Dynamic Class Imbalance Learning for Incremental LPSVM

Lei Zhu

A thesis submitted to Unitec Institute of Technology
in fulfillment of the requirements
for the degree of Master of Computing

March, 2012

Department of Computing
Faculty of Creative Industries & Business

Primary Supervisor: Dr. Shaoning Pang
Secondary Supervisor: Dr. Aaron Chen

Abstract Linear Proximal Support Vector Machines (LPSVM), like decision trees, classic SVM, etc. are originally not equipped to handle drifting data streams that exhibit high and varying degrees of class imbalance. For online classification of data streams with imbalanced class distribution, we propose an incremental LPSVM termed DCIL-IncLPSVM that has robust learning performance under class imbalance. In doing so, we simplify a weighted LPSVM, which is computationally not renewable, as several core matrices multiplying two simple weight coefficients. When data addition and/or retirement occurs, the proposed DCIL-IncLPSVM accommodates current class imbalance by a simple matrix and coefficient updating, meanwhile ensures no discriminative information lost throughout the learning process. Experiments on benchmark datasets indicate that the proposed DCIL-IncLPSVM outperforms batch SVM and LPSVM in terms of F-measure, relative sensitivity and G-mean metrics. Moreover, our application to online face membership authentication shows that the proposed DCIL-IncLPSVM remains effective in the presence of highly dynamic class imbalance, which usually poses serious problems to classic incremental SVM (IncSVM) and incremental LPSVM (IncLPSVM).

Acknowledgments

I would like to thank all people who have helped and inspired me during my master study.

I especially want to thank my supervisors, Dr Paul Pang and Dr Aaron Chen, for their guidance during my research and study at Unitec. Their perpetual energy and enthusiasm for research have motivated all their students, including me. In addition, Dr Paul Pang and Dr Aaron Chen were always accessible and willing to help his students with their research. As a result, research life became smooth and rewarding for me.

All my lab buddies at the DMLI Lab made it a convivial place to work. In particular, I would like to thank Lei Song, Yiming Peng and Peter Zhang for their friendship and help during my thesis.

My deepest gratitude goes to my family for their unflagging love and support throughout my life; this thesis would be simply impossible without them. I am indebted to my father, Jianguo Zhu, for his care and love. As a typical father, he worked industriously to support the family and spared no effort to provide the best possible environment for me to grow up and attend school. He has never complained in spite of all the hardships in his life. I cannot ask for more from my mother, Jianying Wang, as she is simply perfect. I have no suitable word that can fully describe her everlasting love for me. I remember her constant support when I encountered difficulties and I remember, most of all, her delicious dishes.

By the end, my best greetings to Prof. Hari Seldon. May his Foundations and Plan keep the universe in peace.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Research Objectives 2

1.3 Research Contributions 3

1.4 Notation Declaration 3

1.5 Thesis Structure 4

2 Class Imbalance Impacts on SVMs and Solutions **5**

2.1 Support Vector Machine 5

2.2 Linear Proximal SVM 8

2.3 Class Imbalance Problem 11

2.4 Class Imbalance Learning for SVMs 13

2.4.1 Sampling Methods 13

2.4.2 SVM Ensemble 13

2.4.3 Weighting Methods 14

2.4.4 Weighted LPSVM 15

3 A Technical Review of Incremental SVMs **18**

3.1 Incremental SVM 18

3.2 Incremental LPSVM 22

4 Dynamic Class Imbalance Learning for LPSVM **24**

4.1 DCIL Derivation 26

4.2 Algorithm Script 28

4.3 Extension to Multiple Classes 29

5 Experiments and Discussions 32

5.1 Experimental Setup 32

5.2 Performance Measurements 33

5.3 Equality to Batch Learning 34

5.4 Static Class imbalances Robustness Tests 34

5.5 Dynamic Class imbalance Robustness Tests on Face Membership Authentication 36

5.6 Class Imbalance on Multi-class Face Recognition 40

6 Conclusions and Future Work 45

7 Publication Delivery 47

References 48

Chapter 1

Introduction

1.1 Background

Class imbalance occurs frequently in datasets from many real-world applications, such as intrusion detection, medical diagnosis, and web mining etc. (He & Chen, 2008). A great number of researches (Zhou & Liu, 2006; Huang, Yang, King & Lyu, 2006; Wu & Chang, 2005; Hong, Chen & Harris, 2007) have been conducted for investigating the impact of class imbalance on supervised machine learning, on the basis of training a classifier directly from a static set of training data (called batch learning hereafter). However in the context of data stream mining where data samples are available at different times, class imbalance learning involves not just biased class data distribution, but also concept drift, target variable changing over time in unforeseen ways. It remains as an interesting and demanding research topic for many supervised learning approaches. This paper addresses the challenge on classic Linear Proximal Support Vector Machine (LPSVM), and proposes a novel incremental LPSVM for class imbalance robust data stream mining.

Batch LPSVM (G. Fung & Mangasarian, 2001), similar to classic SVM, has a default assumption that training samples are distributed evenly among two classes. In reality, class imbalance is a common phenomenon in training data. One class known as the majority class may have many more samples than the other class, called the minority class in the paper. In the presence of imbalanced class distribution, the separating plane of LPSVM biases easily to the minority class. As a consequence the classification accuracy of the majority class overwhelms that of the minority class. A serious class imbalance may even lead to the classifier completely lose its classification capability, classifying data samples all into the majority class. wLPSVM (G. M. Fung & Mangasarian, 2005; Zhuang et al., 2005) has been proposed to solve the class imbalance problem for batch LPSVM learning, in which the contribution of two classes are balanced by weights calculated on the basis of class distribution ratio.

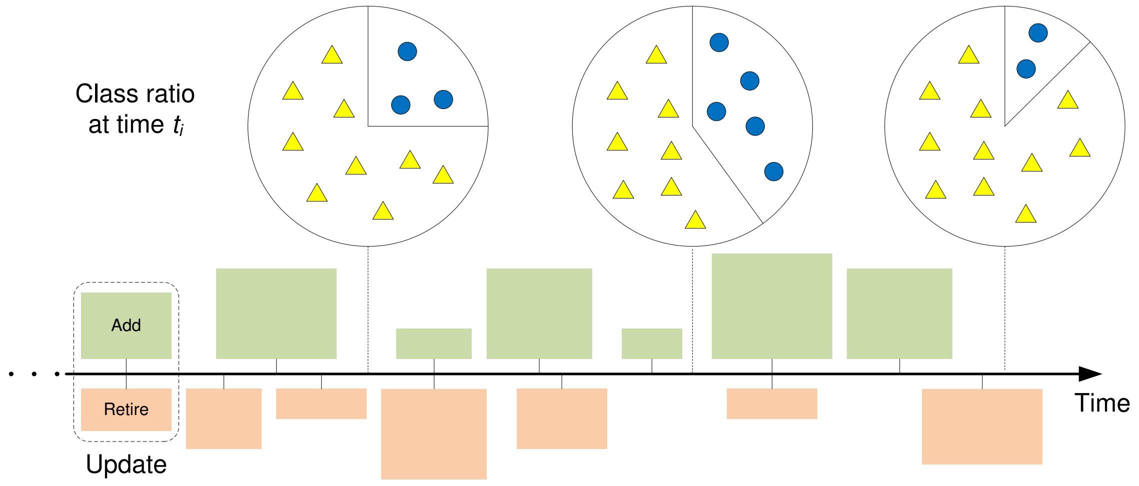


Figure 1.1: Demonstration of class imbalance variation in data stream

In data stream environment as shown in Figure 1.1, dynamic class imbalance occurs in that new data becomes available and old data becomes unwanted continuously over an indefinite (possibly infinite) lifetime, and the class distribution ratio (i.e., number of samples in class 1 / number of samples in class 2) might become high and vary at different time point. Learning from such data stream, incremental learning algorithm is desirable to pose a capability for dynamic class imbalance learning (DCIL), i.e. learning from data to adjust itself adaptively to accommodate varied class imbalances.

1.2 Research Objectives

This work aims to upgrade the wLPSVM for DCIL, the objectives including:

- develop an incremental algorithm which is able to learn continuously from data streams based on batch wLPSVM;
- the model learned from the developed incremental algorithm should be explicitly equals to a batch wLPSVM model on the same accumulated data.

1.3 Research Contributions

To upgrade the wLPSVM for DCIL, it is required to instantly update the LPSVM and its weights whenever data addition/retirement occurred. Updating weights for classic wLPSVM however is a challenge, because weights re-calculation for wLPSVM involves updating several matrix multiplications as detailed in Chapter 4. This work proposes a new incremental learning of wLPSVM for DCIL, where non-stationary imbalanced stream data mining problem is explicitly formalized as learning from data chunks of imbalanced class ratio, which are becoming available in an incremental manner. In the proposed DCIL-IncLPSVM, we derive a new expression of wLPSVM in which the weights are presented as two simple coefficients and LPSVM is represented through four core matrices. This has made the updating of weights crispy and simple.

In general, the proposed DCIL-IncLPSVM updates its weights and LPSVM simultaneously whenever a chunk of data is presented/removed. The theoretical guarantee is that the updated wLPSVM is deterministically identical to the wLPSVM obtained from batch learning. Because the DCIL-IncLPSVM adapts its weights dynamically with the current class imbalance ratio, the proposed DCIL-IncLPSVM effectively cope any dynamic class imbalance throughout the incremental learning cycle.

The contribution of this research can be summarized as follows

- a novel incremental algorithm DCIL-IncLPSVM satisfying above objectives is developed;
- the proposed binary-class DCIL-IncLPSVM is extended into multi-class scenario which improves the multi-class classification accuracy;
- the effectiveness of DCIL-IncLPSVM and its extension are verified by theoretical analysis and real-world application tests.

1.4 Notation Declaration

For the convenience of method derivation and clarity of presentation, we summarize most notations used in this thesis in Table 1.1.

Notation	Descriptions
$\boldsymbol{X}$	instance matrix, $\boldsymbol{X} \in \mathbb{R}^{n \times d}$
$\boldsymbol{Y}$	class label vector, $\boldsymbol{Y} \in \mathbb{R}^{n \times 1}$
d	dimension of instance
n	number of training samples
$\boldsymbol{x}_i$	column vector of ith instance, $\boldsymbol{x}_i^T$ is the ith row of $\boldsymbol{X}$
y_i	class label of instance $\boldsymbol{x}_i$, the ith component of $\boldsymbol{Y}$
$\boldsymbol{I}$	$(d+1) \times (d+1)$ identity matrix
C	trade off between margin maximization and training error minimization
$\boldsymbol{w}, b$	parameters in LPSVM class separating plane
$\boldsymbol{\xi}$	column vector of training errors
$\boldsymbol{O}$	normal direction of LPSVM class separating plane, $\boldsymbol{O} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} \in \mathbb{R}^{(d+1) \times 1}$
$\boldsymbol{D}$	diagonal matrix of class labels, $\boldsymbol{D} = diag(\boldsymbol{Y})$
$\boldsymbol{e}$	column vector of ones
$\boldsymbol{E}$	expanded instance matrix, $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{X} & -\boldsymbol{e} \end{bmatrix}$
$\boldsymbol{N}$	diagonal weighting matrix
$\boldsymbol{M}$	matrix term in LPSVM / DCIL-IncLPSVM model
$\boldsymbol{v}$	vector term in LPSVM / DCIL-IncLPSVM model
σ_+/σ_-	class-wise weight for positive/negative class
l_+/l_-	number of samples in positive/negative class

Table 1.1: NOTATIONS

1.5 Thesis Structure

The rest of this thesis is organized as follows: Chapter 2 reviews the SVMs as well as class imbalance impacts on SVMs and exiting solutions, including wLPSVM which gives the foundation of our work. Chapter 3 presents the technical background of this thesis through a technical review of incremental SVMs. Chapter 4 presents the proposed DCIL-IncLPSVM including proofs and derivations. Experiments and discussion are given in Chapter 5. Finally, we conclude this thesis in Chapter 6 and highlight the publications derived from this research project.

Chapter 2

Class Imbalance Impacts on SVMs and Solutions

2.1 Support Vector Machine

Let $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_1, y_i), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ be a given training dataset, where $\boldsymbol{x}_i \in \mathbb{R}^{d\times 1}$ denotes a input instance and $y_i \in \{+1, -1\}$ is the corresponding class label. We have a instance matrix $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n\end{bmatrix}^T \in \mathbb{R}^{n\times d}$ and a corresponding label vector $\boldsymbol{Y} = \begin{bmatrix}y_1 & y_2 & \cdots & y_n\end{bmatrix}^T \in \{+1, -1\}^{n\times 1}$ on $\boldsymbol{\mathcal{S}}$. A classic linear SVM (Vapnik, 1995) learns a discriminant function

$$f(x) = \boldsymbol{w}^T \boldsymbol{x} + b, \tag{2.1}$$

which separates testing samples by a class separating plane

$$\boldsymbol{x}^T \boldsymbol{w} + b = 0. \tag{2.2}$$

The separating plane (2.2) lies midway between two parallel bounding planes formulated as,

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{w} + b &= +1 \\ \boldsymbol{x}^T \boldsymbol{w} + b &= -1. \end{aligned} \tag{2.3}$$

In practice, bounding planes (2.3) bound two classes often with some non-negative errors ξ_i

$$\begin{aligned} \boldsymbol{x}_i^T\boldsymbol{w} + b + \xi_i &\geq +1 \qquad \text{for} \quad y_i = +1 \\ \boldsymbol{x}_i^T\boldsymbol{w} + b - \xi_i &\leq -1 \qquad \text{for} \quad y_i = -1. \end{aligned} \tag{2.4}$$

Here, the distance between these two planes equals to $\frac{2}{\|\boldsymbol{w}\|}$, which is called the "margin" in literature. The $\boldsymbol{w}$ and b in (2.2) and (2.3) are obtained by solving an optimization problem

$$\begin{aligned} &\min \quad \frac{\|\boldsymbol{w}\|^2}{2} + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} \quad y_i(\boldsymbol{x}_i^T\boldsymbol{w} + b) + \xi_i \geq 1 \quad \xi_i \geq 0 \quad \forall i \in \{1, \cdots, n\}, \end{aligned} \tag{2.5}$$

where $\frac{\|\boldsymbol{w}\|^2}{2}$ is for maximizing the margin, and $\sum_{i=1}^{n}\xi_i$ for minimizing the total training error. The regularization parameter C balances the importance of error and margin (Yamasaki & Ikeda, 2005).

In practice, the dual problem of (2.5) is solved to obtain an SVM classifier (Karasuyama & Takeuchi, 2010; Cauwenberghs & Poggio, 2000). The classifier obtained through solving a linear SVM (2.5) is a linear classifier of the input space, which is insufficient for some linear-inseparable problems.

To overcome the insufficiency of linear SVM in linear-inseparable tasks, a space transformation method based on kernel function Φ is introduced. Through kernel function, any sample $\boldsymbol{x}$ in the d-dimensional input space can be transformed into $\Phi(\boldsymbol{x})$ in a feature space. None-linear SVM turn to solve a linear problem in the transformed feature space to achieve a non-linear classifier in the input space.

The discriminant function for a none-linear SVM is formulated as

$$f(x) = \boldsymbol{w}^T\Phi(\boldsymbol{x}) + b, \tag{2.6}$$

where $\Phi(x)$ denotes the transformation from input space to feature space. Consequently, the optimization of none-linear SVM turns to be

$$\begin{aligned} &\min \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t.} \quad y_i f(\boldsymbol{x}_i) \geq 1 - \xi_i \quad \xi_i \geq 0 \quad \forall i \in \{1, \cdots, n\}, \end{aligned} \tag{2.7}$$

Using Lagrangian, problem (2.7) is converted into a dual problem

$$\begin{aligned} \min \quad & G = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n}\alpha_i + b\sum_{i=1}^{n} y_i\alpha_i \\ \text{s.t.} \quad & 0 \le \alpha_i \le C \quad \forall i \in \{1, \cdots, n\} \end{aligned} \tag{2.8}$$

where $K(x_i, x_j) = \Phi(x_i)^T\Phi(x_j)$ denotes a kernel function. Accordingly, the objective discriminant function (2.6) is expressed as

$$f(x) = \sum_{i=1}^{n}\alpha_i y_i K(x, x_i) + b. \tag{2.9}$$

Define a kernel matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ where $\boldsymbol{Q}_{ij} = K(x_i, x_j)y_i y_j$. The Karush-Kuhn-Tucker condition of function G (2.8) in its optimality can be written as

$$g_i = \frac{\partial G}{\partial \alpha_i} = \sum_{j=1}^{n}\boldsymbol{Q}_{ij}\alpha_j + y_i b - 1 = y_i f(x_i) - 1 \begin{cases} > 0 & \alpha_i = 0 \\ = 0 & 0 \le \alpha_i \le C \\ < 0 & \alpha_i = C \end{cases} \tag{2.10}$$

$$h = \frac{\partial G}{\partial b} = \sum_{j=1}^{n} y_j\alpha_j \equiv 0. \tag{2.11}$$

The training samples thus can be categorized into three sets according to their partial derivatives g_i. Let $\mathcal{M}$ be a set of margin vectors which are samples lying on the margin, $\mathcal{E}$ represent the set of error vectors violating the margin, and set $\mathcal{R}$ denote the reserve vectors exceeding the margin. The definition of $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{R}$ is listed as follow

$$\mathcal{M} = \{i : g_i = y_i f(x_i) - 1 = 0 \quad 0 \le \alpha_i \le C\} \tag{2.12a}$$

$$\mathcal{E} = \{i : g_i = y_i f(x_i) - 1 < 0 \quad \alpha_i = C\} \tag{2.12b}$$

$$\mathcal{R} = \{i : g_i = y_i f(x_i) - 1 > 0 \quad \alpha_i = 0\}. \tag{2.12c}$$

Considering formulation (2.9), it is easy to see that only training examples with nonzero Lagrange multipliers have influence on the discriminant function, thus these examples are defined as support vectors $\mathcal{SV}$, and $\mathcal{SV} = \mathcal{M} \cup \mathcal{E}$.

2.2 Linear Proximal SVM

LPSVM (G. Fung & Mangasarian, 2001) models a binary classification task as a regularized least square problem, which simplifies the above SVM optimization and results in an extremely efficient training algorithm. The optimization of LPSVM is given as,

$$\begin{aligned} \min \quad & \frac{C}{2}\left\|\boldsymbol{\xi}\right\|^2 + \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w} + b^2) \\ \text{s.t.} \quad & \boldsymbol{D}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{e}b) + \boldsymbol{\xi} = \boldsymbol{e}, \end{aligned} \tag{2.13}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{n\times 1}$ refers to the vector of training errors, $\boldsymbol{D} = diag(\boldsymbol{Y}) \in \mathbb{R}^{n\times n}$ denotes a diagonal matrix of class labels, and $\boldsymbol{e} = [1, \ldots, 1]^T \in \mathbb{R}^{n\times 1}$. By (2.13), the LPSVM seeks a class separating plane

$$\boldsymbol{x}^T\boldsymbol{w} - b = 0 \tag{2.14}$$

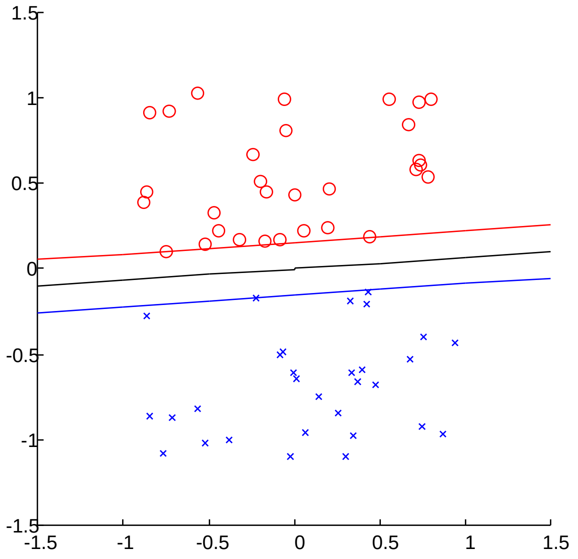
which lies midway between two parallel Proximal planes

$$\begin{aligned} \boldsymbol{x}^T\boldsymbol{w} - b &= +1 \\ \boldsymbol{x}^T\boldsymbol{w} - b &= -1. \end{aligned} \tag{2.15}$$

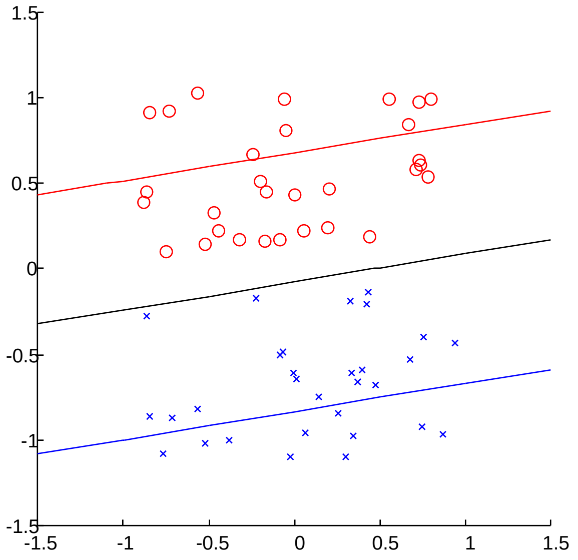
In contrast to classic SVM, LPSVM in (2.13) replaces the inequality constraint of (2.5) with an equality condition. As a result, the planes in (2.15) are not bounding planes anymore, but can be seen as "proximal" planes around which the instances of each class are clustered, as we have

$$\begin{aligned} \boldsymbol{x}_i^T\boldsymbol{w} - b + \xi_i &= +1 \qquad \text{for} \quad y_i = +1 \\ \boldsymbol{x}_i^T\boldsymbol{w} - b - \xi_i &= -1 \qquad \text{for} \quad y_i = -1, \end{aligned} \tag{2.16}$$

and it is not necessary for the error ξ_i to be nonnegative. The planes in (2.15) are pushed as far apart as possible by the term of $(\boldsymbol{w}^T\boldsymbol{w}+b^2)$ in the LPSVM optimization (2.13), and the total training error are minimized by the term of $\left\|\boldsymbol{\xi}\right\|^2$. Geometrical comparison between classic SVM and LPSVM shows that, both algorithms learn a separating plane that lies in the midway of two parallel planes which are pushed as apart as possible. Parallel planes in classic SVM case bound two classes, and in LPSVM these planes behave as "proximal" planes around which the instances of each class are clustered.



(a) SVM



(b) LPSVM

Figure 2.1: A graphical comparison of SVM and LPSVM

Figure 2.1 gives a demonstration of LPSVM and SVM separating planes obtained by learning on a two-dimensional artificial dataset. Red o and blue x denote positive and negative samples respectively. The red line and blue line represent the parallel planes and the black line in midway of these lines denotes the separating plane. It is worth noting that in Figure 2.1a the samples are bounded by the parallel planes while in Figure 2.1b the samples are clustered around the parallel planes.

For optimizing LPSVM, the equality constraint in (2.13) can be substituted into the optimization function to form an unconstrained optimization problem,

$$\min \quad G = \frac{C}{2}\left\|\boldsymbol{D}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{e}b) - \boldsymbol{e}\right\|^2 + \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w} + b^2). \tag{2.17}$$

Set the partial derivatives of G to 0, we have

$$\begin{aligned}
\frac{\partial G}{\partial \boldsymbol{w}} &= C\boldsymbol{X}^T\boldsymbol{D}(\boldsymbol{D}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{e}b) - \boldsymbol{e}) + \boldsymbol{w} \\
&= C\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{w} - C\boldsymbol{X}^T\boldsymbol{e}b - C\boldsymbol{X}^T\boldsymbol{D}\boldsymbol{e} \\
&= 0 \\
\frac{\partial G}{\partial b} &= -C\boldsymbol{e}^T\boldsymbol{D}(\boldsymbol{D}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{e}b) - \boldsymbol{e}) + b \\
&= -C\boldsymbol{e}^T\boldsymbol{X}\boldsymbol{w} + C\boldsymbol{e}^T\boldsymbol{e}b + b + C\boldsymbol{e}^T\boldsymbol{D}\boldsymbol{e} \\
&= 0,
\end{aligned} \tag{2.18}$$

where $\boldsymbol{D}\boldsymbol{D} = \boldsymbol{I}$ ($\boldsymbol{I}$ refers to an identity matrix with arbitrary size) and $\boldsymbol{e}^T\boldsymbol{e} = \|\boldsymbol{e}\|^2 = n$.

Solving the linear system in (2.18), the solution of LPSVM optimization (i.e., (2.13)) then is obtained,

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} &= \begin{bmatrix} \boldsymbol{X}^T\boldsymbol{X} + \frac{\boldsymbol{I}}{C} & -\boldsymbol{X}^T\boldsymbol{e} \\ -\boldsymbol{e}^T\boldsymbol{X} & n + \frac{1}{C} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{X}^T\boldsymbol{D}\boldsymbol{e} \\ -\boldsymbol{e}^T\boldsymbol{D}\boldsymbol{e} \end{bmatrix} \\
&= \left[\frac{\boldsymbol{I}}{C} + \begin{bmatrix} \boldsymbol{X}^T \\ -\boldsymbol{e}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{X} & -\boldsymbol{e} \end{bmatrix}\right]^{-1} \begin{bmatrix} \boldsymbol{X}^T\boldsymbol{D}\boldsymbol{e} \\ -\boldsymbol{e}^T\boldsymbol{D}\boldsymbol{e} \end{bmatrix}.
\end{aligned} \tag{2.19}$$

Let $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{X} & -\boldsymbol{e} \end{bmatrix}$ and $\boldsymbol{O} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$, (2.19) can be reformulated as

$$\boldsymbol{O} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{E})^{-1}\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{e}. \tag{2.20}$$

To keep formula simple, we use $\boldsymbol{M}$ and $\boldsymbol{v}$ to denote the matrix term $\frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{E}$ and vector term $\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{e}$ in (2.20) respectively. Then, (2.20) can be rewritten as

$$\boldsymbol{O} = \boldsymbol{M}^{-1}\boldsymbol{v}, \tag{2.21}$$

and the LPSVM decision function is obtained

$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{w} - b = \begin{bmatrix}\boldsymbol{x}^T & -1\end{bmatrix}\boldsymbol{O} \begin{cases} > 0 & \Rightarrow y = +1 \\ < 0 & \Rightarrow y = -1. \end{cases} \tag{2.22}$$

The algorithm of batch LPSVM is summarized as Algorithm 1 below.

Algorithm 1 Batch LPSVM Algorithm

```
Input: Instance matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$; Class label vector $\boldsymbol{Y} \in \mathbb{R}^{n\times 1}$; Regularization parameter $C$.
Output: $\boldsymbol{O} \in \mathbb{R}^{(d+1)\times 1}$.
 1: Form identity matrix $\boldsymbol{I} \in \mathbb{R}^{(d+1)\times(d+1)}$ and column vector $\boldsymbol{e} = [1, \ldots, 1]^T \in \mathbb{R}^{n\times 1}$;
 2: Expand the instance matrix $\boldsymbol{X}$ into $\boldsymbol{E}$, $\boldsymbol{E} = [\boldsymbol{X} \quad -\boldsymbol{e}]$;
 3: Transform the class label vector $\boldsymbol{Y}$ into diagonal matrix $\boldsymbol{D}$, $\boldsymbol{D} = diag(\boldsymbol{Y})$;
 4: Compute $\boldsymbol{M} = \frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{E}$ and $\boldsymbol{v} = \boldsymbol{E}^T\boldsymbol{D}\boldsymbol{e}$;
 5: Compute $\boldsymbol{O} = \boldsymbol{M}^{-1}\boldsymbol{v}$.
```

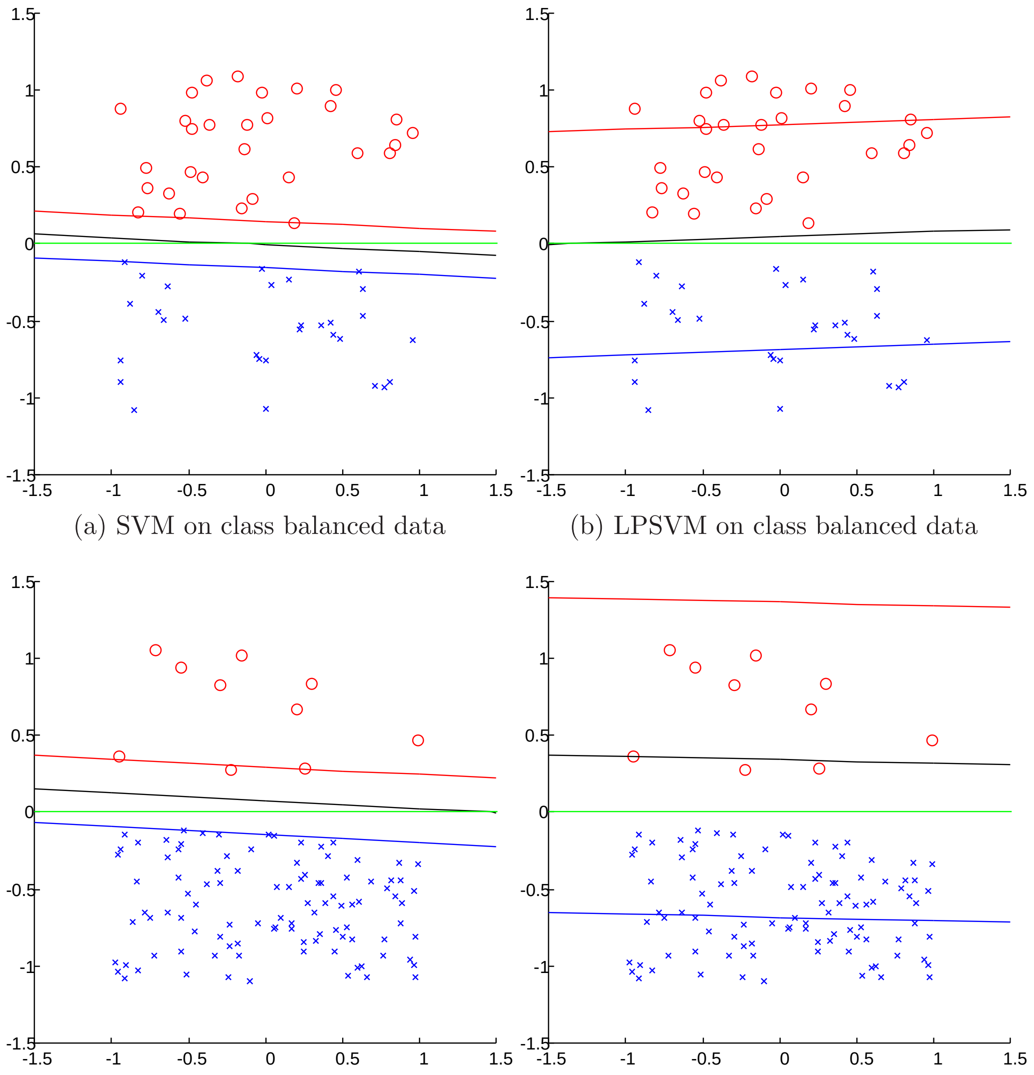
2.3 Class Imbalance Problem

The class imbalance problem arises when samples of the class of interest are relatively rare as compared to the other class. Specifically, for LPSVM the total training error $\|\boldsymbol{\xi}\|^2$ in (2.13) comes from two classes, the error thus can be represented as $\|\boldsymbol{\xi}\|^2 = \|\boldsymbol{\xi}_+\|^2 + \|\boldsymbol{\xi}_-\|^2$. In the presence of class imbalance, the LPSVM optimization (2.13) has $\|\boldsymbol{\xi}_+\|^2 << \|\boldsymbol{\xi}_-\|^2$. As a result, LPSVM shifts its positive class proximal plane away from the separating plane, which enlarges the margin at the price of an augmented $\|\boldsymbol{\xi}_+\|^2$. Consequently, the separating plane biases to the positive class resulting in the worse recognition of the positive class.

On the other hand, as the parallel planes for SVM behave as bounding planes of samples, the total training error of SVM only comes from samples exceeding the parallel planes (i.e., positive/negative samples located in the side of positive/negative plane which towards the separating plane). Thus when learning from dataset with the same level of class imbalance, the imbalance force from total train error for SVM is much smaller than that for LPSVM. Consequently, the impact of class imbalance on SVM is relatively smaller than that on LPSVM.

We now demonstrate the impact of class imbalance on SVM and LPSVM by a simple test on artificial data. Certain number of two-dimensional (i.e., $X1$ and $X2$) data points subject to even distribution are randomly generated in the area of $X1 \in [0.1, 1.1]$ $X2 \in [-1, 1]$ (which are labeled as positive class) and $X1 \in [-1.1, -0.1]$

$X2 \in [-1, 1]$ (which are labeled as negative class) to create class balanced and class imbalanced dataset respectively. Since the data distribution is known in this test, the ideal class separating plane $X1 = 0$ is a priori identified. The learning outcome of SVM and LPSVM are displayed in Figure 2.2.



(a) SVM on class balanced data

(b) LPSVM on class balanced data

(c) SVM on class imbalanced data

(d) LPSVM on class imbalanced data

Figure 2.2: Learning outcome demonstration

As we can see, in the case of class balanced dataset (i.e., Figure 2.2a and Figure 2.2b in which 30 positive to 30 negative samples) both SVM and LPSVM result in separating planes that approximate closely to the ideal plane (i.e., the green line in the graphs). When there is obvious class imbalance, the separating plane of both

SVMs shift towards the minority class and the shift of LPSVM is much more serious in comparison with that of SVM. The above theoretical assumption that LPSVM suffers more from the class imbalance problem thus is testified.

2.4 Class Imbalance Learning for SVMs

In literature, it has been well studied that SVMs can be jammed by high incidences of false negatives when the training samples of the target class are heavily out-numbered by the training samples of a non-target class. The decision boundary of SVM can be skewed towards the minority (positive) class, which causes the generation of a high number of false-negative predictions, and lowers the performance on the positive class (Akbani, Kwek & Japkowicz, 2004)(Wu & Chang, 2005). Existing class imbalance learning methods for reducing the effect of data imbalance can be summarized into three categories: sampling methods, weighting, and SVM ensemble.

2.4.1 Sampling Methods

The use of sampling methods for class imbalance learning considers the modification of an imbalanced data set by some mechanisms in order to provide a balanced distribution (He & Garcia, 2009). A variety of sampling methods have been developed so far. Undersampling/oversampling/re-sampling has the majority/miniority class examples randomly removed/duplicated respectively until a particular class distribution ratio is met (Weiss, 2004),(Liu, Wu & Zhou, 2006)(Liu, Wu & Zhou, 2009). The synthetic minority oversampling technique (SMOTE) (Chawla, Bowyer, Hall & Kegelmeyer, 2002) is a powerful method that has shown a great deal of success in SVM related applications (Wang, 2008)(Gouripeddi et al., 2009)(Maciejewski & Stefanowski, 2011). In general, sampling methods are applicable to various type of SVM algorithms being used (Batuwita & Palade, 2010) for data modeling, thus they are potential choices for SVM class imbalance learning, but they are not able to completely solve the class imbalance problem.

2.4.2 SVM Ensemble

Ensemble is seen as a promising method with the ability to improve the generalization performance of classification algorithms. For solving class imbalance, ensembles of classifiers consist of a set of individually trained classifiers whose predictions are combined to classify unknown samples (Zhai, Yang, Ma & Ruan, 2010). Partitioning ensemble is used in (Yan, Liu, Jin & Hauptmann, 2003; Pang, Kim & Bang, 2003; Pang, Kim & Sung-yang, 2005) and (Dong & Han, 2005) for learning from class imbalanced datasets. Assume the class distribution ratio is $1 : r$ in the data set. The majority class is then divided into r disjunct partitions. With each of these r partitions, the complete minority class is combined to create r balanced training sets. r SVMs are trained based on these training set. The classification outputs of these SVMs are combined by majority voting.

2.4.3 Weighting Methods

Instead of creating balanced data distributions through different sampling strategies, weighting methods tackle the imbalanced learning problem by using different weights to balance the contribution of the minority and majority classes. The weight can be considered as a numerical cost of classifying examples from one class to another. Typically, there is no cost for correct classification of either class and the cost of misclassifying minority examples is higher than the contrary case.

The weighting method can be embedded into the SVM objective function. A modified objective function assigns weights as two misclassification cost values ξ^+ and ξ^- to each class respectively as

$$\begin{aligned} &\min \quad \frac{\|\boldsymbol{w}\|^2}{2} + C(\sigma_+ \sum_{i|y_i=+1}^{n} \xi_i + \sigma_- \sum_{i|y_i=-1}^{n} \xi_i) \\ &\text{s.t.} \quad y_i(\boldsymbol{x}_i^T \boldsymbol{w} + b) + \xi_i \geq 1 \quad \xi_i \geq 0 \quad \forall i \in \{1, \cdots, n\}, \end{aligned} \tag{2.23}$$

has σ_+ and σ_- corresponding to the class distribution ratio. By assigning a higher misclassification cost for the positive (minority) class examples than the negative (majority) class examples, the effect of class imbalance could be reduced. Example methods include Different Error Costs (DEC) (Veropoulos, Campbell & Cristianini, 1999) and fuzzy SVM for Class Imbalance Learning (FSVM-CIL) (Batuwita & Palade, 2010). An alternative weighting method can be applied directly to the SVM decision function as,

$$f(x) = \text{sign}(\sigma_+ \sum_{i|y_i=+1}^{n} \alpha_i y_i K(x_i, x) + \sigma_- \sum_{j|y_j=-1}^{n} \alpha_j y_j K(x_j, x)). \tag{2.24}$$

where σ_+ and σ_- are weights on support vectors. To adjust the decision boundary's bias towards the minority (positive), higher weights are utilized for the positive support vectors (SVs) (Imam, Ting & Kamruzzaman, 2006).

2.4.4 Weighted LPSVM

Weighted LPSVM (wLPSVM) (G. M. Fung & Mangasarian, 2005; Zhuang et al., 2005) is a solution to the class imbalance problem for LPSVM. It can be seen as a example of the weighting method. As the foundation of our work, wLPSVM is introduced in this subsection.

As we mentioned before, the class imbalance problem for LPSVM is mainly caused by the imbalance force of the two-class training error. The idea of wLPSVM is straightforward: balancing the training error from two classes to obtain a separating plane learning that is robust to the class imbalance. This training error balancing is achieved by assigning smaller weight to the training errors from the majority class while assigning larger weight to that of the minority class. In wLPSVM the classic LPSVM optimization (2.13) is extended to,

$$\begin{array}{ll} \min & \frac{C}{2}\boldsymbol{\xi}^T \boldsymbol{N}\boldsymbol{\xi} + \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w} + b^2) \\ \text{s.t.} & \boldsymbol{D}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{e}b) + \boldsymbol{\xi} = \boldsymbol{e}, \end{array} \tag{2.25}$$

where $\boldsymbol{N}$ is a diagonal weighting matrix with,

$$\boldsymbol{N}_{ii} = \begin{cases} \sigma_+ & \text{if } y_i = +1 \\ \sigma_- & \text{if } y_i = -1, \end{cases} \tag{2.26}$$

in which the class-wise weight σ is used, σ_+ for the positive class and σ_- for the negative class, to balance the impacts of two classes to the LPSVM separating plane. In practice, σ is determined by the number of samples for each class. For example in (Tao & Ji, 2007), σ_+ and σ_- are calculated as the ratio of the corresponding class size (i.e., l_- or l_+) to the size of the whole dataset,

$$\begin{aligned} \sigma_+ &= l_-/(l_+ + l_-) \\ \sigma_- &= l_+/(l_+ + l_-). \end{aligned} \tag{2.27}$$

By a similar approach as (2.17)-(2.20), (2.25) can be solved and the wLPSVM solution is obtained as,

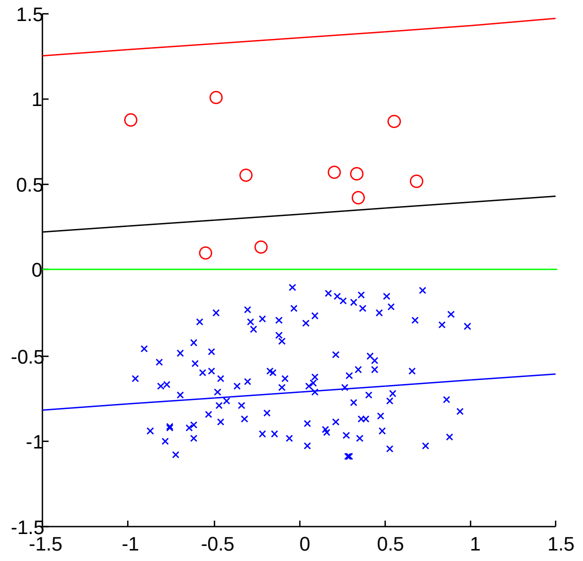
$$\boldsymbol{O} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E})^{-1}\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e}. \tag{2.28}$$

Algorithm 2 presents the batch wLPSVM algorithm that implements (2.28).

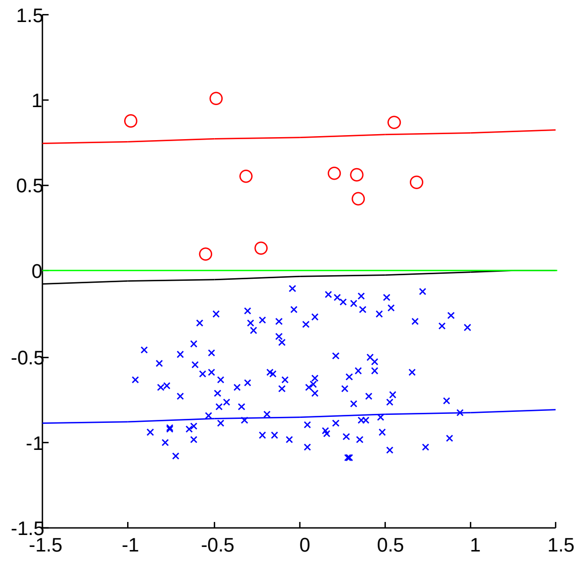
Algorithm 2 Batch wLPSVM Algorithm

```
Input: Instance matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$; Class label vector $\boldsymbol{Y} \in \mathbb{R}^{n\times 1}$; Regularization parameter $C$.
Output: $\boldsymbol{O} \in \mathbb{R}^{(d+1)\times 1}$.
1: Calculate weights $\sigma_+$ and $\sigma_-$ following weighting function such as (2.27);
2: Form weight matrix $\boldsymbol{N}$ following (2.26);
3: Expand the instance matrix $\boldsymbol{X}$ into $\boldsymbol{E}$, $\boldsymbol{E} = [\boldsymbol{X} \quad -\boldsymbol{e}]$;
4: Transform the class label vector $\boldsymbol{Y}$ into diagonal matrix $\boldsymbol{D}$, $\boldsymbol{D} = diag(\boldsymbol{Y})$;
5: Compute $\boldsymbol{M} = \frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E}$ and $\boldsymbol{v} = \boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e}$;
6: Compute $\boldsymbol{O} = \boldsymbol{M}^{-1}\boldsymbol{v}$.
```

Figure 2.3 gives a demonstration of wLPSVM effectiveness, the test setting are similar to that in Section 2.3. As we can see, on a class imbalanced dataset, as the force of two-class training error is balanced, the wLPSVM proximal plane for the minority class is “drawn” back to the center of the minority class cluster. Consequently, the separating plane learned by wLPSVM is closer to the ideal plane than that of LPSVM.



(a) LPSVM



(b) wLPSVM

Figure 2.3: A graphical comparison of LPSVM and wLPSVM on a class imbalanced dataset

Chapter 3

A Technical Review of Incremental SVMs

For learning from continuously coming data, incremental learning algorithms are developed. For the comprehensive and completeness of this thesis, we introduce incremental SVM in section 3.1, however, the incremental LPSVM introduced in section 3.2 is more relevant to our work.

3.1 Incremental SVM

The objective of incremental SVM (Cauwenberghs & Poggio, 2000; Yamasaki & Ikeda, 2005; Karasuyama & Takeuchi, 2010) is to incorporate m new training examples $\{(\boldsymbol{x}_i, y_i) : i \in \{n+1, \cdots, n+m\}\}$ into an initial solution $\{(\boldsymbol{x}_i, y_i), \alpha_i, b : i \in \{1, \cdots, n\}\}$ and obtain a updated solution $\{(\boldsymbol{x}_i, y_i), \alpha_i^*, b^* : i \in \{1, \cdots, n+m\}\}$.

Define index set $\mathcal{U}$ for unlearned examples, Lagrange vector $\boldsymbol{\alpha}$, label vector $\boldsymbol{y}$ and gradient vector $\boldsymbol{g}$ respectively as

$$\mathcal{U} = \{i : i \in \{n+1, \cdots, n+m\}\} \tag{3.1}$$

$$\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_{n+m}]^T \tag{3.2}$$

$$\boldsymbol{y} = [y_1, \cdots, y_{n+m}]^T \tag{3.3}$$

$$\boldsymbol{g} = [g_1, \cdots, g_{n+m}]^T \tag{3.4}$$

Initially, we can set $\alpha_i = 0, \forall i \in \mathcal{U}$, and we initialize $\boldsymbol{g}$ according to (2.10).

Indexes $\{i : g_i > 0, i \in \mathcal{U}\}$ and $\{i : g_i = 0, i \in \mathcal{U}\}$ are directly removed from $\mathcal{U}$ and appended to rest set $\mathcal{R}$ and margin set $\mathcal{M}$ (the definition of $\mathcal{R}$ and $\mathcal{M}$ is given in Section 2.1) respectively, because they already satisfy the KKT condition of the initial solution. By now, examples remaining in $\mathcal{U}$ are all KKT condition violators which will go through adiabatic increment procedure and eventually turn into margin or error examples.

In what follows, index set(s) are used as subscriptions in some expressions. For example, $\boldsymbol{y}_{\mathcal{M}}$ denotes a sub-vector of $\boldsymbol{y}$ and the elements of $\boldsymbol{y}_{\mathcal{U}}$ are indexed by set $\mathcal{M}$. Similarly, two index sets are used as subscription of matrix, such as $\boldsymbol{Q}_{\mathcal{M},\mathcal{U}}$, which refer to a sub-matrix of $\boldsymbol{Q}$ with rows indexed by $\mathcal{M}$ and columns indexed by $\mathcal{U}$.

In adiabatic increments, Lagrangian coefficients of unlearned samples $(\alpha_i, i \in \mathcal{U})$ are increased from 0 to their optimal value until every unlearned examples become margin samples or error samples, while keeping the optimal condition of all learned examples $(\mathcal{L} = \mathcal{M} \cup \mathcal{E} \cup \mathcal{R})$ satisfied. This optimalility maintaining is obtained by adjusting the Lagrangian coefficients of margin samples accordingly. Adiabatic increment is an iterative process. In each iteration there is at least one example that has its identity changed (e.g. i moves from $\mathcal{U}$ to $\mathcal{M}$). Thus, this incremental learning can be seen as a series of adiabatic steps.

Perturbation parameter p and the amount of perturbation in each adiabatic increment $\Delta p*$ are defined to control the incremental learning process, and we have $p = \sum \Delta p*$. When $p = 0$, the solution is initialized as the original solution and knowledge from new data are not included. After a series of adiabatic increments and eventually when $p = 1$, the solution covers all examples and the learning terminates which means both old and new data satisfy the KKT condition indicated in (2.10) and (2.11).

Define coefficient sensitivities to represent the ratio of change in α and b in response to Δp, set

$$\beta_0 = \frac{\Delta b}{\Delta p} \tag{3.5}$$

$$\beta_k = \frac{\Delta \alpha_k}{\Delta p} \quad \forall k \in \mathcal{M} \tag{3.6}$$

$$\lambda_l = \frac{\Delta \alpha_l}{\Delta p} \quad \forall l \in \mathcal{U} \tag{3.7}$$

Set $\lambda_l = C \quad : \quad \forall l \in \mathcal{U}$, in this way, the Lagrangian coefficients of unlearned vectors will change at most by C when p becomes 1. It guarantees that the remaining unlearned vectors can be moved into error vectors.

Before a adiabatic increment, the following constraints for margin vectors are satisfied

$$g_i = \sum_j Q_{ij}\alpha_j + y_i b - 1 = 0 \quad \forall i \in \mathcal{M} \tag{3.8}$$

$$h = \sum_j y_j \alpha_j = 0. \tag{3.9}$$

After a perturbation, the constraints become

$$g_i = \sum_j Q_{ij}\alpha_j + \sum_{k\in\mathcal{M}} Q_{ik}\Delta\alpha_k + \sum_{l\in\mathcal{U}} Q_{il}\Delta\alpha_l + y_i(b+\Delta b) - 1 = 0 \quad \forall i \in \mathcal{M} \tag{3.10}$$

$$h = \sum_j y_j\alpha_j + \sum_{k\in\mathcal{M}} y_k\Delta\alpha_k + \sum_{l\in\mathcal{U}} y_l\Delta\alpha_l = 0. \tag{3.11}$$

Combine constraints (3.8) to (3.11), the following conditions are obtained:

$$\Delta g_i = \sum_{k\in\mathcal{M}} Q_{ik}\Delta\alpha_k + \sum_{l\in\mathcal{U}} Q_{il}\Delta\alpha_l + y_i\Delta b = 0 \quad \forall i \in \mathcal{M} \tag{3.12}$$

$$\Delta h = \sum_{k\in\mathcal{M}} y_k\Delta\alpha_k + \sum_{l\in\mathcal{U}} y_l\Delta\alpha_l = 0 \tag{3.13}$$

Substituting coefficient sensitivities into (3.12) and (3.13), we obtain:

$$\Delta g_i = \sum_{k\in\mathcal{M}} Q_{ik}\beta_k\Delta p + \sum_{l\in\mathcal{U}} Q_{il}\lambda_l\Delta p + y_i\beta_0\Delta p = 0 \quad \forall i \in \mathcal{M} \tag{3.14}$$

$$\Delta h = \sum_{k\in\mathcal{M}} y_k\beta_k\Delta p + \sum_{l\in\mathcal{U}} y_l\lambda_l\Delta p = 0 \tag{3.15}$$

Clearly we have:

$$\frac{\Delta g_i}{\Delta p} = \sum_{k \in \mathcal{M}} Q_{ik}\beta_k + \sum_{l \in \mathcal{U}} Q_{il}\lambda_l + y_i\beta_0 = 0 \quad \forall i \in \mathcal{M} \tag{3.16}$$

$$\frac{\Delta h}{\Delta p} = \sum_{k \in \mathcal{M}} y_k\beta_k + \sum_{l \in \mathcal{U}} y_l\lambda_l = 0 \tag{3.17}$$

The coefficient sensitivities of margin vectors $\{\beta_k : \forall k \in \mathcal{M}\}$ and β_0 can be obtained by solving the linear system represented by (3.16) and (3.17). Using matrix notation, (3.16) and (3.17) can be written as:

$$\begin{bmatrix} 0 & y_{\mathcal{M}}^T \\ y_{\mathcal{M}} & Q_{\mathcal{M},\mathcal{M}} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_{\mathcal{M}} \end{bmatrix} + \begin{bmatrix} y_{\mathcal{U}}^T \\ Q_{\mathcal{M},\mathcal{U}} \end{bmatrix} \lambda_{\mathcal{U}} = 0 \tag{3.18}$$

Note that $\lambda_{\mathcal{U}}$ is a $|\mathcal{U}| \times 1$ sized column vector. Define expanded margin kernel matrix $\begin{bmatrix} 0 & y_{\mathcal{M}}^T \\ y_{\mathcal{M}} & Q_{\mathcal{M},\mathcal{M}} \end{bmatrix}$ as M, thus we have:

$$\begin{bmatrix} \beta_0 \\ \beta_{\mathcal{M}} \end{bmatrix} = -M^T \begin{bmatrix} y_{\mathcal{U}}^T \\ Q_{\mathcal{M},\mathcal{U}} \end{bmatrix} \lambda_{\mathcal{U}} \tag{3.19}$$

Define margin sensitivity γ represents the ratio of gradient change according to Δp, we have:

$$\gamma_i = \frac{\Delta g_i}{\Delta p} = \sum_{k \in \mathcal{M}} Q_{ik}\beta_k + \sum_{l \in \mathcal{U}} Q_{il}\lambda_l + y_i\beta_0 \tag{3.20}$$

And for any $i \in \mathcal{M}$, $\gamma_i = 0$, according to (3.16). As $\{\beta_k : \forall k \in \mathcal{M}\}$ and β_0 have been solved, γ_i for all examples are known. By now, we can compute the $\Delta p*$ for perturbation.

To determine $\Delta p*$ in each adiabatic increment, $\Delta p*$ must: (1) cause at least one example touch the borders between $\mathcal{M}, \mathcal{E}, \mathcal{R}, \mathcal{U}$ and changes its identity, and (2) cause none example crosses the borders. As indicated in (3.5), (3.6) and (3.7), α and g varies linearly according to Δp. Thus we can compute the maximum Δp allowed for each border which not cause border crossing. Border conditions and corresponding

Δp_{max} are listed in Table 3.1.

Original Identity	New Identity	Condition	Δp_{max}
$\mathcal{M}$ $(g_i = 0, \alpha_i \in [0, C])$	$\mathcal{E}$ $(g_i < 0, \alpha_i = C)$	$\alpha_i + \Delta\alpha_i = C$	$\frac{C-\alpha_i}{\beta_i}$
$\mathcal{M}$ $(g_i = 0, \alpha_i \in [0, C])$	$\mathcal{R}$ $(g_i > 0, \alpha_i = 0)$	$\alpha_i + \Delta\alpha_i = 0$	$\frac{-\alpha_i}{\beta_i}$
$\mathcal{E}$ $(g_i < 0, \alpha_i = C)$	$\mathcal{M}$ $(g_i = 0, \alpha_i \in [0, C])$	$g_i + \Delta g_i = 0$	$\frac{-g_i}{\gamma_i}$
$\mathcal{R}$ $(g_i > 0, \alpha_i = 0)$	$\mathcal{M}$ $(g_i = 0, \alpha_i \in [0, C])$	$g_i + \Delta g_i = 0$	$\frac{-g_i}{\gamma_i}$
$\mathcal{U}$ $(g_i < 0, \alpha_i \in [0, C))$	$\mathcal{M}$ $(g_i = 0, \alpha_i \in [0, C])$	$g_i + \Delta g_i = 0$	$\frac{-g_i}{\gamma_i}$
$\mathcal{U}$ $(g_i < 0, \alpha_i \in [0, C))$	$\mathcal{E}$ $(g_i < 0, \alpha_i = C)$	$\alpha_i + \Delta\alpha_i = C$	$\frac{C-\alpha_i}{\lambda_i}$

Table 3.1: Borders

We compute Δp_{max} for each example according to its original identity, and define these Δp_{max} as a set ΔP. The size of ΔP is $2 \times |\mathcal{M}| + |\mathcal{E}| + |\mathcal{R}| + 2 \times |\mathcal{U}|$. The $\Delta p*$ used for perturbation is determined by $\Delta p* = \min \Delta P$, because if we chose any $\Delta p* > \min \Delta P$, there will be example(s) crosses the border and violate the KKT condition.

Once the $\Delta p*$ is computed, we update b, α and g as follow:

$$b \leftarrow b + \Delta b = b + \beta_0 \Delta p* \tag{3.21}$$

$$\alpha_k \leftarrow \alpha_k + \Delta\alpha_k = \alpha_k + \beta_k \Delta p* \quad \forall k \in \mathcal{M} \tag{3.22}$$

$$\alpha_l \leftarrow \alpha_l + \Delta\alpha_l = \alpha_l + \lambda_l \Delta p* \quad \forall l \in \mathcal{U} \tag{3.23}$$

$$g_i \leftarrow g_i + \Delta g_i = g_i + \gamma_i \Delta p* \quad \forall i \in \mathcal{M} \cup \mathcal{E} \cup \mathcal{R} \cup \mathcal{U} \cup \tag{3.24}$$

and update the index sets $\mathcal{M}, \mathcal{E}, \mathcal{R}$ and $\mathcal{U}$ to reflect the example identity change caused by perturbation $\Delta p*$.

After that, we recompute the sensitivities β and γ to conduct a new cycle of adiabatic increments. This iterative procedures continues until $p = 1$ or unlearned set $\mathcal{U}$ becomes empty.

3.2 Incremental LPSVM

The incremental learning of LPSVM is to update existing LPSVM model by retiring old data and/or adding new data simultaneously. Assume that the current LPSVM model on $\boldsymbol{\mathcal{S}}$ is obtained from (2.20), $\boldsymbol{\mathcal{S}_r}$ is an "old" subset of $\boldsymbol{\mathcal{S}}$ that needs to be retired, and $\boldsymbol{\mathcal{S}_a}$ a "new" set of data that needs to be added.

Fung et al. (G. M. Fung & Mangasarian, 2001) proposed an incremental LPSVM model (IncLPSVM) (3.25) to handle the retirement of $\boldsymbol{\mathcal{S}_r}$ and the addition of $\boldsymbol{\mathcal{S}_a}$,

$$\begin{aligned} \boldsymbol{O}' &= (\frac{\boldsymbol{I}}{C} + \boldsymbol{E}^T\boldsymbol{E} - \boldsymbol{E_r}^T\boldsymbol{E_r} + \boldsymbol{E_a}^T\boldsymbol{E_a})^{-1}(\boldsymbol{E}^T\boldsymbol{De} - \boldsymbol{E_r}^T\boldsymbol{D_r e} + \boldsymbol{E_a}^T\boldsymbol{D_a e}) \\ &= (\boldsymbol{M} - \boldsymbol{E_r}^T\boldsymbol{E_r} + \boldsymbol{E_a}^T\boldsymbol{E_a})^{-1}(\boldsymbol{v} - \boldsymbol{E_r}^T\boldsymbol{D_r e} + \boldsymbol{E_a}^T\boldsymbol{D_a e}), \end{aligned} \tag{3.25}$$

where $\boldsymbol{E_r} \in \mathbb{R}^{n_r\times(d+1)}$, $\boldsymbol{D_r} \in \mathbb{R}^{n_r\times n_r}$, $\boldsymbol{E_a} \in \mathbb{R}^{n_a\times(d+1)}$ and $\boldsymbol{D_a} \in \mathbb{R}^{n_a\times n_a}$. This method can update accurately a batch LPSVM, however it considers no dynamic class imbalance of data streams.

For clarity, the steps of incremental LPSVM is stated in Algorithm 3.

Algorithm 3 Incremental LPSVM Algorithm

```
Input: $\boldsymbol{M}$ and $\boldsymbol{v}$ from last stage or batch learning, $\boldsymbol{X_r}$ and $\boldsymbol{Y_r}$ to retire, $\boldsymbol{X_a}$ and $\boldsymbol{Y_a}$ to add.
Output: $\boldsymbol{O}$ for the classifier, $\boldsymbol{M}$ and $\boldsymbol{v}$ for next learning stage.
1: Generate $\boldsymbol{E_r}$, $\boldsymbol{D_r}$, $\boldsymbol{E_a}$ and $\boldsymbol{D_a}$ according to $\boldsymbol{X_r}$, $\boldsymbol{Y_r}$, $\boldsymbol{X_a}$ and $\boldsymbol{Y_a}$ respectively;
2: Update $\boldsymbol{M}$ term as $\boldsymbol{M} = \boldsymbol{M} - \boldsymbol{E_r}^T\boldsymbol{E_r} + \boldsymbol{E_a}^T\boldsymbol{E_a}$, and update $\boldsymbol{v}$ term as $\boldsymbol{v} = \boldsymbol{v} - \boldsymbol{E_r}^T\boldsymbol{D_r e} + \boldsymbol{E_a}^T\boldsymbol{D_a e}$;
3: Compute $\boldsymbol{O} = \boldsymbol{M}^{-1}\boldsymbol{v}$.
```

If we compare SVM and LPSVM, due to their difference in optimization constraint (i.e., inequality constraint for SVM and equality for LPSVM), there is a significant difference in the difficulty of optimization solving, either for batch learning and incremental learning.

For classic SVM, since it is an inequality constrained optimization, we can only seek its solution by perturbations while keeping KKT condition satisfied. In another words, to include more samples satisfying the condition and keep the satisfaction of all learned samples to the condition until all seen samples are included in the solution. This is a iterative process and the number of steps is normally unknown in prior.

For LPSVM, as the equality constraint can be substituted into the objective function, the solution can be directly calculated (for batch learning) and updated (for incremental learning) by simple formulas. This make the LPSVM and IncLPSVM extremely computational efficient thus has considerable potential for high speed data stream learning.

Chapter 4

Dynamic Class Imbalance Learning for LPSVM

For dynamic class imbalance learning (DCIL), incremental wLPSVM (DCIL-IncLPSVM) based on the batch wLPSVM described above is proposed in this section. Since class imbalance varies dynamically and continuously with the presence of new data samples, the idea of the proposed DCIL-IncLPSVM for DCIL is to update the LPSVM model in the meantime its weights.

For the convenience of method derivations, we introduce first two lemmas on basic matrix decomposition:

Lemma 1 *Let* $\begin{bmatrix} \boldsymbol{X} & \boldsymbol{Y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_a & \boldsymbol{Y}_a \\ \boldsymbol{X}_b & \boldsymbol{Y}_b \end{bmatrix}$, $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{E}_a \\ \boldsymbol{E}_b \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}_a & -\boldsymbol{e} \\ \boldsymbol{X}_b & -\boldsymbol{e} \end{bmatrix}$, $\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}_a & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_b \end{bmatrix}$ *and* $\boldsymbol{N} = \begin{bmatrix} \boldsymbol{N}_a & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{N}_b \end{bmatrix}$. *Then,*

$$\begin{aligned} \boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E} &= \boldsymbol{E}_a^T\boldsymbol{N}_a\boldsymbol{E}_a + \boldsymbol{E}_b^T\boldsymbol{N}_b\boldsymbol{E}_b \\ \boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e} &= \boldsymbol{E}_a^T\boldsymbol{D}_a\boldsymbol{N}_a\boldsymbol{e} + \boldsymbol{E}_b^T\boldsymbol{D}_b\boldsymbol{N}_b\boldsymbol{e} \\ \boldsymbol{E}^T\boldsymbol{E} &= \boldsymbol{E}_a^T\boldsymbol{E}_a + \boldsymbol{E}_b^T\boldsymbol{E}_b \\ \boldsymbol{E}^T\boldsymbol{e} &= \boldsymbol{E}_a^T\boldsymbol{e} + \boldsymbol{E}_b^T\boldsymbol{e}. \end{aligned} \tag{4.1}$$

Proof:

$$\begin{aligned}
\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E} &= \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix}^T \begin{bmatrix}\boldsymbol{N}_a & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{N}_b\end{bmatrix} \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix} = \begin{bmatrix}\boldsymbol{E}_a^T \boldsymbol{E}_b^T\end{bmatrix} \begin{bmatrix}\boldsymbol{N}_a & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{N}_b\end{bmatrix} \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix}\\
&= \boldsymbol{E}_a^T\boldsymbol{N}_a\boldsymbol{E}_a + \boldsymbol{E}_b^T\boldsymbol{N}_b\boldsymbol{E}_b\\
\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e} &= \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix}^T \begin{bmatrix}\boldsymbol{D}_a & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{D}_b\end{bmatrix} \begin{bmatrix}\boldsymbol{N}_a & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{N}_b\end{bmatrix} \begin{bmatrix}\boldsymbol{e}\\ \boldsymbol{e}\end{bmatrix} = \begin{bmatrix}\boldsymbol{E}_a^T & \boldsymbol{E}_b^T\end{bmatrix} \begin{bmatrix}\boldsymbol{D}_a & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{D}_b\end{bmatrix} \begin{bmatrix}\boldsymbol{N}_a & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{N}_b\end{bmatrix} \begin{bmatrix}\boldsymbol{e}\\ \boldsymbol{e}\end{bmatrix}\\
&= \boldsymbol{E}_a^T\boldsymbol{D}_a\boldsymbol{N}_a\boldsymbol{e} + \boldsymbol{E}_b^T\boldsymbol{D}_b\boldsymbol{N}_b\boldsymbol{e}\\
\boldsymbol{E}^T\boldsymbol{E} &= \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix}^T \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix} = \begin{bmatrix}\boldsymbol{E}_a^T & \boldsymbol{E}_b^T\end{bmatrix} \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix} = \boldsymbol{E}_a^T\boldsymbol{E}_a + \boldsymbol{E}_b^T\boldsymbol{E}_b\\
\boldsymbol{E}^T\boldsymbol{e} &= \begin{bmatrix}\boldsymbol{E}_a\\ \boldsymbol{E}_b\end{bmatrix}^T \begin{bmatrix}\boldsymbol{e}\\ \boldsymbol{e}\end{bmatrix} = \begin{bmatrix}\boldsymbol{E}_a^T & \boldsymbol{E}_b^T\end{bmatrix} \begin{bmatrix}\boldsymbol{e}\\ \boldsymbol{e}\end{bmatrix} = \boldsymbol{E}_a^T\boldsymbol{e} + \boldsymbol{E}_b^T\boldsymbol{e}.
\end{aligned} \tag{4.2}$$

□

Lemma 2 *Let $\boldsymbol{\mathcal{S}}$ be a dataset with n samples, i.e., $\boldsymbol{\mathcal{S}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots, (\boldsymbol{x}_n, y_n)\}$. $\eta = \begin{bmatrix}1 & 2 & \cdots & i & \cdots & j & \cdots & n\end{bmatrix}$ identifies a sequence of data samples. Applying η to the dataset, we form instance matrix $\boldsymbol{X}^{\boldsymbol{\eta}} = \begin{bmatrix}\boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_i & \cdots & \boldsymbol{x}_j & \cdots & \boldsymbol{x}_n\end{bmatrix}^T$ and label vector $\boldsymbol{Y}^{\boldsymbol{\eta}} = \begin{bmatrix}y_1 & y_2 & \cdots & y_i & \cdots & y_j & \cdots & y_n\end{bmatrix}^T$, and corresponding matrices $\boldsymbol{E}^{\boldsymbol{\eta}}$, $\boldsymbol{D}^{\eta}$ and $\boldsymbol{N}^{\boldsymbol{\eta}}$. Given another sample sequence $\eta' = \begin{bmatrix}1 & 2 & \cdots & j & \cdots & i & \cdots & n\end{bmatrix}$ and corresponding $\boldsymbol{E}^{\boldsymbol{\eta}'}$, $\boldsymbol{D}^{\boldsymbol{\eta}'}$ and $\boldsymbol{N}^{\boldsymbol{\eta}'}$. Then,*

$$\begin{aligned}
&\boldsymbol{E}^{\boldsymbol{\eta}T}\boldsymbol{N}^{\boldsymbol{\eta}}\boldsymbol{E}^{\boldsymbol{\eta}} = \boldsymbol{E}^{\boldsymbol{\eta}'^T}\boldsymbol{N}^{\boldsymbol{\eta}'}\boldsymbol{E}^{\boldsymbol{\eta}'}\\
&\boldsymbol{E}^{\boldsymbol{\eta}T}\boldsymbol{D}^{\boldsymbol{\eta}}\boldsymbol{N}^{\boldsymbol{\eta}}\boldsymbol{e} = \boldsymbol{E}^{\boldsymbol{\eta}'^T}\boldsymbol{D}^{\boldsymbol{\eta}'}\boldsymbol{N}^{\boldsymbol{\eta}'}\boldsymbol{e}\\
&\boldsymbol{E}^{\boldsymbol{\eta}T}\boldsymbol{E}^{\boldsymbol{\eta}} = \boldsymbol{E}^{\boldsymbol{\eta}'^T}\boldsymbol{E}^{\boldsymbol{\eta}'}\\
&\boldsymbol{E}^{\boldsymbol{\eta}T}\boldsymbol{e} = \boldsymbol{E}^{\boldsymbol{\eta}'^T}\boldsymbol{e}
\end{aligned} \tag{4.3}$$

Proof: Let $\boldsymbol{E}_i = \begin{bmatrix}\boldsymbol{x}_i^T & -1\end{bmatrix}$, $\boldsymbol{N}_i = \begin{cases}\sigma_+ & \text{if } y_i = +1\\ \sigma_- & \text{if } y_i = -1\end{cases}$ and $\boldsymbol{D}_i = y_i$ be the terms corresponding to the i-th sample, apply Lemma 1 repeatedly, then

$$\begin{aligned}
\boldsymbol{E}^{\eta T}\boldsymbol{N}^{\eta}\boldsymbol{E}^{\eta} &= \sum_{i=1}^{n}\boldsymbol{E}_i^T\boldsymbol{N}_i\boldsymbol{E}_i = \boldsymbol{E}^{\eta' T}\boldsymbol{N}^{\eta'}\boldsymbol{E}^{\eta'} \\
\boldsymbol{E}^{\eta T}\boldsymbol{D}^{\eta}\boldsymbol{N}^{\eta}\boldsymbol{e} &= \sum_{i=1}^{n}\boldsymbol{E}_i^T\boldsymbol{D}_i\boldsymbol{N}_i = \boldsymbol{E}^{\eta' T}\boldsymbol{D}^{\eta'}\boldsymbol{N}^{\eta'}\boldsymbol{e} \\
\boldsymbol{E}^{\eta T}\boldsymbol{E}^{\eta} &= \sum_{i=1}^{n}\boldsymbol{E}_i^T\boldsymbol{E}_i = \boldsymbol{E}^{\eta' T}\boldsymbol{E}^{\eta'} \\
\boldsymbol{E}^{\eta T}\boldsymbol{e} &= \sum_{i=1}^{n}\boldsymbol{E}_i^T = \boldsymbol{E}^{\eta' T}\boldsymbol{e}
\end{aligned} \tag{4.4}$$

□

Lemma 2 implies that the equations in (4.3) hold with respect to arbitrary sample sequences.

4.1 DCIL Derivation

Given an initial wLPSVM model (2.28) on dataset $\boldsymbol{\mathcal{S}}$. If $\boldsymbol{\mathcal{S}_l}$ denotes the set of data that remains after retiring $\boldsymbol{\mathcal{S}_r}$, and $\boldsymbol{\mathcal{S}'}$ denotes the updated dataset after the addition and retirement, then we have

$$\begin{aligned}
\boldsymbol{\mathcal{S}} &= \boldsymbol{\mathcal{S}}_l \cup \boldsymbol{\mathcal{S}}_r \\
\boldsymbol{\mathcal{S}'} &= \boldsymbol{\mathcal{S}}_l \cup \boldsymbol{\mathcal{S}}_a.
\end{aligned} \tag{4.5}$$

Then, the incremental learning of wLPSVM is to compute a updated wLPSVM model $\boldsymbol{O'}$ based on $\boldsymbol{\mathcal{S}}_a$, $\boldsymbol{\mathcal{S}}_r$, and a trained wLPSVM model (i.e., (2.28)) on the current dataset $\boldsymbol{\mathcal{S}}$.

By (2.28), a batch wLPSVM on the updated dataset $\boldsymbol{\mathcal{S}'}$ can be written as,

$$\boldsymbol{O'} = (\frac{\boldsymbol{I}}{C} + \boldsymbol{E'}^T\boldsymbol{N'}\boldsymbol{E'})^{-1}\boldsymbol{E'}^T\boldsymbol{D'}\boldsymbol{N'}\boldsymbol{e}. \tag{4.6}$$

In (4.6), the weighting matrix $\boldsymbol{N}$ can be simply updated by,

$$\boldsymbol{N'}_{ii} = \begin{cases} \sigma'_+ = l'_-/(l'_+ + l'_-) & \text{if } y'_i = +1 \\ \sigma'_- = l'_+/(l'_+ + l'_-) & \text{if } y'_i = -1, \end{cases} \tag{4.7}$$

in which $l'_- = l_- - l_{r-} + l_{a-}$ and $l'_+ = l_+ - l_{r+} + l_{a+}$. However, its problematic to update the matrix multiplications $\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E}$ and $\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e}$. To overcome this difficulty, we propose to simplify (4.6) by Proposition 1, transforming the weighting matrix $\boldsymbol{N}$ to two simple weight coefficients σ_+ and σ_-.

Proposition 1 *Given a wLPSVM model (2.28) over dataset* $\boldsymbol{S}$*, and let* $\boldsymbol{M}_+ = \boldsymbol{E}_+^T\boldsymbol{E}_+$*,* $\boldsymbol{M}_- = \boldsymbol{E}_-^T\boldsymbol{E}_-$*,* $\boldsymbol{v}_+ = \boldsymbol{E}_+^T\boldsymbol{e}$ *and* $\boldsymbol{v}_- = \boldsymbol{E}_-^T\boldsymbol{e}$*, then the wLPSVM model (2.28) can be reformulated as,*

$$\boldsymbol{O} = (\frac{\boldsymbol{I}}{C} + \sigma_+\boldsymbol{M}_+ + \sigma_-\boldsymbol{M}_-)^{-1}(\sigma_+\boldsymbol{v}_+ - \sigma_-\boldsymbol{v}_-). \tag{4.8}$$

Proof:

As $\boldsymbol{\mathcal{S}}$ is a 2-class dataset decomposable into $\boldsymbol{\mathcal{S}}_+$ and $\boldsymbol{\mathcal{S}}_-$, we apply Lemma 2 and Lemma 1 to the term $\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E}$ in (2.28), and have

$$\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E} = \boldsymbol{E}_+^T\boldsymbol{N}_+\boldsymbol{E}_+ + \boldsymbol{E}_-^T\boldsymbol{N}_-\boldsymbol{E}_-. \tag{4.9}$$

As $\boldsymbol{N}_+ = \sigma_+\boldsymbol{I}$ and $\boldsymbol{N}_- = \sigma_-\boldsymbol{I}$, term $\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E}$ can be further written as,

$$\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E} = \sigma_+\boldsymbol{E}_+^T\boldsymbol{E}_+ + \sigma_-\boldsymbol{E}_-^T\boldsymbol{E}_-. \tag{4.10}$$

Similarly, applying Lemma 2 and Lemma 1 to the term $\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e}$ in (2.28), we have

$$\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e} = \boldsymbol{E}_+^T\boldsymbol{D}_+\boldsymbol{N}_+\boldsymbol{e} + \boldsymbol{E}_-^T\boldsymbol{D}_-\boldsymbol{N}_-\boldsymbol{e}. \tag{4.11}$$

As $\boldsymbol{D}_+ = \boldsymbol{I}$ and $\boldsymbol{D}_- = -\boldsymbol{I}$, also $\boldsymbol{N}_+ = \sigma_+\boldsymbol{I}$ and $\boldsymbol{N}_- = \sigma_-\boldsymbol{I}$, we have

$$\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e} = \sigma_+\boldsymbol{E}_+^T\boldsymbol{e} - \sigma_-\boldsymbol{E}_-^T\boldsymbol{e}. \tag{4.12}$$

Substituting (4.10) and (4.12) into (2.28), and replacing $\boldsymbol{E}_+^T\boldsymbol{E}_+$, $\boldsymbol{E}_-^T\boldsymbol{E}_-$, $\boldsymbol{E}_+^T\boldsymbol{e}$ and $\boldsymbol{E}_-^T\boldsymbol{e}$ with $\boldsymbol{M}_+$, $\boldsymbol{M}_-$, $\boldsymbol{v}_+$ and $\boldsymbol{v}_-$ respectively, we obtain

$$\boldsymbol{O} = (\frac{\boldsymbol{I}}{C} + \sigma_+\boldsymbol{M}_+ + \sigma_-\boldsymbol{M}_-)^{-1}(\sigma_+\boldsymbol{v}_+ - \sigma_-\boldsymbol{v}_-). \tag{4.13}$$

□

Applying Proposition 1 to (4.6), we rewrite the updated wLPSVM model (4.6) as,

$$\boldsymbol{O}' = (\frac{\boldsymbol{I}}{C} + \sigma'_+\boldsymbol{M}'_+ + \sigma'_-\boldsymbol{M}'_-)^{-1}(\sigma'_+\boldsymbol{v}'_+ - \sigma'_-\boldsymbol{v}'_-). \tag{4.14}$$

When data addition and/or retirement is incurred, σ' in (4.14) can be recalculated by (4.7). So, we consider only the updating of $\boldsymbol{M}_+$, $\boldsymbol{M}_-$, $\boldsymbol{v}_+$ and $\boldsymbol{v}_-$.

As $\boldsymbol{\mathcal{S}}_+$ by (4.5) is decomposable into $\boldsymbol{\mathcal{S}}_{l+}$ and $\boldsymbol{\mathcal{S}}_{r+}$, and so for $\boldsymbol{\mathcal{S}}'_+$ decomposable into $\boldsymbol{\mathcal{S}}_{l+}$ and $\boldsymbol{\mathcal{S}}_{a+}$, we can apply Lemma 2 and Lemma 1 to decompose $\boldsymbol{M}_+$ and $\boldsymbol{M}'_+$, and have

$$\boldsymbol{M}_+ = \boldsymbol{E}_+^T\boldsymbol{E}_+ = \boldsymbol{E}_{l+}^T\boldsymbol{E}_{l+} + \boldsymbol{E}_{r+}^T\boldsymbol{E}_{r+} \tag{4.15}$$

$$\boldsymbol{M}'_+ = \boldsymbol{E}_+^{'T}\boldsymbol{E}'_+ = \boldsymbol{E}_{l+}^T\boldsymbol{E}_{l+} + \boldsymbol{E}_{a+}^T\boldsymbol{E}_{a+} \tag{4.16}$$

(4.16) minus (4.15), we obtain the updating of $\boldsymbol{M}_+$ as,

$$\boldsymbol{M}'_+ = \boldsymbol{M}_+ - \boldsymbol{E}_{r+}^T\boldsymbol{E}_{r+} + \boldsymbol{E}_{a+}^T\boldsymbol{E}_{a+}. \tag{4.17}$$

By an analogous process, we can update $\boldsymbol{M}_-$,$\boldsymbol{v}_+$ and $\boldsymbol{v}_-$ respectively as,

$$\begin{aligned} \boldsymbol{M}'_- &= \boldsymbol{M}_- - \boldsymbol{E}_{r-}^T\boldsymbol{E}_{r-} + \boldsymbol{E}_{a-}^T\boldsymbol{E}_{a-} \\ \boldsymbol{v}'_+ &= \boldsymbol{v}_+ - \boldsymbol{E}_{r+}^T\boldsymbol{e} + \boldsymbol{E}_{a+}^T\boldsymbol{e} \\ \boldsymbol{v}'_- &= \boldsymbol{v}_- - \boldsymbol{E}_{r-}^T\boldsymbol{e} + \boldsymbol{E}_{a-}^T\boldsymbol{e}. \end{aligned} \tag{4.18}$$

4.2 Algorithm Script

For DCIL of data streams, the matrix multiplication terms in (2.28) are divided into a set of class-wise matrices in (4.13) whose updating for incremental learning is straightforward. Most importantly, current class imbalance level is modeled here as two simple coefficients, and utilized in every step of LPSVM updating. This empowers the incremental LPSVM with the capability for DCIL. The pseudo-code of the proposed DCIL-IncLPSVM is given in Algorithm 4.

Algorithm 4 The proposed DCIL-IncLPSVM Algorithm

```
Input: initial DCIL-IncLPSVM model:
    $\{\boldsymbol{O}, \boldsymbol{M}_+, \boldsymbol{M}_-, \boldsymbol{v}_+, \boldsymbol{v}_-, l_+, l_-, C\}$; $\boldsymbol{X}_r$ and $\boldsymbol{Y}_r$ to retire;
    $\boldsymbol{X}_a$ and $\boldsymbol{Y}_a$ to add; and regularization parameter $C$.
Output: updated DCIL-IncLPSVM model:
    $\{\boldsymbol{O}', \boldsymbol{M}'_+, \boldsymbol{M}'_-, \boldsymbol{v}'_+, \boldsymbol{v}'_-, l'_+, l'_-, C\}$.
 1: if input model is empty then
 2:     Form $\boldsymbol{E}_{a+}$ and $\boldsymbol{E}_{a-}$ according to $\boldsymbol{X_a}$ and $\boldsymbol{Y_a}$
 3:     Compute $\boldsymbol{M_+} = \boldsymbol{E}_{a+}^T\boldsymbol{E}_{a+}$, $\boldsymbol{M}_- = \boldsymbol{E}_{a-}^T\boldsymbol{E}_{a-}$, $\boldsymbol{v}_+ = \boldsymbol{E}_{a+}^T\boldsymbol{e}$ and $\boldsymbol{v}_- = \boldsymbol{E}_{a-}^T\boldsymbol{e}$;
 4:     Compute $\sigma_+$ and $\sigma_-$ according to (2.27);
 5:     Compute $\boldsymbol{O} = (\frac{\boldsymbol{I}}{C} + \sigma_+\boldsymbol{M_+} + \sigma_-\boldsymbol{M}_-)^{-1}(\sigma_+\boldsymbol{v}_+ - \sigma_-\boldsymbol{v}_-)$ according to (4.8);
 6: else
 7:     Generate $\boldsymbol{E}_{a+}$, $\boldsymbol{E}_{a-}$, $\boldsymbol{E}_{r+}$ and $\boldsymbol{E}_{r-}$ according to $\boldsymbol{X_a}$, $\boldsymbol{Y_a}$, $\boldsymbol{X_r}$ and $\boldsymbol{Y_r}$;
 8:     Compute $\boldsymbol{M}'_+$,$\boldsymbol{M}'_-$, $\boldsymbol{v}'_+$, $\boldsymbol{v}'_-$ according to (4.17) and (4.18)
 9:     Compute $l'_+ = l_+ - l_{r+} + l_{a+}$ and $l'_- = l_- - l_{r-} + l_{a-}$;
10:     Compute $\sigma'_+$ and $\sigma'_-$;according to (4.7);
11:     Compute $\boldsymbol{O}' = (\frac{\boldsymbol{I}}{C} + \sigma'_+\boldsymbol{M}'_+ + \sigma'_-\boldsymbol{M}'_-)^{-1}(\sigma'_+\boldsymbol{v}'_+ - \sigma'_-\boldsymbol{v}'_-)$ according to (4.6).
12: end if
```

4.3 Extension to Multiple Classes

Assume we have k-class streaming data, the previously studied multi-category incremental LPSVM (McIncLPSVM) (Tveit & Hetl, 2003) learns and updates k individual binary IncLPSVMs in a one-against-rest manner without balancing training errors from two classes. The classification of a test sample is conducted by comparing the its decision values on each class. However, those binary one-against-rest IncLPSVMs are all built on a 1-to-$(k-1)$ partition data which poses normally a serious class imbalance, therefore each IncLPSVM suffers from the class imbalance problem and the overall accuracy of McIncLPSVM is affected consequently. To mitigate this difficulty, we extend in the following the above proposed DCIL-IncLPSVM for multi-class classification.

Let $\boldsymbol{\mathcal{S}}$ be a k-class initial training dataset including n samples in which the classes are labeled as $1, 2, \ldots, k$, $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{Y} \in \{1, 2, \ldots, k\}^{n\times 1}$ as its instance matrix and corresponding label vector, respectively. To achieve multi-class classification on $\boldsymbol{\mathcal{S}}$, for any class $j \in \{1, 2, \ldots, k\}$ we use an individual binary DCIL-IncLPSVM to separate the class j from the rest as

$$\boldsymbol{O}_{cj} = (\frac{\boldsymbol{I}}{C} + \sigma_+\boldsymbol{M}_+ + \sigma_-\boldsymbol{M}_-)^{-1}(\sigma_+\boldsymbol{v}_+ - \sigma_-\boldsymbol{v}_-), \tag{4.19}$$

in which samples from class j are taken as positive and the rest samples are taken as negative. Thus, we have

$$\begin{aligned} \boldsymbol{M}_+ &= \boldsymbol{E}_{cj}^T\boldsymbol{E}_{cj} = \boldsymbol{M}_{cj} \\ \boldsymbol{v}_+ &= \boldsymbol{E}_{cj}^T\boldsymbol{e} = \boldsymbol{v}_{cj} \end{aligned} \tag{4.20}$$

and

$$\begin{aligned} \sigma_+ &= (l - l_{cj})/l \\ \sigma_- &= l_{cj}/l. \end{aligned} \tag{4.21}$$

Applying Lemma 2 and Lemma 1 to decompose term $\boldsymbol{M}_-$ and $\boldsymbol{v}_-$ in (4.19), we obtain

$$\begin{aligned} \boldsymbol{M}_- &= \sum_{i=1}^{j-1}\boldsymbol{E}_{ci}^T\boldsymbol{E}_{ci} + \sum_{i=j+1}^{k}\boldsymbol{E}_{ci}^T\boldsymbol{E}_{ci} = \boldsymbol{E}^T\boldsymbol{E} - \boldsymbol{E}_{cj}^T\boldsymbol{E}_{cj} = \boldsymbol{M} - \boldsymbol{M}_{cj} \\ \boldsymbol{v}_- &= \sum_{i=1}^{j-1}\boldsymbol{E}_{ci}^T\boldsymbol{e} + \sum_{i=j+1}^{k}\boldsymbol{E}_{ci}^T\boldsymbol{e} = \boldsymbol{E}^T\boldsymbol{e} - \boldsymbol{E}_{cj}^T\boldsymbol{e} = \boldsymbol{v} - \boldsymbol{v}_{cj}. \end{aligned} \tag{4.22}$$

Substituting (4.20), (4.21) and (4.22) into (4.19), we have the DCIL-IncLPSVM for class j as,

$$\boldsymbol{O}_{cj} = (\frac{\boldsymbol{I}}{C} + \frac{l_{cj}}{l}\boldsymbol{M} + \frac{l - 2l_{cj}}{l}\boldsymbol{M}_{cj})^{-1}(\boldsymbol{v}_{cj} - \frac{l_{cj}}{l}\boldsymbol{v}). \tag{4.23}$$

Given dataset $\boldsymbol{\mathcal{S}}_r$ to be retried and $\boldsymbol{\mathcal{S}}_a$ to be added. The above DCIL-IncLPSVM for incremental learning is supposed to be updated for:
1) any existing class (i.e., $\forall j \in \{1, 2, \dots, k\}$), we update terms in (4.23) as

$$\begin{aligned} \boldsymbol{M}' &= \boldsymbol{M} - \boldsymbol{E}_r^T\boldsymbol{E}_r + \boldsymbol{E}_a^T\boldsymbol{E}_a \\ \boldsymbol{M}'_{cj} &= \boldsymbol{M}_{cj} - \boldsymbol{E}_{cjr}^T\boldsymbol{E}_{cjr} + \boldsymbol{E}_{cja}^T\boldsymbol{E}_{cja} \\ \boldsymbol{v}' &= \boldsymbol{v} - \boldsymbol{E}_r^T\boldsymbol{e} + \boldsymbol{E}_a^T\boldsymbol{e} \\ \boldsymbol{v}'_{cj} &= \boldsymbol{v}_{cj} - \boldsymbol{E}_{cjr}^T\boldsymbol{e} + \boldsymbol{E}_{cja}^T\boldsymbol{e} \\ l' &= l - l_r + l_a \\ l'_{cj} &= l_{cj} - l_{cjr} + l_{cja}, \end{aligned} \tag{4.24}$$

and obtain the updated model as

$$\boldsymbol{O}'_{cj} = (\frac{\boldsymbol{I}}{C} + \frac{l'_{cj}}{l'}\boldsymbol{M}' + \frac{l' - 2l'_{cj}}{l'}\boldsymbol{M}'_{cj})^{-1}(\boldsymbol{v}'_{cj} - \frac{l'_{cj}}{l'}\boldsymbol{v}'), \tag{4.25}$$

separating the class r from the rest.
2) any new class (i.e., $\forall j > k$), we construct a new model $\boldsymbol{O}'_{cj}$ to distinguish the class from the rest as

$$\boldsymbol{O}'_{cj} = (\frac{\boldsymbol{I}}{C} + \frac{l_{cj}}{l}\boldsymbol{M}' + \frac{l' - 2l_{cj}}{l'}\boldsymbol{M}_{cj})^{-1}(\boldsymbol{v}_{cj} - \frac{l_{cj}}{l'}\boldsymbol{v}'), \tag{4.26}$$

where $\boldsymbol{M}_{cj} = \boldsymbol{E}_{cja}^T\boldsymbol{E}_{cja}$ and $\boldsymbol{v}_{cj} = \boldsymbol{E}_{cja}^T\boldsymbol{e}$.

In summary, the pseudo-code of the proposed DCIL-IncLPSVM for multi-class classification is given in Algorithm 5.

Algorithm 5 The proposed multi-class DCIL-IncLPSVM

```
Input: multi-class DCIL-IncLPSVM model, $\boldsymbol{X}_r$ and $\boldsymbol{Y}_r$ to retire, $\boldsymbol{X}_a$ and $\boldsymbol{Y}_a$ to add and regularization parameter $C$.
Output: multi-class DCIL-IncLPSVM model $\{\boldsymbol{O}_{cj}, \boldsymbol{M}_{cj}, \boldsymbol{v}_{cj}, l_{cj}, \boldsymbol{M}, \boldsymbol{v}, l, C, \boldsymbol{I}\}$ or $\{\boldsymbol{O}'_{cj}, \boldsymbol{M}'_{cj}, \boldsymbol{v}'_{cj}, l'_{cj}, \boldsymbol{M}', \boldsymbol{v}', l', C, \boldsymbol{I}\}$.
 1: if input model is empty then
 2:     Generate $\boldsymbol{I}$, $\boldsymbol{E}$ and $l$ according to $\boldsymbol{X}_a$ and $\boldsymbol{Y}_a$;
 3:     Compute $\boldsymbol{M} = \boldsymbol{E}^T\boldsymbol{E}$ and $\boldsymbol{v} = \boldsymbol{E}^T\boldsymbol{e}$;
 4:     for $j = 1$ to $k$ do
 5:         Generate $\boldsymbol{E}_{cj}$ and $l_{cj}$ according to $\boldsymbol{X}_a$ and $\boldsymbol{Y}_a$
 6:         Compute $\boldsymbol{M}_{cj}$ and $\boldsymbol{v}_{cj}$ according to (4.20);
 7:         Compute $\boldsymbol{O}_{cj}$ according to (4.23);
 8:     end for
 9: else
10:     Generate $\boldsymbol{E}_a$, $\boldsymbol{E}_r$, $l_a$ and $l_r$ according to $\boldsymbol{X}_a$, $\boldsymbol{Y}_a$, $\boldsymbol{X}_r$ and $\boldsymbol{Y}_r$;
11:     Update $\boldsymbol{M}$, $\boldsymbol{v}$ and $l$ according to (4.24);
12:     for $j = 1$ to $k$ do
13:         Generate $\boldsymbol{E}_{cja}$, $\boldsymbol{E}_{cjr}$, $l_{cja}$ and $l_{cjr}$ according to $\boldsymbol{X}_a$, $\boldsymbol{Y}_a$, $\boldsymbol{X}_r$ and $\boldsymbol{Y}_r$;
14:         Update $\boldsymbol{M}_{cj}$, $\boldsymbol{v}_{cj}$ and $l_{cj}$ according to (4.24);
15:         Compute $\boldsymbol{O}'_{cj}$ according to (4.25);
16:     end for
17:     for all $j > k$ do
18:         Generate $\boldsymbol{E}_{cja}$ and $l_{cj}$ according to $\boldsymbol{X}_a$ and $\boldsymbol{Y}_a$;
19:         Compute $\boldsymbol{M}_{cj} = \boldsymbol{E}_{cja}^T\boldsymbol{E}_{cja}$ and $\boldsymbol{v}_{cj} = \boldsymbol{E}_{cja}^T\boldsymbol{e}$;
20:         Compute $\boldsymbol{O}'_{cj}$ according to (4.26);
21:     end for
22: end if
```

Chapter 5

Experiments and Discussions

In this chapter, the effectiveness of our algorithm in learning from imbalanced data is testified. We compared the performance of the proposed DCIL-IncLPSVM with other benchmark linear classifiers (i.e., classic SVM with linear kernel and classic LPSVM) on two scenarios: a) static datasets with various degrees of class imbalance and, b) data streams with dynamic class imbalances.

5.1 Experimental Setup

For the purposes of investigating the impact of different levels of class imbalance on the training of the classifier and verifying the ability of proposed DCIL-IncLPSVM to overcome it, samples from original benchmark datasets are grouped and selected artificially to form binary-class experimental datasets with predefined class ratio. All experiments were implemented on a Intel 2.26GHZ Core I5 PC with 4GB Ram.

For incremental learning on each experimental dataset, we first construct initial models using 10% of the data, and divide the remaining 90% training instances equally and arbitrarily into nine subsets, which are presented to the learners sequentially. As the incremental learning proceeds, the learners are fed with subsets of new data in a sequential manner until all data from the dataset are consumed.

With the above experimental setup, class imbalance may occur in two cases: 1) the class distribution of the whole dataset is imbalanced, and the class distribution of data subsets is also imbalanced; and 2) the class distribution of the whole dataset is balanced, but subsets are class imbalanced.

5.2 Performance Measurements

For evaluating the algorithm performance on imbalanced datasets, the normal metric of the overall accuracy is no longer sufficient (He & Garcia, 2009; Chen & He, 2009). Here several performance metrics based on the confusion matrix is utilized to give a more comprehensive and objective assessment. Specifically, for a binary classifier the confusion matrix is consisted of True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN). In our experiments, “positive” refers to the minority class whereas “negative” is taken as the majority class.

In this paper, we evaluate classification performance through five metrics: sensitivity, specificity, F-measure, relative sensitivity (RS) and G-mean. Sensitivity and specificity measure the accuracy on positive and negative samples respectively

$$Sensitivity = \frac{TP}{TP + FN}$$
$$Specificity = \frac{TN}{TN + FP}.$$

Normally, high sensitivity and specificity implies high classification performance. F-measure is defined as

$$F\text{-}measure = \frac{(1+\beta)^2 \times Recall \times Precision}{\beta^2 \times Recall + Precision},$$

where

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$

and β is a coefficient that balances the relative importance of precision and recall. Following common practice, we assign β to 1. RS determines a binary classifier's capability of achieving compatible prediction accuracy on both classes (Su & Hsiao, 2007) and is defined as

$$RS = \frac{Sensitivity}{Specificity}.$$

The bias of the classifier diminishes as RS approaches to 1. G-mean is another metric widely adopted for evaluating classifiers on imbalanced dataset, which is defined as

$$G\text{-}mean = \sqrt{Sensitivity \times Specificity}.$$

5.3 Equality to Batch Learning

To testify the equivalence between the discriminant vector $\boldsymbol{O}$ extracted from the proposed DCIL-IncLPSVM and that from the ground truth wLPSVM, we measure the similarity between the retrieved discriminant vectors on the Breast Wisconsin and Iris dataset (Frank & Asuncion, 2010) at the final steps of the aforementioned 10 chuck incremental learning. The discriminant vector similarity is measured in terms of Euclidean distance and angular deviation. The result is shown in Figure 5.1.

Table 5.1: Similarity of Discriminant Vectors

	Breast Wisconsin	Iris 1 to 2,3	Iris 2 to 1,3	Iris 3 to 1,2
Euclidean Distance	0	7.63×10^{-27}	2.10×10^{-27}	1.04×10^{-26}
Angular Deviation	1.20×10^{-6}	0	0	0

As we can see, the proposed DCIL-IncLPSVM implements incremental learning with significant accuracy, as the obtained discriminant vector is very similar to that of a batch wLPSVM. As mentioned in Chapter 4 that DCIL-IncLPSVM learning output is theoretically equivalent to batch wLPSVM, this rare difference comes from numerical computational errors.

5.4 Static Class imbalances Robustness Tests

In this section, we compare the performance of the final classifier obtained at the end of a complete incremental learning process.

Table 5.2: Datasets Description

Dataset	Class Min. / Maj.	#Var.	Exp.	Imbalance Ratio #Maj./#Min.	Training Set		Testing Set	
					#Pos.	#Neg.	#Pos.	#Neg.
Breast Wisconsin	Abnormal / Normal	9	1	1	20	20	40	40
			2	5	100	20		
			3	10	200	20		
			4	20	400	20		
Car	Very Good / Remainder	6	1	1	20	20	40	40
			2	5	100	20		
			3	20	400	20		
			4	80	1600	20		
Abalone	Less than 5 rings/ Remainder	8	1	1	20	20	40	40
			2	5	100	20		
			3	40	800	20		
			4	200	4000	20		

Three publicly known UCI datasets are utilized to conduct the comparison, Table 5.2 shows the characteristics of these datasets. For each dataset, experiments (i.e., Exp. in Table 5.2) using training set with different degrees of class imbalance are designed. Each experiment consists of 50 rounds of independent tests. In each test, training samples are randomly selected and sequentially presented to the IncSVM (Cauwenberghs & Poggio, 2000)(Karasuyama & Takeuchi, 2010), IncLPSVM and our DCIL-IncLPSVM algorithms for training a classifier. In our experiment, training samples are fed in chucks of random sizes. Testing sets with the same size are used for classifier performance evaluation. Table 5.3 gives the experimental results on Breast-Wisconsin, Car and Abalone datasets, respectively.

Table 5.3: Results of static class imbalance robustness tests.

Dataset	Exp.	Algorithm	Sensitivity	Specificity	F-measure	RS	G-mean
Breast Wisconsin	1	IncSVM	93.80 ± 5.67	97.20 ± 2.35	**95.35** ± 3.09	**0.97** ± 0.07	**95.43** ± 2.95
		IncLPSVM	88.45 ± 5.53	97.95 ± 1.94	92.78 ± 3.28	0.90 ± 0.06	93.03 ± 3.07
		DCIL-IncLPSVM	88.45 ± 5.53	97.95 ± 1.94	92.78 ± 3.28	0.90 ± 0.06	93.03 ± 3.07
	2	SVM	85.00 ± 8.25	98.25 ± 2.04	90.82 ± 5.00	0.87 ± 0.09	91.27 ± 4.55
		LPSVM	87.20 ± 5.93	98.85 ± 1.61	92.49 ± 3.64	0.88 ± 0.06	92.79 ± 3.35
		DCIL-IncLPSVM	88.90 ± 5.44	98.30 ± 2.05	**93.21** ± 3.29	**0.90** ± 0.06	**93.43** ± 3.08
	3	IncSVM	81.30 ± 7.86	97.95 ± 2.35	88.49 ± 4.80	0.83 ± 0.09	89.12 ± 4.29
		IncLPSVM	84.65 ± 6.19	98.15 ± 2.13	90.67 ± 3.54	0.86 ± 0.07	91.07 ± 3.22
		DCIL-IncLPSVM	90.00 ± 5.08	97.80 ± 2.46	**93.59** ± 2.86	**0.92** ± 0.06	**93.77** ± 2.69
	4	IncSVM	72.15 ± 10.20	98.85 ± 2.10	82.87 ± 7.32	0.73 ± 0.10	84.24 ± 6.27
		IncLPSVM	76.85 ± 6.72	98.65 ± 1.90	86.10 ± 4.50	0.78 ± 0.07	86.98 ± 3.97
		DCIL-IncLPSVM	89.70 ± 5.95	97.65 ± 2.74	**93.33** ± 3.70	**0.92** ± 0.06	**93.53** ± 3.50
Car	1	IncSVM	97.70 ± 3.46	87.15 ± 6.60	**92.86** ± 3.39	**1.13** ± 0.10	**92.19** ± 3.88
		IncLPSVM	99.15 ± 2.40	80.95 ± 6.66	90.95 ± 2.88	1.23 ± 0.12	89.50 ± 3.77
		DCIL-IncLPSVM	99.15 ± 2.40	80.95 ± 6.66	90.95 ± 2.88	1.23 ± 0.12	89.50 ± 3.77
	2	IncSVM	85.85 ± 7.15	95.85 ± 3.45	90.24 ± 4.37	**0.90** ± 0.08	90.61 ± 4.04
		IncLPSVM	62.55 ± 13.63	98.05 ± 2.73	75.16 ± 10.88	0.64 ± 0.14	77.77 ± 8.89
		DCIL-IncLPSVM	99.95 ± 0.35	83.00 ± 5.95	**92.21** ± 2.54	1.21 ± 0.09	**91.02** ± 3.28
	3	IncSVM	60.75 ± 7.39	99.15 ± 1.39	74.93 ± 5.62	0.61 ± 0.08	77.45 ± 4.63
		IncLPSVM	0.00 ± 0.00	100.00 ± 0.00	NaN	0.00 ± 0.00	0.00 ± 0.00
		DCIL-IncLPSVM	100.00 ± 0.00	81.75 ± 5.87	**91.70** ± 2.47	**1.23** ± 0.09	**90.36** ± 3.26
	4	IncSVM	11.30 ± 8.98	100.00 ± 0.00	NaN	0.11 ± 0.09	28.58 ± 17.87
		IncLPSVM	0.00 ± 0.00	100.00 ± 0.00	NaN	0.00 ± 0.00	0.00 ± 0.00
		DCIL-IncLPSVM	100.00 ± 0.00	83.00 ± 5.37	**92.22** ± 2.27	**1.21** ± 0.08	**91.06** ± 2.96
Abalone	1	IncSVM	97.05 ± 5.50	83.15 ± 6.88	**90.76** ± 4.37	**1.17** ± 0.11	**89.73** ± 4.75
		IncLPSVM	98.50 ± 2.86	80.85 ± 7.42	90.60 ± 3.44	1.23 ± 0.13	89.14 ± 4.36
		DCIL-IncLPSVM	98.50 ± 2.86	80.85 ± 7.42	90.60 ± 3.44	1.23 ± 0.13	89.14 ± 4.36
	2	IncSVM	94.65 ± 5.49	94.60 ± 3.48	**94.57** ± 3.39	**1.00** ± 0.07	**94.57** ± 3.27
		IncLPSVM	75.20 ± 10.76	98.45 ± 2.08	84.68 ± 7.30	0.76 ± 0.11	85.82 ± 6.38
		DCIL-IncLPSVM	99.50 ± 1.13	81.50 ± 5.93	91.35 ± 2.54	1.23 ± 0.09	89.99 ± 3.34
	3	IncSVM	0.00 ± 0.00	100.00 ± 0.00	NaN	0.00 ± 0.00	0.00 ± 0.00
		IncLPSVM	0.00 ± 0.00	100.00 ± 0.00	NaN	0.00 ± 0.00	0.00 ± 0.00
		DCIL-IncLPSVM	99.55 ± 0.97	83.30 ± 6.32	**92.14** ± 2.67	**1.20** ± 0.10	**90.99** ± 3.48
	4	IncSVM	0.00 ± 0.00	100.00 ± 0.00	NaN	0.00 ± 0.00	0.00 ± 0.00
		IncLPSVM	0.00 ± 0.00	100.00 ± 0.00	NaN	0.00 ± 0.00	0.00 ± 0.00
		DCIL-IncLPSVM	99.10 ± 1.66	82.95 ± 6.34	**91.77** ± 2.83	**1.20** ± 0.09	**90.60** ± 3.57

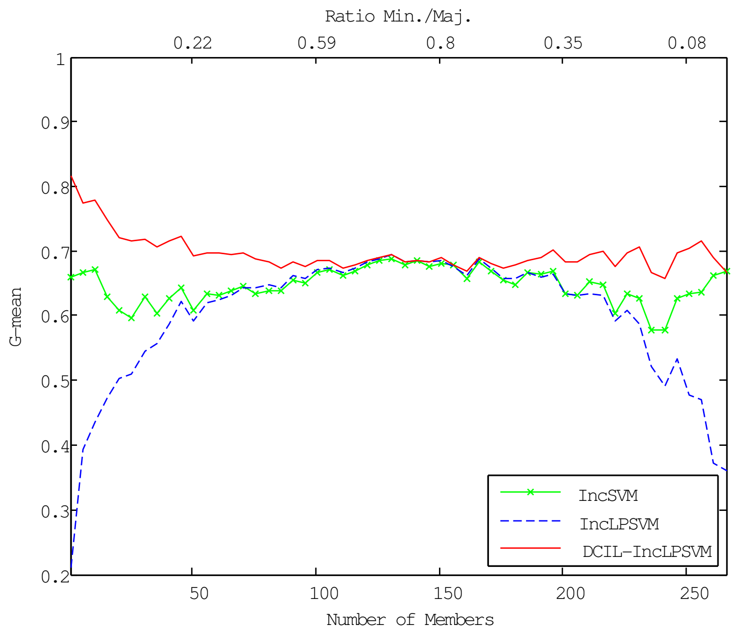
On Breast-Wisconsin dataset, the proposed DCIL-IncLPSVM outperforms batch SVM and LPSVM in the presence of class imbalance. Two classes of training set is balanced in experiment 1. In this case the training results from LPSVM and the proposed DCIL-IncLPSVM are exactly the same since $\sigma_- = \sigma_+ = 1$. For class balanced Breast-Wisconsin data, three algorithms achieve similar accuracy for the negative class, but SVM outperforms slightly for the positive class which leads to better performance in terms of F-measure, RS and G-mean. From experiment 1 in Table 5.3, we can also indicate that negative class in Breast-Wisconsin set is easier to be identified by nature, this may be caused by the data distribution imbalance in the feature space. Taking experiments 2, 3 and 4 into account, we can see that, when the imbalance ratio increases from 1 to 20, the sensitivity drops significantly for both SVM and LPSVM but keeps stable for DCIL-IncLPSVM. Meanwhile the three algorithms obtain comparative specificity. As a result, DCIL-IncLPSVM achieves highest F-measure and G-mean as well as the RS nearest to 1. These phenomena indicate that both SVM and LPSVM are losing classification capability on the minority class when the level of class imbalance is increased. In comparison, the proposed DCIL-IncLPSVM performs stable under varied degrees of class imbalance.

On dataset Car and Abalone, we can see from experiment 1 where the class distribution is balanced that there is a data distribution imbalance effect leading to higher accuracy for positive class. When the class imbalance arising, the effect of class imbalance emerges in the opposite direction drawing the discriminate plane bias to the minority (i.e., positive) class. At certain degree of class imbalance, the interaction of these two opposite effects results in even better performance compare with class balanced circumstances, as evidenced by SVM in experiment 2 on Abalone dataset. Despite being surpassed occasionally, the proposed algorithm maintains stable performance in face of various class imbalance even at the level that peer algorithms completely lost classification capability (e.g., experiment 3 on Car data, experiment 3 and 4 on Abalone data).

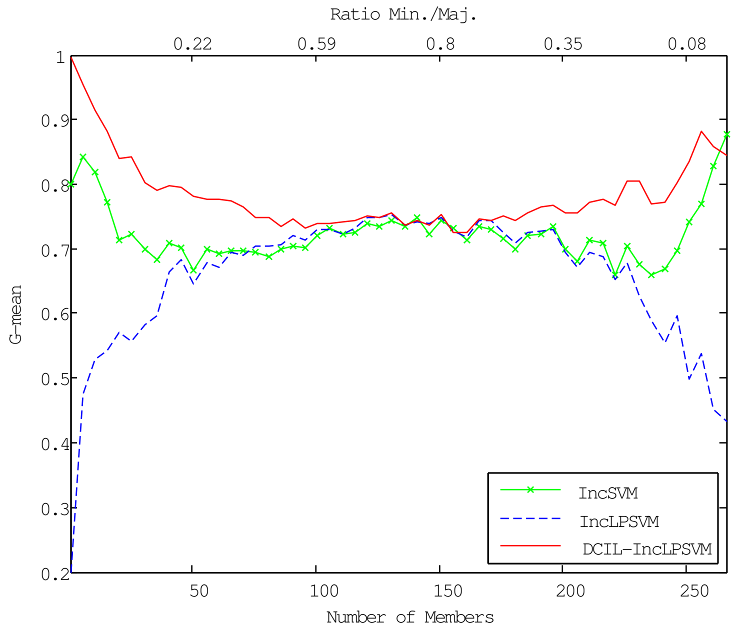
5.5 Dynamic Class imbalance Robustness Tests on Face Membership Authentication

To evaluate the algorithm robustness to dynamic class imbalance, we study the face membership authentication (FMA) problem (Pang et al., 2003). The membership authentication problem is to distinguish the membership class from the nonmembership class based on the personal facial images stored in a database. The FMA problem presents here a comprehensive dynamic class imbalance case, because the membership group is generally much smaller than the nonmembership group, and the size of membership is adjustable. We use dataset from (Kim, Kim & Lee, 2003) which includes images collected from 271 individuals (5 images for each). In the experiment, 4 of 5 images for each person are utilized for training and one image is left for testing. For online FMA, 9 subsets are presented incrementally to IncSVM, IncLPSVM and the proposed DCIL-IncLPSVM algorithm respectively.

We first compare the performance of each classifier under the condition that the



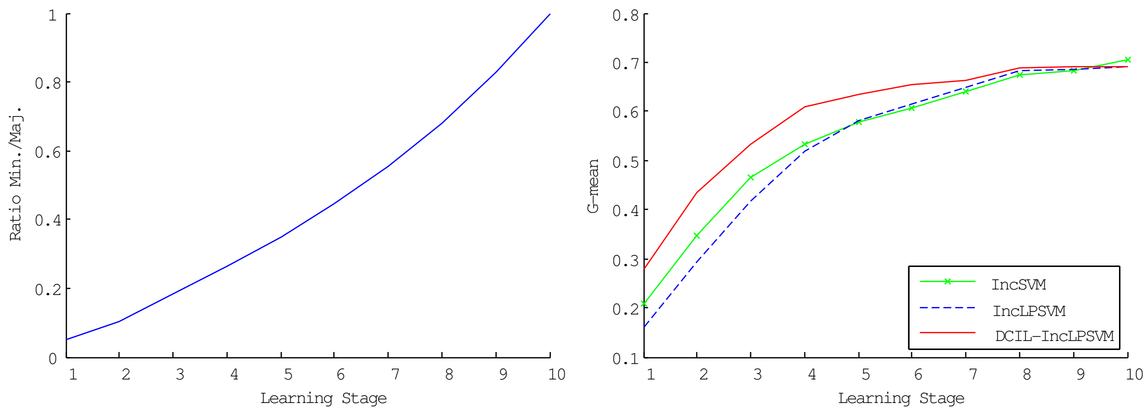
(a) Time course performance



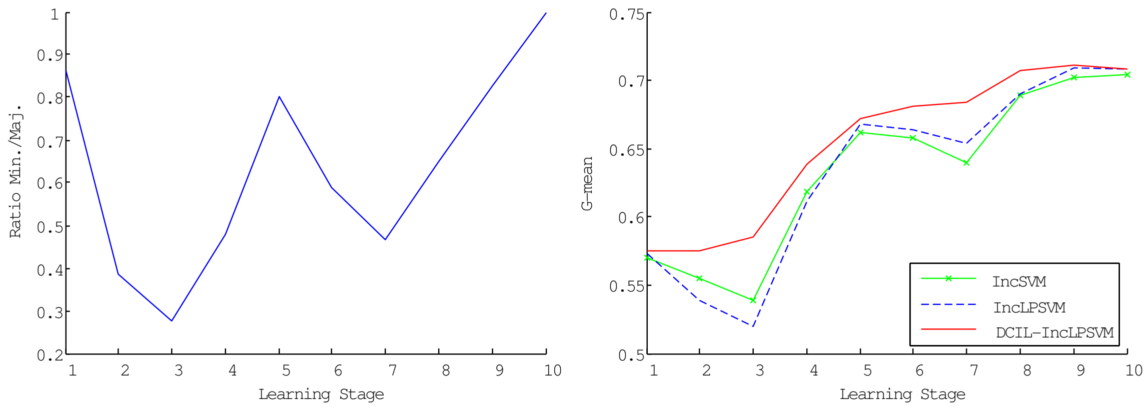
(b) Final performance

Figure 5.1: Robustness tests on online face membership authentication under the condition that the whole dataset has diverse class imbalances

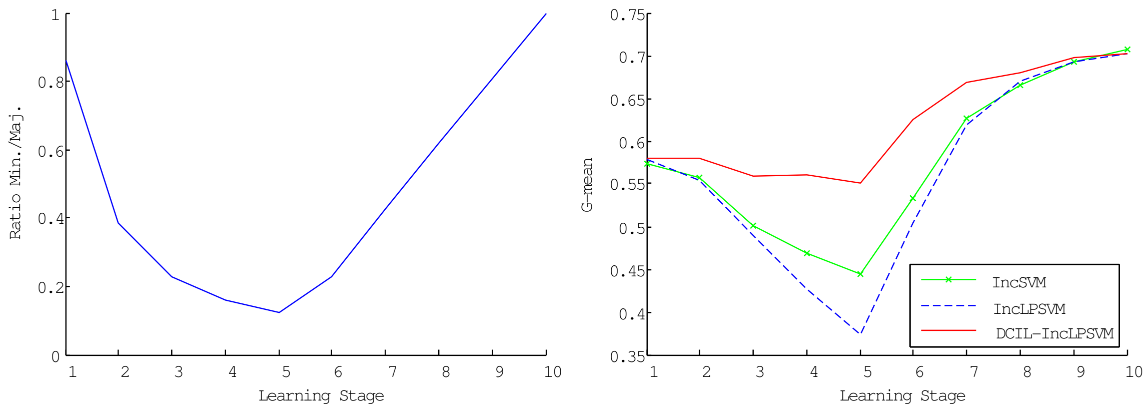
whole dataset has diverse class imbalances. In doing so, the membership group size is set to range from 1 to 270 person with a step size of 5 person. This makes the class imbalance ratio of the whole dataset increase first from 1/270 to 135/136 in



(a) class imbalance consistent decrease



(b) class imbalance random variation



(c) class imbalance increase then decrease

Figure 5.2: Robustness tests on online face membership authentication under the condition that the whole dataset is class balanced and subsets for stage incremental learning is imbalanced.

which the membership group is the minority class, and then decrease from 135/136 back to 1/270 where the non-membership group becomes the minority class. For performance evaluation, we calculate the time course performance as the average of 10 learning stages and the final result as the last incremental stage performance. The obtained results are shown in Figure 5.1a and Figure 5.1b respectively. As we can see for both comparisons, three algorithms perform similarly if the membership sizes are between 100 and 150, and the whole dataset is almost class balanced (i.e., the ratio is close to 1). When the data has noticeable class imbalance, the proposed DCIL-IncLPSVM significantly outperforms other two incremental SVMs. In the case of extreme class imbalance when membership size is below 30 or over 240 (i.e., the ratio is close to 1/270=0.003), IncLPSVM G-mean drops down to under 0.5, but the proposed DCIL-IncLPSVM maintains its superiority to IncSVM with its G-mean growing to above 0.8.

The second robustness test is under the condition that the whole dataset is class balanced, and the subsets at different stages of incremental learning are imbalanced. We are particularly interested in assessing the algorithms' responses to varied class imbalance caused by data additions. In the experiment, we use the overall balanced dataset with 135 members and 136 nonmembers. For online FMA, we change the level of class imbalance according to schemes 'consistent decrease', 'random variation' and 'increase then decrease' respectively. Figure 5.2 gives the experimental results, where left figures plot the real time variation of class imbalance, and right ones summarize the obtained G-means. As we can see, for all three experiments, the performance of the proposed DCIL-IncLPSVM increases consistently through the whole learning lifetime despite of the diverse class imbalance variations. However, for the IncSVM and IncLPSVM, the rising of class imbalance has caused apparent performance degradation. As seen from stages 1 to 3 and 5 to 7 in Figure 5.2b and stages 1 to 5 in Figure 5.2c, the performance of both IncSVM and IncLPSVM drops significantly. However, at the final stage when the class distribution is back to balance, three algorithms give very similar performance. This indicates that for online incremental learning, the effect of class imbalance may overwhelm the contribution of newly added data.

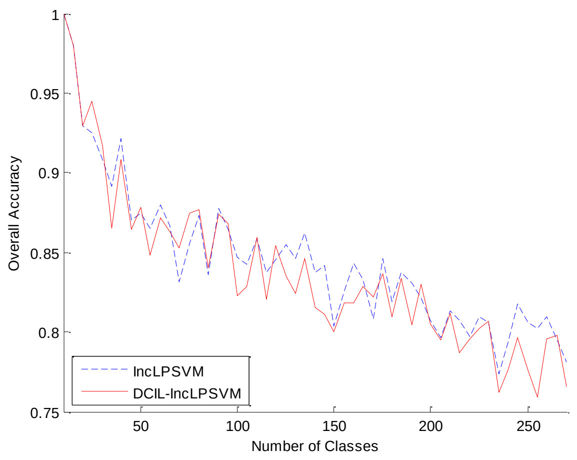
5.6 Class Imbalance on Multi-class Face Recognition

To evaluate the performance of the proposed DCIL-IncLPSVM on multi-class classification, we study the face recognition (FR) task using the facial dataset introduced in last section. Face recognition is a multi-class classification problem in which each person represents one class, and the objective is to correctly classify unlabeled facial images. In the experiment, we use 4 of 5 images for each person for training and the remaining one image for testing.

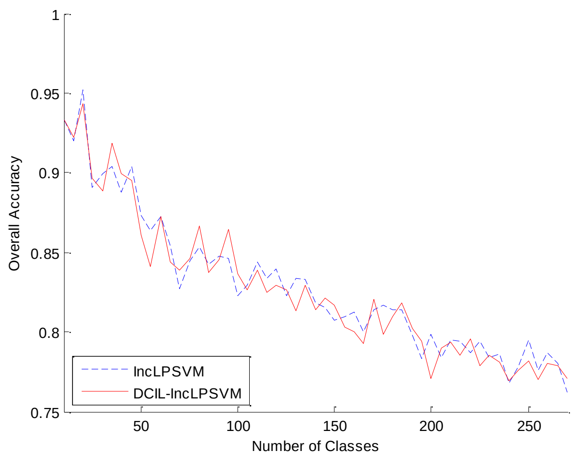
As indicated before, traditional multi-class IncLPSVM is a combination of $k(k-1)$ binary IncLPSVMs, in which class imbalance is incurred for each individual binary IncLPSVM training. The overall accuracy of multi-class classification thus is affected by the performance of each binary IncLPSVM. The proposed DCIL-IncLPSVM for multi-class classification suppresses such impacts by performing real-time DCIL for each binary IncLPSVM. Compared to the traditional IncLPSVM, we can safely assume that with more classes to be classified, the proposed algorithm will achieve bigger improvement.

To testify this assumption, we compare two multi-class classifiers for online face recognition with different number of classes. In doing so, we set the online FR system working on 10 to 270 random selected person with 5 person as the interval. Figure 5.3 and Figure 5.4 shows the average accuracy of both algorithms by the end of each learning stage. These results are summarized as final and time course average accuracy in Figure 5.5. As we can see, All accuracies decrease with the increase of the class number, which indicates that larger number of classes increases the classification difficulty for both algorithms.

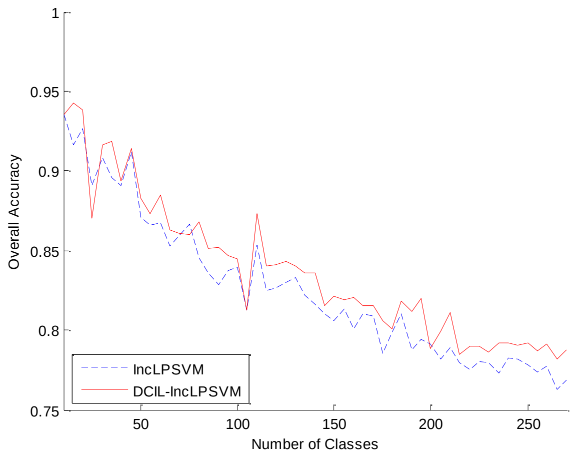
The superiority of the proposed DCIL-IncLPSVM in terms of its final accuracy emerges when the class number is around 30, and it grows steadily with the increase of the class number. When the number of classes rises to 270, such superiority rises to 3.5%. Although the superiority on time course average accuracy is not as much as that on the final accuracy, it is apparent that the consistent increase trend is the same for both evaluations. This indicates that the proposed DCIL-IncLPSVM performs better online face recognition that involves more than 30 person, and consistantly outperform traditional IncLPSVM throughout the whole incremental learning process. Figure 5.6 gives an example of online face recognition for 50 persons, where the



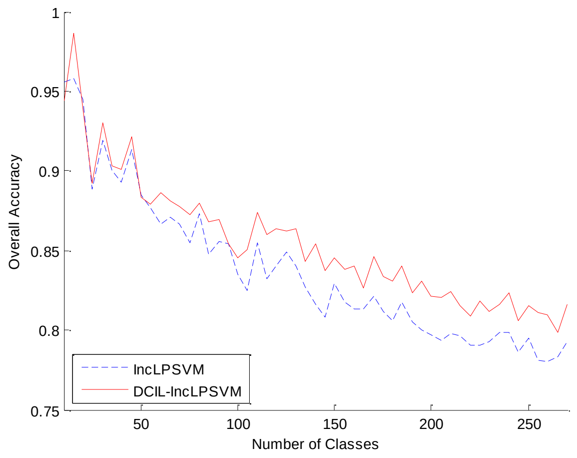
(a) Stage 1



(b) Stage 2



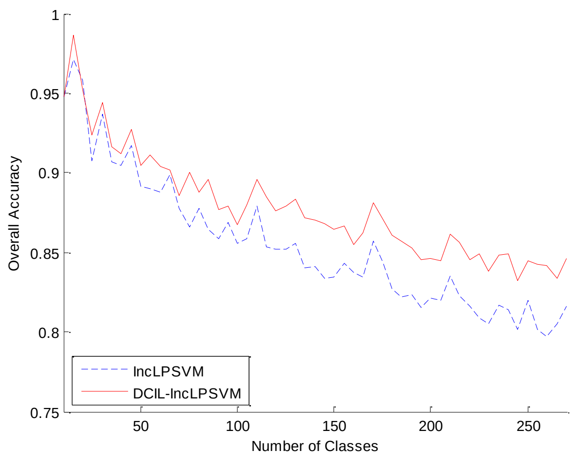
(c) Stage 3



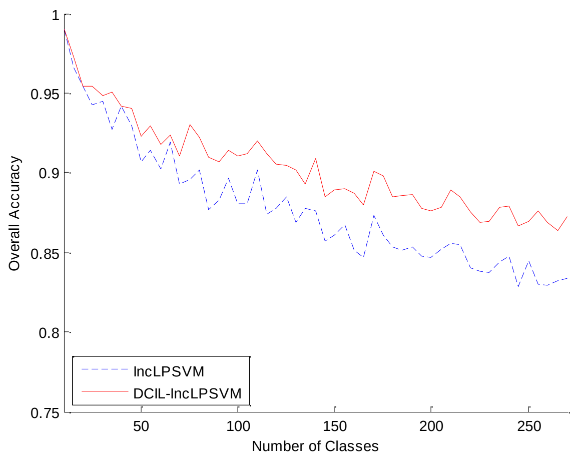
(d) Stage 4

Figure 5.3: Multi-class classification tests on online face recognition in different number of classes - learning stages A.

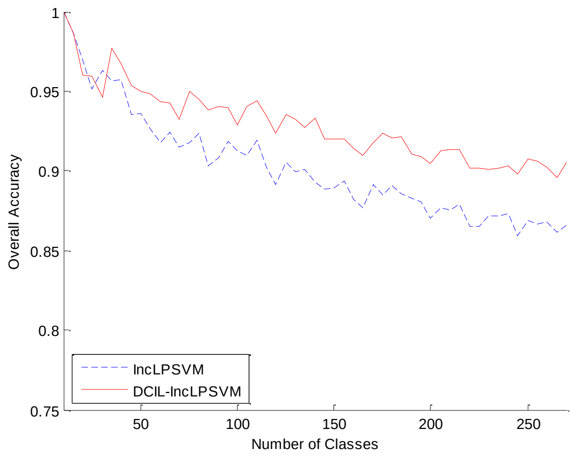
overall accuracy of IncLPSVM and DCIL-IncLPSVM are plotted at 10 incremental stages.



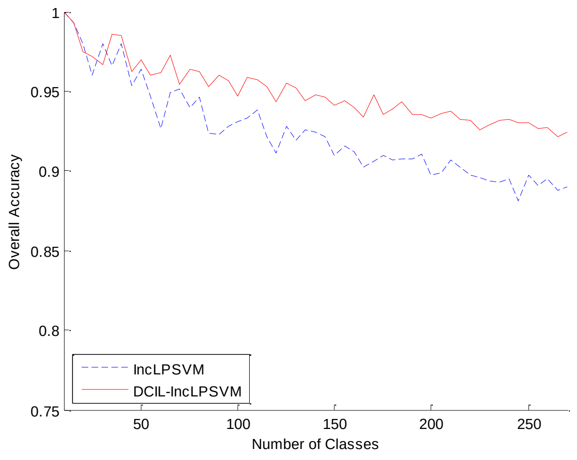
(a) Stage 5



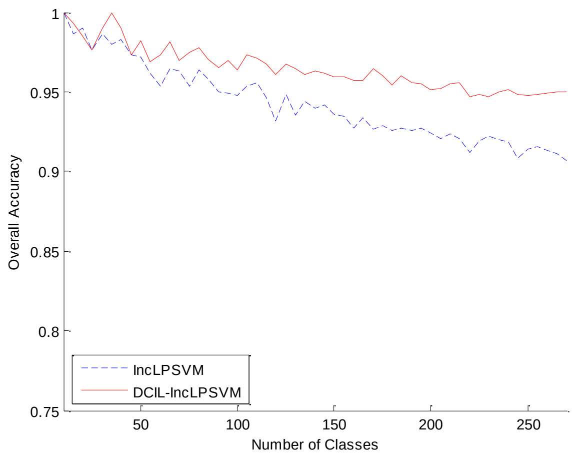
(b) Stage 6



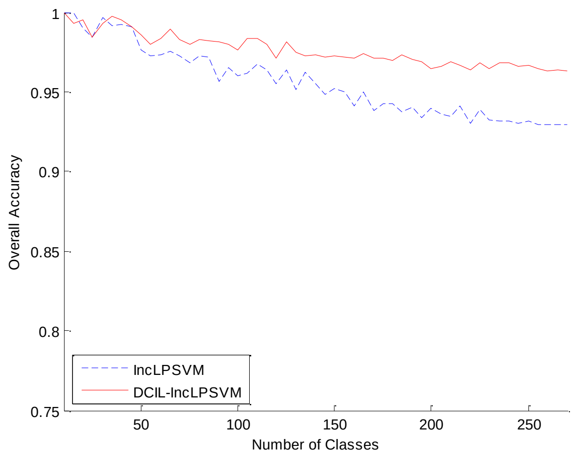
(c) Stage 7



(d) Stage 8

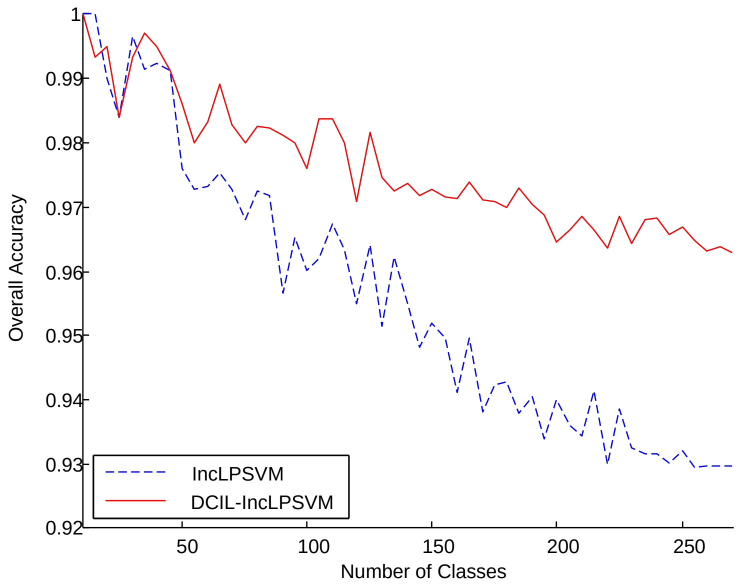


(e) Stage 9

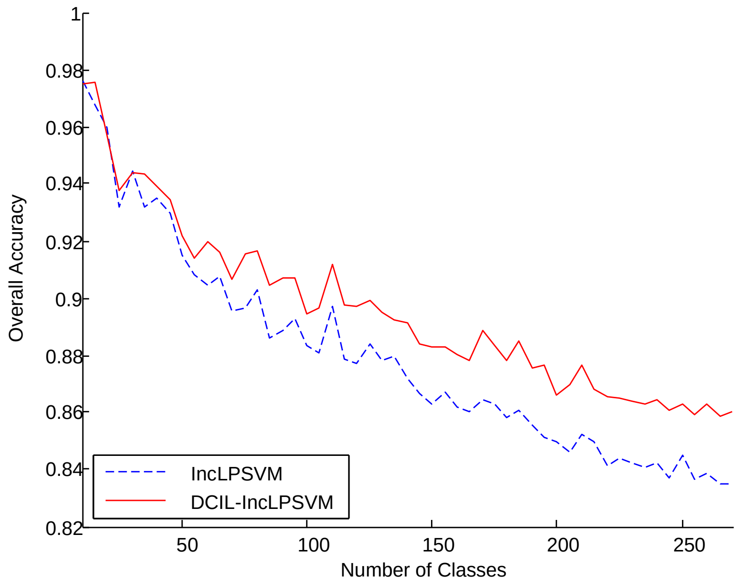


(f) Stage 10

Figure 5.4: Multi-class classification tests on online face recognition in different number of classes - learning stages B.



(a) Final accuracy



(b) Time course average accuracy

Figure 5.5: Multi-class classification tests on online face recognition in different number of classes - summary.

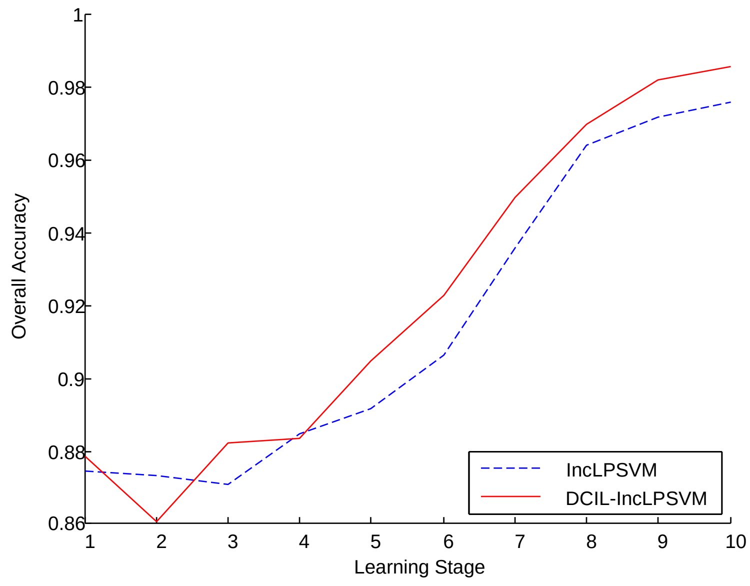


Figure 5.6: Multi-class classification tests on online face recognition in 50 classes.

Chapter 6

Conclusions and Future Work

Aimed at solving the class imbalance problem of LPSVM data stream learning, we successfully developed in this paper a DCIL solution that enables learning incrementally LPSVM and its weights in real time according to the class imbalance ratio of the training data. Our analysis guarantees that the updated LPSVM obtained from proposed DCIL-IncLPSVM (i.e., the model after incremental learning on newly added/retired data) is equivalent to a batch LPSVM model over the updated dataset (i.e., the dataset after data addition/retirement).

For LPSVM batch learning, wLPSVM (G. M. Fung & Mangasarian, 2005) models successfully the class imbalance trade-off by a weighting matrix $\boldsymbol{N}$. For online learning in the difficulty of dynamic class imbalance, a straightforward solution is to update $\boldsymbol{N}$ and LPSVM for incremental learning. However it is problematic to update $\boldsymbol{N}$, because it involves the updating of two matrix multiplications. For the DCIL of LPSVM, we derive, without loss of information, a new expression of wLPSVM. In this expression, matrix multiplication $\boldsymbol{E}^T\boldsymbol{N}\boldsymbol{E}$ is replaced with $\sigma_+\boldsymbol{M}_+ + \sigma_-\boldsymbol{M}_-$ and $\boldsymbol{E}^T\boldsymbol{D}\boldsymbol{N}\boldsymbol{e}$ with $\sigma_+\boldsymbol{v}_+ - \sigma_-\boldsymbol{v}_-$. As a result, the weighting matrix $\boldsymbol{N} \in \mathbb{R}^{n\times n}$ is transformed to two $\mathbb{R}^1$ coefficients σ_+ and σ_-, which can be easily updated upon any data addition/retirement. For both static and dynamic class imbalance learning, the developed DCIL-IncLPSVM demonstrated high robustness that we expect in practice.

For class imbalance learning, wLPSVM measures the bias of class distribution just by the number of samples, but does not consider the state of sample distribution. However it is noticed that for LPSVM or SVM training, even if the number of samples is even for both classes, bias on decision boundary may still happen due to biased data scattering. Hence, taking sample distribution into account to weight LPSVM is an interesting direction for our future research.

Chapter 7

Publication Delivery

Lei Zhu, Shaoning Pang, Gang Chen and Hossein Sarrafzadeh. 'Class Imbalance Robust Incremental LPSVM for Data Streams Learning'. submitted to 2012 International Joint Conference on Neural Networks. Accepted.

Shaoning Pang, Lei Zhu, Gang Chen and Hossein Sarrafzadeh. 'DCIL-IncLPSVM: Dynamic Class Imbalance Learning for Incremental LPSVM'. submitted to ACM Transactions on Knowledge Discovery from Data. Under review.

References

Akbani, R., Kwek, S. & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In *Proceedings of the 15th european conference on machine learning (ecml* (pp. 39–50).

Batuwita, R. & Palade, V. (2010). Fsvm-cil: Fuzzy support vector machines for class imbalance learning. *IEEE Transactions on Fuzzy Systems*, *18*(3), 558 -571.

Cauwenberghs, G. & Poggio, T. (2000). Incremental and decremental support vector machine learning. In *Advances in neural information processing systems nips'2000* (p. 409-415).

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, S. & He, H. (2009). Sera: Selectively recursive approach towards nonstationary imbalanced stream data mining. In *Proceedings of 2009 international joint conference on neural networks* (p. 522 -529).

Dong, Y.-S. & Han, K.-S. (2005). Text classification based on data partitioning and parameter varying ensembles. In *Proceedings of the 2005 acm symposium on applied computing* (p. 1044-1048). New York, NY, USA: ACM.

Frank, A. & Asuncion, A. (2010). *UCI machine learning repository.* Available from `http://archive.ics.uci.edu/ml`

Fung, G. & Mangasarian, O. L. (2001). Proximal support vector machine classifiers. In *Proceedings of the seventh acm sigkdd international conference on knowledge discovery and data mining* (p. 77-86). San Francisco, California.

Fung, G. M. & Mangasarian, O. L. (2001). Incremental support vector machine classification. In *Proceedings of 7th acm sigkdd international conference on knowledge discovery and data mining* (pp. 77–86).

Fung, G. M. & Mangasarian, O. L. (2005). Multicategory proximal support vector machine classifiers. *Machine Learning*, *59*, 77-97.

Gouripeddi, R., Balasubramanian, V., Panchanathan, S., Harris, J., Bhaskaran, A. & Siegel, R. (2009). Predicting risk of complications following a drug eluting stent procedure: A svm approach for imbalanced data. In *Proceedings of 22nd ieee international symposium on computer-based medical systems (cbms)* (p. 1 -7).

He, H. & Chen, S. (2008). Imorl: Incremental multiple-object recognition and localization. *IEEE Transactions on Neural Networks*, *19*(10), 1727 -1738.

He, H. & Garcia, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263 -1284.

Hong, X., Chen, S. & Harris, C. (2007). A kernel-based two-class classifier for imbalanced data sets. *IEEE Transactions on Neural Networks*, *18*(1), 28 -41.

Huang, K., Yang, H., King, I. & Lyu, M. (2006). Imbalanced learning with a biased minimax probability machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,*, *36*(4), 913 -923.

Imam, T., Ting, K. & Kamruzzaman, J. (2006). z-svm: An svm for improved classification of imbalanced data. In A. Sattar & B.-h. Kang (Eds.), *Ai 2006: Advances in artificial intelligence* (Vol. 4304, p. 264-273). Springer Berlin / Heidelberg.

Karasuyama, M. & Takeuchi, I. (2010). Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks*, *21*(7), 1048 -1059.

Kim, M.-S., Kim, D. & Lee, S.-Y. (2003). Face recognition using the embedded hmm with second-order block-specific observations. *Pattern Recognition*, *36*(11), 2723-2735.

Liu, X.-Y., Wu, J. & Zhou, Z.-H. (2006, dec.). Exploratory under-sampling for class-imbalance learning. In *Proceedings of sixth international conference on data mining (icdm)* (p. 965 -969).

Liu, X.-Y., Wu, J. & Zhou, Z.-H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *39*(2), 539 -550.

Maciejewski, T. & Stefanowski, J. (2011). Local neighbourhood extension of smote for mining imbalanced data. In *Proceedings of 2011 ieee symposium on computational intelligence and data mining (cidm)* (p. 104 -111).

Pang, S., Kim, D. & Bang, S.-y. (2003). Membership authentication in the dynamic group by face classification using svm ensemble. *Pattern Recognition Letters*, *24*(1-3), 215-225.

Pang, S., Kim, D. & Sung-yang, B. (2005). Face membership authentication using svm classification tree generated by membership-based lle data partition. *IEEE Transaction on Neural Network*, *16*(2), 436-446.

Su, C.-T. & Hsiao, Y.-H. (2007). An evaluation of the robustness of mts for imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *19*(10), 1321 -1332.

Tao, X. & Ji, H. (2007). A modified psvm and its application to unbalanced data classification. In *Proceedings of third international conference on natural computation* (Vol. 1, p. 488-490).

Tveit, A. & Hetl, M. L. (2003). Multicategory incremental proximal support vector classifiers. In *In proceedings of the 7th international conference on knowledge-based information & engineering systems (kes2003), number 2773 in lecture notes in artificial intelligence (lnai* (pp. 386–392). Springer-Verlag.

Vapnik, V. N. (1995). *The nature of statistical learning theory.* Berlin, Germany: Springer-Verlag.

Veropoulos, K., Campbell, C. & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on ai* (pp. 55–60).

Wang, H.-Y. (2008). Combination approach of smote and biased-svm for imbalanced datasets. In *Proceedings of ieee international joint conference on neural networks* (p. 228 -231).

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *SIGKDD*, *6*, 7–19.

Wu, G. & Chang, E. (2005). Kba: kernel boundary alignment considering imbalanced data distribution. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 786 - 795.

Yamasaki, T. & Ikeda, K. (2005). Incremental svms and their geometrical analyses. In *Proceedings of international conference on neural networks and brain 05* (Vol. 3, p. 1734 -1738).

Yan, R., Liu, Y., Jin, R. & Hauptmann, A. (2003). On predicting rare classes with svm ensembles in scene classification. In *Proceedings of 2003 ieee international conference on acoustics, speech, and signal processing (icassp '03)* (Vol. 3, p. 21-29).

Zhai, Y., Yang, B., Ma, N. & Ruan, D. (2010). New construction of ensemble classifiers for imbalanced datasets. In *Proceedings of 2010 international conference on intelligent systems and knowledge engineering (iske)* (p. 228 -233).

Zhou, Z.-H. & Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, *18*(1), 63 - 77.

Zhuang, D., Zhang, B., Yang, Q., Yan, J., Chen, Z. & Chen, Y. (2005). Efficient text classification by weighted proximal svm. In *Proceedings of fifth ieee international conference on data mining.*