y}_t = y_t = -1$; and (4) *False Negative* (FN) if $\hat{y}_t = -1$ and $y_t = +1$.

We now consider a sequence of training examples $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ for online learning. Then, for convenience, we denote by $\mathcal{M}$ the set of indexes that correspond to the trials of misclassification:

$$\mathcal{M} = \{t \mid y_t \neq \text{sign}(f_t(\mathbf{x}_t)), \forall t \in [T]\},$$

where $[T] = \{1, \dots, T\}$. Similarly, we denote by $\mathcal{M}_p = \{t \mid t \in \mathcal{M} \text{ and } y_t = +1\}$ the set of indexes for false negatives, and $\mathcal{M}_n = \{t \mid t \in \mathcal{M} \text{ and } y_t = -1\}$ the set of indexes for false positives.

Further, we introduce notation $M = |\mathcal{M}|$ to denote the number of mistakes, $M_p = |\mathcal{M}_p|$ to denote the number of false negatives, and $M_n = |\mathcal{M}_n|$ to denote the number of false positives. Also we use notation $\mathcal{I}_T^p = \{i \in [T] \mid y_i = +1\}$ to denote the set of indexes of the positive examples, $\mathcal{I}_T^n = \{i \in [T] \mid y_i = -1\}$ to denote the set of indexes of negative examples, $T_p = |\mathcal{I}_T^p|$ to denote the number of positive examples, and $T_n = |\mathcal{I}_T^n|$ to denote the number of negative examples.

For performance metrics, *sensitivity* is defined as the ratio between the number of true positives $T_p - M_p$ and the number of positive examples; *specificity* is defined as the ratio between $T_n - M_n$ and the number of negative examples; and *accuracy* is defined as the ratio between the number of correctly classified examples and the total number of examples. These can be summarized as:

$$\text{sensitivity} = \frac{T_p - M_p}{T_p}, \quad \text{specificity} = \frac{T_n - M_n}{T_n}, \quad \text{accuracy} = \frac{T - M}{T}.$$

Consider an online binary classification task, without loss of generality, we assume positive class is the rare class, i.e., $T_p \leq T_n$, the number of positive examples is smaller than

the number of negative examples. For simplicity, we also assume that $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$. For traditional online learning, the performance is measured by the prediction accuracy (or mistake rate equivalently) over the sequence of examples. This is inappropriate for imbalanced data because a trivial learner that simply classifies any example as negative could achieve a quite high accuracy for a highly imbalanced dataset. Thus, a more appropriate metric is to measure the *sum* of weighted *sensitivity* and *specificity*, i.e.,

$$sum = \eta_p \times sensitivity + \eta_n \times specificity \quad (6.1)$$

where $\eta_p + \eta_n = 1$ and $0 \leq \eta_p, \eta_n \leq 1$ are two parameters to trade off between sensitivity and specificity. Notably, when $\eta_p = \eta_n = 0.5$, the corresponding *sum* is the well known balanced accuracy. In general, the higher the *sum* value, the better the classification performance. Besides, another approach is to measure the total misclassification cost suffered by the algorithm, which is defined as:

$$cost = c_p \times M_p + c_n \times M_n \quad (6.2)$$

where $c_p + c_n = 1$ and $0 \leq c_p, c_n \leq 1$ are the misclassification cost parameters for positive and negative classes, respectively. The lower the *cost* value, the better the classification performance.

6.2.2 Algorithms

In this section, we propose the Cost-Sensitive Double Updating Online Learning algorithms for cost-sensitive classification by optimizing two cost-sensitive measures. Before presenting our algorithms, we first prove the following important proposition that motivates our solution.

Proposition 6.6: *Consider a cost-sensitive classification problem, the goal of maximizing the weighted sum in (6.1) or minimizing the weighted cost in (6.2) is equivalent to minimizing the following objective:*

$$\sum_{y_t=+1} \theta I_{(y_t f(\mathbf{x}_t) < 0)} + \sum_{y_t=-1} I_{(y_t f(\mathbf{x}_t) < 0)}$$

where $\theta = \frac{\eta_p T_n}{\eta_n T_p}$ for the maximization of the weighted sum, and $\theta = \frac{c_p}{c_n}$ for the minimization of the weighted misclassification cost.

Proof: Firstly, by analyzing the function of the weighted sum in (6.1), we can derive the following:

$$sum = \eta_p \frac{T_p - M_p}{T_p} + \eta_n \frac{T_n - M_n}{T_n} = 1 - \frac{\eta_n}{T_n} \left[\frac{\eta_p T_n}{\eta_n T_p} \sum_{y_t=+1} I_{(y_t f(\mathbf{x}_t) < 0)} + \sum_{y_t=-1} I_{(y_t f(\mathbf{x}_t) < 0)} \right],$$

where I_π is the indicator function that outputs 1 if the statement π holds and 0 otherwise. Thus, maximizing sum is equivalent to minimizing

$$\frac{\eta_p T_n}{\eta_n T_p} \sum_{y_t=+1} I_{(y_t f(\mathbf{x}_t) < 0)} + \sum_{y_t=-1} I_{(y_t f(\mathbf{x}_t) < 0)}.$$

Secondly, by analyzing the function of the weighted cost in (6.2), we can also derive the following:

$$\text{cost} = c_p M_p + c_n M_n = c_n \left[\frac{c_p}{c_n} \sum_{y_t=+1} I_{(y_t f(\mathbf{x}_t) < 0)} + \sum_{y_t=-1} I_{(y_t f(\mathbf{x}_t) < 0)} \right]$$

Thus, minimizing cost is equivalent to minimizing

$$\frac{c_p}{c_n} \sum_{y_t=+1} I_{(y_t f(\mathbf{x}_t) < 0)} + \sum_{y_t=-1} I_{(y_t f(\mathbf{x}_t) < 0)}.$$

Thus, the proposition holds by setting $\theta = \frac{\eta_p T_n}{\eta_n T_p}$ for sum, and $\theta = \frac{c_p}{c_n}$ for cost. □

Proposition 6.6 gives the explicit objective function for optimization, but the indicator function is not convex. To facilitate the online optimization task, we replace the indicator function by its convex surrogate, i.e., the following modified hinge loss function:

$$\ell(f; (\mathbf{x}, y)) = \max(0, (\theta * I_{(y=1)} + I_{(y=-1)}) - yf(\mathbf{x})).$$

As a result, we can formulate the optimization problem for cost-sensitive classification as follows:

$$\mathcal{F}_T(f) = \frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{t=1}^T \ell(f; (\mathbf{x}_t, y_t)), \quad (6.3)$$

where $\|f\|_{\mathcal{H}_\kappa}^2$ is introduced to regularize the complexity of the linear classifier and C is a positive penalty parameter of the cumulative loss. The idea of the above formulation is somewhat similar to the biased formulation of batch SVM for learning with imbalanced datasets [AKJ04]. However our framework is used to optimize two cost-sensitive metrics, while theirs is only to minimize the cumulative cost. In addition, they assume all the examples are given in advance, while we assume the examples comes one by one.

Now our goal is to find an online learning solution to tackle the above convex optimization (6.3). To this end, we propose to explore double updating online learning to tackle this problem. Specifically, we consider trial t in an online learning task where the training example (x_a, y_a) is misclassified (i.e., $y_a f(\mathbf{x}_a) \leq 0$). Similar to DUOL for regular binary classification, we introduce an auxiliary example $(\mathbf{x}_b, y_b)$ from the existing support vectors that obeys the following conditions:

- $y_b f(\mathbf{x}_b) \leq 0$, that is, support vector $(\mathbf{x}_b, y_b)$ is misclassified by the current classifier $f(\mathbf{x})$,
- $k(\mathbf{x}_b, \mathbf{x}_a) y_a y_b \leq -\rho$ where $\rho \in (0, 1)$ is a predetermined threshold, that is, support vector $(\mathbf{x}_b, y_b)$ “**conflicts**” with the new misclassified example $(\mathbf{x}_a, y_a)$.

To facilitate the analysis, we also denote

$$k_a = \kappa(\mathbf{x}_a, \mathbf{x}_a), \quad k_b = \kappa(\mathbf{x}_b, \mathbf{x}_b), \quad k_{ab} = \kappa(\mathbf{x}_a, \mathbf{x}_b), \quad w_{ab} = y_a y_b k_{ab}.$$

According to the assumption of auxiliary example, we have $w_{ab} = k_{ab} y_a y_b \leq -\rho$. Finally, we denote by $\hat{\gamma}_b$ the weight for the auxiliary example $(\mathbf{x}_b, y_b)$ that is used in the current classifier $f(\mathbf{x})$, by γ_a and γ_b the updated weights for $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$, respectively, and by d_{γ_b} the difference $\gamma_b - \hat{\gamma}_b$.

Following the framework of dual formulation for online learning, the following lemma shows how to compute Δ_t , that is, the improvement in the objective function of dual biased SVM by adjusting weights for $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$.

Lemma 6.2 *The maximal improvement in the objective function of dual biased SVM by adjusting weights for $(\mathbf{x}_a, y_a)$ and $(\mathbf{x}_b, y_b)$, denoted by Δ_t , is computed by solving the following optimization problem:*

$$\Delta_t = \max_{\gamma_a, d_{\gamma_b}} \{h(\gamma_a, d_{\gamma_b}) : 0 \leq \gamma_a \leq C, -\hat{\gamma}_b \leq d_{\gamma_b} \leq C - \hat{\gamma}_b\}, \quad (6.4)$$

where

$$h(\gamma_a, d_{\gamma_b}) = \gamma_a(\theta_a - y_a f(\mathbf{x}_a)) + d_{\gamma_b}(\theta_b - y_b f(\mathbf{x}_b)) - \frac{k_a}{2} \gamma_a^2 - \frac{k_b}{2} d_{\gamma_b}^2 - w_{ab} \gamma_a d_{\gamma_b}. \quad (6.5)$$

where $\theta_i = \theta * \mathbf{I}_{(y_i=1)} + \mathbf{I}_{(y_i=-1)}$.

Proof: It is straightforward to verify that the dual function of $\min_{f \in \mathcal{H}_\kappa} \frac{1}{2} \|f\|_{\mathcal{H}_\kappa}^2 + C \sum_{i=1}^t \max(0, \theta_i - y_i f(\mathbf{x}_i))$, denoted by $\mathcal{D}_t(\gamma_1, \dots, \gamma_t)$, is computed as follows,

$$\mathcal{D}_t(\gamma_1, \dots, \gamma_t) = \sum_{i=1}^t \gamma_i \theta_i - \frac{1}{2} \left\| \sum_{i=1}^t \gamma_i y_i \kappa(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}_\kappa}^2,$$

where $\gamma_i \in [0, C]$, $i = 1, \dots, t$ and $f_{t-1} = \sum_{i=1}^{t-1} \gamma_i y_i \kappa(\mathbf{x}_i, \cdot)$ is the current classifier. Thus

$$\begin{aligned} & h(\gamma_a, d_{\gamma_b}) \\ &= \mathcal{D}(\gamma_1, \dots, \hat{\gamma}_b + d_{\gamma_b}, \dots, \gamma_{t-1}, \gamma_a) - \mathcal{D}(\gamma_1, \dots, \hat{\gamma}_b, \dots, \gamma_{t-1}) \\ &= \gamma_a(\theta_a - y_a f_{t-1}(\mathbf{x}_a)) + d_{\gamma_b}(\theta_b - y_b f_{t-1}(\mathbf{x}_b)) - \frac{1}{2} \gamma_a^2 \kappa(\mathbf{x}_a, \mathbf{x}_a) - \frac{1}{2} d_{\gamma_b}^2 \kappa(\mathbf{x}_b, \mathbf{x}_b) - \gamma_a d_{\gamma_b} y_a y_b \kappa(\mathbf{x}_a, \mathbf{x}_b). \end{aligned}$$

The theorem below bounds the bounding constant Δ when C is sufficiently large.

Theorem 13 Assume $C \geq \max(\hat{\gamma}_b + \frac{\theta_b + \rho\theta_a}{1-\rho^2}, \frac{\theta_a + \rho\theta_b}{1-\rho^2})$ with $\rho \in [0, 1)$ for the selected auxiliary example $(\mathbf{x}_b, y_b)$, we have the following bound for the bounding constant Δ :

$$\Delta \geq \frac{\theta_a^2 + 2\rho\theta_a\theta_b + \theta_b^2}{2(1-\rho^2)}.$$

Proof: First, we show $d_{\gamma_b} \geq 0$. This is because for given $\gamma_a \geq 0$, the optimal solution for d_{γ_b} , given by

$$d_{\gamma_b} = \frac{\theta_b - y_b f(\mathbf{x}_b) - w_{ab}\gamma_a}{k_b},$$

is positive because $y_b f(\mathbf{x}_b) \leq 0$ and $w_{ab} \leq -\rho$. Using the fact $k_a, k_b \leq 1$, $\gamma_a, d_{\gamma_b} \geq 0$, $y_a f(\mathbf{x}_a) \leq 0$, $y_b f(\mathbf{x}_b) \leq 0$, and $w_{a,b} \leq -\rho$, we have

$$h(\gamma_a, d_{\gamma_b}) \geq \gamma_a \theta_a + d_{\gamma_b} \theta_b - \frac{1}{2} \gamma_a^2 - \frac{1}{2} d_{\gamma_b}^2 + \rho \gamma_a d_{\gamma_b}.$$

Thus, Δ is bounded as

$$\Delta \geq \max_{\gamma_a \in [0, C], d_{\gamma_b} \in [0, C - \hat{\gamma}_b]} \gamma_a \theta_a + d_{\gamma_b} \theta_b - \frac{1}{2} (\gamma_a^2 + d_{\gamma_b}^2) + \rho \gamma_a d_{\gamma_b}.$$

Under the condition that $C \geq \max(\hat{\gamma}_b + \frac{\theta_b + \rho\theta_a}{1-\rho^2}, \frac{\theta_a + \rho\theta_b}{1-\rho^2})$, it is easy to verify that the optimal solution for the above problem is $\gamma_a = \frac{\theta_a + \rho\theta_b}{1-\rho^2}$ and $d_{\gamma_b} = \frac{\theta_b + \rho\theta_a}{1-\rho^2}$.

We refer to the case as a **strong double update** when the condition of Theorem 13 is satisfied. We have the following theorem for the general case when we only have $C \geq \max(\theta, 1)$.

Theorem 14 Assume $C \geq \max(\theta, 1)$. We have the following bound for Δ when updating the weight for the misclassified example $(\mathbf{x}_a, y_a)$ and the auxiliary example $(\mathbf{x}_b, y_b)$:

$$\Delta \geq \frac{\theta_a^2}{2} + \frac{1}{2} \min((\theta_b + \rho\theta_a)^2, (C - \hat{\gamma})^2).$$

Proof: By setting $\gamma_a = \theta_a$, we have $h(\gamma_a, d_{\gamma_b})$ computed as

$$h(\gamma_a = 1, d_{\gamma_b}) \geq \frac{\theta_a^2}{2} + (\theta_b + \rho\theta_a)d_{\gamma_b} - \frac{1}{2} d_{\gamma_b}^2.$$

Hence, Δ is lower bounded by

$$\Delta \geq \frac{\theta_a^2}{2} + \max_{d_{\gamma_b} \in [0, C - \hat{\gamma}]} \left((\theta_b + \rho\theta_a)d_{\gamma_b} - \frac{1}{2} d_{\gamma_b}^2 \right) \geq \frac{\theta_a^2}{2} + \frac{1}{2} \min((\theta_b + \rho\theta_a)^2, (C - \hat{\gamma})^2).$$

Although Theorem 13 and Theorem 14 show that the double update strategy could significantly improve the bounding constant Δ , it is applicable only when there exists an auxiliary example. Below, we extend the double update strategy to the cases when there is no auxiliary example. Specifically, we relax the condition for performing double update as follows: there exists $(\mathbf{x}_b, y_b) \in \mathcal{D}$ that (i) $w_{ab} \leq -\rho$, (ii) $y_b f_{t-1}(\mathbf{x}_b) \leq \theta_b$, and (iii) $C \geq \max(\hat{\gamma}_b + \frac{\rho\theta_a}{1-\rho^2}, \frac{\theta_a}{1-\rho^2})$. We refer to these cases as **weak double update**.

Theorem 15 Assume $w_{ab} \leq -\rho$, $y_b f_{t-1}(\mathbf{x}_b) \leq \theta_b$ and $C \geq \max(\hat{\gamma}_b + \frac{\rho\theta_a}{1-\rho^2}, \frac{\theta_a}{1-\rho^2})$, we have the following bound for the bounding constant

$$\Delta \geq \frac{\theta_a}{2(1-\rho^2)}.$$

Proof: Following the definitions and assumptions, we have

$$\begin{aligned} \Delta &= \max_{\gamma_a, d_{\gamma_b}} h(\gamma_a, d_{\gamma_b}) \geq h\left(\frac{\theta_a}{1-\rho^2}, \frac{\rho\theta_a}{1-\rho^2}\right) \\ &\geq \frac{\theta_a}{1-\rho^2}\theta_a + 0 - \frac{1}{2}\left(\frac{\theta_a}{1-\rho^2}\right)^2 - \frac{1}{2}\left(\frac{\rho\theta_a}{1-\rho^2}\right)^2 + \rho\frac{\theta_a^2}{1-\rho^2}\frac{\rho\theta_a}{1-\rho^2} = \frac{\theta_a}{2(1-\rho^2)}. \end{aligned}$$

Now solving the optimization problem 6.4 is the key to the double update. The following proposition provides the optimal solution to the problem 6.4.

Proposition 6.7: Denote $\ell_a := \theta_a - y_a f(\mathbf{x}_a)$ and $\ell_b := \theta_b - y_b f(\mathbf{x}_b)$. Assume $\ell_a, \ell_b \geq 0$, $k_a, k_b > 0$ and $w_{ab} \leq 0$, then the solution of optimization problem (6.4) is as follows:

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \text{if } (k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) < 0 \text{ and} \\ & (k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \text{ and} \\ & \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \text{ and} \\ & \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab}\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2}) & \text{if } (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2}) \in [0, C] \times [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}.$$

□

We skip the proof for Proposition 6.7, since it is similar with the one for regular binary DUOL. Algorithm 7 summarizes the proposed Cost-Sensitive Double Updating Online Learning (CSDUOL) algorithm. In this algorithm, to efficiently find the auxiliary example $(\mathbf{x}_b, y_b)$, we introduce a variable f_t^i for each support vector to keep track of its classification score. Parameter ρ is used to trade off between efficiency and efficacy for CSDUOL: the smaller ρ the more double updates will be performed.

Finally, we give the bound analysis for the CSDUOL algorithm. We denote by $\mathcal{M}_d^s(\rho)$, $\mathcal{M}_d^w(\rho)$ and $\mathcal{M}_s$ the sets of indexes for the cases of *strong double*, *weak double* and *single* updating, respectively, that is,

$$\begin{aligned} \mathcal{M}_d^s(\rho) &= \{t | \exists \text{ auxiliary example } (\mathbf{x}_b, y_b) \text{ s.t. } C \geq \max(\hat{\gamma}_b + \frac{\theta_b + \rho\theta_t}{1-\rho^2}, \frac{\theta_t + \rho\theta_b}{1-\rho^2}) \text{ for } (\mathbf{x}_t, y_t), t \in \mathcal{M}\}, \\ \mathcal{M}_d^w(\rho) &= \{t \in \mathcal{M} / \mathcal{M}_d^s(\rho) | \exists (\mathbf{x}_b, y_b) \text{ s.t. } w_{ab} \leq -\rho, y_b f_{t-1}(\mathbf{x}_b) \leq \theta_b \text{ and } C \geq \max(\hat{\gamma}_b + \frac{\rho\theta_t}{1-\rho^2}, \frac{\theta_t}{1-\rho^2})\}, \\ \mathcal{M}_s &= \mathcal{M} / [\mathcal{M}_d^s(\rho) \cup \mathcal{M}_d^w(\rho)]. \end{aligned}$$

Note that in set $\mathcal{M}_d^s(\rho)$, for the convenience of analysis, we only consider the subset of strong updates when the condition $C \geq \max(\hat{\gamma}_b + \frac{\theta_b + \rho\theta_t}{1-\rho^2}, \frac{\theta_t + \rho\theta_b}{1-\rho^2})$ is satisfied.

Algorithm 7 The proposed Cost-Sensitive Double Updating Online Learning (CS-DUOL) algorithms.

Initialize $S_0 = \emptyset$, $f_0 = 0$ bias parameter $\theta = \frac{\eta_p T_n}{\eta_m T_p}$ for “sum” and $\theta = \frac{c_p}{c_n}$ for “cost”;
for $t = 1, 2, \dots, T$ **do**
 Receive a new instance $\mathbf{x}_t$;
 Predict $\hat{y}_t = \text{sign}(f_{t-1}(\mathbf{x}_t))$;
 Receive its label y_t and compute $\theta_t = \theta * \mathbf{I}_{(y_t=1)} + \mathbf{I}_{(y_t=-1)}$;
 $\ell_t = \max\{0, \theta_t - y_t f_{t-1}(\mathbf{x}_t)\}$;
if $\ell_t > 0$ **then**
 $w_{\min} = \infty$;
 for $\forall i \in S_{t-1}$ **do**
 if $(f_{t-1}^i \leq \theta_i)$ **then**
 if $(y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) \leq w_{\min})$ **then**
 $w_{\min} = y_i y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
 $(\mathbf{x}_b, y_b) = (\mathbf{x}_i, y_i)$;
 end if
 end if
 end for
 $f_{t-1}^t = y_t f_{t-1}(\mathbf{x}_t)$;
 $S_t = S_{t-1} \cup \{t\}$;
 if $(w_{\min} \leq -\rho)$ **then**
 Compute γ_t and d_{γ_b} by solving the optimization (6.4);
 for $\forall i \in S_t$ **do**
 $f_t^i \leftarrow f_{t-1}^i + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t) + y_i d_{\gamma_b} y_b \kappa(\mathbf{x}_i, \mathbf{x}_b)$;
 end for
 $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot) + d_{\gamma_b} y_b \kappa(\mathbf{x}_b, \cdot)$;
 else
 $\gamma_t = \min(C, \ell_t / \kappa(\mathbf{x}_t, \mathbf{x}_t))$;
 for $\forall i \in S_t$ **do**
 $f_t^i \leftarrow f_{t-1}^i + y_i \gamma_t y_t \kappa(\mathbf{x}_i, \mathbf{x}_t)$;
 end for
 $f_t = f_{t-1} + \gamma_t y_t \kappa(\mathbf{x}_t, \cdot)$;
 end if
 else
 $f_t = f_{t-1}$; $S_t = S_{t-1}$;
 for $\forall i \in S_t$ **do**
 $f_t^i \leftarrow f_{t-1}^i$;
 end for
 end if
end for
 Return f_T, S_T

Theorem 16 *Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t , and assume $C \geq \max(\theta, 1)$. Then the proposed CSDUOL algorithm satisfies the following inequality:*

$$\theta^2 M_p + M_n \leq \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2}.$$

where $\rho \in [0, 1)$.

Proof: According to Theorem 13 and 15, we have

$$\min_{t \in \mathcal{M}_d^s(\rho)} \Delta_t \geq \frac{\theta_t^2 + 2\rho\theta_t\theta_b + \theta_b^2}{2(1 - \rho^2)}, \quad \min_{t \in \mathcal{M}_d^w(\rho)} \Delta_t \geq \frac{\theta_t^2}{2(1 - \rho^2)}.$$

Moreover, according to Theorem 14, we have $\Delta_t \geq \theta_t^2/2, \forall t \in \mathcal{M}$. Combining these results with Theorem 1, we have

$$\sum_{t \in \mathcal{M}_s} \frac{\theta_t^2}{2} + \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{\theta_t^2 + 2\rho\theta_t\theta_b + \theta_b^2}{2(1 - \rho^2)} + \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2}{2(1 - \rho^2)} \leq \min_{f \in \mathcal{H}_\kappa} \mathcal{F}_T(f).$$

Using the fact $\mathcal{M} = \mathcal{M}_s \cup \mathcal{M}_d^w(\rho) \cup \mathcal{M}_d^s(\rho)$,

$$\sum_{t \in \mathcal{M}} \frac{\theta_t^2}{2} \leq \min_{f \in \mathcal{H}_\kappa} \mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{2(1 - \rho^2)} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{2(1 - \rho^2)}.$$

We complete the proof using $\theta_t = \theta$, if $y_t = +1$, otherwise $\theta_t = 1$.

Now our goal is to analyze the performance of the proposed algorithm in terms of the imbalance metrics. We first consider the weighted sum of sensitivity and specificity, i.e., $sum = \eta_p \times sensitivity + \eta_n \times specificity$, where $\eta_p + \eta_n = 1$ and $\eta_p \geq \eta_n > 0$. The following theorem gives the bound on the sum by the proposed CSDUOL algorithm.

Theorem 17 *Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t . By setting $\theta = \frac{\eta_p T_n}{\eta_n T_p} \geq 1$, and assuming $C \geq \max(\theta, 1)$, for any $f \in \mathcal{H}_\kappa$, we have the following bound for the proposed CSDUOL algorithm:*

$$sum \geq 1 - \frac{\eta_n}{T_n} \left\{ \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2} \right\}.$$

Proof: Following the result of Theorem 16, we have

$$\begin{aligned}
 & \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2} \\
 & \geq (\theta^2 M_p + M_n) \geq (\theta M_p + M_n) \geq \left(\frac{\eta_p T_n}{\eta_n T_p} M_p + M_n \right) = \frac{T_n}{\eta_n} \left(\eta_p \frac{M_p}{T_p} + \eta_n \frac{M_n}{T_n} \right) \\
 & = \frac{T_n}{\eta_n} (\eta_p (1 - \text{sensitivity}) + \eta_n (1 - \text{specificity})) \\
 & = \frac{T_n}{\eta_n} (1 - \eta_p \text{sensitive} - \eta_n \text{specificity}).
 \end{aligned}$$

Rearranging the above inequality leads to the following conclusion:

$$\begin{aligned}
 & \eta_p \times \text{sensitive} + \eta_n \times \text{specificity} \\
 & \geq 1 - \frac{\eta_n}{T_n} \left\{ \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2} \right\}.
 \end{aligned}$$

In the above approach, the parameter θ is set to $\frac{\eta_p T_n}{\eta_n T_p}$, in which the ratio $\frac{T_n}{T_p}$ may be unavailable in advance. To alleviate this issue, we consider another approach using the cost based performance metric. Specifically, we propose to set $\theta = \frac{c_p}{c_n}$, where c_p and c_n are the predefined cost parameters of false negative and false positive, respectively. We assume $c_p + c_n = 1$ and $0 \leq c_n < c_p$ since we would prefer to improve the accuracy of predicting the rare positive examples. By this setting, the following theorem gives us the cost bound of the proposed CSDUOL algorithm.

Theorem 18 *Let $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_T, y_T)$ be a sequence of examples, where $\mathbf{x}_t \in \mathbb{R}^d$, $y_t \in \{-1, +1\}$ and $\kappa(\mathbf{x}_t, \mathbf{x}_t) \leq 1$ for all t . By setting $\theta = \frac{c_p}{c_n} \geq 1$, and assuming $C \geq \max(\theta, 1)$, for any $f \in \mathcal{H}_\kappa$, the overall cost made by the proposed CSDUOL algorithm over this sequence of examples is bounded as follows:*

$$\text{cost} \leq c_n \left\{ \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2} \right\}.$$

Proof: Following the result of Theorem 16, we have

$$\begin{aligned}
 & \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2} \\
 & \geq (\theta^2 M_p + M_n) \geq (\theta M_p + M_n) = \left(\frac{c_p}{c_n} M_p + M_n \right) = \frac{1}{c_n} (c_p M_p + c_n M_n)
 \end{aligned}$$

Therefore, we have

$$c_p M_p + c_n M_n \leq c_n \left\{ \min_{f \in \mathcal{H}_\kappa} 2\mathcal{F}_T(f) - \sum_{t \in \mathcal{M}_d^s(\rho)} \frac{2\rho\theta_t\theta_b + \theta_b^2 + \theta_t^2\rho^2}{1 - \rho^2} - \sum_{t \in \mathcal{M}_d^w(\rho)} \frac{\theta_t^2\rho^2}{1 - \rho^2} \right\}.$$

6.3 Experiments of Cost-Sensitive Online Classification

This section is to evaluate the empirical performance of the two proposed algorithms. To facilitate our discussions, we denote by CSDUOL_{sum} the proposed CSDUOL algorithm that aims to maximize the weighted sum of sensitivity and specificity, and CSDUOL_{cost} the proposed CSDUOL algorithm that aims to minimize the overall misclassification cost.

6.3.1 Experimental Testbed and Setup

We compare two CSDUOL algorithms with the regular DUOL algorithm and a number of state-of-the-art online learning algorithms, including:

- “Perceptron”: the kernel Perceptron algorithm [Ros58],
- “ $\text{ALMA}_p(\alpha)$ ”: the **A**pproximate **L**arge **M**argin **A**lgorithm [Gen01],
- “ROMMA”: the **R**elaxed **O**nline **M**aximum **M**argin **A**lgorithm [LL99],
- “PA-I”: the PA-I version of online **P**assive-**A**ggressive algorithm [CDK⁺06],
- “PAUM”: the **P**erceptron **A**lgorithm with **U**neven **M**argin [LZH⁺02], and
- “ CPA_{ML} ”: the **C**ost-sensitive **P**assive-**A**ggressive algorithm based on **M**ax-**L**oss update method [CDK⁺06].

To examine the performance, we test all the algorithms on a number of benchmark datasets from web machine learning repositories. All of them can be downloaded from LIBSVM website ¹. For space limitation, we randomly choose some of them for our following discussions, which are listed in Table 6.1.

To make a fair comparison, all algorithms adopt the same experimental setup. In particular, for all the compared algorithms, we set the penalty parameter C as 10 and use the same Gaussian kernel with $\sigma = 8$. For the $\text{ALMA}_p(\alpha)$ algorithm, $p = 2$ and

¹<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

Table 6.1: List of binary datasets in our experiments.

dataset	#Examples	#Features	#Pos:#Neg
a7a	16100	123	1:3.1
german	1000	24	1:2.3
spambase	4601	57	1:1.5
w7a	24692	300	1:32.4

$\alpha = 0.9$. For the proposed CSDUOL_{sum} algorithm, we set $\eta_p = \eta_n = 1/2$ and $\theta = \frac{\eta_p T_n}{\eta_n T_p}$ for all cases, while for CSDUOL_{cost} , we set $c_p = 0.95$ and $c_n = 0.05$ and $\theta = \frac{c_p}{c_n}$. For PAUM, the parameters are set as $\tau_+ = \theta$, $\tau_- = 1$ and $\eta = 1$; for CPA_{ML} , $\rho(-1, 1)$ is set to 1 and $\rho(1, -1)$ is set to θ . The threshold ρ of DUOL and CSDUOL is set to 0.

All the experiments are conducted over 20 random permutations for each dataset. The results are reported by averaging over these 20 runs. We evaluate the online classification performance by several metrics: **sensitivity**, **specificity**, the weighted **sum** of sensitivity and specificity, and the weighted **cost**.

6.3.2 Evaluation of Weighted Sum Performance

We first evaluate the weighted sum performance. Table 6.2 summarizes the results of the algorithms. From the results, we can draw several observations.

First of all, by examining the *sum* results, we found that CSDUOL_{sum} always achieves the best on all the datasets, which significantly outperforms all the online algorithms, including two cost-sensitive online algorithms (PAUM and CPA) and the regular DUOL. This shows the proposed CSDUOL_{sum} algorithm is effective in optimization of weighted sum.

Secondly, by examining both *sensitivity* and *specificity* metrics, we found that CSDUOL_{sum} not only achieves the best *sensitivity* for all cases, but also can produce a fairly good specificity performance for most cases. This shows that the proposed approach for CSDUOL is effective in improving the accuracy of predicting the examples from the rare class, while at the same time does not sacrifice a lot for the prediction on the other class.

Third, the number of support vectors of CSDUOL_{sum} is comparable with the regular DUOL algorithm, less than the PA and CPA_{ML} algorithms, while more than the other algorithms. This shows that the proposed technique does not suffer storing more support vectors as a cost for improving the performance.

Fourth, according to the results of running time, we observe that CSDUOL_{sum} is overall efficient as compared with the state-of-the-art online learning algorithms.

Finally, Figure 6.1 shows the changes of online average *sum* performance, from which we observe that the CSDUOL_{sum} algorithms consistently outperform the other algorithms in the entire online learning process.

Table 6.2: Evaluation of the performance of CSDUOL_{sum} and other existing algorithms.

Algorithm	a7a				
	Sum(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	70.125 $\pm$ 0.337	54.793 $\pm$ 0.512	85.456 $\pm$ 0.163	3542.95 $\pm$ 39.866	2.132
ALMA ₂ (0.9)	72.256 $\pm$ 0.213	57.583 $\pm$ 0.312	86.928 $\pm$ 0.146	3586.15 $\pm$ 23.002	2.227
ROMMA	72.129 $\pm$ 0.440	59.185 $\pm$ 0.704	85.074 $\pm$ 0.238	3417.40 $\pm$ 51.542	2.175
PA-I	70.017 $\pm$ 0.436	54.398 $\pm$ 0.714	85.637 $\pm$ 0.243	6768.95 $\pm$ 57.628	4.444
DUOL	70.746 $\pm$ 0.355	55.084 $\pm$ 0.742	86.408 $\pm$ 0.196	6299.85 $\pm$ 49.036	9.122
PAUM	73.832 $\pm$ 0.330	63.352 $\pm$ 0.595	84.311 $\pm$ 0.257	5659.75 $\pm$ 22.627	3.717
CPA _{ML}	71.759 $\pm$ 0.350	61.974 $\pm$ 0.548	81.544 $\pm$ 0.236	6769.75 $\pm$ 57.819	4.691
CSDUOL _{sum}	74.115 $\pm$ 0.337	68.910 $\pm$ 0.597	79.320 $\pm$ 0.220	6425.10 $\pm$ 50.054	9.494
Algorithm	german				
	Sum(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	58.867 $\pm$ 1.116	42.933 $\pm$ 1.588	74.800 $\pm$ 0.738	347.60 $\pm$ 9.467	0.017
ALMA ₂ (0.9)	59.786 $\pm$ 0.994	42.950 $\pm$ 2.158	76.621 $\pm$ 0.831	394.75 $\pm$ 9.244	0.030
ROMMA	59.739 $\pm$ 1.178	45.900 $\pm$ 1.914	73.579 $\pm$ 1.021	347.25 $\pm$ 10.088	0.031
PA-I	59.750 $\pm$ 1.258	41.650 $\pm$ 2.159	77.850 $\pm$ 1.001	721.10 $\pm$ 12.994	0.027
DUOL	61.339 $\pm$ 1.120	46.150 $\pm$ 2.371	76.529 $\pm$ 1.205	656.90 $\pm$ 10.208	0.085
PAUM	56.621 $\pm$ 1.270	34.750 $\pm$ 2.611	78.493 $\pm$ 1.339	566.40 $\pm$ 9.1850	0.023
CPA _{ML}	60.658 $\pm$ 1.276	50.117 $\pm$ 2.164	71.200 $\pm$ 0.964	721.75 $\pm$ 12.674	0.045
CSDUOL _{sum}	62.213 $\pm$ 1.392	57.333 $\pm$ 2.568	67.093 $\pm$ 1.133	657.90 $\pm$ 9.846	0.087
Algorithm	spambase				
	Sum(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	74.061 $\pm$ 0.515	68.323 $\pm$ 0.647	79.799 $\pm$ 0.420	1137.50 $\pm$ 22.596	0.207
ALMA ₂ (0.9)	75.464 $\pm$ 0.602	70.386 $\pm$ 0.929	80.542 $\pm$ 0.531	1544.10 $\pm$ 19.620	0.329
ROMMA	75.102 $\pm$ 0.566	70.028 $\pm$ 0.741	80.176 $\pm$ 0.491	1096.10 $\pm$ 24.809	0.262
PA-I	76.789 $\pm$ 0.474	71.605 $\pm$ 0.758	81.973 $\pm$ 0.425	2854.00 $\pm$ 29.088	0.486
DUOL	79.571 $\pm$ 0.382	75.350 $\pm$ 0.803	83.791 $\pm$ 0.399	2432.30 $\pm$ 27.058	0.974
PAUM	72.591 $\pm$ 0.464	70.985 $\pm$ 0.956	74.198 $\pm$ 0.599	2275.50 $\pm$ 18.724	0.382
CPA _{ML}	77.015 $\pm$ 0.466	74.239 $\pm$ 0.724	79.792 $\pm$ 0.395	2851.45 $\pm$ 28.956	0.559
CSDUOL _{sum}	79.913 $\pm$ 0.354	78.977 $\pm$ 0.861	80.848 $\pm$ 0.492	2443.85 $\pm$ 24.314	0.991
Algorithm	w7a				
	Sum(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	65.305 $\pm$ 0.831	32.682 $\pm$ 1.613	97.928 $\pm$ 0.049	994.40 $\pm$ 23.569	1.134
ALMA ₂ (0.9)	66.062 $\pm$ 0.685	33.703 $\pm$ 1.361	98.422 $\pm$ 0.053	1031.05 $\pm$ 15.330	1.305
ROMMA	68.888 $\pm$ 0.600	40.216 $\pm$ 1.174	97.560 $\pm$ 0.071	1026.75 $\pm$ 21.511	1.337
PA-I	63.155 $\pm$ 0.405	27.899 $\pm$ 0.789	98.412 $\pm$ 0.042	2842.60 $\pm$ 39.400	2.708
DUOL	68.617 $\pm$ 0.626	38.182 $\pm$ 1.248	99.052 $\pm$ 0.054	2228.40 $\pm$ 41.031	2.732
PAUM	57.285 $\pm$ 0.481	15.547 $\pm$ 0.967	99.023 $\pm$ 0.081	1477.95 $\pm$ 0.605	1.520
CPA _{ML}	73.241 $\pm$ 0.699	51.297 $\pm$ 1.349	95.184 $\pm$ 0.099	2841.50 $\pm$ 39.910	2.955
CSDUOL _{sum}	74.609 $\pm$ 0.616	54.291 $\pm$ 1.213	94.928 $\pm$ 0.110	2675.95 $\pm$ 41.082	3.448

 Figure 6.1: Evaluation of online “sum” performance of the $\text{CSDUOL}_{\text{sum}}$ algorithm.

6.3.3 Evaluation of Weighted Cost Performance

We further evaluate the performance of the $\text{CSDUOL}_{\text{cost}}$ algorithm. Table 6.3 summarizes the results of total cost evaluation.

From the results, we can also draw several observations.

First of all, among all the algorithms, we found that the proposed $\text{CSDUOL}_{\text{cost}}$ algorithm achieves significantly less total misclassification *cost* than the other algorithms for all the cases. For example, on the dataset “spambase”, the total misclassification cost made by $\text{CSDUOL}_{\text{cost}}$ is about a half of those made by Perceptron algorithm. This demonstrates the proposed technique can effectively minimize the cost measure.

Further, by examining both *sensitivity* and *specificity* metrics, we found that $\text{CSDUOL}_{\text{cost}}$ often achieves the best *sensitivity* result, but does not always guarantee the best results for *specificity*. This shows the proposed algorithm can effectively improve the prediction accuracy for the instances of the rare class.

Third, the number of support vectors of $\text{CSDUOL}_{\text{cost}}$ is comparable with the regular DUOL algorithm, less than the PA and CPA_{ML} algorithms, while more than the other algorithms. As for running time, we observe that $\text{CSDUOL}_{\text{cost}}$ is overall efficient as compared with the state-of-the-art online learning algorithms.

Finally, Figure 6.2 shows the changes of online average *cost* performance, from which we observe that the $\text{CSDUOL}_{\text{cost}}$ algorithms consistently outperform the other algorithms in the entire online learning process.

6.4 Application to Online Anomaly Detection

The proposed cost-sensitive online classification technique can be potentially applied to a wide range of real-world applications. In this section, we demonstrate an application of the proposed cost-sensitive online classification algorithms to tackle online anomaly detection tasks. Below we first introduce the related application domains, and then show our empirical evaluation results.

6.4.1 Application Domains and Testbeds.

We apply the proposed algorithm to solve problems in the following domains:

- **Bioinformatics:** We apply our algorithm to solve a bioinformatics problem using the “Cod-RNA” dataset [UKM06]. The objective of this task is to develop a computational method to detect novel non-coding RNAs from some large sequenced genomes. Non-coding RNAs are defined as anomalies and others are considered as normal instances.

Table 6.3: Evaluation of the performance of $\text{CSDUOL}_{\text{cost}}$ and other existing algorithms.

Algorithm	a7a				
	Cost	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	1771.255 $\pm$ 20.021	54.793 $\pm$ 0.512	85.456 $\pm$ 0.163	3542.950 $\pm$ 39.866	2.157
ALMA ₂ (0.9)	1654.950 $\pm$ 12.299	57.583 $\pm$ 0.312	86.928 $\pm$ 0.146	3586.150 $\pm$ 23.002	2.276
ROMMA	1621.060 $\pm$ 26.830	59.185 $\pm$ 0.704	85.074 $\pm$ 0.238	3417.400 $\pm$ 51.542	2.211
PA-I	1783.005 $\pm$ 26.927	54.398 $\pm$ 0.714	85.637 $\pm$ 0.243	6768.950 $\pm$ 57.628	4.400
DUOL	1749.400 $\pm$ 25.573	55.084 $\pm$ 0.742	86.408 $\pm$ 0.196	6299.850 $\pm$ 49.036	8.948
PAUM	1218.215 $\pm$ 15.904	72.735 $\pm$ 0.476	78.921 $\pm$ 0.248	6527.700 $\pm$ 21.796	4.417
CPA _{ML}	1405.200 $\pm$ 20.894	67.848 $\pm$ 0.546	77.715 $\pm$ 0.261	6769.500 $\pm$ 56.947	4.666
$\text{CSDUOL}_{\text{cost}}$	1108.005 $\pm$ 17.175	77.231 $\pm$ 0.474	74.954 $\pm$ 0.268	6442.150 $\pm$ 42.521	9.528
Algorithm	german				
	Cost	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	171.720 $\pm$ 4.719	42.933 $\pm$ 1.588	74.800 $\pm$ 0.738	347.600 $\pm$ 9.467	0.017
ALMA ₂ (0.9)	170.400 $\pm$ 5.626	42.950 $\pm$ 2.158	76.621 $\pm$ 0.831	394.750 $\pm$ 9.244	0.030
ROMMA	164.565 $\pm$ 5.367	45.900 $\pm$ 1.914	73.579 $\pm$ 1.021	347.250 $\pm$ 10.088	0.031
PA-I	173.050 $\pm$ 5.977	41.650 $\pm$ 2.159	77.850 $\pm$ 1.001	721.100 $\pm$ 12.994	0.027
DUOL	161.825 $\pm$ 6.148	46.150 $\pm$ 2.371	76.529 $\pm$ 1.205	656.900 $\pm$ 10.208	0.086
PAUM	183.715 $\pm$ 8.171	38.583 $\pm$ 3.296	74.443 $\pm$ 1.301	599.300 $\pm$ 1.559	0.024
CPA _{ML}	132.580 $\pm$ 4.509	61.150 $\pm$ 1.688	60.450 $\pm$ 1.222	719.750 $\pm$ 13.408	0.045
$\text{CSDUOL}_{\text{cost}}$	116.025 $\pm$ 5.627	68.333 $\pm$ 2.066	56.393 $\pm$ 1.591	668.450 $\pm$ 13.032	0.090
Algorithm	spambase				
	Cost	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	573.190 $\pm$ 11.575	68.323 $\pm$ 0.647	79.799 $\pm$ 0.420	1137.500 $\pm$ 22.596	0.207
ALMA ₂ (0.9)	537.460 $\pm$ 15.673	70.386 $\pm$ 0.929	80.542 $\pm$ 0.531	1544.100 $\pm$ 19.620	0.328
ROMMA	544.330 $\pm$ 13.056	70.028 $\pm$ 0.741	80.176 $\pm$ 0.491	1096.100 $\pm$ 24.809	0.261
PA-I	513.580 $\pm$ 12.683	71.605 $\pm$ 0.758	81.973 $\pm$ 0.425	2854.000 $\pm$ 29.088	0.491
DUOL	447.400 $\pm$ 12.762	75.350 $\pm$ 0.803	83.791 $\pm$ 0.399	2432.300 $\pm$ 27.058	0.985
PAUM	339.665 $\pm$ 14.697	86.944 $\pm$ 1.050	54.579 $\pm$ 0.947	2943.350 $\pm$ 13.112	0.508
CPA _{ML}	362.100 $\pm$ 13.095	82.838 $\pm$ 0.748	70.565 $\pm$ 0.604	2839.800 $\pm$ 29.433	0.565
$\text{CSDUOL}_{\text{cost}}$	293.560 $\pm$ 12.845	87.678 $\pm$ 0.786	66.822 $\pm$ 0.586	2636.000 $\pm$ 24.917	1.127
Algorithm	w7a				
	Cost	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	497.960 $\pm$ 11.905	32.682 $\pm$ 1.613	97.928 $\pm$ 0.049	994.400 $\pm$ 23.569	1.191
ALMA ₂ (0.9)	479.345 $\pm$ 9.348	33.703 $\pm$ 1.361	98.422 $\pm$ 0.053	1031.050 $\pm$ 15.330	1.295
ROMMA	456.595 $\pm$ 8.542	40.216 $\pm$ 1.174	97.560 $\pm$ 0.071	1026.750 $\pm$ 21.511	1.325
PA-I	518.230 $\pm$ 5.825	27.899 $\pm$ 0.789	98.412 $\pm$ 0.042	2842.600 $\pm$ 39.400	2.688
DUOL	434.400 $\pm$ 8.467	38.182 $\pm$ 1.248	99.052 $\pm$ 0.054	2228.400 $\pm$ 41.031	2.720
PAUM	585.855 $\pm$ 6.526	15.547 $\pm$ 0.967	99.023 $\pm$ 0.081	1477.950 $\pm$ 0.605	1.576
CPA _{ML}	458.700 $\pm$ 8.042	43.419 $\pm$ 1.140	96.582 $\pm$ 0.069	2843.100 $\pm$ 41.551	2.920
$\text{CSDUOL}_{\text{cost}}$	405.640 $\pm$ 9.594	53.797 $\pm$ 1.257	95.911 $\pm$ 0.107	2547.750 $\pm$ 32.301	3.227

 Figure 6.2: Evaluation of online “cost” performance of the CSDUOL_{cost} algorithm.

- Finance: We apply our algorithm to a credit card approval problem in finance domain. In particular, we consider the well-known Australia credit card data set with 690 instances from an Australian credit company, in which the task is to distinguish credit-worthy from non credit-worthy customers.
- Medical Imaging: We apply our algorithms to solve medical image anomaly detection problem. We consider the “KDDCUP08” breast cancer dataset ², in which the task is to develop a computational method for early detection of breast cancer from X-ray images of the breast. The class “benign” is assigned as the normal class, and the class “malignant” is the anomaly class.

Table 6.4 summarizes the details of the data sets for online anomaly detection.

Table 6.4: Data Sets for Online Anomaly Detection.

Dataset Name	#Examples	#Features	#Outlier:#Normal
Cod-RNA	271617	8	1:2.00
Australian	690	14	1:1.25
KDDCUP08	102294	117	1:163.19

6.4.2 Empirical Evaluation Results.

We apply our algorithm to solve anomaly detection tasks on the real-world datasets as shown in Table 6.4. For performance metric, we evaluate the anomaly detection performance using the *balanced accuracy*, which is able to avoid inflated performance estimates on imbalanced datasets which are very common in anomaly detection tasks.

Table 6.5 summarizes the experimental results. From the results, we can draw several observations as follows.

First of all, among all the existing algorithms, the two cost-sensitive algorithms (PAUM and CPA_{ML}) generally perform better than their regular versions. However, the improvements are not always consistent and significant over different datasets. Such observations indicate the importance of studying more effective cost-sensitive algorithms.

Second, among all the compared algorithms, it is obvious to see that the proposed CSDUOL_{sum} significantly outperforms the other algorithms for all the datasets. This promising result shows the advantage of the proposed algorithm for a real-world online anomaly detection task which is often class-imbalanced.

²<http://www.sigkdd.org/kddcup/>

Table 6.5: Evaluation of balanced accuracy performance for online anomaly detection.

Algorithm	Cod-RNA				
	Balanced Accuracy(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	85.137 $\pm$ 0.260	80.154 $\pm$ 0.349	90.120 $\pm$ 0.172	1318.700 $\pm$ 23.052	0.567
ALMA ₂ (0.9)	87.087 $\pm$ 0.241	82.684 $\pm$ 0.390	91.490 $\pm$ 0.221	1330.100 $\pm$ 20.870	0.614
ROMMA	87.923 $\pm$ 0.227	83.552 $\pm$ 0.318	92.294 $\pm$ 0.193	1060.650 $\pm$ 20.717	0.546
PA-I	86.940 $\pm$ 0.293	81.997 $\pm$ 0.474	91.883 $\pm$ 0.237	3347.250 $\pm$ 34.860	1.226
DUOL	89.371 $\pm$ 0.209	85.549 $\pm$ 0.378	93.194 $\pm$ 0.174	2134.250 $\pm$ 31.217	1.251
PAUM	86.547 $\pm$ 0.294	84.604 $\pm$ 0.463	88.491 $\pm$ 0.272	2830.050 $\pm$ 27.594	1.143
CPA _{ML}	87.670 $\pm$ 0.254	85.520 $\pm$ 0.390	89.820 $\pm$ 0.240	3347.350 $\pm$ 35.006	1.360
CSDUOL _{sum}	90.248 $\pm$ 0.273	89.876 $\pm$ 0.415	90.620 $\pm$ 0.204	2262.900 $\pm$ 31.792	1.367
Algorithm	Australian				
	Balanced Accuracy(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	76.623 $\pm$ 1.294	74.055 $\pm$ 1.442	79.191 $\pm$ 1.153	159.350 $\pm$ 8.821	0.009
ALMA ₂ (0.9)	79.486 $\pm$ 1.098	77.117 $\pm$ 1.250	81.854 $\pm$ 1.286	160.800 $\pm$ 7.223	0.017
ROMMA	78.245 $\pm$ 0.908	75.993 $\pm$ 1.205	80.496 $\pm$ 1.177	148.400 $\pm$ 6.278	0.019
PA-I	77.707 $\pm$ 1.251	75.505 $\pm$ 1.537	79.909 $\pm$ 1.512	351.350 $\pm$ 8.536	0.013
DUOL	79.524 $\pm$ 1.218	77.638 $\pm$ 1.407	81.410 $\pm$ 1.552	283.250 $\pm$ 11.350	0.033
PAUM	77.343 $\pm$ 1.043	75.326 $\pm$ 1.620	79.360 $\pm$ 1.483	281.450 $\pm$ 6.669	0.012
CPA _{ML}	77.800 $\pm$ 1.340	76.840 $\pm$ 1.588	78.760 $\pm$ 1.532	351.500 $\pm$ 8.757	0.024
CSDUOL _{sum}	80.214 $\pm$ 0.950	79.788 $\pm$ 1.807	80.640 $\pm$ 1.523	285.700 $\pm$ 10.352	0.034
Algorithm	KDDCUP08				
	Balanced Accuracy(%)	Sensitivity(%)	Specificity (%)	Support Vectors (#)	Time (s)
Perceptron	61.834 $\pm$ 0.260	24.159 $\pm$ 2.586	99.508 $\pm$ 0.020	935.850 $\pm$ 33.809	1.696
ALMA ₂ (0.9)	60.942 $\pm$ 0.243	22.168 $\pm$ 2.427	99.717 $\pm$ 0.027	1424.685 $\pm$ 26.281	3.209
ROMMA	63.424 $\pm$ 0.257	27.566 $\pm$ 2.556	99.281 $\pm$ 0.026	1146.480 $\pm$ 34.336	2.906
PA-I	60.375 $\pm$ 0.243	20.929 $\pm$ 2.424	99.821 $\pm$ 0.023	5582.970 $\pm$ 127.582	4.497
DUOL	64.864 $\pm$ 0.258	30.752 $\pm$ 2.566	98.977 $\pm$ 0.050	8650.620 $\pm$ 307.092	8.086
PAUM	57.032 $\pm$ 0.243	14.115 $\pm$ 2.427	99.948 $\pm$ 0.014	1161.525 $\pm$ 2.806	2.006
CPA _{ML}	70.542 $\pm$ 0.331	43.274 $\pm$ 3.297	97.810 $\pm$ 0.057	5294.055 $\pm$ 147.446	5.580
CSDUOL	70.728 $\pm$ 0.314	44.159 $\pm$ 3.064	97.297 $\pm$ 0.127	4223.310 $\pm$ 158.363	4.485

6.5 Summary

To directly optimize cost-sensitive metrics, this chapter investigated a new framework of Cost-Sensitive Online Classification. We proposed two effective algorithms based on double updating online learning techniques, theoretically analyzed their cost-sensitive bounds, and finally examined their empirical performance extensively, including their applications to tackle online anomaly detection tasks. Our encouraging results show that the proposed algorithms considerably outperform the existing online algorithms for cost-sensitive online classification tasks.

Finally, all the previous proposed algorithms only learn from one single target domain, which ignore any possible existing knowledge learnt from other related source domain. This obviously is a waste of resource. To tackle this problem, we investigate an on-line learning framework called “Online Transfer Learning” (OTL) that aims to transfer knowledge from some source domain to an online learning task on a target domain.

Chapter 7

Online Transfer Learning

Although being extensively studied, previous kernel based online learning algorithms only learn knowledge from one single domain. However, in the real applications, there may be well-learned knowledge from other source domain, which can be transferred to improve the online learning in the target domain. This setting is important for many applications where training data in a target domain may be limited or too expensive to collect. Although the above setting is somehow similar with transfer learning, most existing work on transfer learning were often studied in an offline learning fashion, which has to assume training data in the target domain is given a priori.

Unlike the existing transfer learning studies, in this chapter, we propose a new framework of Online Transfer Learning (OTL), which could improve an online learning by transferring knowledge from other domains. As the first attempt to this problem, we address some OTL challenges in two different settings.

In the first setting, we study the *single-view OTL* where the target domain shares the same feature space as the old/source one. In the second setting, we address the challenge of *multi-view OTL* where the feature space of the source domain is a subset of that of the target domain. We propose algorithms to solve both problems, and theoretically analyze their mistake bounds. Finally, we empirically examine their performance on several challenging OTL tasks.

The rest of this chapter is organized as follows. Section 7.1 reviews related work. Section 7.2 presents the proposed framework. Section 7.3 and Section 7.4 address the single-view and multi-view OTL tasks, respectively. Section 7.5 gives our experimental results and Section 7.6 summarizes this chapter.

7.1 Related Work

Transfer learning (TL) has been actively studied. The goal of TL is to extract knowledge from one or more source tasks and then apply it to a target task. According to different

learning types, the existing TL settings can be roughly classified into three categories: *inductive*, *transductive*, and *unsupervised* approaches. A comprehensive survey on transfer learning can be found in [PY09].

Inductive TL [NMC07] aims to induce the model of labeled data in the target domain with the aid of knowledge transferred from the source domains. When the source instances are not labeled and the feature spaces of both domains are the same, self taught learning [RBL⁺07] is proposed to learn new bases for the source instances through sparse coding method and then transform the target instances to new representation under the new bases. Finally, the labeled new representations of the target instances can be dealt with those traditional supervised learning methods. When source instances are labeled and the feature spaces of both domains are the same, the main problem is that the source domain data can not be reused directly due to potential data distribution difference between these two domains. To solve this problem, Adaptive SVM (ASVM [YYH07] proposed to learn a classifier which is near to the source classifier and can minimize the cumulative loss on the target examples at the same time.

Transductive TL [ANC07, DM06] aims to extract the knowledge from labeled data in source domain to improve the target tasks on unlabeled data in the target domain. Transductive TL mainly consists of two specific problems: domain adaptation and covariance shift. Domain adaptation problem is that the feature representations across domains are different. To tackle this problem, it is crucial to discover a good feature representation across domains. For this purpose, Pan et al. proposed a transfer component analysis (T-CA) algorithm [PTKY11] to learn some transfer components across domains in a RKHS. In the subspace spanned by these transfer components, data distributions of different domains are close to each other, and then traditional learning methods could be applied. To solve the task of multiple source domains adaptation, Duan et al. proposed Domain Adaption Machine (DAM) [DXT12], whose main idea is through adjusting the weights of a set of source classifiers from multiple source domains to learn a robust classifier for the target domain. Covariance shift is the setting that the marginal distributions of the training instances and testing instances are different, which may be caused by different sampling processes. To tackle this issue, Huang et al. [HSG⁺06], proposed the kernel mean match (KMM) algorithm to estimate the ratio between the distributions of the training and test instances. Then, this ratio is used to re-weight the source instances for traditional supervised learning.

Unsupervised TL [WSZ08, DYXY08] aims to resolve unsupervised learning tasks in target domain with the help of unlabeled data from source domain. One well known unsupervised TL algorithm is the self taught clustering algorithm [DYXY08], which aims to cluster a small unlabeled target dataset with the help of a large unlabeled source dataset. Its main idea is to cluster the target and source data simultaneously so that the

target task can be influenced by the source task through a common set of features. By this way, the clustering on the target task can be improved.

Although both online learning and transfer learning have been actively studied, to the best of our knowledge, few existing work has formally addressed online learning by an transfer learning framework. Besides, we notice that [QPS09] proposed a stochastic (online) algorithm to solve a covariance shift problem, which is a transductive TL problem. However, the problems we studied are inductive TL problems, where we further assume the source classifiers are fixed. Furthermore, at every step, their algorithm requires a pair of examples and a pair of instances from the source dataset and target dataset, respectively, while our setting only provides one example from the target domain. Because of these differences, their algorithm can not be applied to our problems. Finally, we note that OTL is also different from *online multi-task learning* [DLS07], which aims to learn multiple tasks in parallel in an online learning framework.

7.2 Problem Formulation

Let us denote by $\mathcal{X}_1 \times \mathcal{Y}_1$ the source/old data space, where $\mathcal{X}_1 = \mathbb{R}^m$ and $\mathcal{Y}_1 = \{-1, +1\}$. Since our task aims to learn a kernel classifier, we thus denote by $\kappa_1(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ the kernel function to be used in the source classifier. Assume that a source classifier $h(\mathbf{x})$ can be represented as:

$$h(\mathbf{x}) = \sum_{s=1}^S \alpha_s y_{1_s} \kappa_1(\mathbf{x}_{1_s}, \mathbf{x}),$$

where $\{(\mathbf{x}_{1_s}, y_{1_s}) \in \mathcal{X}_1 \times \mathcal{Y}_1 | s = 1, \dots, S\}$ is the set of support vectors for the source training data set, and α_s are the coefficients of support vectors. Typically the source classifier $h(\mathbf{x})$ can be obtained by applying existing learning techniques, such as online learning via the Perceptron algorithm [Ros58, FS99] or regular learning by support vector machines (SVM).

For an online transfer learning (OTL) task, our goal is to transfer the learnt knowledge from source domain to minimize the online mistake rates of the online learnt function $f \in \mathcal{H}_\kappa$ on a sequence of examples $\{(\mathbf{x}_{2_t}, y_{2_t}) | t = 1, \dots, T\}$ in some source space $\mathcal{X}_2 \times \mathcal{Y}_2$.

Specifically, at the t -th trial of the OTL task, the learner receives an instance $\mathbf{x}_{2_t}$, and the goal of online transfer learning is to find a good prediction function such that the predicted class label can match its truth class label y_{2_t} . The key challenge of OTL is how to effectively transfer the knowledge from the old/source domain to the new/target domain for improving the online learning performance. Next, we study OTL in two different settings: single-view vs. multi-view OTL.

7.3 Single-view Online Transfer Learning

We start by studying the single-view OTL, in which we assume the source domain and the target domain have the same feature space, i.e., $\mathcal{X}_2 = \mathcal{X}_1$. One key challenge of this task is to address the *covariance shift* issue that often occurs in this scenario. Specifically, the covariance shift means the marginal distributions of the training instances and testing instances are different, which may be caused by different sampling processes. This raises the challenge of transferring knowledge from source domain to target domain.

The basic idea of our OTL solution is based on the ensemble learning approach to learn a weighted combination of the source classifier and online target classifier. But our In particular, we first construct an entirely new prediction function f only from the data in the target domain in an *online* fashion, and then learn an ensemble prediction function that is the mixture of both the source and the target prediction functions, i.e., h and f , which thus can transfer the knowledge from the source domain. The remaining issue is then how to effectively combine the two prediction functions for handling the covariance shift issue.

To combine the two prediction functions $h(\mathbf{x})$ and $f_t(\mathbf{x})$ at the t -trial of the online learning task, we introduce two weight parameters, $w_{1,t}$ and $w_{2,t}$, for the two prediction functions respectively. At the t -th step, the predicted label of $\mathbf{x}_{2_t}$ is:

$$\hat{y}_{2_t} = \text{sign}\left(w_{1,t}\Pi(h(\mathbf{x}_{2_t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2_t})) - \frac{1}{2}\right), \quad (7.1)$$

where $\Pi(x)$ is a truncation function, i.e., $\Pi(x) = \max(0, \min(1, \frac{x+1}{2}))$. At the beginning of the OTL task, we simply set $w_{1,1} = w_{2,1} = \frac{1}{2}$. In order to effectively transfer, for the subsequent trials of the OTL task, in addition to updating the function $f_{t+1}(\mathbf{x})$ by some online learning methods, e.g. the PA algorithm [CDK⁺06], we expect the two weights of both prediction functions, i.e., $w_{1,t}$ and $w_{2,t}$, can be adjusted dynamically. We suggest the following exponentially weighted updating scheme [CBL06] for adjusting the weights:

$$w_{1,t+1} = \frac{w_{1,t} * s_t(h)}{w_{1,t} * s_t(h) + w_{2,t} * s_t(f_t)}, \quad (7.2)$$

$$w_{2,t+1} = \frac{w_{2,t} * s_t(f_t)}{w_{1,t} * s_t(h) + w_{2,t} * s_t(f_t)}, \quad (7.3)$$

where $s_t(g) = \exp\{-\eta\ell^*(\Pi(g(\mathbf{x}_{2_t})), \Pi(y_{2_t}))\}$, $\forall g \in \mathcal{H}_\kappa$ and $\ell^*(z, y)$ is a loss function which is set to $\ell^*(z, y) = (z - y)^2$ in our approach. Finally, Algorithm 8 summarizes the proposed OTL algorithm.

Next we analyze the mistake bound of the algorithm. We first introduce the following proposition.

Algorithm 8 The Online Transfer Learning (OTL) algorithm.

INPUT: the source classifier $h(\mathbf{x})$ and initial trade off C and weights $w_{1,1} = w_{2,1} = \frac{1}{2}$.
 Initialize $f_1 = 0$;
for $t = 1, 2, \dots, T$ **do**
 Receive instance: $\mathbf{x}_{2_t} \in \mathcal{X}_2$;
 Predict $\hat{y}_{2_t} = \text{sign}\left(w_{1,t}\Pi(h(\mathbf{x}_{2_t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2_t})) - \frac{1}{2}\right)$;
 Receive correct label: $y_{2_t} \in \{-1, +1\}$;
 Compute $w_{1,t+1} = \frac{w_{1,t}s_t(h)}{w_{1,t}s_t(h) + w_{2,t}s_t(f_t)}$, $w_{2,t+1} = \frac{w_{2,t}s_t(f_t)}{w_{1,t}s_t(h) + w_{2,t}s_t(f_t)}$;
 Suffer loss: $\ell_t = [1 - y_{2_t}f_t(\mathbf{x}_{2_t})]_+$;
 if $\ell_t > 0$ **then**
 $\tau_t = \min\{C, \ell_t/\kappa_2(\mathbf{x}_{2_t}, \mathbf{x}_{2_t})\}$;
 $f_{t+1} = f_t + \tau_t y_{2_t} \kappa_2(\mathbf{x}_{2_t}, \cdot)$;
 end if
end for

Proposition 7.8: Let $(\mathbf{x}_{2_t}, y_{2_t}), t = 1, \dots, T$ be a sequence of examples, where $\mathbf{x}_{2_t} \in \mathbb{R}^m$ and $y_{2_t} \in \{-1, +1\}$ for all t . When using the square loss $\ell^*(z, y) = (z - y)^2$ for $z \in [0, 1]$ and $y \in \{0, 1\}$ and the above exponentially weighting update method and setting $\eta = 1/2$, we have the bound of the OTL algorithm as:

$$\begin{aligned}
 & \sum_{t=1}^T \ell^*(w_{1,t}\Pi(h(\mathbf{x}_{2_t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2_t})), \Pi(y_{2_t})) \\
 & \leq \min \left\{ \sum_{t=1}^T \ell^*(\Pi(h(\mathbf{x}_{2_t})), \Pi(y_{2_t})), \sum_{t=1}^T \ell^*(\Pi(f_t(\mathbf{x}_{2_t})), \Pi(y_{2_t})) \right\} + 2 \ln 2.
 \end{aligned}$$

□

The proof of this Proposition is given in Appendix A.

By the above proposition, we derive the mistake bound of the OTL algorithm as follows.

Theorem 19 Let $(\mathbf{x}_{2_t}, y_{2_t}), t = 1, \dots, T$ be a sequence of examples, where $\mathbf{x}_{2_t} \in \mathbb{R}^m$ and $y_{2_t} \in \{-1, +1\}$ for all t . Let us denote by M the number of mistakes made by the OTL algorithm on this sequence, we then have M bounded from above by:

$$M \leq 4 \min \left\{ \sum_{t=1}^T \ell^*(\Pi(h(\mathbf{x}_{2_t})), \Pi(y_{2_t})), \sum_{t=1}^T \ell^*(\Pi(f_t(\mathbf{x}_{2_t})), \Pi(y_{2_t})) \right\} + 8 \ln 2.$$

The proof of this Theorem is given in Appendix A.

Remark. To better understand the mistake bound, we denote by M_h and M_f the mistake bound of model h and f_t , respectively. First, we note that $\ell^*(\Pi(h(\mathbf{x}_{2_t})), \Pi(y_{2_t}))$

is the upper bound of $\frac{1}{4}M_h$ instead of M_h (because ℓ is a square loss and both $\Pi(h(\mathbf{x}_{2_t}))$ and $\Pi(y_{2_t})$ are normalized to $[0, 1]$); similarly, $\ell^*(\Pi(f_t(\mathbf{x}_{2_t})), \Pi(y_{2_t}))$ is the upper bound of $\frac{1}{4}M_f$. Further, if we assume $\ell^*(\Pi(h(\mathbf{x}_{2_t})), \Pi(y_{2_t})) \approx \frac{1}{4}M_h$ and $\ell^*(\Pi(f_t(\mathbf{x}_{2_t})), \Pi(y_{2_t})) \approx \frac{1}{4}M_f$, we have the result: $M \leq \min\{M_h, M_f\} + 8 \ln 2$. This gives a safe theoretical support for the OTL algorithm. If $M_h < M_f$, then $M \leq M_f + 8 \ln 2$, which indicates M is comparable with M_f , since $8 \ln 2$ is a small constant compared with the size of the real word problem.

7.4 Multi-view Online Transfer Learning

In this section, we study the Multi-view OTL problem, where the feature space of the source domain is a subset of that of the target domain. Due to the difference of the two feature spaces, we cannot directly apply the algorithm in the previous section. Below we introduce a co-regularization approach for solving the challenges in this case.

Formally, we denote the data on the target domain as: $\{(\mathbf{x}_{2_t}, y_{2_t}) | t = 1, \dots, T\}$, where $\mathbf{x}_{2_t} \in \mathcal{X}_2 = \mathbb{R}^n \supset \mathbb{R}^m$ and $y_{2_t} \in \{-1, +1\}$. Without loss of generality, we assume the feature subspace of the first m dimensions of $\mathcal{X}_2$ is the same with the source feature space $\mathcal{X}_1$. In the multi-view setting, we split each data instance $\mathbf{x}_{2_t}$ into two sub-instances (views) $\mathbf{x}_{2_t}^{(1)} \in \mathcal{X}_1$ and $\mathbf{x}_{2_t}^{(2)} \in \mathcal{X}_2/\mathcal{X}_1$. For the second view, we introduce a new kernel function $\kappa_2(\cdot, \cdot) : \mathbb{R}^{n-m} \times \mathbb{R}^{n-m} \rightarrow \mathbb{R}$.

The key idea of our multi-view OTL method is to adopt a co-regularization principle of online learning two classifiers $f_t^{(1)}$ and $f_t^{(2)}$ simultaneously from the two views, and predict an unseen example on the target domain by

$$\hat{y}_t = \text{sign} \left(\frac{1}{2} (f_t^{(1)}(\mathbf{x}_{2_t}^{(1)}) + f_t^{(2)}(\mathbf{x}_{2_t}^{(2)})) \right).$$

For the specific algorithm, we initialize the classifier for the first view by setting $f_1^{(1)} = h$, and setting $f_1^{(2)} = 0$ for the second view. For a new example in the online learning task, we update the new functions $f_{t+1}^{(1)}$ and $f_{t+1}^{(2)}$ by the following *co-regularization* optimization:

$$(f_{t+1}^{(1)}, f_{t+1}^{(2)}) = \arg \min_{f^{(1)} \in \mathcal{H}_{\kappa_1}, f^{(2)} \in \mathcal{H}_{\kappa_2}} \frac{\gamma_1}{2} \|f^{(1)} - f_t^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|f^{(2)} - f_t^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + C\ell_t, \quad (7.4)$$

where γ_1 , γ_2 and C are positive parameters, and the loss term ℓ_t is defined below:

$$\ell_t = [1 - y_{2_t} \frac{1}{2} (f^{(1)}(\mathbf{x}_{2_t}^{(1)}) + f^{(2)}(\mathbf{x}_{2_t}^{(2)}))]_{+}. \quad (7.5)$$

Intuitively, the above updating method aims to make the updated ensemble classifier be able to classify the new observed example $(\mathbf{x}_{2_t}, y_{2_t})$ correctly, and to force the two-view

classifiers without deviating too much from the previous classifiers $(f_t^{(1)}, f_t^{(2)})$ via the first two regularization terms.

The above optimization enjoys a closed-form solution as shown in the following proposition. To simplify our discussion, we introduce notation $k_t^1 = \kappa_1(\mathbf{x}_{2_t}^{(1)}, \mathbf{x}_{2_t}^{(1)})$ and $k_t^2 = \kappa_2(\mathbf{x}_{2_t}^{(2)}, \mathbf{x}_{2_t}^{(2)})$.

Proposition 7.9: *For the optimization problem (7.4), its solution can be expressed as follows:*

$$f_{t+1}^{(i)} = f_t^{(i)} + \frac{\tau_t}{2\gamma_i} \kappa_i(\mathbf{x}_{2_t}^{(i)}, \cdot) \quad i = 1, 2,$$

where $\tau_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\}$. □

The proof of the proposition is given in the Appendix A. By this proposition, we summarize the proposed “Co-regularized Online Transfer Learning” (COTL) algorithm in Algorithm 9.

Algorithm 9 The Co-regularized Online Transfer Learning.

INPUT: the source classifier $h(\mathbf{x})$ and parameters γ_1, γ_2 and C .

Initialize $f_1^{(1)} = h$ and $f_1^{(2)} = 0$;

for $t = 1, 2, \dots, T$ **do**

 Receive instance: $\mathbf{x}_{2_t} \in \mathcal{X}_2$;

 Predict: $\hat{y}_t = \text{sign}\left(\frac{1}{2}(f_t^{(1)}(\mathbf{x}_{2_t}^{(1)}) + f_t^{(2)}(\mathbf{x}_{2_t}^{(2)}))\right)$;

 Receive correct label: $y_{2_t} \in \{-1, +1\}$;

 Suffer loss: $\ell_t = \left[1 - y_{2_t} \left(\frac{1}{2}(f_t^{(1)}(\mathbf{x}_{2_t}^{(1)}) + f_t^{(2)}(\mathbf{x}_{2_t}^{(2)}))\right)\right]_+$;

if $\ell_t > 0$ **then**

$\tau_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\}$;

$f_{t+1}^{(1)}(\cdot) = f_t^{(1)}(\cdot) + \frac{\tau_t}{2\gamma_1} y_{2_t} \kappa_1(\mathbf{x}_{2_t}^{(1)}, \cdot)$;

$f_{t+1}^{(2)}(\cdot) = f_t^{(2)}(\cdot) + \frac{\tau_t}{2\gamma_2} y_{2_t} \kappa_2(\mathbf{x}_{2_t}^{(2)}, \cdot)$;

end if

end for

Before we prove the mistake bound for the COTL algorithm, we first introduce a lemma.

Lemma 7.3 *Let $(\mathbf{x}_{2_t}, y_{2_t}), t = 1, \dots, T$ be a sequence of examples, where $\mathbf{x}_{2_t} \in \mathbb{R}^n$ and $y_{2_t} \in \{-1, +1\}$ for all t . After we split the instance $\mathbf{x}_{2_t}$ into two views $(\mathbf{x}_{2_t}^{(1)}, \mathbf{x}_{2_t}^{(2)})$, for any $g^{(1)} \in \mathcal{H}_{\kappa_1}$ and $g^{(2)} \in \mathcal{H}_{\kappa_2}$, we have the following bound:*

$$\sum_{t=1}^T \tau_t \left(\ell_t - \ell(g^{(1)}, g^{(2)}; t) - \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \right) \leq \frac{\gamma_1}{2} \|h - g^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|g^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2,$$

where ℓ_t is given in Eqn. (7.5) and $\ell(g^{(1)}, g^{(2)}; t) = [1 - y_{2_t} \frac{1}{2}(g^{(1)}(\mathbf{x}_{2_t}^{(1)}) + g^{(2)}(\mathbf{x}_{2_t}^{(2)}))]_+$.

The proof of the Lemma is given in the Appendix A. Using the above lemma, we can show the following theorem for the mistake bound of the proposed COTL algorithm.

Theorem 20 *Let $(\mathbf{x}_{2t}, y_{2t}), t = 1, \dots, T$ be a sequence of examples, where $\mathbf{x}_{2t} \in \mathbb{R}^n$ and $y_{2t} \in \{-1, +1\}$ for all t . In addition $k_t^1 \leq R_1$ and $k_t^2 \leq R_2$, $t = 1, \dots, T$. And we split the instance $\mathbf{x}_{2t}$ into two views $(\mathbf{x}_{2t}^{(1)}, \mathbf{x}_{2t}^{(2)})$. Then for any $g^{(1)} \in \mathcal{H}_{\kappa_1}$ and $g^{(2)} \in \mathcal{H}_{\kappa_2}$, the number of mistakes M made by the proposed COTL algorithm is bounded from above by:*

$$M \leq \frac{1}{\tau} \left(\gamma_1 \|h - g^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \gamma_2 \|g^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + 2C \sum_{t=1}^T \ell(g^{(1)}, g^{(2)}; t) \right),$$

where $\tau = \min\{C, \frac{4\gamma_1\gamma_2}{R_1\gamma_2 + R_2\gamma_1}\}$.

Proof: Since $\tau_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\} \leq C$, $\tau_t \ell(g^{(1)}, g^{(2)}; t) \leq C \ell(g^{(1)}, g^{(2)}; t)$. In addition, $\tau_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\} \leq \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}$, we thus have

$$\begin{aligned} & \sum_{t=1}^T \tau_t \left(\ell_t - \ell(g^{(1)}, g^{(2)}; t) - \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \right) \\ &= \sum_{t=1}^T \tau_t \ell_t - \sum_{t=1}^T \tau_t \ell(g^{(1)}, g^{(2)}; t) - \sum_{t=1}^T \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t^2 \\ &\geq \sum_{t=1}^T \tau_t \ell_t - \sum_{t=1}^T C \ell(g^{(1)}, g^{(2)}; t) - \sum_{t=1}^T \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1} \\ &= \sum_{t=1}^T \tau_t \ell_t - C \sum_{t=1}^T \ell(g^{(1)}, g^{(2)}; t) - \frac{1}{2} \sum_{t=1}^T \tau_t \ell_t = \frac{1}{2} \sum_{t=1}^T \tau_t \ell_t - C \sum_{t=1}^T \ell(g^{(1)}, g^{(2)}; t). \end{aligned}$$

Combining the above inequality with the conclusion of Lemma 7.3, we have

$$\frac{1}{2} \sum_{t=1}^T \tau_t \ell_t \leq \frac{\gamma_1}{2} \|h - g^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|g^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + C \sum_{t=1}^T \ell(g^{(1)}, g^{(2)}; t).$$

Furthermore, when a mistake occurs, $\ell_t \geq 1$; thus $\tau_t \ell_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\} * \ell_t \geq \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\} \geq \min\{C, \frac{4\gamma_1\gamma_2}{R_1\gamma_2 + R_2\gamma_1}\} = \tau$. Combining this observation with the inequality above, we have

$$\frac{1}{2} M \times \tau \leq \frac{\gamma_1}{2} \|h - g^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|g^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + C \sum_{t=1}^T \ell(g^{(1)}, g^{(2)}; t).$$

The theorem follows directly by multiplying $2/\tau$ on both sides of the above inequality.

7.5 Experimental Results

In this section, we evaluate the empirical performance of the proposed two kinds of OTL algorithms.

7.5.1 Experimental Testbed and Setup for Single-view OTL

Our first experiment is to evaluate the performance of OTL. We compare our OTL technique against other popular online learning techniques, including:

- “PA”: the Passive-Aggressive algorithms(“PA”) [CDK⁺06] without exploiting any knowledge from the source domain;
- “OASVM”: an Online version of Adaptive SVM [YYH07], which is adapted from Adaptive SVM using online passive-aggressive technique;
- “OTL(fixed)”: a variant of OTL, which is implemented by fixing the ensemble weights of the OTL algorithm to 1/2. This helps us to examine the efficacy of the proposed weighting strategy.

We evaluate the proposed algorithms on six benchmark datasets for transfer learning as listed in Table 7.1.

Table 7.1: Datasets used in the Single-view OTL tasks.

Dataset	# examples	# features	# source instances
books-dvd	4,000	473,857	2,000
dvd-books	4,000	473,857	2,000
electronics-kitchen	4,000	473,857	2,000
kitchen-electronics	4,000	473,857	2,000
landmine1	14,820	9	8,535
landmine2	14,820	9	6,285

These datasets were created based on the “sentiment” and “landmine” datasets, which can be downloaded from the Sentiment data Website¹ and the Transfer Learning Website², which are popularly used for benchmark transfer learning algorithms. The first four datasets are named in the form of “name1-name2”. This means data “name1” from “sentiment” is used as training data for the source domain; and data “name2” from “sentiment” is treated as test data for online learning in the target domain. The last

¹<http://www.cs.jhu.edu/~mdredze/datasets/sentiment/>.

²<http://www.cse.ust.hk/TL/index.html>.

two datasets were created from “landmine”, which consists of 19 tasks, where 1-10 were collected at foliated regions and 11-19 were collected at regions that are bare earth or desert. For “landmine1”, data from 1-10 tasks are used as the source data and the rest as target data; while for “landmine2”, data from 11-19 tasks are used as source data and the rest as target data.

Finally, we adopt the PA algorithm to run on the source dataset and adopt the average classifier as the source classifier, which generally enjoys better generalization ability [CBCG04]. Further, we draw 20 times random permutation of the instances in the target domain in order to obtain stable results by averaging over the 20 trials. All the algorithms adopt linear kernel. In addition, parameter C is set to 5 for all algorithms on every dataset. We evaluate the predictive accuracy of online learning methods by measuring the standard mistake rate, and time efficiency by calculating average time costs.

7.5.2 Evaluation of Single-view OTL Tasks

Table 7.2 summarizes the performance of the compared algorithms.

Table 7.2: Evaluation of Single-view OTL Tasks.

Algorithm	books-dvd		dvd-books	
	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
PA:	33.388 $\pm$ 0.7138	6.911 $\pm$ 0.1680	35.063 $\pm$ 0.8434	7.041 $\pm$ 0.1906
OASVM:	28.460 $\pm$ 0.5377	6.488 $\pm$ 0.1741	29.977 $\pm$ 0.6500	6.664 $\pm$ 0.1962
OTL(fixed):	27.040 $\pm$ 0.5942	6.947 $\pm$ 0.1675	28.607 $\pm$ 0.6288	7.121 $\pm$ 0.2208
OTL:	24.760 $\pm$ 0.0754	6.926 $\pm$ 0.1626	25.833 $\pm$ 0.0799	7.104 $\pm$ 0.2096
Algorithm	electronics-kitchen		kitchen-electronics	
	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
PA:	26.508 $\pm$ 0.6632	6.246 $\pm$ 0.1480	28.620 $\pm$ 0.7378	6.348 $\pm$ 0.1267
OASVM:	21.390 $\pm$ 0.5873	5.647 $\pm$ 0.1593	23.333 $\pm$ 0.6677	5.743 $\pm$ 0.1275
OTL(fixed):	19.800 $\pm$ 0.5237	6.289 $\pm$ 0.1589	22.148 $\pm$ 0.5533	6.389 $\pm$ 0.1375
OTL:	18.387 $\pm$ 0.0958	6.272 $\pm$ 0.1578	20.850 $\pm$ 0.0946	6.398 $\pm$ 0.1634
Algorithm	landmine1		landmine2	
	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
PA:	12.866 $\pm$ 0.2578	0.082 $\pm$ 0.0029	9.040 $\pm$ 0.1666	0.099 $\pm$ 0.0005
OASVM:	12.887 $\pm$ 0.2357	0.076 $\pm$ 0.0021	9.021 $\pm$ 0.1685	0.094 $\pm$ 0.0007
OTL(fixed):	8.196 $\pm$ 0.1290	0.102 $\pm$ 0.0017	6.019 $\pm$ 0.0615	0.128 $\pm$ 0.0005
OTL:	7.290 $\pm$ 0.0107	0.134 $\pm$ 0.0055	5.226 $\pm$ 0.0038	0.168 $\pm$ 0.0004

Several observations can be drawn from the experimental results. First of all, for most of the datasets, we found that the PA algorithm performs the worst, which implies

the necessity to study online transfer learning. Secondly, OASVM achieves better performance than PA on the first four datasets, while has no improvement on the final two datasets, which demonstrates the importance to develop more sophisticated algorithms. In addition, the proposed OTL algorithm achieved the best performance among all datasets, which implies that the proposed algorithm could effectively transfer the learnt knowledge from source task to improve the performance of online learning for the target task. Furthermore, OTL outperform OTL(fixed) on all the datasets, which tells the proposed weights updating strategy can effectively help knowledge transfer.

7.5.3 Experimental Testbed and Setup for Multi-view OTL

We now evaluate the empirical performance of the proposed COTL algorithm for multi-view OTL tasks. We compare COTL algorithm with:

- “PA”: it does not exploit knowledge from the source domain;
- “OASVM”: an Online version of Adaptive SVM [YYH07], which is adapted from Adaptive SVM using online passive-aggressive technique. Because Adaptive SVM requires that the instances are from single view, this OASVM is applied only on the first-view;
- “COTL0”: a variant of COTL, by initializing its first-view classifier as zero function. This method enables us to examine the importance of engaging the h function learnt from the source domain;
- “Ensemble”: a baseline algorithm simply adapted from OTL algorithm, by augmenting the source classifier with zero vector so that its dimension matches the target domain.

We evaluate all the algorithms extensively on several benchmark datasets, as shown in Table 7.3.

These datasets are the same with those for single-view tasks. However, to satisfy the setup of multi-view OTL task, the source-domain data is randomly associated with only half of the feature space while the target-domain data include the whole feature space.

All the algorithms adopt linear models. For fair comparison and simplicity, for all the datasets and algorithms, we set $\gamma_1 = \gamma_2 = 1$ for the two views. In addition, parameter C is set to 5 for all the algorithms on every dataset. We conducted 20 different random permutations to obtain the average results. We evaluate the performance of online learning methods by calculating the mistake rates. We also evaluate the time cost of the compared algorithms.

Table 7.3: Summary of data sets used for multi-view OTL tasks.

Datasets	source/old domain		target/new domain	
	Number	Dimension	Number	Dimension
books-dvd	2,000	236,928	2,000	473,857
dvd-books	2,000	236,928	2,000	473,857
electronics-kitchen	2,000	236,928	2,000	473,857
kitchen-electronics	2,000	236,928	2,000	473,857
landmine1	8,535	5	6,285	9
landmine2	6,285	5	8,535	9

7.5.4 Evaluation of Multi-view OTL Tasks

Table 7.4 summarizes the evaluation results of multi-view OTL tasks.

Several observations can be drawn from the above results. First of all, we found that among all the algorithms, the PA algorithm without exploiting knowledge from source domain achieved very high mistake rate in most cases. This shows that it is important

Table 7.4: Evaluation of Multi-view OTL Tasks.

Algorithm	books-dvd		dvd-books	
	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
PA:	23.630 $\pm$ 0.5979	6.979 $\pm$ 0.1860	24.980 $\pm$ 0.7272	7.075 $\pm$ 0.2255
OASVM:	21.910 $\pm$ 0.6242	2.503 $\pm$ 0.0908	23.065 $\pm$ 0.6337	2.523 $\pm$ 0.0879
COTL0:	23.547 $\pm$ 0.6546	6.895 $\pm$ 0.1565	24.898 $\pm$ 0.5926	6.979 $\pm$ 0.2115
Ensemble:	22.433 $\pm$ 0.4966	7.024 $\pm$ 0.1492	23.950 $\pm$ 0.7037	7.067 $\pm$ 0.2004
COTL:	20.602 $\pm$ 0.5001	6.499 $\pm$ 0.1638	21.820 $\pm$ 0.4057	6.589 $\pm$ 0.2085
Algorithm	electronics-kitchen		kitchen-electronics	
	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
PA:	17.605 $\pm$ 0.5475	6.035 $\pm$ 0.1249	20.018 $\pm$ 0.6919	6.340 $\pm$ 0.1433
OASVM:	16.470 $\pm$ 0.3878	2.170 $\pm$ 0.0636	17.820 $\pm$ 0.5442	2.221 $\pm$ 0.0704
COTL0:	17.443 $\pm$ 0.4985	5.982 $\pm$ 0.1620	20.102 $\pm$ 0.6221	6.228 $\pm$ 0.1398
Ensemble:	17.172 $\pm$ 0.3201	6.105 $\pm$ 0.1350	19.015 $\pm$ 0.3031	6.563 $\pm$ 0.2036
COTL:	14.860 $\pm$ 0.4352	5.554 $\pm$ 0.0941	16.858 $\pm$ 0.4772	5.828 $\pm$ 0.1398
Algorithm	landmine1		landmine2	
	Mistake (%)	Time (s)	Mistakes (%)	Time (s)
PA:	25.365 $\pm$ 0.4176	0.095 $\pm$ 0.0007	35.305 $\pm$ 0.4243	0.139 $\pm$ 0.0008
OASVM:	24.613 $\pm$ 0.4087	0.088 $\pm$ 0.0006	32.301 $\pm$ 0.2987	0.131 $\pm$ 0.0005
COTL0:	20.105 $\pm$ 0.2897	0.129 $\pm$ 0.0007	31.050 $\pm$ 0.2855	0.198 $\pm$ 0.0010
Ensemble:	20.481 $\pm$ 0.0240	0.137 $\pm$ 0.0025	35.279 $\pm$ 0.4355	0.188 $\pm$ 0.0016
COTL:	19.756 $\pm$ 0.3307	0.128 $\pm$ 0.0009	30.545 $\pm$ 0.3931	0.196 $\pm$ 0.0006

for studying knowledge transfer in an OTL task. Second, compared with PA, OASVM achieves better performance on all the datasets, which implies the importance of transferring source knowledge to improve the target task. Third, for all the datasets, we found that the COTL algorithm has the smallest mistake rate. This validates the proposed OTL technique is effective for knowledge transfer in the online learning tasks. Fourth, compared with COTL0, the proposed COTL algorithm achieves better performance on all the datasets, which indicates the importance of utilizing the source classifier to improve the target task. By examining the running time cost, we found that the COTL techniques usually consumes comparable time with the other baselines, which clearly demonstrates the efficiency of the proposed COTL technique.

7.6 Summary

In this chapter, we studied the new problem of **Online Transfer Learning** (OTL), which aims to improve an online learning task on a target domain by transferring knowledge from a source domain. We addressed two OTL tasks in different settings and presented two novel OTL algorithms. We offered theoretical analysis on the mistake bounds of the proposed OTL algorithms, and extensively examined their empirical performance. Encouraging results show the proposed algorithms are effective.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Kernel based online learning plays an important role in machine learning, which is more suitable for large scale real world applications, such as classification, regression, ranking, and metric learning. In the previous chapters, we conducted a survey on online learning and tried to solve several open challenges: one is to improve the efficiency of kernel based online updating; another is to keep the number of stored SVs less than a pre-fixed budget; the third is to directly online optimize the cost-sensitive metrics; and the final one is to improve the kernel based online learning efficiency by transferring knowledge learnt from another domain.

For the first challenge, based on the primal-dual framework, we extended some existing single updating online learning approach to “double updating” approach to online learning named as “DUOL”, which not only updates the weight of the newly added support vector, but also adjusts the weight of one existing support vector that the most seriously conflicts with the new support vector. We show that the mistake bounds for kernel based online binary and multiclass classification tasks can be significantly reduced by the proposed DUOL algorithms. We have conducted two extensive sets of experiments by comparing with a number of competing algorithms on both binary and multiclass online classification tasks. Promising empirical results validate the effectiveness of our technique.

For the second challenge, we presented a new framework of kernel based double updating online learning on a fixed budget, which requires the number of support vectors associated with the prediction function is always bounded by a predefined budget. We theoretically analyzed its performance, which reveals that its effectiveness is tightly connected with the dual ascent achieved by the model reduction for budget maintenance. Based on the theoretical analysis, we proposed three budget maintenance strategies:

removal, projection, and nearest neighbor. We evaluate the proposed algorithms extensively on benchmark datasets. The promising empirical results show that the proposed algorithms outperform the state-of-the-art budget online learning algorithms in terms of mistake rates.

For the third one, we investigated a new framework of Cost-Sensitive Online Classification, which aims to directly optimize cost-sensitive measures for online classification tasks. We proposed two effective algorithms based on double updating online learning techniques, theoretically analyzed their cost-sensitive bounds, and finally examined their empirical performance extensively, including their applications to tackle online anomaly detection tasks. Our encouraging results show that the proposed algorithms considerably outperform the existing online algorithms for cost-sensitive online classification tasks. Through this study, we hope to inspire researchers in machine learning to further explore other rich theory of cost-sensitive online classification, and researchers in data mining to explore the application of new cost-sensitive online learning techniques to tackle a variety of emerging challenges in real-world applications.

For the final challenge, we propose the new framework Online Transfer Learning (OTL), which aims to improve online learning on a target domain by transferring knowledge from a source domain. We addressed two OTL tasks in different settings and presented two novel OTL algorithms. We offered theoretical analysis on the mistake bounds of the proposed OTL algorithms, and extensively examined their empirical performance. Encouraging results show the proposed algorithms are effective.

8.2 Future Directions

In this section, we try to list some potential problems for future research.

8.2.1 Online Learning with Concept Drifting

Although most existing kernel based online learning algorithms [CDK⁺06, SSK08, WZH12] do not make any assumption on the distribution of the data sequence, their theoretical performance is usually compared with the best batch classifier, for which dataset is typically assumed from a fixed distribution in the previous research. From this perspective, most existing online learning algorithms are more suitable for data sequence from a fixed distribution.

But, as we know, the interest of an individual usually will change as times goes, as a result the interested concept will change over time, which is termed as concept drift [BM10]. More formally, concept drift means that the target variable to be predicted changes over time in the learning process.

Because traditional kernel based online learning algorithms do not take the existence of concept drift into consideration, they are not very suitable for concept drift tasks. So it is an interesting problem to develop new online learning algorithm for concept drift dataset, where the main difficulty is to detect the drift in time.

8.2.2 Active Online Learning

Most of the traditional kernel based online learning algorithms [Ros58, CDK⁺06, CKD09, WZH12] will always assume the label of every instance can be requested, but in reality the cost of labeling an instance may be very high. To minimize the cost of labeling instances, active online learning is proposed, where the label is received only when it is requested. More formally, we assume each request of feedback is either costly or time consuming and the objective of active online learning is to learn a classification model by asking the oracle a small number of labels.

According to the criterion for requesting labels, existing active online learning algorithms can be roughly assigned into two groups: confidence based algorithms [CBCG03, CBGZ04, CBGZ06, VCB08, CCBG11] and distribution based algorithms [CBGO09, DGS10, OCB11]. However, it is still an open challenge to organically combine the confidence and the distribution to propose more effective active online learning algorithms.

8.2.3 Application of Kernel Based Online Learning

Kernel based online learning shows strengths in a broad range of applications, such as classification, regression, ranking, and metric learning. Here we list some possible fields that are suitable to use online learning.

Sponsored advertising on webpages. This problem actually belongs to the bandit learning category [KSST08], for which only partial information on the true label is offered. Specifically, by using information about a user (such as, browse history and search history), an advertising system can provide its recommendation, for instance, some ads or some songs are presented for the user to click or to buy; Observing the suggestion, the user will provide his feedback, clicking it or omitting it. The key challenge for this setting is that no information is provided for if other recommendations are offered. This problem is clearly more challenging, since which faces balancing exploration and exploitation.

Stock market prediction. The goal is to maximize the final return from the stock market. Suppose there are n stocks and the total number of money is supposed to be \$ 1 at the beginning. The learner firstly initializes an allocation strategy. At the beginning of every step the learner will allocate the money on the n stocks according to the current strategy, while at the end of every step the market will provide the return value of every stock. Based on the return value of every stock, the learner will update its allocation strategy .

8.3 Summary

This chapter firstly concludes this thesis, and then presents several further directions and potential applications for kernel based online learning. There are so many fields that kernel based online learning can work well that it is worthy of further study. Finally, I hope that more and more researchers will join the study of kernel based online learning and apply it in our daily lives.

Appendix A

Supplementary Proofs

A.1 A Pure Online Version of LASVM

Algorithm 10 Online LASVM.

Initialize $\mathcal{S} = \emptyset$, $b = 0$, $C > 0$ and $\tau > 0$.

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

 Receive a new instance $\mathbf{x}_t$;

 Predict $\hat{y}_t = \text{sign}(\sum_{s \in \mathcal{S}} \alpha_s \kappa(\mathbf{x}_s, \mathbf{x}_t) + b)$;

 Receive its true label y_t ;

$\alpha_t \leftarrow 0$, $g_t \leftarrow y_t - \sum_{s \in \mathcal{S}} \alpha_s \kappa(\mathbf{x}_t, \mathbf{x}_s)$, $A_t \leftarrow \min(0, Cy_t)$, $B_t \leftarrow \max(0, Cy_t)$, $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$;

if $y_t = +1$ **then**

$i \leftarrow t$, $j \leftarrow \arg \min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$;

else

$j \leftarrow t$, $i \leftarrow \arg \max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$;

end if

if $\alpha_i < B_i$, $\alpha_j > A_j$ and $g_i - g_j > \tau$ **then**

$\lambda \leftarrow \min\{\frac{g_i - g_j}{\kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_j, \mathbf{x}_j) - 2\kappa(\mathbf{x}_i, \mathbf{x}_j)}, B_i - \alpha_i, \alpha_j - A_j\}$

$\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$, $g_s \leftarrow g_s - \lambda(\kappa(\mathbf{x}_i, \mathbf{x}_s) - \kappa(\mathbf{x}_j, \mathbf{x}_s)) \forall s \in \mathcal{S}$

end if

$i \leftarrow \arg \max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$, $j \leftarrow \arg \min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$;

if $\alpha_i < B_i$, $\alpha_j > A_j$ and $g_i - g_j > \tau$ **then**

$\lambda \leftarrow \min\{\frac{g_i - g_j}{\kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_j, \mathbf{x}_j) - 2\kappa(\mathbf{x}_i, \mathbf{x}_j)}, B_i - \alpha_i, \alpha_j - A_j\}$

$\alpha_i \leftarrow \alpha_i + \lambda$, $\alpha_j \leftarrow \alpha_j - \lambda$, $g_s \leftarrow g_s - \lambda(\kappa(\mathbf{x}_i, \mathbf{x}_s) - \kappa(\mathbf{x}_j, \mathbf{x}_s)) \forall s \in \mathcal{S}$

$i \leftarrow \arg \max_{s \in \mathcal{S}} g_s$ with $\alpha_s < B_s$, $j \leftarrow \arg \min_{s \in \mathcal{S}} g_s$ with $\alpha_s > A_s$;

for $s \in \mathcal{S}$ such that $\alpha_s = 0$ **do**

if ($y_s = -1$ and $g_s \geq g_i$) or ($y_s = +1$ and $g_s \leq g_j$) **then**

$\mathcal{S} \leftarrow \mathcal{S} - s$

end if

end for

$b \leftarrow (g_i + g_j)/2$, $\delta \leftarrow g_i - g_j$

end if

end for

A.2 Proofs for the Chapter 3

A.2.1 The Proof for Proposition 3.2

Proof: The optimization (3.1) can be rewritten to the following equivalent optimization:

$$\min_{\gamma_a, d_{\gamma_b}} \quad \frac{k_a}{2}\gamma_a^2 + \frac{k_b}{2}d_{\gamma_b}^2 + w_{ab}\gamma_a d_{\gamma_b} - \ell_a\gamma_a - \ell_b d_{\gamma_b}, \quad (\text{A.1})$$

$$\gamma_a - C \leq 0, \quad (\text{A.2})$$

$$-\gamma_a \leq 0, \quad (\text{A.3})$$

$$d_{\gamma_b} - C + \hat{\gamma}_b \leq 0, \quad (\text{A.4})$$

$$-d_{\gamma_b} - \hat{\gamma}_b \leq 0, \quad (\text{A.5})$$

where $k_a, k_b > 0$, $w_{ab} \leq 0$, $\ell_a = 1 - y_a f(\mathbf{x}_a) \geq 0$, $\ell_b = 1 - y_b f(\mathbf{x}_b) \geq 0$ and $\hat{\gamma}_b > 0$. With $\lambda_1, \lambda_2, \lambda_3$ and λ_4 as Lagrange multipliers, the KKT conditions for this problem consist of the constraints (A.2)-(A.5), the nonnegativity constraints $\lambda_i \geq 0, \forall i$, the complementary slackness conditions

$$\lambda_1(\gamma_a - C) = 0, \quad \lambda_2(-\gamma_a) = 0, \quad \lambda_3(d_{\gamma_b} - C + \hat{\gamma}_b) = 0, \quad \lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$$

and zero gradient conditions:

$$k_a\gamma_a + w_{ab}d_{\gamma_b} - \ell_a + \lambda_1 - \lambda_2 = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}\gamma_a - \ell_b + \lambda_3 - \lambda_4 = 0.$$

We will discuss every possible condition to compute the closed-form solution. Firstly, we will discuss the case $\lambda_1 \neq 0$:

A.2.1.1 Case 1. If $\lambda_1 \neq 0$

Since $\lambda_1(\gamma_a - C) = 0$, we have $\gamma_a = C$; further, because $\lambda_2(-\gamma_a) = 0$, we have $\lambda_2 = 0$. Under the condition $\lambda_1 \neq 0$, we will discuss $\lambda_3 \neq 0$ and $\lambda_3 = 0$ separately as follows:

Sub-case 1.1. If $\lambda_3 \neq 0$ Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, we have $d_{\gamma_b} = C - \hat{\gamma}_b$, as a result $\lambda_4(-C) = 0$, so $\lambda_4 = 0$. Plugging the results $\gamma_a = C$, $\lambda_2 = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and $\lambda_4 = 0$ into the zero gradient condition, we have

$$k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b + \lambda_3 = 0.$$

Thus, we have

$$\lambda_1 = -[k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a] \quad \text{and} \quad \lambda_3 = -[k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b].$$

As a result, if

$$-(k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) > 0 \quad \text{and} \quad -(k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) > 0,$$

then KKT conditions are satisfied, $(\gamma_a, d_{\gamma_b}) = (C, C - \hat{\gamma}_b)$ is the unique solution.

Sub-case 1.2. If $\lambda_3 = 0$ When $\lambda_3 = 0$, we only conclude $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$.

Under the conditions $\lambda_1 \neq 0$ and $\lambda_3 = 0$, we will discuss the two cases $\lambda_4 \neq 0$ and $\lambda_4 = 0$, respectively as follows.

Sub-case 1.2.1. If $\lambda_4 \neq 0$. Since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, we have $d_{\gamma_b} = -\hat{\gamma}_b$. Plugging the results $\lambda_2 = 0$, $\gamma_a = C$, $\lambda_3 = 0$ and $d_{\gamma_b} = -\hat{\gamma}_b$ in to the zero gradient conditions:

$$k_a C + w_{ab}(-\hat{\gamma}_b) - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) + w_{ab}C - \ell_b - \lambda_4 = 0 \quad (\text{A.6})$$

But since $k_b(-\hat{\gamma}_b) < 0$, $w_{ab}C \leq 0$ and $\ell_b, \lambda_4 \geq 0$, $k_b(-\hat{\gamma}_b) + w_{ab}C - \ell_b - \lambda_4 < 0$, which contradicts the equation above.

Sub-case 1.2.2. If $\lambda_4 = 0$. Plugging the conditions $\gamma_a = C$, $\lambda_2 = 0$, $\lambda_3 = 0$ and $\lambda_4 = 0$ into the zero gradient equations:

$$k_a C + w_{ab}d_{\gamma_b} - \ell_a + \lambda_1 = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab}C - \ell_b = 0.$$

Solving the above equations leads to the following:

$$\lambda_1 = \frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} \quad \text{and} \quad d_{\gamma_b} = \frac{\ell_b - w_{ab}C}{k_b}.$$

If $\frac{w_{ab}^2 C - w_{ab}\ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0$ and $\frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$, then the KKT conditions are all satisfied; as a result, $(\gamma_a, d_{\gamma_b}) = (C, \frac{\ell_b - w_{ab}C}{k_b})$ is the unique optimal solution.

Next we will discuss the situation with the condition $\lambda_1 = 0$.

A.2.1.2 Case 2. If $\lambda_1 = 0$

Under the condition $\lambda_1 = 0$, we only conclude $\gamma_a \in [0, C]$. We will discuss the cases $\lambda_2 \neq 0$ and $\lambda_2 = 0$ under the condition $\lambda_1 = 0$, respectively.

Sub-case 2.1. If $\lambda_2 \neq 0$ Since $\lambda_2(-\gamma_a) = 0$, we conclude $\gamma_a = 0$. Under the conditions $\lambda_1 = 0$ and $\lambda_2 \neq 0$, we will discuss the cases $\lambda_3 \neq 0$ and $\lambda_3 = 0$:

Sub-case 2.1.1. If $\lambda_3 \neq 0$. Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, plugging the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and $\lambda_4 = 0$ into the zero gradient conditions:

$$w_{ab}(C - \hat{\gamma}_b) - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) - \ell_b + \lambda_3 = 0.$$

Since $w_{ab} \leq 0$, $C - \hat{\gamma}_b \geq 0$ and $\ell_a \geq 0$, we conclude

$$\lambda_2 = w_{ab}(C - \hat{\gamma}_b) - \ell_a \leq 0.$$

But $\lambda_2 \geq 0$ and $\lambda_2 \neq 0$, conclude $\lambda_2 > 0$, which contradicts the inequality above.

Sub-case 2.1.2. If $\lambda_3 = 0$. Under these known conditions, we only know $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$. Below, we will discuss the cases $\lambda_4 \neq 0$ and $\lambda_4 = 0$, under the conditions $\lambda_1 = 0$, $\lambda_2 \neq 0$ and $\lambda_3 = 0$.

- If $\lambda_4 \neq 0$, since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$. From the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $\lambda_3 = 0$ and $d_{\gamma_b} = -\hat{\gamma}_b$ and the zero gradient conditions, we have

$$w_{ab}(-\hat{\gamma}_b) - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) - \ell_b - \lambda_4 = 0.$$

Since $k_b, \hat{\gamma}_b > 0$ and $\ell_b \geq 0$, we conclude

$$\lambda_4 = k_b(-\hat{\gamma}_b) - \ell_b < 0.$$

But the equation above contradicts $\lambda_4 > 0$.

- Else if $\lambda_4 = 0$, from the conditions $\lambda_1 = 0$, $\gamma_a = 0$, $\lambda_3 = 0$ and $\lambda_4 = 0$ and the zero gradient conditions, we have

$$w_{ab}d_{\gamma_b} - \ell_a - \lambda_2 = 0 \quad \text{and} \quad k_b d_{\gamma_b} - \ell_b = 0.$$

Since $w_{ab} \leq 0$, $\ell_b, \ell_a \geq 0$ and $k_b > 0$,

$$\lambda_2 = w_{ab} \frac{\ell_b}{k_b} - \ell_a \leq 0,$$

which contradicts $\lambda_2 > 0$ (Since $\lambda_2 \neq 0$).

A.2.1.3 Sub-case 2.2. If $\lambda_2 = 0$

Under the conditions $\lambda_1 = \lambda_2 = 0$, we only know $\gamma_a \in [0, C]$. Below, we will discuss the two cases $\lambda_3 \neq 0$ and $\lambda_3 = 0$, under the conditions $\lambda_1 = \lambda_2 = 0$.

Sub-case 2.2.1. If $\lambda_3 \neq 0$. Since $\lambda_3[d_{\gamma_b} - (C - \hat{\gamma}_b)] = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$, as a result $\lambda_4(-C) = 0$, so $\lambda_4 = 0$. From the conditions $\lambda_1 = \lambda_2 = \lambda_4 = 0$, $d_{\gamma_b} = C - \hat{\gamma}_b$ and zero gradient conditions:

$$k_a \gamma_a + w_{ab}(C - \hat{\gamma}_b) - \ell_a = 0 \quad \text{and} \quad k_b(C - \hat{\gamma}_b) + w_{ab} \gamma_a - \ell_b + \lambda_3 = 0.$$

As a result, if

$$\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \quad \text{and} \quad \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab} \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0,$$

the unique optimal solution is $(\gamma_a, d_{\gamma_b}) = (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b)$.

Sub-case 2.2.2. If $\lambda_3 = 0$. According to $\lambda_1 = \lambda_2 = \lambda_3 = 0$, we only conclude $d_{\gamma_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b]$

- If $\lambda_4 \neq 0$, since $\lambda_4(-d_{\gamma_b} - \hat{\gamma}_b) = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$. From $\lambda_1 = \lambda_2 = \lambda_3 = 0$, $d_{\gamma_b} = -\hat{\gamma}_b$ and zero gradient conditions:

$$k_a \gamma_a + w_{ab}(-\hat{\gamma}_b) - \ell_a = 0 \quad \text{and} \quad k_b(-\hat{\gamma}_b) + w_{ab} \gamma_a - \ell_b - \lambda_4 = 0,$$

since $\lambda_4 = k_b(-\hat{\gamma}_b) + w_{ab} \gamma_a - \ell_b < 0$ which contradicts with the condition $\lambda_4 > 0$

- If $\lambda_4 = 0$, from $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$ and zero gradient conditions:

$$k_a \gamma_a + w_{ab} d_{\gamma_b} - \ell_a = 0 \quad \text{and} \quad k_b d_{\gamma_b} + w_{ab} \gamma_a - \ell_b = 0.$$

As a result, if γ_a and d_{γ_b} satisfy the following:

$$\gamma_a = \frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2} \in [0, C] \quad \text{and} \quad d_{\gamma_b} = \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b],$$

then $(\gamma_a, d_{\gamma_b}) = (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2})$ is the unique optimal solution.

Summary: The final closed-form solution to the optimization is summarized as:

$$(\gamma_a, d_{\gamma_b}) = \begin{cases} (C, C - \hat{\gamma}_b) & \text{if } (k_a C + w_{ab}(C - \hat{\gamma}_b) - \ell_a) < 0 \text{ and} \\ & (k_b(C - \hat{\gamma}_b) + w_{ab}C - \ell_b) < 0 \\ (C, \frac{\ell_b - w_{ab}C}{k_b}) & \text{if } \frac{w_{ab}^2 C - w_{ab} \ell_b - k_a k_b C + k_b \ell_a}{k_b} > 0 \text{ and} \\ & \frac{\ell_b - w_{ab}C}{k_b} \in [-\hat{\gamma}_b, C - \hat{\gamma}_b] \\ (\frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a}, C - \hat{\gamma}_b) & \text{if } \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} \in [0, C] \text{ and} \\ & \ell_b - k_b(C - \hat{\gamma}_b) - w_{ab} \frac{\ell_a - w_{ab}(C - \hat{\gamma}_b)}{k_a} > 0 \\ (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2}) & \text{if } (\frac{k_b \ell_a - w_{ab} \ell_b}{k_a k_b - w_{ab}^2}, \frac{k_a \ell_b - w_{ab} \ell_a}{k_a k_b - w_{ab}^2}) \in [0, C] \times [-\hat{\gamma}_b, C - \hat{\gamma}_b] \end{cases}.$$

A.3 Proofs for the Chapter 4

A.3.1 The Proof for Proposition 4.3

Proof: First of all, the product $H(Y_a) \cdot H(Y_b)$ can be simplified as:

$$H(Y_a) \cdot H(Y_b) = \sum_{i=1}^k \sigma(i, a) \sigma(i, b) = \sigma(r_a, a) \sigma(r_a, b) + \sigma(s_a, a) \sigma(s_a, b) = \sigma(r_a, b) - \sigma(s_a, b).$$

We can check the value of $\sigma(r_a, b) - \sigma(s_a, b)$ by examining all possible cases as follows:

- 1 If $r_a = r_b$ that implies that $\mathbf{x}_a$ and $\mathbf{x}_b$ have the same relevant labels, then we should have $H(Y_a) \cdot H(Y_b) = 1 - \sigma(s_a, b) \geq 1$ (either 1 or 2).
- 2 If $r_a \neq r_b$, then

2.1 If $r_a = s_b$, then $H_{Y_a} \cdot H_{Y_b} = \sigma(s_b, b) - \sigma(s_a, b) = -1 - \sigma(s_a, b) \leq -1$.

2.2 If $r_a \neq s_b$, then $H_{Y_a} \cdot H_{Y_b} = \sigma(r_a, b) - \sigma(s_a, b) = 0 - \sigma(s_a, b)$,

2.2.1 If $s_a = s_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(s_b, b) = 1$;

2.2.2 If $s_a = r_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(r_b, b) = -1$;

2.2.3 If $s_a \neq s_b$ and $s_a \neq r_b$, then $H_{Y_a} \cdot H_{Y_b} = -\sigma(s_a, b) = 0$.

We thus have the fact that $H(Y_a) \cdot H(Y_b) < 0$ holds if and only if $(r_a = s_b)$ or $(s_a = r_b)$.

A.3.2 The Proof for Proposition 4.4

In this appendix, we will derive the dual ascent by the multiclass double updating approach. Our approach to the proofs is mainly inspired by the study in [SS07], but our problem is different from their study.

For the convenience of our presentation, we introduce the following notation for our derivation. We denote the loss function for a training example $(\mathbf{x}, Y)$ as follows:

$$g(\bar{f}) = \ell(\bar{f}(\mathbf{x}), Y) = \max_{r \in Y, s \notin Y} \left[1 - (f_r(\mathbf{x}) - f_s(\mathbf{x})) \right]_+.$$

We order all the classes r in the assigned set Y as $r_1, \dots, r_{\|Y\|}$, and the class s in the unassigned set $\mathcal{Y} \setminus Y$ as $s_1, \dots, s_{\|[k]/Y\|}$. We slightly abuse our notations by simplifying $\langle f, g \rangle_{\mathcal{H}_\kappa}$ as $\langle f, g \rangle$ and $\|f\|_{\mathcal{H}_\kappa}$ as $\|f\|$ when there is no ambiguity about the space for computing dot product and norm.

We first give a lemma that shows the Fenchel conjugate of the above loss function g .

Lemma A.4 *Let $\mathcal{Y} = [k]$ be the possible labels set. $Y \subseteq \mathcal{Y}$ is relevant labels set for $x \in R^n$. $\bar{f} = (f_1, \dots, f_k)^T$, where $\forall i \in [k]$, $f_i \in \mathcal{H}_\kappa$. And the loss function is defined as follows:*

$$g(\bar{f}) = \max_{r \in Y, s \notin Y} \left[1 - (f_r(\mathbf{x}) - f_s(\mathbf{x})) \right]_+.$$

Then for any $\bar{\lambda} = (\lambda_1, \dots, \lambda_k)^T$, where $\forall i \lambda_i \in \mathcal{H}_\kappa$, we have g 's Fenchel conjugate as:

$$g^*(\bar{\lambda}) = \begin{cases} -\sum_{i,j} \alpha_{ij} & \text{if } \lambda_{r_i} + \sum_j \alpha_{ij} \kappa(\mathbf{x}, \cdot) = 0 \text{ and } \lambda_{s_j} - \sum_i \alpha_{ij} \kappa(\mathbf{x}, \cdot) = 0 \\ \infty & \text{otherwise} \end{cases}.$$

where $\bar{\alpha} = (\alpha_{ij}) \in \mathcal{A} = [A | A \in R_+^{\|Y\|} \times R_+^{(k-\|Y\|)}, \|A\|_1 \leq 1]$ and $(r_i \times s_j) \in \mathcal{B} = Y \times ([k]/Y)$.

Proof: The approach of our proof is similar to the method for proving the “Max-of-hinge” in [SS07]. First of all, it is not difficult to show that the loss function can be re-formulated as follows:

$$\begin{aligned} g(\bar{f}) &= \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \left[1 - (f_{r_i}(\mathbf{x}) - f_{s_j}(\mathbf{x})) \right] \\ &= \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \left[1 - (\langle f_{r_i}(\cdot), \kappa(\mathbf{x}, \cdot) \rangle - \langle f_{s_j}(\cdot), \kappa(\mathbf{x}, \cdot) \rangle) \right]. \end{aligned}$$

As a result, we have:

$$\begin{aligned} g^*(\bar{\lambda}) &= \max_{\bar{f}} \left\{ \langle \bar{\lambda}, \bar{f} \rangle - g(\bar{f}) \right\} \\ &= \max_{\bar{f}} \left\{ \sum_{n=1}^k \langle \lambda_n, f_n \rangle - \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \left[1 - (\langle f_{r_i}(\cdot), \kappa(\mathbf{x}, \cdot) \rangle - \langle f_{s_j}(\cdot), \kappa(\mathbf{x}, \cdot) \rangle) \right] \right\}. \end{aligned}$$

For any $f_n, \lambda_n \in \mathcal{H}_\kappa$, they can be written as: $f_n = \beta_n \kappa(\mathbf{x}, \cdot) + f_n^\perp$, $\lambda_n = \gamma_n \kappa(\mathbf{x}, \cdot) + \lambda_n^\perp$, where $f_n^\perp, \lambda_n^\perp \in \mathcal{V}^\perp$, $\mathcal{V} = \text{span}\{\kappa(\mathbf{x}, \cdot)\}$. As a result, we have

$$\begin{aligned} g^*(\bar{\lambda}) &= \max_{\bar{f}} \left\{ \sum_{n=1}^k (\langle \lambda_n^\perp, f_n^\perp \rangle + \beta_n \gamma_n \kappa(\mathbf{x}, \mathbf{x})) - \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \left[1 - (\beta_{r_i} \kappa(\mathbf{x}, \mathbf{x}) - \beta_{s_j} \kappa(\mathbf{x}, \mathbf{x})) \right] \right\}. \end{aligned}$$

When $\lambda_n^\perp \neq 0$, the $\max_{f^\perp} \langle \lambda_n^\perp, f_n^\perp \rangle$ will be ∞ , resulting $g^*(\bar{\lambda}) = \infty$. Otherwise, if $\lambda_n^\perp = 0$, $\forall n$, the term $f_n^\perp$ does not take effect for the objective; as a result, the optimal f_n can be written in the form of $\beta_n \kappa(\mathbf{x}, \cdot)$ and the conjugate is computed as follows:

$$\begin{aligned} g^*(\bar{\lambda}) &= \max_{\beta_n} \left\{ \sum_{n=1}^k \beta_n \gamma_n \kappa(\mathbf{x}, \mathbf{x}) - \max_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \sum_{i,j} \alpha_{ij} \left[1 - (\beta_{r_i} \kappa(\mathbf{x}, \mathbf{x}) - \beta_{s_j} \kappa(\mathbf{x}, \mathbf{x})) \right] \right\} \\ &= \max_{\beta_n} \min_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \left\{ \sum_{n=1}^k \langle \lambda_n, \beta_n \kappa(\mathbf{x}, \cdot) \rangle - \sum_{i,j} \alpha_{ij} \left[1 - (\langle \beta_{r_i} \kappa(\mathbf{x}, \cdot), \kappa(\mathbf{x}, \cdot) \rangle - \langle \beta_{s_j} \kappa(\mathbf{x}, \cdot), \kappa(\mathbf{x}, \cdot) \rangle) \right] \right\} \\ &= \min_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \max_{\beta_n} \left\{ \sum_{n=1}^k \langle \lambda_n, \beta_n \kappa(\mathbf{x}, \cdot) \rangle - \sum_{i,j} \alpha_{ij} \left[1 - (\langle \beta_{r_i} \kappa(\mathbf{x}, \cdot), \kappa(\mathbf{x}, \cdot) \rangle - \langle \beta_{s_j} \kappa(\mathbf{x}, \cdot), \kappa(\mathbf{x}, \cdot) \rangle) \right] \right\} \\ &= \min_{\bar{\alpha} \in \mathcal{A}, (r_i \times s_j) \in \mathcal{B}} \left\{ -\sum_{i,j} \alpha_{ij} + \max_{\beta_n} \left[\sum_{r_i} \langle \beta_{r_i} \kappa(\mathbf{x}, \cdot), \lambda_{r_i} \rangle + \sum_j \alpha_{ij} \kappa(\mathbf{x}, \cdot) + \sum_{s_j} \langle \beta_{s_j} \kappa(\mathbf{x}, \cdot), \lambda_{s_j} \rangle - \sum_i \alpha_{ij} \kappa(\mathbf{x}, \cdot) \right] \right\}. \end{aligned}$$

The fourth equality is guaranteed by the strong max-min property [BV04], and more importantly, we can see that only when $\bar{\alpha}$ satisfies $\lambda_{r_i} + \sum_j \alpha_{ij} \kappa(\mathbf{x}, \cdot) = 0$ and $\lambda_{s_j} - \sum_i \alpha_{ij} \kappa(\mathbf{x}, \cdot) = 0$, the second term in the equation above will be zero; otherwise, it will be ∞ . Therefore, we have the resulting Fenchel conjugate of $g(\bar{f})$ as follows:

$$g^*(\bar{\lambda}) = \begin{cases} -\sum_{i,j} \alpha_{ij} & \lambda_{r_i} + \sum_j \alpha_{ij} \kappa(\mathbf{x}, \cdot) = 0 \text{ and } \lambda_{s_j} - \sum_i \alpha_{ij} \kappa(\mathbf{x}, \cdot) = 0 \\ \infty & \text{otherwise} \end{cases}.$$

Given the above Fenchel dual of loss function, we can derive the dual for the optimization problem given on the right-hand side of Eq. (4.3), as given in the following lemma.

Lemma A.5 *Suppose the complexity measure function is given as $F(\bar{f}) = \sum_{i=1}^k \frac{1}{2} \|f_i\|_{\mathcal{H}_\kappa}^2$, and we set α_{ij} to zeros for $\forall (i, j) \in \{[Y_t \times ([k]/Y_t)]/(r_t, s_t)\}$, where (r_t, s_t) is defined in Eq. (4.1). Then the dual objective function for optimization given on the right-hand side of Eq. (4.3) can be expressed as follows:*

$$D(\gamma_1, \dots, \gamma_T) = - \sum_{i=1}^k \frac{1}{2} \left\| \sum_{t=1}^T \sigma(i, t) \gamma_t \kappa(\mathbf{x}_t, \cdot) \right\|^2 + \sum_{t=1}^T \gamma_t,$$

$$\text{where } \gamma_t \in [0, C] \text{ and } \sigma(i, t) = \begin{cases} 1 & \text{if } i = r_t \\ -1 & \text{if } i = s_t \\ 0 & \text{otherwise} \end{cases}.$$

Proof: The proof here resembles the one in the section 3.2 of [SS07]. Firstly, we note that the problem (4.3) is equivalent to the following:

$$\inf_{\bar{f}_0, \bar{f}_1, \dots, \bar{f}_T} \left(F(\bar{f}_0) + \sum_{t=1}^T C g_t(\bar{f}_t) \right) \text{ s.t. } \bar{f}_0, \bar{f}_t \in \bar{\mathcal{H}}_\kappa \text{ and } \forall t \in [T], \bar{f}_t = \bar{f}_0.$$

By introducing T function vectors $\bar{\lambda}_1, \dots, \bar{\lambda}_T$, in which each $\bar{\lambda}_t = (\lambda_{t,1}, \dots, \lambda_{t,k}) \in \bar{H}_\kappa$ is a Lagrange multipliers for the constraint $\bar{f}_t = \bar{f}_0$, we can obtain the following Lagrangian:

$$\mathcal{L}(\bar{f}_0, \dots, \bar{f}_T, \bar{\lambda}_1, \dots, \bar{\lambda}_T) = F(\bar{f}_0) + \sum_{t=1}^T C g_t(\bar{f}_t) + \sum_{t=1}^T \langle \bar{\lambda}_t, \bar{f}_0 - \bar{f}_t \rangle.$$

The dual objective function can be derived as follows:

$$\begin{aligned} D(\bar{\lambda}_1, \dots, \bar{\lambda}_T) &= \inf_{\bar{f}_0, \bar{f}_1, \dots, \bar{f}_T} \mathcal{L}(\bar{f}_0, \dots, \bar{f}_T, \bar{\lambda}_1, \dots, \bar{\lambda}_T) \\ &= - \sup_{\bar{f}_0} \left[\langle \bar{f}_0, - \sum_{t=1}^T \bar{\lambda}_t \rangle - F(\bar{f}_0) \right] - \sum_{t=1}^T \sup_{\bar{f}_t} [\langle \bar{f}_t, \bar{\lambda}_t \rangle - C g_t(\bar{f}_t)] \\ &= -F^* \left(- \sum_{t=1}^T \bar{\lambda}_t \right) - \sum_{t=1}^T (C g_t)^*(\bar{\lambda}_t) = -F^* \left(- \sum_{t=1}^T \bar{\lambda}_t \right) - \sum_{t=1}^T C g_t^* \left(\frac{\bar{\lambda}_t}{C} \right). \end{aligned}$$

Because $F(\bar{f}) = \sum_{i=1}^k \frac{1}{2} \|f_i\|_{\mathcal{H}_\kappa}^2$, we have $F^* = F$. The dual problem thus becomes:

$$D(\bar{\lambda}_1, \dots, \bar{\lambda}_T) = - \sum_{i=1}^k \frac{1}{2} \left\| - \sum_{t=1}^T \lambda_{t,i} \right\|_{\mathcal{H}_\kappa}^2 - \sum_{t=1}^T C g_t^* \left(\frac{\bar{\lambda}_t}{C} \right).$$

Because we want to maximize the dual objective, according to Lemma A.4, we should set

$$\frac{\lambda_{t,r_i^t}}{C} + \sum_j \alpha_{ij}^t k(\mathbf{x}_t, \cdot) = 0, \quad \frac{\lambda_{t,s_j^t}}{C} - \sum_i \alpha_{ij}^t k(\mathbf{x}_t, \cdot) = 0,$$

where $(\alpha_{ij}^t) \in \mathcal{A}_t$, $(r_i^t \times s_j^t) \in \mathcal{B}_t$, $\mathcal{A}_t = [A | A \in R_+^{\|Y_t\|} \times R_+^{(k-\|Y_t\|)}, \|A\|_1 \leq 1]$ and $\mathcal{B}_t = Y_t \times ([k]/Y_t)$. Furthermore, we set α_{ij} to zeros for $\forall (i, j) \in \{[Y_t \times ([k]/Y_t)]/(r_t, s_t)\}$. For simplicity, we denote α_{r_t, s_t}^t as $\frac{\gamma_t}{C}$. As a result, the dual objective function becomes

$$D(\gamma_1, \dots, \gamma_T) = - \sum_{i=1}^k \frac{1}{2} \left\| \sum_{t=1}^T \sigma(i, t) \gamma_t k(\mathbf{x}_t, \cdot) \right\|^2 + \sum_{t=1}^T \gamma_t,$$

where $\gamma_t \in [0, C]$ and $\sigma(i, t) = \begin{cases} 1 & \text{if } i = r_t \\ -1 & \text{if } i = s_t \\ 0 & \text{otherwise} \end{cases}$.

By applying Lemma A.5, we thus have the dual objective function for the t -th step as:

$$D_t(\gamma_1, \dots, \gamma_t) = - \sum_{i=1}^k \frac{1}{2} \left\| \sum_{j=1}^t \sigma(i, j) \gamma_j k(\mathbf{x}_j, \cdot) \right\|^2 + \sum_{j=1}^t \gamma_j. \quad (\text{A.7})$$

Now our goal is to derive the dual ascent guaranteed by the proposed double updating scheme. When pair $(\mathbf{x}_a, Y_a)$ is misclassified by the prediction function $\bar{f}_t = (f_{t,1}, \dots, f_{t,k})$, we will perform the update on the prediction function. Assume we conduct a double updating for $(\mathbf{x}_a, Y_a)$ and some auxiliary example $(\mathbf{x}_b, Y_b)$, we can prove Proposition 4.4 as follows.

Proof: According Eqn (A.7) obtained by Lemma A.5, before performing the double updating, the value of the dual function is expressed as:

$$D_{t-1} = - \sum_{i=1}^k \frac{1}{2} \left\| \sum_{j=1}^{t-1} \sigma(i, j) \hat{\gamma}_j k(\mathbf{x}_j, \cdot) \right\|^2 + \sum_{j=1}^{t-1} \hat{\gamma}_j = - \sum_{i=1}^k \frac{1}{2} \|f_{t-1,i}\|^2 + \sum_{j=1}^{t-1} \hat{\gamma}_j,$$

where $\hat{\gamma}_j$'s denote the weights of the prediction function $\bar{f}_{t-1}$ before the updating. After performing the dual update, the value of the new dual function can be written as:

$$D_t = - \sum_{i=1}^k \frac{1}{2} \|f_{t-1,i} + \sigma(i, a) \gamma_a k(\mathbf{x}_a, \cdot) + \sigma(i, b) d_{\gamma_b} k(\mathbf{x}_b, \cdot)\|^2 + \sum_{j=1}^{t-1} \hat{\gamma}_j + \gamma_a + d_{\gamma_b},$$

Hence, the dual ascent is computed as follows:

$$\begin{aligned} \Delta D &= D_t - D_{t-1} \\ &= \gamma_a \left(1 - (f_{t-1,r_a}(\mathbf{x}_a) - f_{t-1,s_a}(\mathbf{x}_a)) \right) + d_{\gamma_b} \left(1 - (f_{t-1,r_b}(\mathbf{x}_b) - f_{t-1,s_b}(\mathbf{x}_b)) \right) - \gamma_a^2 s_a \\ &\quad - d_{\gamma_b}^2 s_b - \sum_{i=1}^k \sigma(i, a) \sigma(i, b) \gamma_a d_{\gamma_b} k(\mathbf{x}_a, \mathbf{x}_b). \end{aligned}$$

A.4 Proofs for the Chapter 7

A.4.1 The Proof for Proposition 7.8

Proof: To facilitate the analysis, we denote $p_t = w_{1,t}\Pi(h(\mathbf{x}_{2_t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2_t}))$, $p_{1,t} = \Pi(h(\mathbf{x}_{2_t}))$, and $p_{2,t} = \Pi(f_t(\mathbf{x}_{2_t}))$.

It is straightforward to show that $F(z) = \exp(-\frac{1}{2}\ell^*(z, y))$ is concave with respect to z for all y . Then according to Jensen's inequality

$$\exp(-\frac{1}{2}\ell^*(p_t, \Pi(y_{2_t}))) \geq \frac{\sum_{i=1}^2 w_{i,t} \exp(-\frac{1}{2}\ell^*(p_{i,t}, \Pi(y_{2_t})))}{\sum_{i=1}^2 w_{i,t}}.$$

Denoting $r_{i,t} = \ell^*(p_t, \Pi(y_{2_t})) - \ell^*(p_{i,t}, \Pi(y_{2_t}))$ and rearranging the above inequality result in:

$$\sum_{i=1}^2 w_{i,t} \exp(\frac{1}{2}r_{i,t}) \leq \sum_{i=1}^2 w_{i,t}.$$

Combining the equalities (7.2) and (7.3) with the above inequality, we have

$$\sum_{i=1}^2 \exp(-\frac{1}{2} \sum_{j=1}^{t-1} \ell^*(p_{i,j}, \Pi(y_{2_j}))) \exp(\frac{1}{2}r_{i,t}) \leq \sum_{i=1}^2 \exp(-\frac{1}{2} \sum_{j=1}^{t-1} \ell^*(p_{i,j}, \Pi(y_{2_j}))).$$

Multiplying the two sides of the above inequality with $\exp(\frac{1}{2} \sum_{j=1}^{t-1} \ell^*(p_j, \Pi(y_{2_j})))$ results in

$$\sum_{i=1}^2 \exp(\frac{1}{2} \sum_{j=1}^t r_{i,j}) \leq \sum_{i=1}^2 \exp(\frac{1}{2} \sum_{j=1}^{t-1} r_{i,j}),$$

which further implies

$$\sum_{i=1}^2 \exp(\frac{1}{2} \sum_{j=1}^T r_{i,j}) \leq \sum_{i=1}^2 \exp(\frac{1}{2} \sum_{j=1}^{T-1} r_{i,j}) \leq \dots \sum_{i=1}^2 \exp(\frac{1}{2} \sum_{j=1}^0 r_{i,j}) = 2 \ln(2),$$

Finally,

$$\sum_{t=1}^T \ell^*(p_t, \Pi(y_{2t})) - \min_{i=1,2} \sum_{t=1}^T \ell^*(p_{i,t}, \Pi(y_{2t})) = \max_{i=1,2} \sum_{t=1}^T r_{i,t} \leq 2 \ln \left(\sum_{i=1}^2 \exp \left(\frac{1}{2} \sum_{t=1}^T r_{i,t} \right) \right) \leq 2 \ln(2).$$

Plugging $p_t = w_{1,t}\Pi(h(\mathbf{x}_{2t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2t}))$, $p_{1,t} = \Pi(h(\mathbf{x}_{2t}))$, and $p_{2,t} = \Pi(f_t(\mathbf{x}_{2t}))$ into the above inequality concludes the claim.

A.4.2 The Proof for Theorem 19

Proof: First notice that whenever there is a mistake at some t -th step, we should have $|w_{1,t}\Pi(h(\mathbf{x}_{2t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2t})) - \Pi(y_{2t})| \geq \frac{1}{2}$. Thus, we have

$$\begin{aligned} & \sum_{t=1}^T \ell^*(w_{1,t}\Pi(h(\mathbf{x}_{2t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2t})), \Pi(y_{2t})) \\ &= \sum_{t=1}^T (w_{1,t}\Pi(h(\mathbf{x}_{2t})) + w_{2,t}\Pi(f_t(\mathbf{x}_{2t})) - \Pi(y_{2t}))^2 \geq \frac{1}{4}M. \end{aligned}$$

Combining the above fact with Proposition 7.8, we have

$$\frac{1}{4}M \leq \min \left\{ \sum_{t=1}^T \ell^*(\Pi(h(\mathbf{x}_{2t})), \Pi(y_{2t})), \sum_{t=1}^T \ell^*(\Pi(f_t(\mathbf{x}_{2t})), \Pi(y_{2t})) \right\} + 2 \ln 2.$$

The theorem follows directly by multiplying 4 both sides of the above inequality.

A.4.3 The Proof for Proposition 7.9

Proof: It is easy to see that the optimization problem 7.4 is equivalent to the following problem

$$\begin{aligned} & \min_{f^{(1)} \in \mathcal{H}_{\kappa_1}, f^{(2)} \in \mathcal{H}_{\kappa_2}} \frac{\gamma_1}{2} \|f^{(1)} - f_t^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|f^{(2)} - f_t^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + C\xi \\ & s.t. \quad 1 - y_{2t} \frac{1}{2} \left(f^{(1)}(\mathbf{x}_{2t}^{(1)}) + f^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) \leq \xi \quad \text{and} \quad \xi \geq 0. \end{aligned}$$

The Lagrangian of the above optimization is:

$$\begin{aligned} & \mathcal{L}(f^{(1)}, f^{(2)}, \xi, \tau_t, \lambda) \\ &= \frac{\gamma_1}{2} \|f^{(1)} - f_t^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|f^{(2)} - f_t^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + C\xi + \tau_t \left(1 - y_{2t} \frac{1}{2} \left(f^{(1)}(\mathbf{x}_{2t}^{(1)}) + f^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) - \xi \right) - \lambda \xi \\ &= \frac{\gamma_1}{2} \|f^{(1)} - f_t^{(1)}\|_{\mathcal{H}_{\kappa_1}}^2 + \frac{\gamma_2}{2} \|f^{(2)} - f_t^{(2)}\|_{\mathcal{H}_{\kappa_2}}^2 + \xi(C - \tau_t - \lambda) + \tau_t \left(1 - y_{2t} \frac{1}{2} \left(f^{(1)}(\mathbf{x}_{2t}^{(1)}) + f^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) \right), \end{aligned} \tag{A.8}$$

where $\tau_t \geq 0$ and $\lambda \geq 0$ are Lagrange multipliers. We now find the minimum of the Lagrangian with respect to $f^{(1)}$, $f^{(2)}$ and ξ by setting their partial derivatives to zeros. We get $f^{(i)} = f_t^{(i)} + \frac{\tau_t}{2\gamma_i} y_{2t} \kappa_i(\mathbf{x}_{2t}^{(i)}, \cdot)$ for $i = 1, 2$ and $C - \tau_t - \lambda = 0$. And since $\lambda \geq 0$, we conclude $C \geq \tau_t$. We thus have $\tau_t \in [0, C]$.

Plugging the three equations $f^{(i)} = f_t^{(i)} + \frac{\tau_t}{2\gamma_i} y_{2t} \kappa_i(\mathbf{x}_{2t}^{(i)}, \cdot)$ (where $i = 1, 2$) and $C - \tau_t - \lambda = 0$ into Eq. (A.8), we have

$$\mathcal{L}(\tau_t) = -\tau_t^2 \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) + \tau_t \ell_t.$$

By setting the partial derivative of the above equation to zero, we have

$$\tau_t = \ell_t / \left(\frac{k_t^1}{4\gamma_1} + \frac{k_t^2}{4\gamma_2} \right) = \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}.$$

Finally, combining the result $\tau_t \in [0, C]$, we thus have the solution: $\tau_t = \min\{C, \frac{4\gamma_1\gamma_2\ell_t}{k_t^1\gamma_2 + k_t^2\gamma_1}\}$.

A.4.4 The Proof for Lemma 7.3

Proof: Let $\Delta_t = \frac{\gamma_1}{2} \left(\|f_t^{(1)} - g^{(1)}\|^2 - \|f_{t+1}^{(1)} - g^{(1)}\|^2 \right) + \frac{\gamma_2}{2} \left(\|f_t^{(2)} - g^{(2)}\|^2 - \|f_{t+1}^{(2)} - g^{(2)}\|^2 \right)$, then

$$\begin{aligned} & \sum_{t=1}^T \Delta_t \\ &= \sum_{t=1}^T \left\{ \frac{\gamma_1}{2} \left(\|f_t^{(1)} - g^{(1)}\|^2 - \|f_{t+1}^{(1)} - g^{(1)}\|^2 \right) + \frac{\gamma_2}{2} \left(\|f_t^{(2)} - g^{(2)}\|^2 - \|f_{t+1}^{(2)} - g^{(2)}\|^2 \right) \right\} \\ &= \frac{\gamma_1}{2} \left(\|h - g^{(1)}\|^2 - \|f_{T+1}^{(1)} - g^{(1)}\|^2 \right) + \frac{\gamma_2}{2} \left(\|f_1^{(2)} - g^{(2)}\|^2 - \|f_{T+1}^{(2)} - g^{(2)}\|^2 \right) \\ &\leq \frac{\gamma_1}{2} (\|h - g^{(1)}\|^2) + \frac{\gamma_2}{2} (\|g^{(2)}\|^2). \end{aligned}$$

Second, when $\ell_t = 0$, $f_{t+1}^{(i)} = f_t^{(i)}$ for $i = 1, 2$, it is clear $\Delta_t = 0$; when $\ell_t > 0$, $f_{t+1}^{(i)} = f_t^{(i)} + \frac{\tau_t}{2\gamma_i} y_{2t} \kappa_i(\mathbf{x}_{2t}^{(i)}, \cdot)$, we compute Δ_t as:

$$\begin{aligned} & \Delta_t \\ &= \frac{\gamma_1}{2} \left(\|f_t^{(1)} - g^{(1)}\|^2 - \|f_{t+1}^{(1)} - g^{(1)}\|^2 \right) + \frac{\gamma_2}{2} \left(\|f_t^{(2)} - g^{(2)}\|^2 - \|f_{t+1}^{(2)} - g^{(2)}\|^2 \right) \\ &= \tau_t \left\{ -\frac{y_{2t}}{2} \left(f_t^{(1)}(\mathbf{x}_{2t}^{(1)}) + f_t^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) + \frac{y_{2t}}{2} \left(g^{(1)}(\mathbf{x}_{2t}^{(1)}) + g^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) - \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \right\}. \quad (\text{A.9}) \end{aligned}$$

We also have $\ell_t = 1 - y_{2t} \left(\frac{1}{2} (f_t^{(1)}(\mathbf{x}_{2t}^{(1)}) + f_t^{(2)}(\mathbf{x}_{2t}^{(2)})) \right)$ since $\ell_t > 0$. This is equivalent to the following:

$$\frac{y_{2t}}{2} (f_t^{(1)}(\mathbf{x}_{2t}^{(1)}) + f_t^{(2)}(\mathbf{x}_{2t}^{(2)})) = 1 - \ell_t.$$

In addition,

$$\ell(g^{(1)}, g^{(2)}; t) = \left[1 - y_{2t} \frac{1}{2} \left(g^{(1)}(\mathbf{x}_{2t}^{(1)}) + g^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) \right]_+ \geq 1 - y_{2t} \frac{1}{2} \left(g^{(1)}(\mathbf{x}_{2t}^{(1)}) + g^{(2)}(\mathbf{x}_{2t}^{(2)}) \right),$$

we thus have

$$\frac{y_{2t}}{2} \left(g^{(1)}(\mathbf{x}_{2t}^{(1)}) + g^{(2)}(\mathbf{x}_{2t}^{(2)}) \right) \geq 1 - \ell(g^{(1)}, g^{(2)}; (\mathbf{x}_{2t}, y_{2t})).$$

Combining these two facts and inequality (A.9), we thus have the following result:

$$\begin{aligned} \Delta_t &\geq \tau_t \left(-(1 - \ell_t) + 1 - \ell(g^{(1)}, g^{(2)}; (\mathbf{x}_{2t}, y_{2t})) - \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \right) \\ &= \tau_t \left(\ell_t - \ell(g^{(1)}, g^{(2)}; (\mathbf{x}_{2t}, y_{2t})) - \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \right). \end{aligned}$$

Hence, we have the following conclusion:

$$\sum_{t=1}^T \tau_t \left(\ell_t - \ell(g^{(1)}, g^{(2)}; (\mathbf{x}_{2t}, y_{2t})) - \left(\frac{k_t^1}{8\gamma_1} + \frac{k_t^2}{8\gamma_2} \right) \tau_t \right) \leq \frac{\gamma_1}{2} \|h - g^{(1)}\|^2 + \frac{\gamma_2}{2} \|g^{(2)}\|^2.$$

Appendix B

Publication List

- **Peilin Zhao**, Steven C.H. Hoi. Cost-Sensitive Online Active Learning with Application to Malicious URL Detection. 19th Conference on Knowledge Discovery and Data Mining (**KDD2013**), Chicago, Illinois, USA, 11-14, August, 2013. To Appear.
- **Peilin Zhao**, Steven C.H. Hoi, Jinfeng Zhuang. Active Learning with Expert Advice. The Conference on Uncertainty in Artificial Intelligence (**UAI2013**), Bellevue, Washington, USA, 11-15, July, 2013. To Appear.
- Jialei Wang, Steven C.H. Hoi, **Peilin Zhao**, Jinfeng Zhuang, Zhi-Yong Liu. Large Scale Online Kernel Classification. International Joint Conference On Artificial Intelligence (**IJCAI2013**), Beijing, China, 03-09 August, 2013. To appear.
- Jialei Wang, **Peilin Zhao**, Steven C.H. Hoi, Rong Jin. Online Feature Selection and Its Applications. IEEE Transactions on Knowledge and Data Engineering (**TKDE**), 2013. To appear.
- **Peilin Zhao**, Steven C.H. Hoi. Cost-Sensitive Double Updating Online Learning and Its Application to Online Anomaly Detection. SIAM International Conference on Data Mining (**SDM2013**), Austin, Texas, USA, 2-4 May 2013. To appear.
- Bin Li, Steven C.H. Hoi, **Peilin Zhao**, Vivekanand Gopalkrishnan. Confidence Weighted Mean Reversion Strategy for On-Line Portfolio Selection. Transactions on Knowledge Discovery from Data (**TKDD**), 2012.
- Steven C.H. Hoi, Rong Jin, **Peilin Zhao**, Tianbao Yang. Online Multiple Kernel Classification. Machine Learning (**ML**), 2012.
- Jialei Wang, **Peilin Zhao**, Steven C.H. Hoi. Cost-Sensitive Online Classification. IEEE International Conference on Data Mining (**ICDM2012**), Brussels, Belgium, 10 - 13 December, 2012.

- **Peilin Zhao**, Steven C.H. Hoi. BDUOL: Double Updating Online Learning on a Fixed Budget. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (**ECML-PKDD2012**): 810-826, Bristol, UK, 24-28 September 2012.
- Steven C.H. Hoi, Jialei Wang, **Peilin Zhao**, Rong Jin. Online Feature Selection for Big Data Mining. The 1st ACM **SIGKDD** Workshop on Big Data Mining (**BigMine**) Beijing, P.R. China Aug, 2012.
- Jialei Wang, **Peilin Zhao**, Steven C.H. Hoi. Exact Soft Confidence-Weighted Learning. The 29th International Conference on Machine Learning (**ICML2012**), Edinburgh, Scotland, 26 June - 1 July 2012.
- **Peilin Zhao**, Jialei Wang, Pengcheng Wu, Rong Jin, Steven C.H. Hoi. Fast Bounded Online Gradient Descent Algorithms for Scalable Kernel-Based Online Learning. The 29th International Conference on Machine Learning (**ICML2012**), Edinburgh, Scotland, 26 June - 1 July 2012.
- Bin Li, **Peilin Zhao**, Steven C. H. Hoi, Vivekanand Gopalkrishnan. PAMR: Passive aggressive mean reversion strategy for portfolio selection. Machine Learning (**ML**) 87(2): 221-258, 2012.
- **Peilin Zhao**, Steven C.H. Hoi, Rong Jin, Tianbao Yang. Online AUC Maximization. The 28th International Conference on Machine Learning (**ICML2011**): 233-240, Bellevue, Washington, USA, 28 June - 2 July 2011.
- **Peilin Zhao**, Steven C.H. Hoi, Rong Jin. Double Updating Online Learning. Journal of Machine Learning Research (**JMLR**) 12: 1587-1615, May 2011.
- Bin Li, Steven C.H. Hoi, **Peilin Zhao**, Vivekanand Gopalkrishnan. Confidence Weighted Mean Reversion Strategy for On-Line Portfolio Selection. Journal of Machine Learning Research (JMLR), W&CP 10, vol. 10, (**AISTATS2011**): 434-442, Fort Lauderdale, FL, USA, 11-13 April 2011.
- Pengcheng Wu, Steven C.H. Hoi, **Peilin Zhao**, Ying He. Mining Social Images with Distance Metric Learning for Automated Image Tagging. The fourth ACM International Conference on Web Search and Data Mining (**WSDM2011**): 197-206, Hong Kong, China, 9-12 February 2011.
- **Peilin Zhao** and Steven C.H. Hoi. OTL: A Framework of Online Transfer Learning. The 27th International Conference on Machine Learning (**ICML2010**): 1231-1238, Haifa, Israel, 21-24 June 2010.

- Shenghua Gao, Ivor W. Tsang, Liang-Tien Chia, **Peilin Zhao**. Local Features Are Not Lonely - Laplacian Sparse Coding for Image Classification. Proceedings of the 23rd IEEE Computer Society Conference on Computer Vision and Pattern Recognition (**CVPR2010**): 3555-3561, San Francisco, CA, USA, 13-18 June 2010.
- **Peilin Zhao**, Steven C.H. Hoi, Rong Jin. DUOL: A Double Updating Approach for Online Learning. Advances in Neural Information Processing Systems (**NIPS2009**): 2259-2267, Vancouver, B.C., Canada, 7-10 December 2009.
- **Peilin Zhao**, Steven C.H. Hoi. Online Transfer Learning: Towards Knowledge Transfer via Online learning. Advances in Neural Information Processing Systems (**NIPS2009**) Workshop on Transfer Learning, December 7-10, 2009.

References

- [ABR64] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
- [ABR07] Jacob Abernethy, Peter L. Bartlett, and Alexander Rakhlin. Multitask learning with expert advice. In *COLT*, pages 484–498, 2007.
- [ABRT08] Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal strategies and minimax lower bounds for online convex games. In *COLT*, pages 415–424, 2008.
- [ACBG02] Peter Auer, Nicolò Cesa-Bianchi, and Claudio Gentile. Adaptive and self-confident on-line learning algorithms. *J. Comput. Syst. Sci.*, 64(1):48–75, 2002.
- [Agm54] Shmuel Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3):382–392, 1954.
- [AKJ04] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *ECML*, pages 39–50, 2004.
- [ANC07] Andrew Arnold, Ramesh Nallapati, and William W. Cohen. A comparative study of methods for transductive transfer learning. In *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, pages 77–82, Washington, DC, USA, 2007.
- [Aro50] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68, 1950.
- [BB07] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *NIPS*, 2007.

REFERENCES

- [BBGW07] Antoine Bordes, Léon Bottou, Patrick Gallinari, and Jason Weston. Solving multiclass support vector machines with larank. In *ICML*, pages 89–96, 2007.
- [BEWB05] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.
- [BGV92] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, pages 144–152, 1992.
- [BHR07] Peter L. Bartlett, Elad Hazan, and Alexander Rakhlin. Adaptive online gradient descent. In *NIPS*, 2007.
- [BL03] Léon Bottou and Yann LeCun. Large scale online learning. In *NIPS*, 2003.
- [BM10] Stephen H. Bach and Marcus A. Maloof. A bayesian approach to concept drift. In *NIPS*, pages 127–135, 2010.
- [BOSB10] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M. Buhmann. The balanced accuracy and its posterior distribution. In *ICPR*, pages 3121–3124, 2010.
- [Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.
- [BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [CBCG03] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. Learning probabilistic linear-threshold classifiers via selective sampling. In *COLT*, pages 373–387, 2003.
- [CBCG04] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Inf. Theory*, 50(9):2050–2057, 2004.
- [CBCG05] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *SIAM J. Comput.*, 34(3):640–668, 2005.

REFERENCES

- [CBG08] Nicolò Cesa-Bianchi and Claudio Gentile. Improved risk tail bounds for on-line algorithms. *IEEE Transactions on Information Theory*, 54(1):386–390, 2008.
- [CBGO09] Nicolò Cesa-Bianchi, Claudio Gentile, and Francesco Orabona. Robust bounds for classification via selective sampling. In *ICML*, page 16, 2009.
- [CBGZ04] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In *NIPS*, 2004.
- [CBGZ06] Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear classification. *Journal of Machine Learning Research*, 7:1205–1230, 2006.
- [CBHK02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *JAIR*, 16:321–357, 2002.
- [CBL06] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.
- [CCBG07] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167, 2007.
- [CCBG08] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. In *COLT*, pages 251–262, 2008.
- [CCBG11] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Learning noisy linear classifiers via adaptive and selective sampling. *Machine Learning*, 83(1):71–102, 2011.
- [CDK⁺06] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res. (JMLR)*, 7:551–585, 2006.
- [CDK09] Koby Crammer, Mark Dredze, and Alex Kulesza. Multi-class confidence weighted algorithms. In *EMNLP*, pages 496–504, 2009.

REFERENCES

- [CDP08] Koby Crammer, Mark Dredze, and Fernando Pereira. Exact convex confidence-weighted learning. In *NIPS*, pages 345–352, 2008.
- [CG11] Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. In *ICML*, pages 273–280, 2011.
- [CKD09] Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.
- [CKS03] Koby Crammer, Jaz S. Kandola, and Yoram Singer. Online classification on a budget. In *Advances in Neural Information Processing Systems*, 2003.
- [CM03] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *NIPS*, 2003.
- [CP00] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In *NIPS*, pages 409–415, 2000.
- [CS01] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.
- [CS02] Koby Crammer and Yoram Singer. A new family of online algorithms for category ranking. In *SIGIR*, pages 151–158, 2002.
- [CS03] Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res. (JMLR)*, 3:951–991, 2003.
- [CS05] Koby Crammer and Yoram Singer. Loss bounds for online category ranking. In *The 18th Annual Conference on Learning Theory (COLT2005)*, pages 48–62, 2005.
- [CST00] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines : and other kernel-based learning methods*. Cambridge University Press, 1 edition, March 2000.
- [CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

REFERENCES

- [CVS⁺06] Li Cheng, S. V. N. Vishwanathan, Dale Schuurmans, Shaojun Wang, and Terry Caelli. Implicit online learning with kernels. In *Advances in Neural Information Processing Systems*, pages 249–256, 2006.
- [DCP08] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *ICML*, pages 264–271, 2008.
- [DGS10] Ofer Dekel, Claudio Gentile, and Karthik Sridharan. Robust selective sampling from single and multiple teachers. In *COLT*, pages 346–358, 2010.
- [DH03] Chris Drummond and Robert C. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In *ICML ’03 Workshop on Learning from Imbalanced Data Sets*, pages 1–8, 2003.
- [DHS11] 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.
- [DKJ⁺07] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007.
- [DLS06] Ofer Dekel, Philip M. Long, and Yoram Singer. Online multitask learning. In *COLT*, pages 453–467, 2006.
- [DLS07] Ofer Dekel, Philip M. Long, and Yoram Singer. Online learning of multiple tasks with a shared loss. *J. Mach. Learn. Res.*, 8:2233–2264, 2007.
- [DM06] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.
- [Dom99] Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *KDD’99*, pages 155–164, San Diego, CA, USA, 1999. ACM.
- [DSSS05] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a fixed budget. In *Advances in Neural Information Processing Systems*, 2005.
- [DSSS08] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. The forgetron: A kernel-based perceptron on a budget. *SIAM J. Comput.*, 37(5):1342–1372, 2008.

REFERENCES

- [DXT12] Lixin Duan, Dong Xu, and Ivor Wai-Hung Tsang. Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Trans. Neural Netw. Learning Syst.*, 23(3):504–518, 2012.
- [DYXY08] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Self-taught clustering. In *ICML*, pages 200–207, 2008.
- [Elk01] Charles Elkan. The foundations of cost-sensitive learning. In *IJCAI*, pages 973–978, 2001.
- [EMM02] Yaakov Engel, Shie Mannor, and Ron Meir. Sparse online greedy support vector regression. In *ECML*, pages 84–96, 2002.
- [FKM05] Abraham Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *SODA*, pages 385–394, 2005.
- [FS99] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.
- [FSSSU06] Michael Fink, Shai Shalev-Shwartz, Yoram Singer, and Shimon Ullman. Online multiclass learning by interclass hypothesis sharing. In *Proceedings of International Conference on Machine Learning*, pages 313–320, 2006.
- [Gen01] Claudio Gentile. A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res. (JMLR)*, 2:213–242, 2001.
- [Gen03] Claudio Gentile. The robustness of the p-norm algorithms. *Machine Learning*, 53(3):265–299, 2003.
- [GLS97] Adam J. Grove, Nick Littlestone, and Dale Schuurmans. General convergence results for linear discriminant updates. In *The Annual Conference on Learning Theory (COLT’97)*, pages 171–183, 1997.
- [Her01] Mark Herbster. Learning additive models online with fast evaluating kernels. In *COLT/EuroCOLT*, pages 444–460, 2001.
- [HK00] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

REFERENCES

- [HK11] Elad Hazan and Satyen Kale. Newtron: an efficient bandit algorithm for online multiclass prediction. In *NIPS*, pages 891–899, 2011.
- [HKKA06] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *COLT*, pages 499–513, 2006.
- [HSG⁺06] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *NIPS*, pages 601–608, 2006.
- [HW01] Mark Herbster and Manfred K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.
- [JKDG08] Prateek Jain, Brian Kulis, Inderjit S. Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *NIPS*, pages 761–768, 2008.
- [KSST08] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of International Conference on Machine Learning*, pages 440–447, 2008.
- [KSW04] Jyrki Kivinen, Alexander J. Smola, and Robert C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 52(8):2165–2176, 2004.
- [KT08] Sham M. Kakade and Ambuj Tewari. On the generalization ability of online strongly convex programming algorithms. In *NIPS*, pages 801–808, 2008.
- [KV05] Adam Tauman Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005.
- [KW97] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, 132(1):1–63, 1997.
- [KW01] Jyrki Kivinen and Manfred K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.
- [LA08] Aurélie C. Lozano and Naoki Abe. Multi-class cost-sensitive boosting with p-norm loss functions. In *KDD’08*, pages 506–514, Las Vegas, Nevada, USA, 2008. ACM.

REFERENCES

- [Lit87] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm (extended abstract). In *FOCS*, pages 68–77, 1987.
- [LKZ10] Yu-Feng Li, James T. Kwok, and Zhi-Hua Zhou. Cost-sensitive semi-supervised support vector machine. In *AAAI*, 2010.
- [LL99] Yi Li and Philip M. Long. The relaxed online maximum margin algorithm. In *NIPS*, pages 498–504, 1999.
- [LPS09] Gábor Lugosi, Omiros Papaspiliopoulos, and Gilles Stoltz. Online multi-task learning with hard constraints. *CoRR*, abs/0902.3526, 2009.
- [LW86] Nick Littlestone and Manfred K. Warmuth. Relating data compression and learnability. Technical report, 1986.
- [LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.
- [LZH⁺02] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola. The perceptron algorithm with uneven margins. In *ICML*, pages 379–386, 2002.
- [LZHG12] Bin Li, Peilin Zhao, Steven C. H. Hoi, and Vivekanand Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 87(2):221–258, 2012.
- [NMC07] Alexandru Niculescu-Mizil and Rich Caruana. Inductive transfer for bayesian network structure learning. In *AISTATS2007*, 2007.
- [Nov62] Albert B. Novikoff. On convergence proofs on perceptrons. pages 615–622, 1962.
- [OBM64] E.M. Braverman O. Bashkirov and I.E. Muchnik. Potential function algorithms for pattern recognition learning machines. *Automation and Remote Control*, 25:692C695, 1964.
- [OC10] Francesco Orabona and Koby Crammer. New adaptive algorithms for online classification. In *NIPS*, pages 1840–1848, 2010.

REFERENCES

- [OCB11] Francesco Orabona and Nicolò Cesa-Bianchi. Better algorithms for selective sampling. In *ICML*, pages 433–440, 2011.
- [OKC08] Francesco Orabona, Joseph Keshet, and Barbara Caputo. The projectron: a bounded kernel-based perceptron. In *Proceedings of International Conference on Machine Learning*, pages 720–727, 2008.
- [OKC09] Francesco Orabona, Joseph Keshet, and Barbara Caputo. Bounded kernel-based online learning. *Journal of Machine Learning Research*, 10:2643–2666, 2009.
- [PTKY11] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.
- [PY09] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2009.
- [QPS09] Novi Quadrianto, James Petterson, and Alex J. Smola. Distribution matching for transduction. In *NIPS*, pages 1500–1508, 2009.
- [RBL⁺07] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, pages 759–766, 2007.
- [Ros58] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.
- [SC08] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer Publishing Company, Incorporated, 1st edition, 2008.
- [SS01] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001.
- [SS07] Shai Shalev-Shwartz. Online learning: Theory, algorithms, and applications. In *Ph.D thesis*, 2007.

REFERENCES

- [SSK08] Shai Shalev-Shwartz and Sham M. Kakade. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, pages 1457–1464, 2008.
- [SSM97] Bernhard Schölkopf, Alex J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ICANN*, pages 583–588, 1997.
- [SSS05] Shai Shalev-Shwartz and Yoram Singer. A new perspective on an old perceptron algorithm. In *The 18th Annual Conference on Learning Theory (COLT2005)*, pages 264–278, 2005.
- [SSS06a] Shai Shalev-Shwartz and Yoram Singer. Convex repeated games and fenchel duality. In *NIPS*, pages 1265–1272, 2006.
- [SSS06b] Shai Shalev-Shwartz and Yoram Singer. Online learning meets optimization in the dual. In *The 19th Annual Conference on Learning Theory (COLT2006)*, pages 423–437, 2006.
- [SsS07a] Shai Shalev-shwartz and Yoram Singer. Logarithmic regret algorithms for strongly convex repeated games. In *The Hebrew University*, 2007.
- [SSS07b] Shai Shalev-Shwartz and Yoram Singer. A primal-dual perspective of online learning algorithms. *Machine Learning*, 69(2-3):115–142, 2007.
- [SSS07c] Shai Shalev-Shwartz and Yoram Singer. A unified algorithmic approach for efficient online label ranking. *Journal of Machine Learning Research - Proceedings Track*, 2:452–459, 2007.
- [SSSN04] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *ICML*, 2004.
- [SSSS07] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pages 807–814, 2007.
- [STY07] K. Slavakis, S. Theodoridis, and I. Yamada. Online sparse kernel-based classification by projections. In *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 294–299, 2007.
- [Tan93] Ming Tan. Cost-sensitive learning of classification knowledge and its applications in robotics. *Mach. Learn.*, 13(1):7–33, October 1993.

REFERENCES

- [Tur95] Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *JAIR*, 2:369–409, 1995.
- [UKM06] Andrew V. Uzilov, Joshua M. Keegan, and David H. Mathews. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics*, 7:173, 2006.
- [Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [VCB08] Giorgio Valentini and Nicolò Cesa-Bianchi. Hcgene: a software tool to support the hierarchical classification of genes. *Bioinformatics*, 24(5):729–731, 2008.
- [VCC99] K. Veropoulos, C. Campbell, and N. Cristianini. Controlling the sensitivity of support vector machines. In *IJCAI*, pages 55–60, 1999.
- [VL63] V.N. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. In *Automation and Remote Control*, volume 24, page 774C780, 1963.
- [Vov97] Volodya Vovk. Competitive on-line linear regression. In *NIPS*, 1997.
- [WB05] Jason Weston and Antoine Bordes. Online (and offline) on an even tighter budget. In *AISTATS*, pages 413–420, 2005.
- [WCV10] Zhuang Wang, Koby Crammer, and Slobodan Vucetic. Multi-class pegasos on a budget. In *ICML*, pages 1143–1150, 2010.
- [WJV10] Shijun Wang, Rong Jin, and Hamed Valizadegan. A potential-based framework for online multi-class learning with partial feedback. *Journal of Machine Learning Research - Proceedings Track*, 9:900–907, 2010.
- [WSZ08] Zheng Wang, Yangqiu Song, and Changshui Zhang. Transferred dimensionality reduction. In *ECML/PKDD’08*, pages 550–565, Antwerp, Belgium, 2008.
- [WV10] Zhuang Wang and Slobodan Vucetic. Online passive-aggressive algorithms on a budget. *Journal of Machine Learning Research - Proceedings Track*, 9:908–915, 2010.

REFERENCES

- [WZH12] Jialei Wang, Peilin Zhao, and Steven C. H. Hoi. Exact soft confidence-weighted learning. In *ICML*, 2012.
- [YYH07] Jun Yang, Rong Yan, and Alexander G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *ACM Multimedia*, pages 188–197, 2007.
- [ZH01] Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *NIPS*, pages 1081–1088, 2001.
- [ZHJ09] Peilin Zhao, Steven C. H. Hoi, and Rong Jin. Duol: A double updating approach for online learning. In *NIPS 22*, pages 2259–2267, 2009.
- [ZHJ11] Peilin Zhao, Steven C. H. Hoi, and Rong Jin. Double updating online learning. *Journal of Machine Learning Research*, 12:1587–1615, 2011.
- [ZHJY11] Peilin Zhao, Steven C. H. Hoi, Rong Jin, and Tianbao Yang. Online auc maximization. In *ICML*, pages 233–240, 2011.
- [Zin03] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In Tom Fawcett and Nina Mishra, editors, *Proceedings of International Conference on Machine Learning*, volume 20, 2003.