วิธีการจำแนกประเภทข้อมูลแบบสตรีมมิ่ง โดยใช้วงรีหลายมิติที่สามารถปรับขนาดได้ พร้อมกับอัตราส่วนระยะทางฉายแบบดิสครีมิแนนต์เชิงเส้น

นายพีรศุษม์ รุ้งจรัสแสง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2561

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

บทคัดย่อและแฟ้มข้อมูลฉบับเต็มของวิทยานิพนธ์ตั้งแต่ปีการศึกษา 2554 ที่ให้บริการในคลังปัญญาจุฬาฯ (CUIR)
เป็นแฟ้มข้อมูลของนิสิตเจ้าของวิทยานิพนธ์ที่ส่งผ่านทางบัณฑิตวิทยาลัย
The abstract and full text of theses from the academic year 2011 in Chulalongkorn University Intellectual Repository(CUIR)
are the thesis authors' files submitted through the Graduate School.

STREAMING DATA CLASSIFICATION METHOD USING SCALABLE HYPER-ELLIPSOIDS WITH LINEAR DISCRIMINANT PROJECTION DISTANCE RATIO

Mr. Perasut Rungcharassang

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Applied Mathematics and
Computational Science
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2018
Copyright of Chulalongkorn University

Dissertation Title	STREAMING DATA CLASSIFICATION METHOD USING SCALABLE HYPER-ELLIPSOIDS WITH LINEAR DISCRIMINANT PROJECTION DISTANCE RATIO
By	Mr. Perasut Rungcharassang
Field of Study	Applied Mathematics and Computational Science
Dissertation Advisor	Professor Chidchanok Lursinsap, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Doctoral Degree

. . . . Dean of the Faculty of Science
(Professor Polkit Sangvanich, Ph.D.)

DISSERTATION COMMITTEE

. . . . Chairman
(Assistant Professor Krung Sinapiromsaran, Ph.D.)

. . . . Dissertation Advisor
(Professor Chidchanok Lursinsap, Ph.D.)

. . . . Examiner
(Assistant Professor Khamron Mekchay, Ph.D.)

. . . . Examiner
(Assistant Professor Kitiporn Plaimas, Ph.D.)

. . . . External Examiner
(Assistant Professor Saichon Jaiyen, Ph.D.)

พีรศุษม์ รุ้งจรัสแสง : วิธีการจำแนกประเภทข้อมูลแบบสตรีมมิ่ง โดยใช้วงรีหลายมิติที่สามารถปรับขนาดได้ พร้อมกับอัตราส่วนระยะทางฉายแบบดิสครีมิแนนต์เชิงเส้น. (STREAMING DATA CLASSIFICATION METHOD USING SCALABLE HYPER-ELLIPSOIDS WITH LINEAR DISCRIMINANT PROJECTION DISTANCE RATIO) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ศ.ดร.ชิดชนก เหลือสินทรัพย์, 77 หน้า.

การเรียนรู้ข้อมูลแบบสตรีมมิ่งด้วยหน่วยความจำที่จำกัดกลายเป็นปัญหาที่น่าสนใจ แม้ว่าวิธีการเรียนรู้หลายวิธีถูกนำเสนอเร็ว ๆ นี้ตามแนวคิดของการละทิ้งข้อมูลหลังเรียนรู้ อย่างไรก็ตามความเร็วในการเรียนรู้ จำนวนนิวรอนเกินจำเป็น และความแม่นยำในการจำแนกของวิธีการเหล่านี้ สามารถปรับปรุงได้ดียิ่งขึ้นในแง่ของความเร็วในการเรียนรู้ที่เร็วขึ้น จำนวนนิวรอนที่น้อยลง และความแม่นยำที่สูงขึ้น แนวความคิดใหม่ที่ถูกนำเสนอในงานวิทยานิพนธ์นี้ประกอบด้วย 4 ส่วนดังนี้ (1) โครงสร้างใหม่ของฟังก์ชันวงรีหลายมิติที่สามารถปรับขนาดได้ (เชฟ) สามารถจัดการกับปัญหาที่มีจำนวนมิติมากกว่าจำนวนข้อมูล โดยใช้เรกูลาไรเซชันพารามิเตอร์กับเมทริกซ์ความแปรปรวนร่วมของฟังก์ชันวงรีข้างต้น (2) ฟังก์ชันเวียนบังเกิดแบบใหม่ เพื่อปรับปรุงเมทริกซ์ความแปรปรวนร่วมของฟังก์ชันวงรีตามข้อมูลที่เข้ามาเป็นกลุ่ม (3) ความเร็วและเงื่อนไขที่ง่ายต่อการทดสอบการซ้อนทับของฟังก์ชันวงรีสองตัวและ (4) ตัววัดระยะทางใหม่สำหรับการระบุประเภทของข้อมูลโดยใช้การฉายระยะทางลงบนเวกเตอร์ดิสคริมิแนนต์ วิธีการที่นำเสนอมีผลการทดลองที่ประสิทธิภาพเพิ่มขึ้น เมื่อเปรียบเทียบกับผลการทดลองของวิธีการอื่น ๆ

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์
ลายมือชื่อนิสิต ....
ลายมือชื่อ อ.ที่ปรึกษาหลัก ....
สาขาวิชา คณิตศาสตร์ประยุกต์และวิทยาการคณนา
ปีการศึกษา 2561

5572864623 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

KEYWORDS : STREAMING DATA / CLASSIFICATION / LDA

PERASUT RUNGCHARASSANG : STREAMING DATA CLASSIFICATION METHOD USING SCALABLE HYPER-ELLIPSOIDS WITH LINEAR DISCRIMINANT PROJECTION DISTANCE RATIO. ADVISOR : PROF. CHIDCHANOK LURSINSAP, Ph.D., 77 pp.

Learning streaming data with limited size of memory storage becomes an interesting problem. Although there have been several learning methods recently proposed, based on the interesting concept of *discard-after-learn*, the performance of these issues: the learning speed, number of redundant neurons, and classification accuracy of these methods can be further improved in terms of faster speed, less number of neurons, and higher accuracy. The following new four concepts and approaches were proposed in this dissertation: (1) a more generic structure of hyper-ellipsoidal function called *Scalable Hyper-Ellipsoidal Function* (SHEF) capable of handling the problem of curse of dimensionality by introducing a regularization parameter into the covariance matrix of SHEF; (2) a new recursive function to update the covariance matrix of SHEF based on only the incoming data chunk; (3) a fast and easy conditions to test the states of being overlapped, inside, or touch of two SHEFs; (4) a new distance measure for determining the class of a queried datum based on the projected distance on LDA discriminant vector. The experimental results show the significant improvement when compared with other methods.

Department	: Mathematics and	Student's Signature	....
	Computer Science	Advisor's Signature	....
Field of Study	: Applied Mathematics and		
	Computational Science		
Academic Year	: 2018		

ACKNOWLEDGEMENTS

In the completion of this dissertation, I am very grateful to my advisor, Professor Dr. Chidchanok Lursinsap for his advices and many interesting ideas. Without his great suggestions, this dissertation could not be completed.

Sincere thanks to my dissertation committee, Assistant Professor Dr. Krung Sinapiromsaran, Assistant Professor Dr. Khamron Mekchay, Assistant Professor Dr. Kitiporn Plaimas, and Assistant Professor Dr. Saichon Jaiyen, for their interesting questions and suggestions on the suitability of the dissertation content.

I sincerely thank all of my lecturers at Department of Mathematics and Computer Science, Chulalongkorn University, who have given good advices and knowledge for living as a Ph.D. student.

I would like to thank Teaching Assistant Scholarship, Faculty of Science, Chulalongkorn University for partial support of this work.

I would like to thank Mr. Kittisak Chumpong for his suggestions about proving Theorem 3.2.1, and special thank to Miss Pornchanit Supwilai for being so caring, and understanding all of me.

Moreover, I would like to thank my friends for their support and some important advices about literature reviews. Last but not least, I am very grateful to my family who is the most important supporter.

CONTENTS

Page

ABSTRACT IN THAI iv

ABSTRACT IN ENGLISH v

ACKNOWLEDGEMENTS vi

CONTENTS vii

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER

1 INTRODUCTION **1**

1.1 Studied Problems and Constraints 5

2 RELEVANT BACKGROUND **8**

2.1 Basic Concept of Standard Hyper-ellipsoid Function 8

2.2 Concept of LDA with Multiple Classes and Binary Classes 9

2.3 Concept of Versatile Elliptic Basis Function (VEBF) 11

2.4 Checking the Intersection of Two Hyper-ellipsoids 13

2.5 Distance from a Sample Point to a Hyper-ellipsoid 14

2.5.1 Euclidean Distance 14

2.5.2 Mahalanobis Distance 14

2.5.3 The Versatile Elliptic Basis Function Value 15

2.5.4 Boundary Distance 15

3 METHODOLOGY **18**

3.1 Proposed Structure of Capturing Function and Parameter Updating 18

3.1.1 Structure of Scalable Hyper-Ellipsoidal Function 18

3.1.2 Updating Parameters of SHEF 21

3.2 New Learning Method Using Scalable Hyper-ellipsoidal Function 22

3.2.1 Initializing SHEF Widths and Threshold Distance for Introducing New SHEF 23

3.2.2 Condition of Intersection of Two Scalable Hyper-Ellipsoids 24

3.2.3 Learning Algorithm of SHEF 26

CHAPTER Page

3.2.4 Time Complexity of the Learning Algorithm 28
3.2.5 The Illustration of Learning Algorithm in Two-dimensional 29
3.3 Identifying Classes of Testing Data 34
3.3.1 Projecting Width of SHEF onto Discriminant Vector 35
3.3.2 Measuring Distance from a Point to SHEF along Discriminant Vector 37
3.3.3 Determining Class of Queried Data Point Based on Projection Ratio 39
4 EXPERIMENTS AND RESULTS 41
4.1 Data sets 41
4.2 Accuracy Measurements 41
4.3 Parameter Settings 43
4.4 Comparison Results 45
4.4.1 Test Accuracy 48
4.4.2 Average number of generated neurons 50
4.4.3 Computational time 51
4.5 Improve Test Accuracy on VEBF and CIL using Projection Ratio in Testing 55
5 DISCUSSION AND CONCLUSION 63
5.1 Suggestion on Parameter Settings for the Proposed Method 66
REFERENCES 69
APPENDICES 72
BIOGRAPHY 77

LIST OF TABLES

Table Page

1.1 Characteristics of the Compared Methods 4
4.1 Experimental data sets and their attributes. 43
4.2 Parameter settings of VLLDA, VEBF, and CIL. 44
4.3 Number of training data in each fold for each streaming chunk. 47
4.4 Comparison of accuracy (%) in the five-fold validation. 49
4.5 Comparison of the number of neurons in VEBF, CIL, and SHEF methods at the end of training process. 52
4.6 Computational time (seconds) for synthetic data 53
4.7 Computational time (seconds) for real-world data 54
4.8 Comparison of accuracy (%) of VEBF, CIL, VEBF+, and CIL+ 56

LIST OF FIGURES

Figure Page

2.1 Approximate distance from a point to an ellipse by Zimmermann and Svoboda. 16

2.2 Approximate distance from a point to an ellipse by Wattanakitrungroj *et.al.* . 17

3.1 Relationship between eigenvalues and eigenvectors of the covariance matrix to size and shape of hyper-ellipsoid in two-dimensional, respectively. 19

3.2 The data set in the first chunk for calculating the initial width in each class. . 30

3.3 Feed the first training data with class 1 (blue circle). 30

3.4 Feed the second training data with class 2 (red square). 30

3.5 Feed the third training data with class 1. 31

3.6 Introduce a new SHEF then discard the captured data and feed the forth training data with class 2. 31

3.7 Feed the fifth training data with class 2. 31

3.8 Introduce a new SHEF then discard the captured data and feed the sixth training data with class 2. 32

3.9 Update parameters of SHEF_2. 32

3.10 Four SHEFs of the first chunk 32

3.11 Feed a new training data with class 2. 33

3.12 Update parameters of SHEF_2. 33

3.13 Merge two overlapping SHEF into the new SHEF. 33

3.14 An example of wrong interpretation of closeness between testing datum and two SHEFs. The closeness is determined by measuring Euclidean distance from the datum to the centroids of both SHEFs. 34

3.15 The closeness distances measured with respect to the boundary of SHEF_1 and SHEF_2. The datum is close to SHEF_1 instead of SHEF_2. 34

3.16 Projection of a SHEF onto the discriminant vector $\mathbf{w}$ in a 2-dimensional space. 36

3.17 The concept for measuring distance from $\mathbf{x}$ to a SHEF by the ratio of $|\mathbf{w}_p^T(\mathbf{x}-\mathbf{c})|$ and $r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$ along the discriminant vector $\mathbf{w}_p$ defined in equation (3.24). 37

Figure Page

3.18 An example of projections of two SHEFs and $\mathbf{x}$ onto a discriminant vector $\mathbf{w}$ defined in equation (2.8). There are two *projection ratios*, one for each SHEF. 39

4.1 Seven synthetic data sets used in the experiments. 42

4.2 The results of learning seven synthetic data sets using SHEFs. Only 80% of data in each synthetic data were used for training. 46

4.3 The chunk-wise accuracy (%) for six methods after training in each chunk on seven synthetic data sets 57

4.4 The chunk-wise accuracy (%) for six methods after training in each chunk on seven real-world data sets 58

4.5 The average cumulative chunk-wise accuracy (%) for six methods after training in each chunk on seven synthetic data sets 59

4.6 The average cumulative chunk-wise accuracy (%) for six methods after training in each chunk on seven real-world data sets 60

4.7 The number of generated neurons for three methods after training in each chunk on seven synthetic data sets 61

4.8 The number of generated neurons for three methods after training in each chunk on seven real-world data sets 62

CHAPTER I

INTRODUCTION

Problem of classifying streaming data with some specific characteristics has been an interesting topic in many fields such as business, academia, and medical information; especially, where valuable information is tremendously and continuously generated in the internet. However, the speed of hardware technology to increase the memory size to catch up with generated data is obviously much slower than the speed of generated data. This type of data is known as streaming data [1]. This situation leads to a very challenging development of new neural learning algorithms to cope with the problems of data overflow, fast learning speed, as well as limited computing resources or low energy consumption. For classifying streaming data, the learning speed of classifiers must be faster than the speed of incoming data with limited resources and also be capable of achieving the classification accuracy for queried data at any time period.

Generally, when the number of dimensions exceeds the number of data in each class, it is hard to efficiently classify those data. This is known as *Curse of Dimensionality* [2]. Thus, a dimensionality reduction method (DRM) or a feature reduction method (FRM) such as principal component analysis (PCA) [3] and linear discriminant analysis (LDA) [4] are applied in several classification techniques as a pre-processing step to reduce the dimensions of the original data space. These methods transform data in the original space into a new space by either rotating all bases or projecting data onto a discriminant vector. Furthermore, these methods are based on the assumption that the whole data set must be presented prior to the dimensional reduction. Obviously, this assumption cannot be applied to the situation of streaming data where data gradually flow into the learning process. To cope with streaming data, the incremental versions of both PCA and LDA were proposed.

LDA is a suitable method for a classification problem more than PCA. Unlike PCA, LDA emphasizes on the minimum degree of overlap of two individual class after projecting both classes onto a discriminant vector. On the contrary, PCA emphasizes on finding the actual direction of data distribution such that the maximum variance in all dimensions. It does not consider each individual class as LDA does. In 2005, Pang *et.al.* proposed an incremental linear discriminant analysis (ILDA) [5] which makes LDA applicable to the streaming data scenario. Several versions of incremental LDA were developed to improve computational speed [6–8]. However LDA and its incremental version have the same major limitation. If data are nonlinear separable, they will not work properly.

Handling a nonlinear classification problem has three main approaches which are a nonlinear kernel to the classifiers such as a radial basis function kernel [9], local hyperplanes [10], and local data structure to classify some part of data such as a versatile elliptic basis function [11, 12], k-nearest neighbors [13], and k-mean clustering [14]. Those methods are not seriously considered according to the characteristics of streaming data. In [13, 14], the whole data set is used for calculating a distance. Even though both [11, 12] introduced the algorithm for classifying streaming data, but some parameters of their methods must be derived from incoming data. This will add more unnecessary computational time to a learning process.

Besides the efficiency of the learning process, determining the class of queried datum is also a very significant step to achieve the highest classification accuracy. Generally, the class of queried datum is decided by finding a cluster having the nearest distance measured from either the centroid of the cluster or the boundary of the cluster to queried datum. Although this approach is very practical and rather efficient, the accuracy of classifying datum depends strongly upon the shape of data distribution of the cluster. A new improvement of measuring the nearest distance based on local feature reduction was proposed in this study.

Unlike the traditional feature reduction method where the global feature reduction is emphasized, our approach concerns only the local features of the considered cluster. This is based on the observation that data distribution of each cluster is different which implies that the importance of each feature in different clusters should be different as well. Therefore, our approach focuses on the local feature reduction of each cluster more than global feature reduction. One significant application of emphasizing the local feature reduction is to determine the class of queried datum based on the nearest distance between datum and the candidate cluster.

Several methods have been proposed to solve the streaming classification problem where nonlinear classifying functions, amount of data, and learning time complexity are involved. The characteristics of all compared methods in this study are shown in Table 1.1. Six characteristics are categorized. First, *Sequential incremental* means that the method learns one by one of incoming data. Second, *Chunk incremental* means that the method learns chunk by chunk of incoming data. Third, *One-pass* means that the method will discard any data after learned. Fourth, *Stream* means the method must process without knowing data in advance. Besides, the method does not use that data in advance to set any initial parameters. Fifth, *FRM* means that the method uses the feature reduction method for the classification problem. Finally, *Local* means that the method uses only the information from incoming data distribution.

Table 1.1: Characteristics of the Compared Methods

Methods	Incremental		One-pass	Stream	FRM	Local	Tuned parameters (determined by user)
	Sequential	Chunk					
ILDA [5]	✓	-	-	✓	✓	-	No tuning
LOL [10]	✓	-	$\star$	✓	-	✓	k, λ, and C
VEBF [11]	✓	-	✓	-	-	✓	δ
CIL [12]	-	✓	✓	$\star\star$	-	✓	δ
VLLDA [13]	-	-	-	-	✓	✓	k
Proposed method	✓	-	✓	✓	✓	✓	No tuning

$\star$ LOL is the true one-pass learning for only binary classification problem. When LOL is applied to classify multi-class data, it uses one-vs-all approach to classify one class at a time. $\star\star$ The initial width of a hyper-ellipsoid in CIL for any class is computed from the first 20% of total training data. (✓refers Yes and - refers No)

For ILDA (Incremental LDA), Pang *et.al.* introduced two strategies for directly updating $\mathbf{S}_B$ and $\mathbf{S}_W$ (between-class and within-class scatter matrix, respectively) which are one-by-one strategy called *sequential ILDA* and chunk-by-chunk strategy called *chunk ILDA*. The equations for parameter updating of $\mathbf{S}_B$ and $\mathbf{S}_W$ were rather complex. Definitely, this method failed when data were not linearly separable. Only the sequential ILDA is considered for performance comparison with our method. Although ILDA employs incremental learning, it still needs to retain whole incoming data during the classification of testing data. This implies that all training data cannot be discarded from the memory to clear the storage space for receiving next new incoming data chunk.

For LOL (Local Online Learning), an incremental updating for multiple hyperplanes was proposed to deal with streaming data. The Passive Aggressive algorithm was used to update a nonlinear decision boundary of local hyperplanes [15]. However, in multi-class problem, this method adopted one-vs-all strategy to determine a sample class. All incoming data at any time must be retained during the class determining step. Finding the optimal number of prototypes (k) was not easy in order to achieve the best performance in each data set.

For VEBF (Versatile Elliptic Basis Function), Jaiyen *et.al.* proposed a sequential incremental learning based on local-shape function and discard-after-learned concept. However, this method could not be applied to the streaming environment because the initial width of a hyper-ellipsoid was computed using the whole training data set. Besides, the same initial width of each hyper-ellipsoid was pre-defined and applied to all classes without analyzing data distribution of each individual class. This could decrease the classification accuracy when data distribution in each class was different.

For CIL (Class-wise Incremental Learning), Junsawang *et.al.* modified VEBF from sequential incremental learning to chunk incremental learning based on discard-after-learned concept in order to increase the speed of learning and to reduce the effect of an order of incoming datum which VEBF had the problem. They introduced two learning scenarios consisting of static learning and streaming learning. Similarly VEBF, CIL could not be applied to *Stream* directly because the initial width of a hyper-ellipsoid must be determined in advance from the first 20% of total training data. The initial width of each hyper-ellipsoid is pre-defined by the same width as VEBF without analyzing data distribution of each individual class.

For VLLDA (Vector based Local LDA), the k-nearest neighbors algorithm was applied to find k local training data of a testing sample used to classify that sample class. This method was capable of handling high dimensional data and a non-linear classification problem. But it could not be applied to the environment of streaming data because the whole incoming data set must be used for calculating the Euclidean distance in order to find k-nearest neighbors for determining the class of queried data.

1.1 Studied Problems and Constraints

The objectives of our study are to propose a new classifier method to deal with streaming data classification problem and to design a new dissimilarity measure based on distribution of data. Constraints and studied problems imposed on these objectives are the followings.

Constraints:

1. New incoming data gradually flow into the learning process. The memory size is large enough to hold the incoming data.
2. The distribution probability of data set in each class is unknown in advance.
3. The incoming class sequence is unknown.
4. Learned data are assumed to have no class drift or class characteristic change.
5. Incoming data are completely discarded from the learning process after being leaned.
6. The computing memory unit is assumed to be fixed throughout the learning and testing processes.
7. Only one fixed processing unit is deployed for neural learning.

Although the structure of VEBF previously proposed and modified in [11,12] are rather efficient to cope with the discard-after-learn approach in terms of epochs and accuracy, its structure is not consistent with our concept. Both [11, 12] generate and update their generated hyper-ellipsoidal functions to cover all training data after finishing learn while our hyper-ellipsoidal functions are not required covering all of the training data. Thus this study applies the representation structure of captured data based on *versatile hyper-elliptic function* (VHEF) in [16] to each class and updates some parameters based on [11, 12]. However VEBF, CIL, and VHEF still encounter the following problems:

1. During learning process, VHEF needs to compute an inverse of a covariance matrix. If the number of features is greater than the number of training

data (the condition of *curse of dimensionality*), then the covariance matrix becomes singular. This situation does happen in the early stages of learning.

2. Expanding a VEBF or introducing a new VEBF to capture new incoming data as proposed in [11,12] requires a pre-defined threshold distance between incoming data and an existing VEBF. If the threshold distance is too large, then the expanded VEBF may cover some data in other classes possibly entering the learning process in the future. This expansion results in a misclassification.

3. Determining the actual class of queried datum based on the nearest distance between datum and a capturing function is not appropriate. The previous approaches such as [11,12,16] measure this distance by computing the value of capturing function with the datum vector. This value indicates how far datum is from the center of the capturing function. In fact, the nearest distance should be measured from datum to the capturing function by taking distribution of data of the capturing function into account.

The rest of the dissertation is organized as follows. Chapter II summarizes the relevant concept and background used in this study. Chapter III proposes our methodology consisting of discussion on the new structure of capturing function and the equations for updating the parameters of the structure, explanation about the proposed learning steps of the proposed scalable hyper-ellipsoidal function, and presentation on the new nearest distance measure and its concept. Chapter IV gives the experimental results and the comparison of performance evaluation with other methods. Chapter V discusses the rationale behind the results and concludes the dissertation including the limitation of this proposed method.

CHAPTER II

RELEVANT BACKGROUND

Our proposed method is related to the structure of hyper-ellipsoid function in terms of covariance matrix and the concept of linear discriminant analysis. Our learning process mainly develops from VEBF. The summary of related issues are given in the following sections.

2.1 Basic Concept of Standard Hyper-ellipsoid Function

Let $\mathbf{x}_i \in \mathbb{R}^d,\ 1 \leq i \leq N$, be the i^{th} d-dimensional data vector written in the form of column vector. Suppose a set of data vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ belongs to class A. The distribution directions of all vectors in set $\mathbf{X}$ and the variance of data in each direction can be captured using the covariance matrix of set $\mathbf{X}$. This covariance matrix can be easily computed by the following equation. Let $\mathbf{S}$ denote this covariance matrix.

$$\mathbf{S} = \mathbf{E}[(\mathbf{X} - \mathbf{E}[\mathbf{X}])(\mathbf{X} - \mathbf{E}[\mathbf{X}])^T] \tag{2.1}$$

where $\mathbf{E}[\cdot]$ represents the expected value. To realize the concept of *discard-after-learn*, it would be better to compute matrix $\mathbf{S}$ in the form of summation as defined in equation (2.3).

$$\mathbf{c} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{2.2}$$

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{c})(\mathbf{x}_i - \mathbf{c})^T \tag{2.3}$$

where $\mathbf{c} \in \mathbb{R}^d$ is the mean or centroid of data vectors in $\mathbf{X}$. The distribution directions of all data vectors in set $\mathbf{X}$ are the set of eigenvectors of $\mathbf{S}$, denoted by $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d\}$ such that each $||\mathbf{u}_i|| = 1$. The data variances of all eigenvectors are the set of corresponding eigenvalues, denoted by $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_d\}$.

Given a set of data vector $\mathbf{X}$, the standard form of the hyper-ellipsoid function centered at $\mathbf{c}$ can be constructed to capture all vectors in $\mathbf{X}$ by employing the following equation.

$$(\mathbf{x} - \mathbf{c})^T \mathbf{S}^{-1} (\mathbf{x} - \mathbf{c}) = 1. \tag{2.4}$$

2.2 Concept of LDA with Multiple Classes and Binary Classes

Linear Discriminant Analysis (LDA) is a popular supervised dimensionality reduction method to reduce a dimension of an original data set by projecting onto a discriminant space. The basic idea of LDA is to find the suitable discriminant space. When the original data set is projected on that space, projected data can be classified by some criteria for better classifying their classes. Therefore, LDA is an efficient method for solving the high dimensional classification problem. LDA is a data pre-processing, in other words LDA is a step of preparing the original data set before applying them to any classifier such as the nearest neighbor, the decision trees, the neural network, etc. The one of the most popular LDA is Fisher's LDA or another name, Fisher discriminant analysis (FDA). Several researches applied Fisher's LDA to classify their problems such as [6,7,13,14,17,18]. The concept of Fisher's LDA is to find the discriminant space maximizing the distance between projected class means and minimizing the projected within class variance onto that space called Fisher's criterion by Sir Ronald Aylmer Fisher [19].

Suppose $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is a set of N data vectors with K classes. Let $\mathcal{C}_k$ denote class k and n_k be $|\mathcal{C}_k|$, for $1 \leq k \leq K$. The covariance matrix of each class C_k is denoted by $\mathbf{S}_k$. The centroid of $\mathbf{X}$ is at $\mathbf{c}$ and the centroid of each class k is at $\mathbf{c}_k$. The traditional LDA aims to find $(K-1)$ discriminant vectors $\mathbf{w}_i$ formed as a d-by-$(K-1)$ projection matrix $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_{K-1}]$ in order to maximize the following Fisher's criterion [19].

$$\begin{aligned} &\underset{\mathbf{W}}{\text{maximize}} \quad J(\mathbf{W}) := \frac{|\mathbf{W}^T \mathbf{S}_B \mathbf{W}|}{|\mathbf{W}^T \mathbf{S}_W \mathbf{W}|} \\ &\text{subject to} \quad ||\mathbf{w}_i|| = 1 \text{ where } i = 1, ..., K-1. \end{aligned} \tag{2.5}$$

Note that $J(\mathbf{W})$ is the Fisher's criterion. $|\cdot|$ represents the determinant of the matrix. $||\cdot||$ represents Euclidean norm. Between-class scatter matrix $\mathbf{S}_B$ and within-class scatter matrix $\mathbf{S}_W$ are defined as follows:

$$\mathbf{S}_B = \sum_{k=1}^{K} n_k (\mathbf{c}_k - \mathbf{c})(\mathbf{c}_k - \mathbf{c})^T \tag{2.6}$$

$$\mathbf{S}_W = \sum_{k=1}^{K} \mathbf{S}_k. \tag{2.7}$$

The maximization problem in equation (2.5) can be solved by transforming to the generalized eigenvalue problem $\mathbf{S}_W^{-1}\mathbf{S}_B\mathbf{W} = J(\mathbf{W})\mathbf{W}$. In order to solve $\mathbf{W}$, eigenvalues $J(\mathbf{W})$ of $\mathbf{S}_W^{-1}\mathbf{S}_B$ is computed and then choose the eigenvectors $\mathbf{W}$ corresponding to the non-zero eigenvalues.

For a special case, when LDA is only used for a binary class problem (two classes, $K = 2$), the projection matrix $\mathbf{W}$ consists of only one discriminant vector $\mathbf{w}$ of size 1. The value of $\mathbf{w}$ can be simply computed by the following equation. Let $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$. Assume $\mathbf{S}_W$ is invertible, then

$$\mathbf{w} = \frac{\mathbf{S}_W^{-1}(\mathbf{c}_1 - \mathbf{c}_2)}{||\mathbf{S}_W^{-1}(\mathbf{c}_1 - \mathbf{c}_2)||}. \tag{2.8}$$

LDA has an important limitation. If distribution of data in each class is not Gaussian distribution, LDA may not efficiently classify. There are many LDA methods to solve this limitation, overall concepts are called Local LDA. The idea of local LDA is to divide the data set into subgroups, to use k-nearest neighbor method [13], or to get weights of each data depending on their local structures [17] to detect local sample data in order to calculate LDA. The advantages of using local LDA are 1) when the data is non-linear separable, local LDA can handle them with local linear discriminant vectors and 2) it uses only some parts of data for calculating LDA.

Above LDA methods work with a static data that knows the whole data, sizes, and characteristic in advance. Since practical applications need rework for streaming data, several researchers proposed the algorithm of LDA for streaming data as the incremental learning of LDA [5, 7]. The same as the traditional LDA, all incremental learnings of LDA confront the same limitation of LDA.

In our study, a binary class LDA is applied to design a new improvement of measuring the nearness based on local shape-function and used to determine class of testing data based on the new measuring distance. From equation (2.8), centroids and covariance matrices of two class groups are used to find the discriminant vector $\mathbf{w}$ for the binary class. This is consistent with updating the parameters of our proposed method based on the concept of *discard-after-learn* similar to VEBF.

2.3 Concept of Versatile Elliptic Basis Function (VEBF)

The versatile elliptic basis function (VEBF) was introduced by Jaiyen *et.al.* [11] in 2010. They proposed a sequential incremental learning based on local-shape function and discard-after-learn concept. There are two parts of VEBF. The first part is the part of introducing the local-shape function (called neuron) based on covered incoming datum . The second part is the learning part or the part of updating neurons. VEBF proposed a general hyper-ellipsoidal function that can be expanded, shrinked, or rotated according to distribution of the data set in order to cover incoming datum during the learning process. The generalization of the d-dimensional hyper-ellipsoidal function is defined by the following equation.

$$\sum_{i=1}^{d} \frac{((\mathbf{x}-\mathbf{c})^T \mathbf{u}_i)^2}{a_i^2} = 1 \tag{2.9}$$

where $\mathbf{u}_i$ is a i^{th} eigenvector of covariance matrix of $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ for determining a direction of the hyper-ellipsoid. The constant a_i is the width along the direction of the i^{th} eigenvector, $\mathbf{x}$ is a data vector, and $\mathbf{c}$ is a center of $\mathbf{X}$.

From equation (2.9), the versatile elliptic basis function is defined as follows.

$$\psi(\mathbf{x}) = \sum_{i=1}^{d} \frac{((\mathbf{x}-\mathbf{c})^T \mathbf{u}_i)^2}{a_i^2} - 1. \tag{2.10}$$

Since VEBF is the sequential incremental learning, updating the parameters such as mean and variance need a recursive computation. Therefore, a recursive mean and a recursive covariance matrix were introduced. Our proposed method also used these recursive functions to update the parameters.

Due to the new neurons can be automatically generated during learning process, some neurons may be redundant. As the reason above, merging strategy should be included in the learning process. VEBF defined the *merging function* of any two neurons in terms of the distance between centers of two neurons $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$.

$$\phi(\mathbf{c}^{(1)}, \mathbf{c}^{(2)}) = \sum_{i=1}^{d} \frac{\left((\mathbf{c}^{(1)}-\mathbf{c}^{(2)})^T \mathbf{u}_i^{(2)}\right)^2}{(a_i^{(2)})^2} - 1. \tag{2.11}$$

In the learning algorithm of VEBF, two neurons will be merged when they satisfy the merging criterion $\phi(\mathbf{c}^{(1)}, \mathbf{c}^{(2)}) \leq \theta$ or $\phi(\mathbf{c}^{(2)}, \mathbf{c}^{(1)}) \leq \theta$. The constant θ was set to 0 for all experiments in [11]. This means if the center of neuron 1 is inside or on the boundary of neuron 2 or the center of neuron 2 is inside or on the boundary of neuron 1, respectively, then both neurons are merged into the new neuron with the new width based on Gaussian distribution.

2.4 Checking the Intersection of Two Hyper-ellipsoids

Checking the intersection of two hyper-ellipsoids is essential in order to merge two hyper-elliptic structures of the same class into a larger one. This dissertation modified the method of checking touch of two ellipsoids at a single point by Alfano and Greer [20]. Let **A** and **B** be the represented matrices of the first and the second ellipsoids, respectively. Suppose **X** and **Y** are data vectors for the first and the second ellipsoids, respectively. The equations of both ellipsoids can be written as follows.

$$\mathbf{X}\mathbf{A}\mathbf{X}^T = 0 \tag{2.12}$$

$$\mathbf{Y}\mathbf{B}\mathbf{Y}^T = 0. \tag{2.13}$$

Assume that **X** is in both ellipsoids when they intersect each other. Thus, we have

$$\mathbf{X}\mathbf{A}\mathbf{X}^T = 0 \tag{2.14}$$

$$\mathbf{X}\mathbf{B}\mathbf{X}^T = 0. \tag{2.15}$$

Testing overlap of both ellipsoids can be transformed into the process of formulating eigenvalues by these steps. A constant λ is multiplied to matrix **A** in equation (2.14) first.

$$\mathbf{X}(\lambda\mathbf{A})\mathbf{X}^T = 0. \tag{2.16}$$

Then subtract (2.16) and (2.15) to obtain these equations.

$$\mathbf{X}(\lambda\mathbf{A} - \mathbf{B})\mathbf{X}^T = 0 \tag{2.17}$$

$$\mathbf{X}\mathbf{A}(\lambda\mathbf{I} - \mathbf{A}^{-1}\mathbf{B})\mathbf{X}^T = 0. \tag{2.18}$$

Hence, the relation $|\lambda\mathbf{I} - \mathbf{A}^{-1}\mathbf{B}| = 0$ is the condition for testing the intersection of two ellipsoids. $|\cdot|$ represents the determinant of a matrix.

2.5 Distance from a Sample Point to a Hyper-ellipsoid

There are several strategies to measure the closeness between a sample point and a hyper-ellipsoid. A proper measurement is a key factor on a criterion of a decision function for classifying with hyper-ellipsoids. Distance measurements relating with hyper-ellipsoid shape are mentioned as follows. Suppose $\mathbf{x}$ is a sample point in d-dimensional space, $\mathbf{S}$ is a covariance matrix corresponding to a hyper-ellipsoid in d-dimensional space, and $\mathbf{c}$ is a centroid of a hyper-ellipsoid in d-dimensional space. A hyper-ellipsoid is defined in equation (2.4).

2.5.1 Euclidean Distance

A easy way to measure the closeness is to compute the Euclidean distance between a sample point $\mathbf{x}$ and a centroid of a hyper-ellipsoid $\mathbf{c}$ by the following equation.

$$\mathrm{ED}(\mathbf{x}, \mathbf{c}) := ||\mathbf{x} - \mathbf{c}|| := \sqrt{(\mathbf{x} - \mathbf{c})^T(\mathbf{x} - \mathbf{c})} \tag{2.19}$$

where $||\cdot||$ represents Euclidean norm and T represents a transpose.

2.5.2 Mahalanobis Distance

Mahalanobis distance is a distance measure between a sample point $\mathbf{x}$ and distribution of captured data of a hyper-ellipsoid where $\mathbf{S}$ and $\mathbf{c}$ are updated including the new point $\mathbf{x}$. Let $\mathbf{c}_m$ be a new centroid, and $\mathbf{S}_m$ be a new covariance matrix, Mahalanobis distance is defined by the following equation.

$$\mathrm{MD}(\mathbf{x}, \mathbf{c}_m, \mathbf{S}_m) := \sqrt{(\mathbf{x} - \mathbf{c}_m)^T \mathbf{S}_m^{-1} (\mathbf{x} - \mathbf{c}_m)} \tag{2.20}$$

where $\mathbf{S}_m^{-1}$ represents the inverse of $\mathbf{S}_m$.

2.5.3 The Versatile Elliptic Basis Function Value

VEBF [11] and CIL [12] use their shape-function value as a decision function to measure the closeness between a sample point to VEBF. Their versatile elliptic basis function of the k^{th} neuron is defined as follows.

$$\psi_k(\mathbf{x}) = \sum_{i=1}^{d} \frac{((\mathbf{x} - \mathbf{c})^T \mathbf{u}_i)^2}{a_i^2} - 1 \tag{2.21}$$

where $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d\}$ are eigenvectors of a covariance matrix of covered data and a_i is the width of each axis of VEBF.

2.5.4 Boundary Distance

Measuring the distance with respect to the boundary of the standard form of a hyper-ellipsoid is rather complex, hence an approximate distance between a sample point to the boundary of the standard hyper-ellipsoid was proposed.

In 2005, Zimmermann and Svoboda [21] proposed the approximate distance between the sample point to the nearest boundary of an ellipse, it also can be applied to a high dimensional space or a hyper-ellipsoid. This distance is measured on the line connecting the sample point and the centroid. The line intersects the boundary of the ellipse at a specific point. The actual distance is measured from the intersection point to the sample point. Instead of using the ellipse,

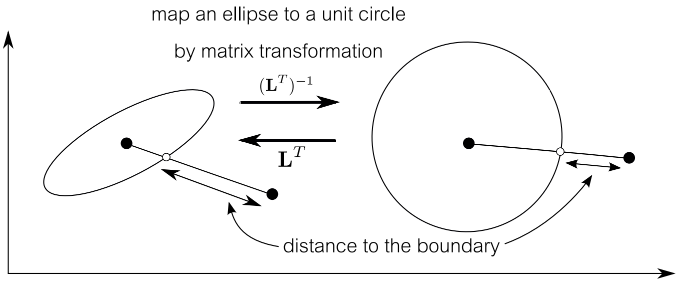


Figure 2.1: Approximate distance from a point to an ellipse by Zimmermann and Svoboda.

Zimmermann and Svoboda transformed the shape of ellipse as a unit circle by matrix transformation which is much simpler as shown in Figure 2.1. The concept of this strategy is described as follows. First, the original ellipse and the sample point are transformed by the inverse of the matrix $\mathbf{L}^T$ become a unit circle where this matrix is obtained by factorizing the covariance matrix representing an ellipse with a Cholesky factorization ($\mathbf{S} = \mathbf{L}\mathbf{L}^T$). The distance of the sample point to the unit circle is easy to compute using the Euclidean distance. After that, the distance value will be retransformed to the original shape using the matrix $\mathbf{L}^T$. Approximate boundary distance according to the above concept is defined by the following equation.

$$\mathrm{BD}_1(\mathbf{x}, \mathbf{c}, \mathbf{S}) := ||(\mathbf{x} - \mathbf{c}) - \frac{(\mathbf{x} - \mathbf{c})}{||(\mathbf{L}^T)^{-1}(\mathbf{x} - \mathbf{c})||}|| \tag{2.22}$$

where $(\cdot)^{-1}$ represents the inverse of the matrix and $|| \cdot ||$ represents Euclidean norm.

In 2017, Wattanakitrungroj *et al.* [16] also proposed a method to compute the distance between the boundary of full micro-cluster and a data point by solving equations following the concept in Figure 2.2. Although both methods [16,21] deploy different definitions of ellipsoid, they end up with the same distance approximation. However, [16] takes less computation time and calculation steps

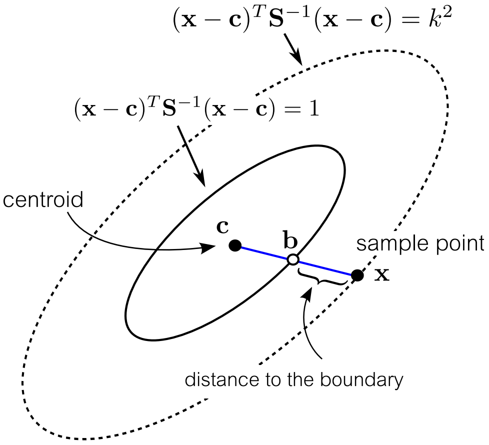

$(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c}) = k^2$
$(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c}) = 1$
centroid
$\mathbf{c}$
$\mathbf{b}$
sample point
$\mathbf{x}$
distance to the boundary

Figure 2.2: Approximate distance from a point to an ellipse by Wattanakitrungroj *et.al.*

than [21]. Approximate boundary distance according to [16] is defined by the following equation.

$$\mathrm{BD}_2(\mathbf{x}, \mathbf{c}, \mathbf{S}) := ||\mathbf{x} - \mathbf{c}||(1 - \frac{1}{\sqrt{(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})}}). \tag{2.23}$$

In the next Chapter, these knowledge backgrounds will be adapted to our proposed method either the part of the learning process or the part of the testing process.

CHAPTER III

METHODOLOGY

The new structure of capturing function and the equations for updating the parameters were introduced in this Chapter. The proposed learning algorithm, its time complexity, and its illustrations in two-dimensional space were provided in the part of the learning process. In the part of the testing process, the new distance measure based on LDA discriminant vector and its concept were proposed. Determining the classes of our method was also based on our new distance measure. The details of our proposed method are given in the following sections.

3.1 Proposed Structure of Capturing Function and Parameter Updating

To cope with the problems previously addressed, a new structure of capturing function and its parameter updating process were proposed. The new structure processes data faster than those structures used in discard-after-learn approaches in [11, 12]. The details of these issues are the followings.

3.1.1 Structure of Scalable Hyper-Ellipsoidal Function

Expanding or shrinking the size of the previously proposed structure of standard hyper-ellipsoid function requires the computations of eigenvectors and eigenvalues first. To reduce this prior computations, the following generic form of standard hyper-ellipsoid function was used in our approach. Instead of setting the right-hand side of standard hyper-ellipsoid function to a constant of one, this constant is replaced by a positive r. This constant r makes the structure of stan-

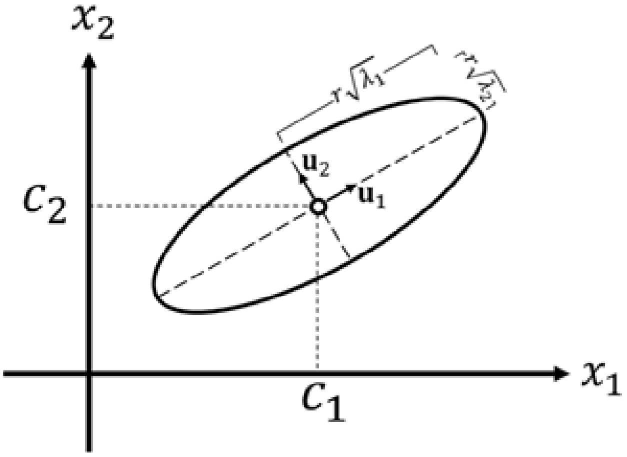


Figure 3.1: Relationship between eigenvalues and eigenvectors of the covariance matrix to size and shape of hyper-ellipsoid in two-dimensional, respectively.

dard hyper-ellipsoid function easy to be scaled. The equation of this new scalable hyper-ellipsoidal function (SHEF) is defined as follows.

$$(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c}) = r^2 \tag{3.1}$$

where $\mathbf{x} \in \mathbf{X}$ is a data vector, $\mathbf{c}$ is the center of $\mathbf{X}$, and $\mathbf{S}$ is the covariance matrix of $\mathbf{X}$.

The sizes of semi-axis in each dimension can be assigned by $r\sqrt{\lambda_i}$. In other words, the sizes of semi-axis is set to be the scale of the standard deviation of the data along the directions of each eigenvector as illustrated in Figure 3.1.

In several applications, the covariance matrix $\mathbf{S}$ is singular. To avoid this condition, the concept of regularization [18,22] was adapted to the scalable hyper-ellipsoidal function by adding a small positive constant ϵ to the covariance matrix $\mathbf{S}$ as shown in the following equation.

$$(\mathbf{x}-\mathbf{c})^T(\mathbf{S}+\epsilon\mathbf{I})^{-1}(\mathbf{x}-\mathbf{c}) = r^2. \tag{3.2}$$

I is a *d*-by-*d* identity matrix.

Lemma 1. *Let* $\mathbf{S}$ *and* $\mathbf{S}^*$ *be two covariance matrices such that* $\mathbf{S}^* = \mathbf{S} + \epsilon\mathbf{I}$*. Covariance matrix* $\mathbf{S}^*$ *has the same set of eigenvectors as those of* $\mathbf{S}$ *and each eigenvalue* $\lambda_i^* = \lambda_i + \epsilon$*.*

Proof: The covariance matrix $\mathbf{S}$ can be factorized in the term of $\mathbf{U}$ and Λ as follows:

$$\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^T. \tag{3.3}$$

Substitute equation (3.3) into $\mathbf{S}^* = \mathbf{S} + \epsilon\mathbf{I}$, we obtain the following equation.

$$\mathbf{S}^* = \mathbf{U}\Lambda\mathbf{U}^T + \epsilon\mathbf{I}. \tag{3.4}$$

Since $\mathbf{U}$ is an orthogonal matrix, so $\mathbf{U}\mathbf{U}^T = \mathbf{I}$. Hence,

$$\begin{aligned} \mathbf{S}^* &= \mathbf{U}\Lambda\mathbf{U}^T + (\epsilon\mathbf{I})\mathbf{U}\mathbf{U}^T \\ &= \mathbf{U}\Lambda\mathbf{U}^T + \mathbf{U}(\epsilon\mathbf{I})\mathbf{U}^T \\ &= \mathbf{U}(\Lambda + \epsilon\mathbf{I})\mathbf{U}^T \end{aligned} \tag{3.5}$$

and

$$\begin{aligned} \Lambda^* = \Lambda + \epsilon\mathbf{I} &= \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix} + \epsilon \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 + \epsilon & & \\ & \ddots & \\ & & \lambda_d + \epsilon \end{bmatrix}. \qquad \square \end{aligned} \tag{3.6}$$

In case of zero covariance matrix (or there is only one datum in SHEF), $\mathbf{S}$ becomes singular. Hence, the initial width of SHEF in each dimension was set to $\sqrt{\epsilon}$ instead.

3.1.2 Updating Parameters of SHEF

Each SHEF contains four parameters: the number of captured data (n), a centroid of captured data ($\mathbf{c}$), a covariance matrix of captured data ($\mathbf{S}$), and a class of captured data (z). Since the training process is based on the concept of *discard-after-learn*. Incoming datum will be discarded after being captured by any SHEF, so the first three parameters of that SHEF must be updated accordingly to recently incoming data.

Assume that incoming datum $\mathbf{x}^{new} \in \mathbb{R}^d$ is captured by the j^{th} SHEF of the same class. Let n_j^{old}, $\mathbf{c}_j^{old}$, $\mathbf{c}_j^{new}$, $\mathbf{S}_j^{old}$, and $\mathbf{S}_j^{new}$ be the current number of data points, the current centroid, the updated centroid, the current covariance matrix, and the updated covariance matrix, respectively. To cope with the possibility of data overflow and to preserve the time and space complexities when employing the concept of *discard-after-learn*, the following set of recursive functions for computing new centriod and covariance matrix were previously proposed in [11, 12, 16].

$$\mathbf{c}_j^{new} = \frac{n_j^{old}\mathbf{c}_j^{old} + \mathbf{x}^{new}}{n_j^{old} + 1} \tag{3.7}$$

$$\mathbf{S}_j^{new} = \frac{n_j^{old}\left(\mathbf{S}_j^{old} + \mathbf{c}_j^{old}(\mathbf{c}_j^{old})^T\right) + \mathbf{x}^{new}(\mathbf{x}^{new})^T}{n_j^{old} + 1} - \mathbf{c}_j^{new}(\mathbf{c}_j^{new})^T. \tag{3.8}$$

Although these recursive functions efficiently support the concept of *discard-after-learn*, it is possible to speed up the updating process of covariance matrix by rewriting equation (3.8) as stated in the following theorem.

Theorem 3.1.1. A new covaraince matrix $\mathbf{S}_j^{new}$ can be computed by the following recursive function.

$$\mathbf{S}_j^{new} = \frac{n_j^{old}}{n_j^{old}+1}\left(\mathbf{S}_j^{old} + \frac{(\mathbf{c}_j^{old} - \mathbf{x}^{new})(\mathbf{c}_j^{old} - \mathbf{x}^{new})^T}{n_j^{old}+1}\right). \tag{3.9}$$

The proof of equation (3.9) is given in Appendix A. Note that the time spent on computing $\mathbf{c}_j^{old}(\mathbf{c}_j^{old})^T$, $\mathbf{x}^{new}(\mathbf{x}^{new})^T$, and $\mathbf{c}_j^{new}(\mathbf{c}_j^{new})^T$ in equation (3.8) is reduced by computing only $(\mathbf{c}_j^{old} - \mathbf{x}^{new})(\mathbf{c}_j^{old} - \mathbf{x}^{new})^T$ instead in equation (3.9). Although **Theorem 3.1.1** addresses only one incoming datum, equation (3.9) can be adapted to an incoming data chunk by updating the covariance matrix with one datum at a time.

3.2 New Learning Method Using Scalable Hyper-ellipsoidal Function

The streaming data flow into the learning process in one multi-class chunk at a time. Let $\Omega = (\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots)$ be the sequence of the streaming data chunk $X^{(t)}$ at different time t. Each $\mathbf{X}^{(t)} = ((\mathbf{x}_1^{(t)}, y_1^{(t)}), \cdots, (\mathbf{x}_{N_t}^{(t)}, y_{N_t}^{(t)}))$ consists of a set of N_t pairs of datum $\mathbf{x}_i^{(t)}$ and its target class $y_i^{(t)}$. The capturing process focuses one datum at a time with the following main steps. Assume that class $y_i^{(t)} = k$ is being considered.

1. Capturing an incoming data chunk by introducing a new SHEF or by expanding some existing SHEF of the same class k. The criteria for performing each operation depend upon: (1) the minimum distance and median distance among data within each class and (2) an adaptive threshold distance based on the number of SHEFs of the same class k and the amount of data in each SHEF of class k.

2. Merging two SHEFs of the same class k into one larger SHEF to reduce

the number of SHEFs of class k. The merging criteria are based on degree of overlap between two nearest SHEFs of the same class k using Euclidean distance from a centroid to other centroids.

Prior to the learning algorithm based on these two main steps, the computational detail in each step is discussed first in the following sections.

3.2.1 Initializing SHEF Widths and Threshold Distance for Introducing New SHEF

At the starting step of learning process, the initial size of the first SHEF for capturing the first data chunk of a class, say class k, must be defined. If the class has only one datum, then a constant ϵ as introduced in **Lemma 1** is deployed as the initial width of SHEF in all dimensions. Otherwise the width of SHEF in each dimension is computed by the following equation. Suppose $\mathbf{x}_a^{(1)}$ is in class k and the amount of data in this class is n_k. Let $dist(\mathbf{x}_a^{(1)})$ be the Euclidean distance from $\mathbf{x}_a^{(1)}$ to its nearest neighbour of the same class in the first incoming chunk. The initial width, denoted as $dist\_init_k$, of SHEF in class k is set up as follows.

$$dist\_init_k = \begin{cases} \underset{\mathbf{x}_j^{(1)} \in \text{ class } k}{median} \; (dist(\mathbf{x}_j^{(1)}) & n_k > 1 \\ \sqrt{\epsilon} \text{ in Lemma 1} & n_k = 1 \end{cases} \tag{3.10}$$

Note that this initial width is used as the initial value for each axis i of SHEF in class k. The initial widths of all new classes appearing after the first chunk were set to $\sqrt{\epsilon}$.

The value of threshold distance is used to determine whether a new SHEF of the same class should be introduced to capture new incoming datum or not. This threshold distance is used to control the number of SHEFs generated during the learning process. If there are too many SHEFs, then the *over-fit* problem is occurred and the computational time obviously is increased. But if there are too few SHEFs, then the misclassification of queried data may be imminent. The distance concerns two factors. The first factor is the amount of data in each SHEF of the same class. The second factor is the number of existing SHEFs of the same class. A merging threshold distance is defined based on these two factors as shown in the following paragraph.

Let M be the predefined minimum amount data allowed within each SHEF of class k. Suppose there are m_k SHEFs whose amount of data in each SHEF is less than M. The threshold distance of class k, denoted as $dist\_ths_k$, is defined as follows. Note that $dist\_ths_k = dist\_init_k$ in the first chunk.

$$dist\_ths_k = \begin{cases} dist\_ths_k & \text{if } m_k \leq \text{half of} \\ & \text{existing SHEFs in class } k \\ 2 \times dist\_ths_k & \text{if } m_k > \text{half of} \\ & \text{existing SHEFs in class } k \end{cases} \tag{3.11}$$

From equation (3.11), its concept is that if there exist many inefficiently generated SHEFs (each SHEF captured data less than M), the threshold distance having an effect to an introducing new SHEF condition should be scaled up.

3.2.2 Condition of Intersection of Two Scalable Hyper-Ellipsoids

The structure of scalable hyper-ellipsoids in this study is different from the structure studied by Alfano and Greer [20]. Their structure is based on the standard elliptic function, where the right-hand side of elliptic equation is set to zero but SHEF employs a scaling constant r^2 as defined in equation (3.1) instead. However, their technique of deriving the intersecting condition was adapted to our scenario. Suppose two SHEFs, SHEF$_\alpha$ and SHEF$_\beta$ intersect. The covariance matrix of SHEF can be computed from the covariance matrix of captured data vectors by the following steps. Let $\widetilde{\mathbf{S}}$ be the covariance matrix of SHEF and $\mathbf{S}$ be the covariance matrix of captured data vectors. Both $\widetilde{\mathbf{S}}$ and $\mathbf{S}$ are computed from the same data set. From equation (3.3), we have

$$\begin{aligned}
\widetilde{\mathbf{S}} &= \mathbf{U}\widetilde{\Lambda}\mathbf{U}^T && (3.12)\\
&= \mathbf{U}\begin{bmatrix} \tilde{\lambda}_1 & & \\ & \ddots & \\ & & \tilde{\lambda}_d \end{bmatrix}\mathbf{U}^T && (3.13)\\
&= \mathbf{U}\begin{bmatrix} (r\sqrt{\lambda_1})^2 & & \\ & \ddots & \\ & & (r\sqrt{\lambda_d})^2 \end{bmatrix}\mathbf{U}^T && (3.14)\\
&= r^2\mathbf{U}\Lambda\mathbf{U}^T && (3.15)\\
&= r^2\mathbf{S} && (3.16)
\end{aligned}$$

Thus,

$$\widetilde{\mathbf{S}}^{-1} = \frac{1}{r^2}\mathbf{S}^{-1}. \tag{3.17}$$

The following theorem states the conditions of intersection of two scalable hyper-ellipsoids modified Alfano and Greer's method [20].

Theorem 3.2.1. Both of SHEF$_\alpha$ and SHEF$_\beta$ do not overlap each other if all eigenvalues of the following matrix $\mathbf{P}$ are distinct and all real numbers with some negative values, otherwise they either overlap, are inside, or touch.

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{D} & -\mathbf{D}\mathbf{c}_\beta + \mathbf{c}_\alpha \\ \hline \mathbf{F} & -\mathbf{F}\mathbf{c}_\beta + 1 \end{array}\right], \tag{3.18}$$

where $\mathbf{F} = (-\mathbf{c}_\alpha^T + \mathbf{c}_\beta^T)\widetilde{\mathbf{S}}_\beta^{-1}$ and $\mathbf{D} = \widetilde{\mathbf{S}}_\alpha \widetilde{\mathbf{S}}_\beta^{-1} + \mathbf{c}_\alpha \mathbf{F}$. Centroids $\mathbf{c}_\alpha$ and $\mathbf{c}_\beta$ are of SHEF$_\alpha$ and SHEF$_\beta$, respectively.

The proof of modifying in **Theorem 3.2.1** is given in Appendix B. If two SHEFs of the same class satisfy the conditions in **Theorem 3.2.1**, then both of them are merged into a larger SHEF$_\gamma$ and all relevant parameters are updated using the following equations.

$$n_\gamma = n_\alpha + n_\beta \tag{3.19}$$

$$\mathbf{c}_\gamma = \frac{n_\alpha \mathbf{c}_\alpha + n_\beta \mathbf{c}_\beta}{n_\gamma} \tag{3.20}$$

$$\mathbf{S}_\gamma = \frac{1}{n_\gamma}\left(n_\alpha \mathbf{S}_\alpha + n_\beta \mathbf{S}_\beta + \frac{n_\alpha n_\beta}{n_\gamma}(\mathbf{c}_\alpha - \mathbf{c}_\beta)(\mathbf{c}_\alpha - \mathbf{c}_\beta)^T\right) \tag{3.21}$$

3.2.3 Learning Algorithm of SHEF

The learning process of SHEF consists of three main procedures. The first procedure is initializing the width of the first SHEF based on the condition stated in Section 3.2.1 and equation (3.10). The second procedure is checking the condition for introducing a new SHEF to capture new incoming data based on the threshold distance discussed in Section 3.2.1 and defined in equation (3.11). The last procedure is merging two SHEFs of the same class according to the overlap constraints in Section 3.2.2 and **Theorem 3.2.1** using equations (3.19), (3.20), and (3.21). The detail of learning algorithm is given in the next page.

Algorithm 1 Learning procedure of SHEF for current incoming data chunk

Input: (1) a set of N pairs of datum and target
$\mathbf{X} = ((\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N))$ for incoming data chunk at any time.
(2) a constant ϵ.
(3) a set of SHEFs from previous learning procedure
(if the incoming chunk is not the first chunk.)
(4) a constant M denoting the minimum of data in any SHEF.
Output: a set of SHEFs and their updated parameters.

```
If the first data chunk then
  Initialize $dist\_ths_{y_j} = dist\_init_{y_j}$ using equation (3.10)
  for every class $y_j$ in chunk $\mathbf{X}$.
EndIf
For each pair $(\mathbf{x}_i, y_i)) \in \mathbf{X}$ do
    If there exists a set of SHEFs of class $y_i$ then
      Let $\mathbf{c}_j$, $n_j$, and $\mathbf{S}_j$ be the centroid, amount data,
      and covariance matrix of captured data of SHEF$_j$
      of class $y_i$, respectively.
      Let $\xi = \arg\min_j(||\mathbf{x}_i - \mathbf{c}_j||)$.
      If $||\mathbf{x}_i - \mathbf{c}_\xi|| > dist\_ths_{y_i}$ then
        Introduce a new SHEF of class $y_i$ to capture
        $\mathbf{x}_i$ and update $dist\_ths_{y_i}$ using equation (3.11).
      else
        Put $\mathbf{x}_i$ in SHEF$_\xi$ and update parameters
        using equations (3.7) and (3.9). Update $n_j = n_j + 1$.
      EndIf
      Discard pair $(\mathbf{x}_i, y_i)$ from the training set.
      Let SHEF$_\alpha$ be the SHEF capturing $\mathbf{x}_i$.
      If $n_\alpha \geq M$ then
        Deploy the conditions in Theorem 3.2.1 to test
        overlap SHEF$_\alpha$ and the nearest SHEF$_\beta$ of class $y_i$.
        If SHEF$_\beta$ overlaps SHEF$_\alpha$ then
          Merge SHEF$_\alpha$ and SHEF$_\beta$ into a larger SHEF$_\gamma$.
          Update parameters of SHEF$_\gamma$ using equations
          (3.19), (3.20), and (3.21).
        EndIf
      EndIf
    else
      Introduce a new SHEF of class $y_i$ to capture $\mathbf{x}_i$
      and update $dist\_ths_{y_i}$ using equation (3.11).
      Discard pair $(\mathbf{x}_i, y_i)$ from the training set.
    EndIf
EndFor
```

3.2.4 Time Complexity of the Learning Algorithm

There are one step for initialization the width of SHEF and two main steps for learning algorithm using SHEF. The worst case time complexity of our learning algorithm can be analyzed as follows. Assume that there is only one class in the first chunk of streaming data with N_1 d-dimensional data points. Initialization the width consists of finding the nearest Euclidean distance of each data points and finding the median of them, so the time complexity of this step is $O(dN_1^2)$. In learning step, the time complexity was analyzed as one datum at a time. Assume that class of the training datum is k. Let h is the number of existing SHEFs of class k. The time complexity of line 7 is $O(dh)$. The worst case of checking the condition of line 8 is it always introduces the new SHEF (line 9), the time complexity of line 9 is dominated by the term of updating the threshold distance using equation (3.11). Therefore, the time complexity of this step is $O(h)$. The worst case of merging strategy in lines 15 - 21 is it always satisfies the condition of line 15 and line 17. The time complexity of this step depends on the dimension of the data set due to the computation of the inverse of covariance matrix and to find its eigenvalues. Thus, the time complexity of merging step is $O(d^3)$. Hence, the time complexity of **Algorithm 1** is $O(dN_1)+[O(dh)+O(h)+O(d^3)] = O(dN_1)+[O(dh)+O(d^3)]$.

Assume that there are N training data points, $N_1 << N$, $d << N$, and $h << N$. So the time complexity becomes $O(dN_1) + \sum_{i=1}^{N}[O(dh) + O(d^3)] = O(dN_1) + O(dhN) + O(d^3N) = O(N)$.

3.2.5 The Illustration of Learning Algorithm in Two-dimensional

The example of learning algorithm of SHEF according to **Algorithm 1** is illustrated in a two-dimensional space in order to explain how the algorithm works. Suppose there are 11 samples in the first chunk of streaming data separated into two classes denoted by five blue circles (class 1) and six red squares (class 2) as shown in Figure 3.2. Assume M is set to 2. First, the initial width of SHEF in each class is calculated using equation (3.10). Figure 3.3 and Figure 3.4 show the first and second training data of the first chunk with an initial width of class 1 and class 2, respectively. The third training data with class 1 is fed, see Figure 3.5, and finds the nearest centroid of the same class checking the criteria for introducing a new SHEF or expanding some existing SHEF. If the Euclidean distance between the nearest centroid of SHEF and the incoming datum of the same class is less than or equal to the threshold distance as shown in Figure 3.5, that SHEF will be updated with that incoming datum. As in the example, the computed distance is greater than the threshold distance, the new group is introduced following Figure 3.6. The new SHEFs including updating their parameters of the next incoming datum in the first chunk are illustrated in Figure 3.7 - Figure 3.9. The captured data into any SHEF will be discarded forever denoted by a shape with dashed boundary. Finally, four SHEFs are genearted after learning all of the first chunk as shown in Figure 3.10. New training data in the second chunk is fed and updated parameters of the capturing SHEF as shown in Figure 3.11 and Figure 3.12, respectively. Notice that there exists two SHEF overlapping each other, thus those SHEFs will be merged into the new larger SHEF to reduce the number of SHEFs as shown in Figure 3.13.

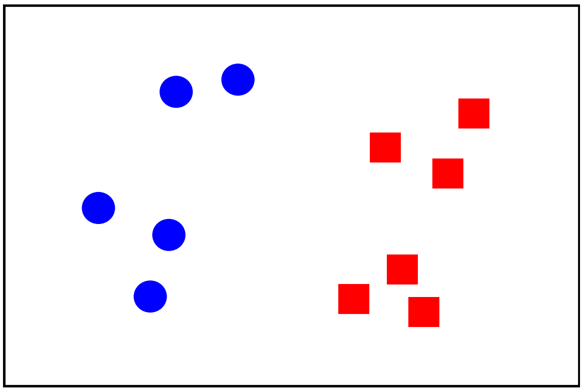


Figure 3.2: The data set in the first chunk for calculating the initial width in each class.

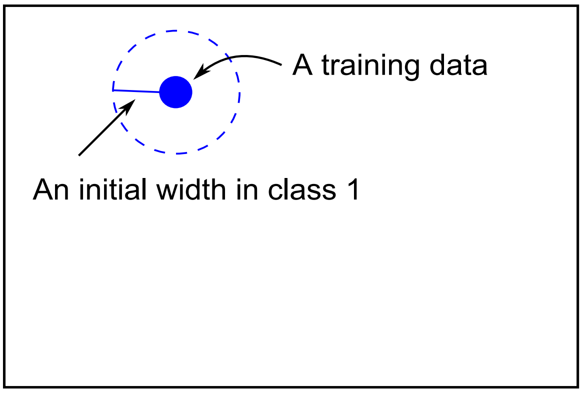


Figure 3.3: Feed the first training data with class 1 (blue circle).

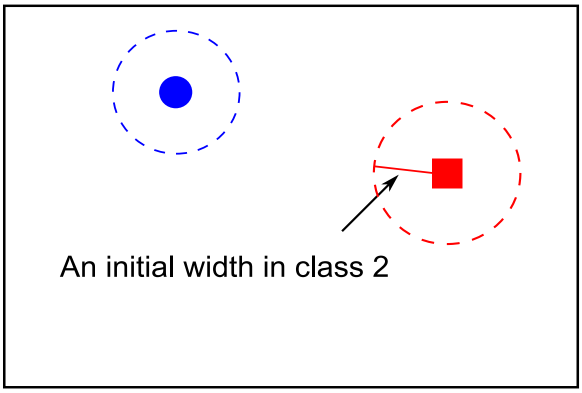


Figure 3.4: Feed the second training data with class 2 (red square).

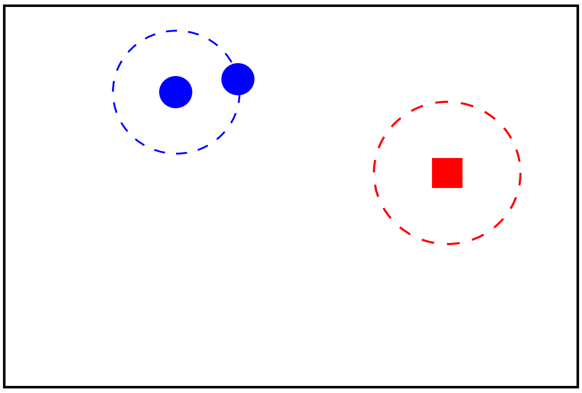


Figure 3.5: Feed the third training data with class 1.

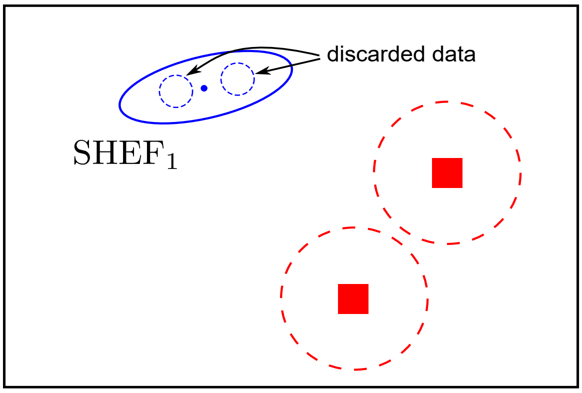


Figure 3.6: Introduce a new SHEF then discard the captured data and feed the forth training data with class 2.

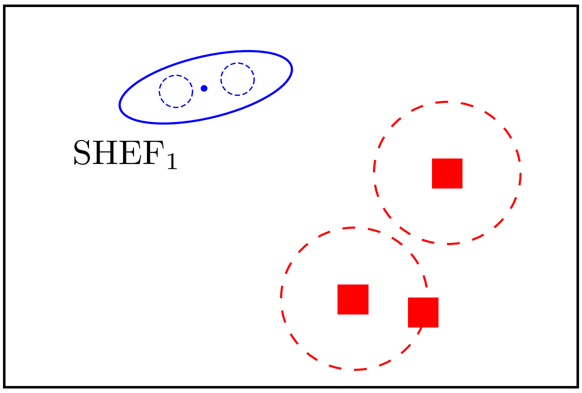


Figure 3.7: Feed the fifth training data with class 2.

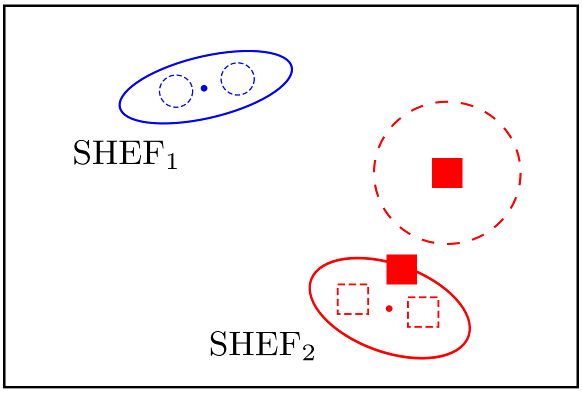


Figure 3.8: Introduce a new SHEF then discard the captured data and feed the sixth training data with class 2.

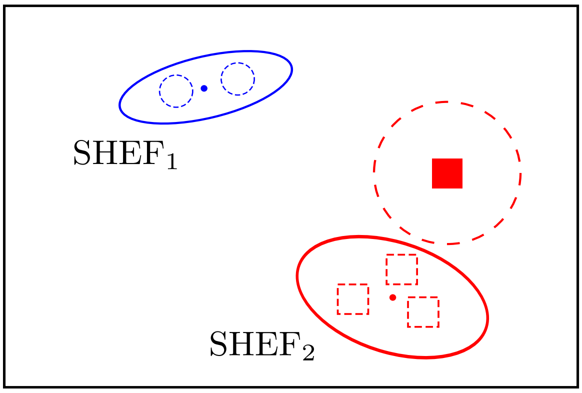


Figure 3.9: Update parameters of SHEF$_2$.

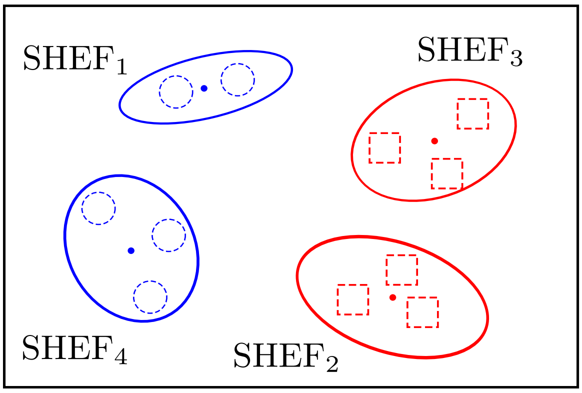


Figure 3.10: Four SHEFs of the first chunk

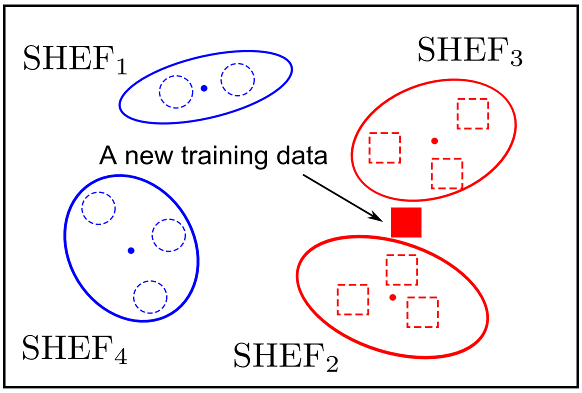


Figure 3.11: Feed a new training data with class 2.

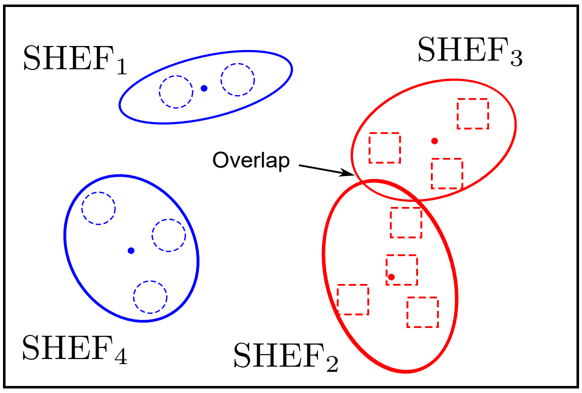


Figure 3.12: Update parameters of $SHEF_2$.

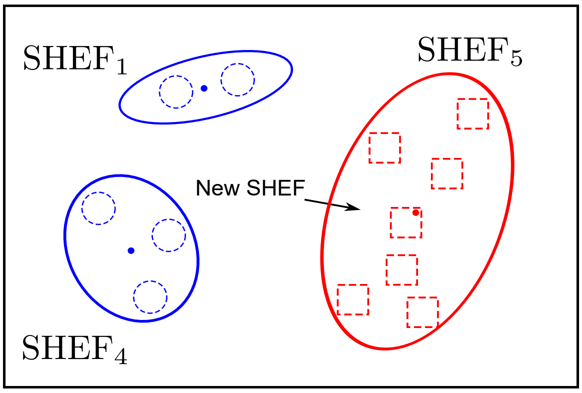


Figure 3.13: Merge two overlapping SHEF into the new SHEF.

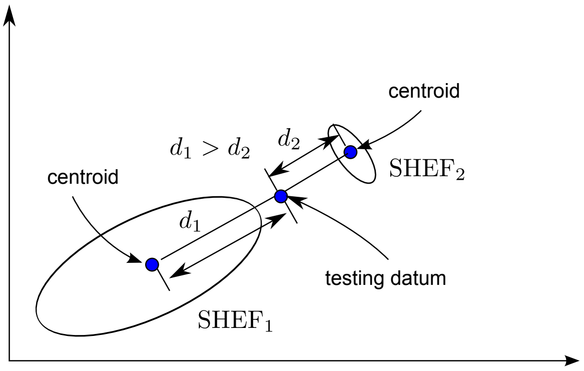


Figure 3.14: An example of wrong interpretation of closeness between testing datum and two SHEFs. The closeness is determined by measuring Euclidean distance from the datum to the centroids of both SHEFs.

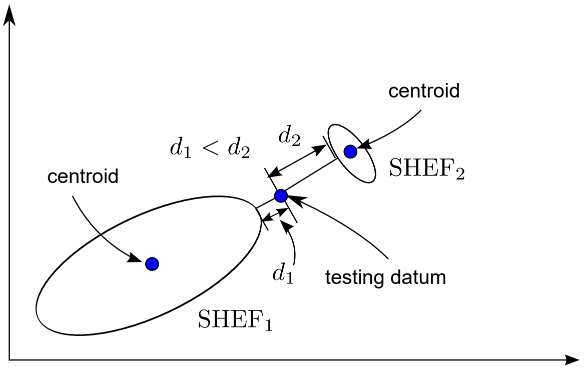


Figure 3.15: The closeness distances measured with respect to the boundary of SHEF$_1$ and SHEF$_2$. The datum is close to SHEF$_1$ instead of SHEF$_2$.

3.3 Identifying Classes of Testing Data

The class of testing datum is identified according to the class of two closest SHEFs. Usually, the closeness between testing datum and SHEF can be easily determined in terms of Euclidean distance either from datum to the centroid or from datum to the boundary of SHEF. But the simplicity may lead to the wrong interpretation. Figure 3.14 shows an example of the Euclidean distances from testing datum to the centroids of SHEF$_1$ and SHEF$_2$. The distance from datum to the centroid of SHEF$_1$ is longer the distance to the centroid of SHEF$_2$. Thus, datum must be assigned to SHEF$_2$ instead of SHEF$_1$. But in fact, the correct class of datum in this example is the class of SHEF$_1$ because it is closer to SHEF$_1$ than SHEF$_2$. To achieve the correct class identification, the distance must be measured with respect to the boundaries of SHEF$_1$ and SHEF$_2$ as shown in Figure 3.15. However, measuring the distance with respect to the boundary of SHEF in a high dimensional space is rather complex. Recently, there were several attempts to measure the closest distance to the boundary of a hyper-ellipsoid. Zimmermann and Svoboda [21] and Wattanakitrungroj *et al.* [16] proposed the approximate distance between a sample point to the boundary of an ellipse as mentioned before in Chapter 2, Section 2.5.4. However, this approximate distance measure only focuses on the shape of hyper-ellipsoid without taking into account distribution of data which is different from our purpose. Thus, a new distance measure was proposed in this study.

3.3.1 Projecting Width of SHEF onto Discriminant Vector

Instead of using the direct distance between testing datum to the boundary of SHEF, our closeness distance is defined as the distance between the projected SHEF and the data point onto the linear-discriminant-analysis (LDA) vector derived from the close SHEF and testing datum. The projection of SHEF boundary and its center onto the LDA vector can be computed as follows.

Let $\mathbf{w}$ be the discriminant vector. According to the concept of LDA as described in equation (2.5), there must be two classes of data given first. Then, the discriminant vector $\mathbf{w}$ is computed using the cost function in equation (2.5). To explain our proposed closeness distance, it is assumed that $\mathbf{w}$ is already computed and SHEF in one class is projected onto this $\mathbf{w}$ as shown in Figure 3.16. Suppose points A and B are the projected boundary points of SHEF onto $\mathbf{w}$. Centroid $\mathbf{c}$

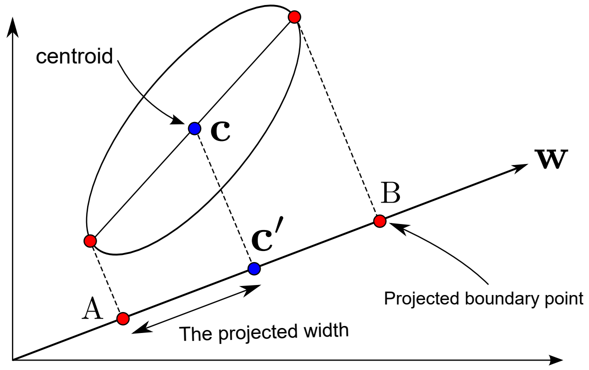


Figure 3.16: Projection of a SHEF onto the discriminant vector $\mathbf{w}$ in a 2-dimensional space.

is projected onto $\mathbf{w}$ at position $\mathbf{c}'$ using equation (3.22).

$$\mathbf{c}' = \mathbf{w}^T\mathbf{c}. \tag{3.22}$$

Note that $||A-\mathbf{c}'||$ and $||B-\mathbf{c}'||$ are equal and they can be defined as the projected width of SHEF onto the discriminant vector $\mathbf{w}$. Actually, the projected width can be computed without knowing the locations of points A and B. In each eigenvector of SHEF, the width of SHEF is equal to $r\sqrt{\lambda_i}$. This value is actually the eigenvalue of data along the eigenvector computed from the covariance matrix $\widetilde{\mathbf{S}}$ of SHEF as in equation (3.12). Thus, the projected width of $||A-\mathbf{c}'||$ and $||B-\mathbf{c}'||$ can be computed by the following equation.

$$||A-\mathbf{c}'|| = ||B-\mathbf{c}'|| = \sqrt{\mathbf{w}^T\widetilde{\mathbf{S}}\mathbf{w}} = r\sqrt{\mathbf{w}^T\mathbf{S}\mathbf{w}}, \tag{3.23}$$

where $\mathbf{w}^T\mathbf{S}\mathbf{w}$ is a non-negative scalar. $||\cdot||$ represents Euclidean norm. Note that, our projection method is similar to Pope's idea [23] using the different definition of hyper-ellipsoid function.

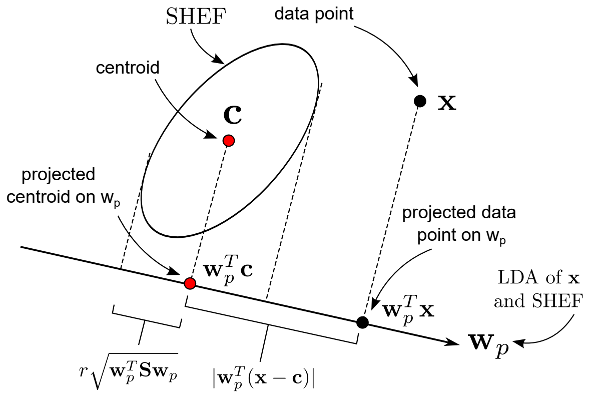

SHEF
data point
centroid
$\mathbf{c}$
$\mathbf{x}$
projected centroid on w$_p$
projected data point on w$_p$
$\mathbf{w}_p^T\mathbf{c}$
$\mathbf{w}_p^T\mathbf{x}$
LDA of $\mathbf{x}$ and SHEF
$\mathbf{w}_p$
$r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$
$|\mathbf{w}_p^T(\mathbf{x}-\mathbf{c})|$

Figure 3.17: The concept for measuring distance from $\mathbf{x}$ to a SHEF by the ratio of $|\mathbf{w}_p^T(\mathbf{x}-\mathbf{c})|$ and $r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$ along the discriminant vector $\mathbf{w}_p$ defined in equation (3.24).

3.3.2 Measuring Distance from a Point to SHEF along Discriminant Vector

A new distance measured from queried datum to SHEF along the discriminant vector was proposed. Based on the concept of LDA, a discriminant vector must be derived from two classes of data to maximize the degree of overlap between projected data in both classes onto the discriminant vector. Moreover, LDA assumes that each class must have more than one datum to make the variance greater than zero. Obviously, the discriminant vector of LDA cannot be directly adapted to our new proposed distance measure because queried datum is just a single point. To solve this problem, the following concept was then developed.

Let $\mathbf{S}$ and $\mathbf{c}$ be the covariance matrix and the centroid of captured data of SHEF, respectively. Suppose $\mathbf{x}$ is a single queried data point. In order to find the discriminant vector for both SHEF and $\mathbf{x}$, data point $\mathbf{x}$ is reconsidered as the centroid of another SHEF. Using equation (2.8) with this scenario, the discriminant vector can be derived as follows.

$$\mathbf{w}_p = \frac{\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})}{||\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})||}. \tag{3.24}$$

Figure 3.17 shows an example of discriminant vector $\mathbf{w}_p$ and the projected width of SHEF as well as the projected location of data point $\mathbf{x}$ onto $\mathbf{w}_p$. There are two significant distances to be used. The first distance $|\mathbf{w}_p^T(\mathbf{x} - \mathbf{c})|$ is the projected distance from the projected centroid to the projected location of $\mathbf{x}$. The second distance $r\sqrt{\mathbf{w}_p^T \mathbf{S} \mathbf{w}_p}$ is the projected width of SHEF. The new distance from data point $\mathbf{x}$ to SHEF can be measured by the value of the following *projection ratio.*

$$D(\mathbf{x}, \mathbf{c}, \mathbf{S}) = \frac{|\mathbf{w}_p^T(\mathbf{x} - \mathbf{c})|}{r\sqrt{\mathbf{w}_p^T \mathbf{S} \mathbf{w}_p}}. \tag{3.25}$$

The value of the projection ratio can be interpreted as follows. If $D(\mathbf{x}, \mathbf{c}, \mathbf{S}) < 1$, the projection location of data point $\mathbf{x}$ is *inside* projected SHEF onto $\mathbf{w}_p$. If $D(\mathbf{x}, \mathbf{c}, \mathbf{S}) > 1$, the projection location of data point $\mathbf{x}$ is *outside* projected SHEF onto $\mathbf{w}_p$. And if $D(\mathbf{x}, \mathbf{c}, \mathbf{S}) = 1$, the projection location of data point $\mathbf{x}$ is *on the boundary* of projected SHEF onto $\mathbf{w}_p$.

Based on this projection ratio, determining whether data point $\mathbf{x}$ is inside or outside SHEF can be easily done. The following theorem states the conditions to indicate the location of $\mathbf{x}$ with respect to SHEF.

Theorem 3.3.1. Let $\mathbf{S}$ and $\mathbf{c}$ be a covariance matrix and a centroid of captured data of SHEF in a d-dimensional space. Suppose $\mathbf{x}$ is a single queried data point and a discriminant vector $\mathbf{w}_p$ is defined in equation (3.24).

1. If $D(\mathbf{x}, \mathbf{c}, \mathbf{S}) < 1$ then $\mathbf{x}$ is inside SHEF.
2. If $D(\mathbf{x}, \mathbf{c}, \mathbf{S}) > 1$ then $\mathbf{x}$ is outside SHEF.
3. If $D(\mathbf{x}, \mathbf{c}, \mathbf{S}) = 1$ then $\mathbf{x}$ is on the boundary of SHEF.

The proof of **Theorem 3.3.1** is given in APPENDIX C.

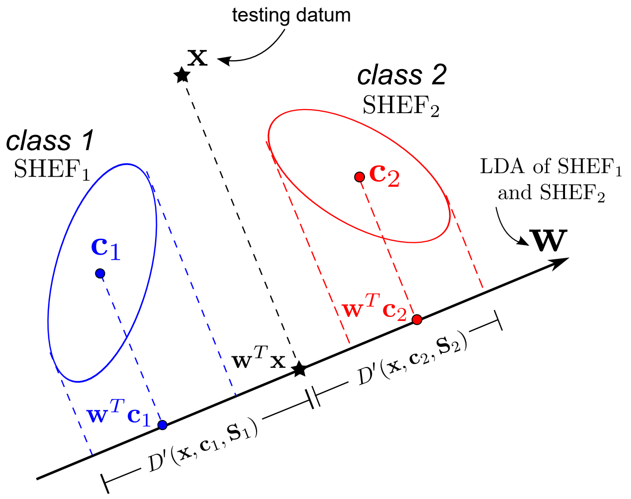

testing datum
x
class 2
SHEF$_2$
class 1
SHEF$_1$
$\mathbf{c}_2$
$\mathbf{c}_1$
LDA of SHEF$_1$ and SHEF$_2$
w
$\mathbf{w}^T\mathbf{c}_2$
$\mathbf{w}^T\mathbf{x}$
$\mathbf{w}^T\mathbf{c}_1$
$D'(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$
$D'(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1)$

Figure 3.18: An example of projections of two SHEFs and $\mathbf{x}$ onto a discriminant vector $\mathbf{w}$ defined in equation (2.8). There are two *projection ratios*, one for each SHEF.

3.3.3 Determining Class of Queried Data Point Based on Projection Ratio

Data point $\mathbf{x}$ may encounter two possible scenarios when its projection ratio is deployed to determine its appropriate class. SHEFs whose amount of data is less than M will be ignored for assigning the class of data point $\mathbf{x}$. Suppose two SHEFs are close to $\mathbf{x}$. The projection ratios of both SHEF$_1$ and SHEF$_2$ are $D(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1)$ and $D(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$, respectively.

1. If both SHEFs represent the same class, then $\mathbf{x}$ is assigned to the class of either SHEF$_1$ or SHEF$_2$.
2. If both SHEFs represent the different classes, then the appropriate class of $\mathbf{x}$ is determined as follows.
   (a) $D(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1) \leq 1 < D(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$. This implies that $\mathbf{x}$ is inside or on the boundary only SHEF$_1$ following **Theorem 3.3.1**. Thus, $\mathbf{x}$ should be assigned to the class of SHEF$_1$ rather than that of SHEF$_2$.

(b) Otherwise, *the projection ratio* is deployed to both SHEFs onto a new discriminant vector $\mathbf{w}$ defined in equation (2.8) instead as shown in Figure 3.18. Denoted by $D'(\mathbf{x}, \mathbf{c}, \mathbf{S}) = \dfrac{|\mathbf{w}^T(\mathbf{x} - \mathbf{c})|}{r\sqrt{\mathbf{w}^T\mathbf{S}\mathbf{w}}}$, the appropriate class of $\mathbf{x}$ is determined as follows.

   i. If $D'(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1) < D'(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$, then $\mathbf{x}$ should be assigned to the class of SHEF_1.

   ii. If $D'(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1) = D'(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$, then the appropriate class of $\mathbf{x}$ is indeterminate.

CHAPTER IV

EXPERIMENTS AND RESULTS

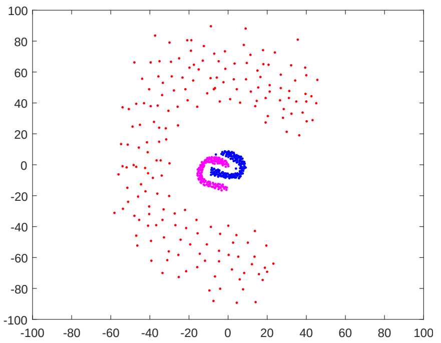
In this Chapter, the performance of SHEF was evaluated on synthetic data sets and real-world data sets compared to other methods. Parameter settings of each method were described. The details of these issues are given in the following sections.

4.1 Data sets

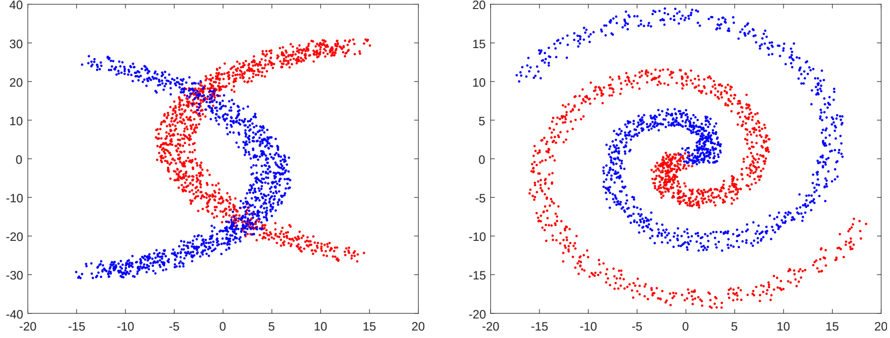
All experiments were tested with seven 2-dimensional synthetic data sets (see Figure 4.1) consisting of a differently distributed data set in each class (Set1), nonlinear separable data sets (Set2 - Set6), and a Guassian distributed data set (Set7) and seven real-world data sets from the University of California at Irvine (UCI) Repository of the machine learning database [24] which were selected by varying the number of features, the number of data, and the number of classes. Letter, Shuttle_trn, and Kddcup99 data set were selected to represent streaming data. For Kddcup99, four symbolic features which are more than two categories were removed. So there are 38 features in this data set. The description of all data sets were given in Table 4.1.

4.2 Accuracy Measurements

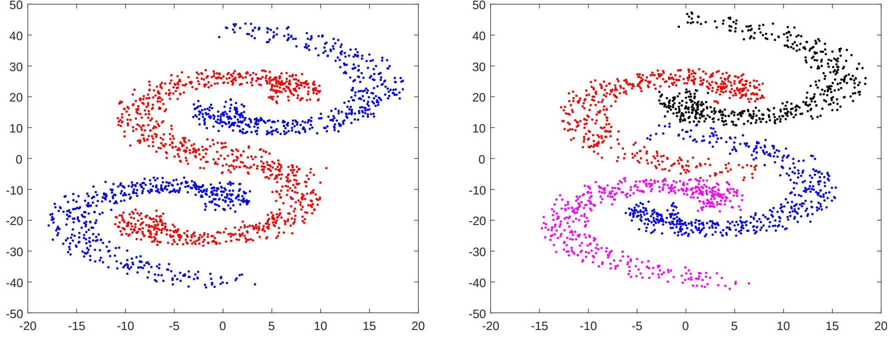
Since the proposed SHEF learning was designed to cope with streaming data environment, a sequence of incoming data chunks was alternatively assigned as a training chunk and a testing chunk. Only the accuracy of the testing chunk is evaluated. The accuracy of each testing chunk is cumulated as *Cumulative*



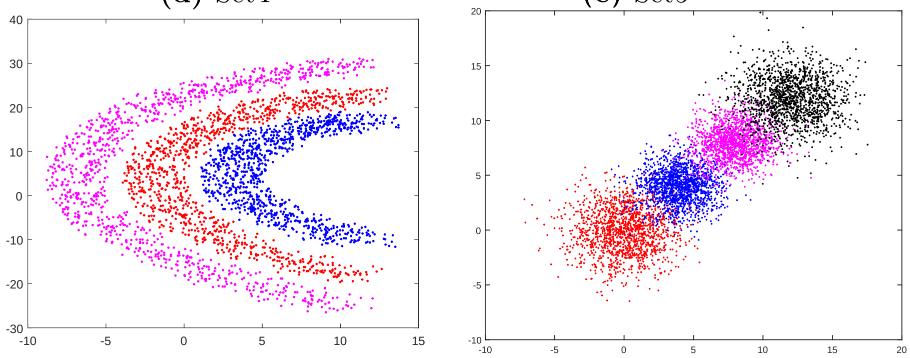
(a) Set1



(b) Set2 **(c)** Set3



(d) Set4 **(e)** Set5



(f) Set6 **(g)** Set7

Figure 4.1: Seven synthetic data sets used in the experiments.

Table 4.1: Experimental data sets and their attributes.

Data set	**# of features**	**# of data**	**# of classes**
% Synthetic data sets			
Set1	2	496	3
Set2	2	2000	2
Set3	2	2000	2
Set4	2	2000	2
Set5	2	2000	4
Set6	2	3000	3
Set7	2	6000	4
% Real-world data sets			
Segment	19	2310	7
Spambase	57	4601	2
Waveform	21	5000	3
Satimage	36	6435	6
Letter	16	20000	26
Shuttle_ trn	9	43500	7
Kddcup99	38	494020	23

Accuracy. The SHEF was also evaluated with non-streaming data. There is only one training chunk and one testing chunk. The accuracy of this evaluating scheme is called *Test Accuracy.* For streaming data evaluation, we assume that training data is divided into T chunks. Let ACC_t, for $1 \leq t \leq T$, be the evaluation accuracy of the $(t+1)^{th}$ testing chunk after training the t^{th} training chunk. Therefore, the test accuracy is defined as ACC_T. The average cumulative accuracy after training t chunk ACA_t is defined as follows.

$$\mathrm{ACA}_t = \frac{1}{t}\sum_{i=1}^{t} \mathrm{ACC}_i, \tag{4.1}$$

where $t = 1, \cdots, T-1$.

4.3 Parameter Settings

The performance of the proposed method was compared to VEBF [11], CIL [12], ILDA [5], VLLDA based on k-NN [13], and LOL [10]. Since each compared method uses different parameters, the settings of parameters of each method must be separately done.

Table 4.2: Parameter settings of VLLDA, VEBF, and CIL.

Data set	VLLDA (k)	VEBF (δ)	CIL (δ)
Set1	5	0.2	0.2
Set2	5	0.1	0.1
Set3	5	0.2	0.1
Set4	5	0.1	0.1
Set5	5	0.1	0.1
Set6	5	0.1	0.1
Set7	10	0.4	0.5
Segment	10	1	0.7
Spambase	10	1	0.4
Waveform	5	1	0.3
Satimage	10	0.9	0.7
Letter	5	0.7	0.7
Shuttle_trn	10	20	1
Kddcup99	10	2	2

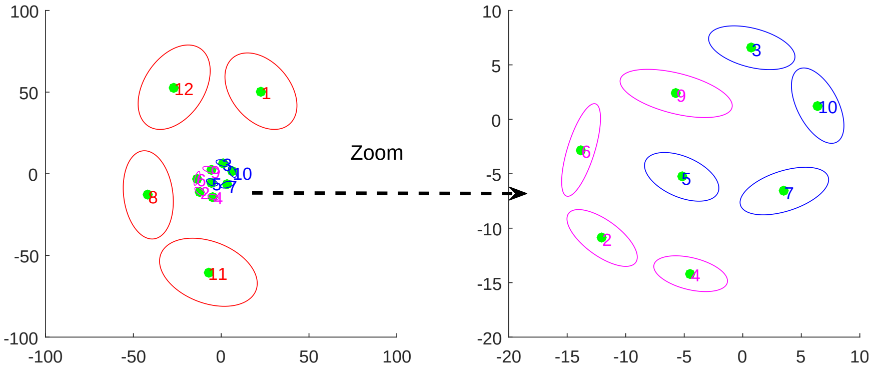
For VEBF and CIL, the constant δ was set to scale the initial width of VEBF shape function calculated from the average distance. Parameter settings for the data sets of Segment, Spambase, Waveform, and Letter were referred from [12]. VEBF uses whole training data to calculate their initial average distance while CIL uses the first 20% of training data. Unfortunately, the number of training data in Shuttle_trn and Kddcup99 is too huge to be calculated in a short time. Hence the only first 10,000 training data of Shuttle_trn and Kddcup99 were calculated for VEBF and the only first 10,000 training data of Kddcup99 were calculated for CIL. For LOL, the authors claimed that LOL is not very sensitive to the parameters and suggested the settings of three parameters, namely the number of prototypes k was set to 60; the balancing parameter λ was set to 1.0; and the aggressive parameter C was set to 1.0 for all experiments. For VLLDA, the parameter k must be set to be congruent with the k-nearest neighbour method. For ILDA and VLLDA which use LDA for classification, the number of selected discriminant vectors was set according to the number of non-zero eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$.

For our proposed method, the following parameters were set: constant r was set to 1.5; regularization parameter ϵ in **Lemma 1** was set to 0.0001; minimum number of data M in a SHEF was set to 3 for all experiments without tuning. Parameter settings of VLLDA, VEBF, and CIL in this dissertation are shown in Table 4.2.

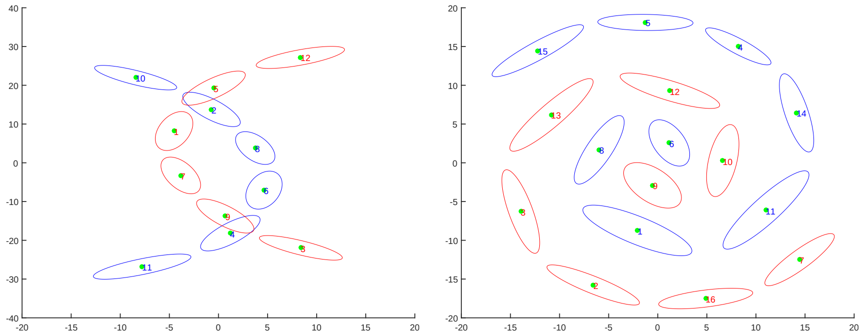
4.4 Comparison Results

A 5-fold cross validation was used in all experiments for each method. In order to create the environment of streaming data, training data in each data set were divided into several chunks of different sizes. These chunks were sequentially fed into the training process. The amount of data in each chunk was randomly set within a pre-defined range called *chunk-size range*. Table 4.3 summarizes the chunk-size range of each data set, the number of training chunks in each fold, and the average of the number of training data in each chunk. In order to study the effect of an incoming order of data chunks in each data set, ten groups of different shuffle patterns of data chunks in each data set were created for each fold. All experiments were conducted on a desktop PC with 8 GB RAM, Intel Core i7-4770, 3.4 GHz with licensed Matlab code.

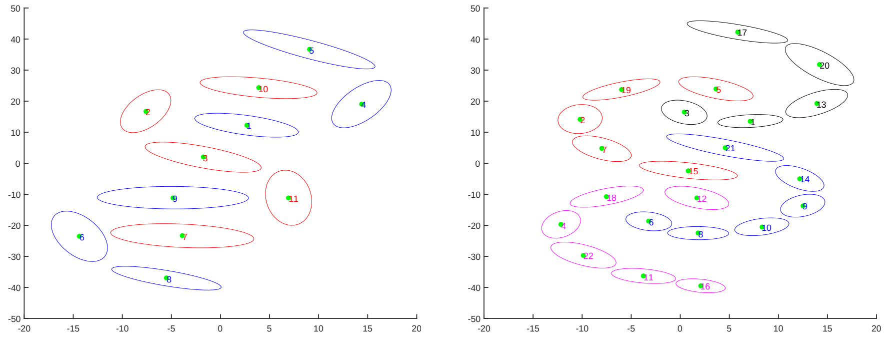
Figure 4.2 illustrates the generated SHEFs by our method with the synthetic data set. SHEFs captured local streaming data structures in each class according to our purpose quite well being compared to original data in Figure 4.1. The size, direction, and number of SHEFs in each class were appropriately updated according to distribution of original data. The independent t-test was adopted to measure the statistically significant difference between the average value of the proposed method and other methods. The value with asterisk (*) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$). The experimental results concerned the following issues.



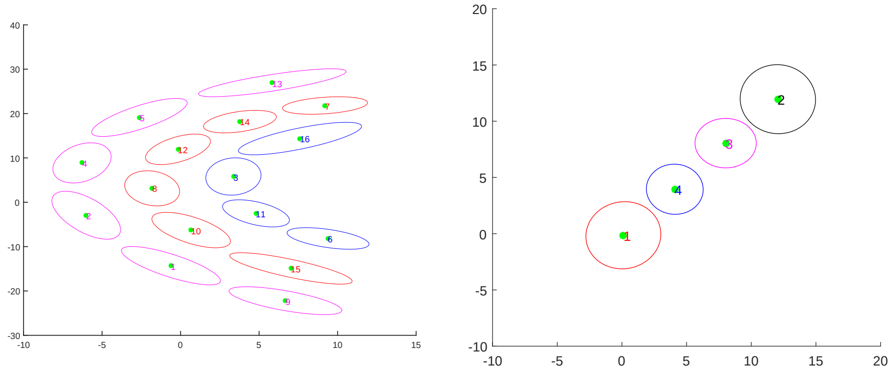
(a) Set1



(b) Set2 **(c)** Set3



(d) Set4 **(e)** Set5



(f) Set6 **(g)** Set7

Figure 4.2: The results of learning seven synthetic data sets using SHEFs. Only 80% of data in each synthetic data were used for training.

Table 4.3: Number of training data in each fold for each streaming chunk.

Data set	Chunk-size range	# of training chunks in each fold					Avg and Stdv of # of training data ($\bar{\mathbf{x}} \pm$sd) in each chunk				
		fold1	fold2	fold3	fold4	fold5	fold1	fold2	fold3	fold4	fold5
Set1	[20,50]	11	12	11	12	10	$36.09_{\pm 8.9}$	$33.08_{\pm 7.44}$	$36.09_{\pm 9.51}$	$33_{\pm 7.9}$	$39.7_{\pm 9.72}$
Set2	[100,200]	10	11	11	11	10	$160_{\pm 28.67}$	$145.45_{\pm 19.61}$	$145.45_{\pm 21.78}$	$145.45_{\pm 21.57}$	$160_{\pm 26.83}$
Set3	[100,200]	11	11	10	10	10	$145.45_{\pm 21.04}$	$145.45_{\pm 21.77}$	$160_{\pm 27.53}$	$160_{\pm 35.38}$	$160_{\pm 22.03}$
Set4	[100,200]	10	11	11	10	10	$160_{\pm 31.12}$	$145.45_{\pm 24.68}$	$145.45_{\pm 29.04}$	$160_{\pm 25.39}$	$160_{\pm 31.36}$
Set5	[100,200]	10	9	10	10	10	$160_{\pm 31.22}$	$177.78_{\pm 22.61}$	$160_{\pm 27.93}$	$160_{\pm 32.07}$	$160_{\pm 23.41}$
Set6	[100,200]	15	16	15	17	16	$160_{\pm 18.94}$	$150_{\pm 29.8}$	$160_{\pm 23.37}$	$141.18_{\pm 31.52}$	$150_{\pm 24.83}$
Set7	[100,200]	31	30	33	32	31	$154.84_{\pm 26.71}$	$160_{\pm 23.74}$	$145.45_{\pm 31.1}$	$150_{\pm 24.63}$	$154.84_{\pm 32.88}$
Segment	[100,200]	12	13	12	12	13	$154_{\pm 31.2}$	$142.15_{\pm 28.8}$	$154_{\pm 30.54}$	$154_{\pm 25.89}$	$142.15_{\pm 32.18}$
Spambase	[100,200]	24	25	25	24	24	$153.33_{\pm 28.86}$	$147.28_{\pm 28.81}$	$147.28_{\pm 28.83}$	$153.33_{\pm 29.67}$	$153.33_{\pm 28.94}$
Waveform	[100,200]	26	26	28	27	25	$153.85_{\pm 28.21}$	$153.85_{\pm 31.83}$	$142.79_{\pm 24.38}$	$148.19_{\pm 25.76}$	$160.04_{\pm 27.69}$
Satimage	[100,200]	35	35	35	34	35	$147.09_{\pm 29.91}$	$147.09_{\pm 29.13}$	$147.11_{\pm 31.22}$	$151.41_{\pm 31.09}$	$147.06_{\pm 27.6}$
Letter	[100,200]	109	108	107	104	105	$146.8_{\pm 27.92}$	$148.14_{\pm 28.44}$	$149.52_{\pm 28.23}$	$153.82_{\pm 27.14}$	$152.42_{\pm 27.87}$
Shuttle_trn	[100,200]	230	231	232	236	232	$151.31_{\pm 30.18}$	$150.65_{\pm 29.61}$	$150_{\pm 29.6}$	$149.46_{\pm 30.3}$	$149.99_{\pm 29.31}$
Kddcup99	[1000,2000]	263	272	260	258	268	$1502.7_{\pm 269.85}$	$1453_{\pm 285.57}$	$1520.1_{\pm 287.42}$	$1531.8_{\pm 276.97}$	$1474.7_{\pm 285.18}$

4.4.1 Test Accuracy

The average test accuracy and the standard deviation of all methods in each data set are reported in Table 4.4. The compared methods used different approaches to learning data. The first approach is retaining all training data throughout the training and testing processes as implemented in VLLDA and ILDA. But the second approach is using the concept of discard-after-learned. Each datum is learned in one pass and discarded afterwards. No need to retain any incoming datum throughout the training and testing processes as implemented in LOL, VEBF, CIL, and our proposed method. To obtain a clear comparison based on these two approaches, the testing was conducted into two categories according to each approach.

Without tuning any parameters, our method called **SHEF** achieved better accuracy in 11 out of 14 data sets compared to the methods using the second approach and in 5 out of 14 data set when compared to the methods using the first approach. The better accuracy is shown in bold numbers and underlined numbers. Notice that there is a statistically significant difference between the average test accuracy of SHEF and other methods (LOL, VEBF, and CIL) on every data set, except for VEBF on Set2, Set4, and Set5. LOL has the lowest accuracy on Set2, Set3, Set4, Segment, Spambase, and Waveform. VEBF has the lowest accuracy with large standard deviation on Shuttle_trn. CIL has the lowest accuracy on Satimage, Letter, and Kddcup99. For Kddcup99 data set, VLLDA and ILDA could not finish the learning process within an hour.

Table 4.4: Comparison of accuracy (%) in the five-fold validation.

Data set	Approach 1		Approach 2			
(# features, # classes)	VLLDA	ILDA	LOL	VEBF	CIL	**SHEF**
Set1	99.798*	99.798*	91.930	86.592	89.291	**99.738**
(2, 3)	±0.408	±0.408	±9.280	±4.024	±2.863	±0.447
Set2	92.600*	72.550	55.135	**93.325***	89.445	92.780
(2, 2)	±1.853	±1.913	±1.978	±1.640	±5.128	±1.922
Set3	99.600	67.850	50.150	97.820	97.625	**98.750**
(2, 2)	±0.258	±1.602	±2.163	±1.512	±3.244	±1.680
Set4	100.000	62.200	49.960	**99.730***	93.255	99.500
(2, 2)	±0.000	±4.521	±0.999	±0.307	±4.840	±0.791
Set5	99.850	99.850	95.750	98.865*	92.430	**98.950**
(2, 4)	±0.202	±0.202	±6.893	±0.839	±4.830	±0.941
Set6	99.733	99.767	97.307	98.897	92.710	**99.253**
(2, 3)	±0.172	±0.172	±1.857	±0.582	±3.928	±0.548
Set7	90.050	89.633	88.220	90.733	85.122	**93.093**
(2, 4)	±1.055	±0.793	±2.788	±3.881	±3.537	±1.074
Segment	95.065	95.714	79.320	82.498	88.563	**93.844**
(19, 7)	±1.511	±1.479	±2.938	±2.935	±2.305	±0.848
Spambase	85.773	87.433	60.533	71.902	90.239	**90.845**
(57, 2)	±0.588	±0.324	±1.391	±8.121	±1.847	±0.775
Waveform	81.920	82.019	80.926	84.116	81.388	**86.480**
(21, 3)	±0.679	±1.217	±2.142	±4.201	±1.139	±0.907
Satimage	88.163	86.869	80.634	83.081	58.176	**84.881**
(36, 6)	±1.037	±0.585	±2.950	±4.454	±2.809	±0.903
Letter	95.552	95.723	61.205	65.985	26.432	**84.918**
(16, 26)	±0.279	±0.328	±2.332	±5.730	±16.741	±0.378
Shuttle_trn	99.862	99.883	**97.753**	38.343	97.280	96.236
(9, 7)	±0.055	±0.029	±0.677	±30.677	±0.324	±1.391
Kddcup99	N/A	N/A	98.757	78.919	19.93	**99.348**
(38, 23)			±0.394	±0.38	±0.189	±0.055

1. **Bold** number indicates the maximum average accuracy in the part of approach 2.
2. Underlined number indicates the maximum average accuracy between SHEF and approach 1.
3. N/A indicates that method could not finish the learning process within an hour.
4. The value with asterisk (*) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$).
5. Average test accuracy (%) and standard deviation ($\bar{x}$±sd) of 50 replications is reported.

Figure 4.3 and Figure 4.4 show the chunk-wise accuracy for six methods after training in each chunk on seven synthetic and seven real-world data sets, respectively. One of 50 experiments was picked up to show the accuracy at each chunk of streaming data. Since VLLDA and ILDA used and retained all incoming training chunks to classify testing data, thus instead of measuring chunk-wise accuracy after each testing chunk, the average cumulative accuracy was measured in this case. It is remarkable that LOL has a wide swinging range of accuracy on Set1, Set5, Set6, Set7, Waveform, Satimage, and Kddcup99. VEBF has a wide swinging range of the accuracy on Spambase and Shuttle_trn. There is a sudden change of the accuracy value in VEBF on Shuttle_trn data set to be discussed in the next section. CIL has a wide swinging range of the accuracy on Set7, Waveform, and Letter. For Kddcup99, CIL has the low accuracy every chunk of data. The average cumulative accuracy of the results in Figure 4.3 and Figure 4.4 calculated by equation (4.1) are shown in Figure 4.5 and Figure 4.6, respectively. This type of accuracy indicates the trend of accuracy as the result of incrementally learning the incoming chunk after chunk. For the proposed method, the average cumulative accuracy trends to increase in all data sets when SHEFs are incrementally trained.

4.4.2 Average number of generated neurons

Table 4.5 shows the number of neurons generated by VEBF, CIL, and SHEF based on hyper-ellipsoidal shape function to capture incoming data. SHEF at the end of training all incoming chunks, the number of generated SHEFs is less than or equal to those of the others in 11 out of 14 data sets. There is statistically significant difference between the average number of neurons in our proposed method and the other methods (VEBF and CIL) in every data set, except for VEBF on Set7 and Waveform. There are six data sets, i.e. Set7, Segment, Spambase, Waveform, Satimage, and Letter, where our method generated the number of SHEFs close to the number of classes. However, there is one data set, namely Kddcup99 data set, where the number of generated SHEFs is less than the number of classes (23 classes). They are discussed in the next section.

Figure 4.7 and Figure 4.8 show the number of generated neurons for three methods (VEBF, CIL, and proposed SHEF) during training in each chunk seven synthetic and seven real-world data sets, respectively. VEBF and SHEF merge two overlapped hyper-ellipsoids. Therefore the number of neurons during the training process may be increased or decreased according to their approaches. Since CIL does not have any merging strategy, the number of neurons is always increased.

4.4.3 Computational time

Computational time of training and testing processes is reported in Table 4.6 and Table 4.7. There is a statistically significant difference between the average of computational time (both training and testing process) of the proposed method and the other five methods, except for the testing time of LOL on Set1. For VLLDA, they used only the testing process so only the testing time is reported. The training time of VEBF, CIL, and our proposed method included the computational time of calculating the initial distance of hyper-ellipsoids. When the number of data is not huge, the computational time of all methods may not be different. However, the last three data sets (Letter, Shuttle_trn, and Kddcup99) are quite big, our method was obviously faster than others methods for both training and testing. For example, our method spent about 18 seconds for 395,216 training data and about 93 seconds for 98,804 testing data in Kddcup99 data set while the other methods spent for a much longer time to train and test, especially for VLLDA and ILDA. They could not finish the process within an hour. Actually, VLLDA and ILDA spent about four hours and three days, respectively.

Table 4.5: Comparison of the number of neurons in VEBF, CIL, and SHEF methods at the end of training process.

Data set	VEBF	CIL	**SHEF**
Set1	29.960 ±2.878	17.980 ±3.431	**12.280** ±1.485
Set2	**9.020** ±0.845	109.920 ±13.471	12.480 ±1.568
Set3	16.640 ±1.120	186.640 ±11.374	**15.360** ±1.575
Set4	15.320 ±1.115	133.140 ±10.208	**13.380** ±1.413
Set5	17.080 ±1.291	99.280 ±17.473	**16.320** ±1.609
Set6	**12.220** ±1.314	218.960 ±19.283	12.800 ±1.457
Set7	**4.260*** ±0.600	43.720 ±3.881	4.500 ±1.909
Segment	14.320 ±0.978	86.040 ±3.812	**7.120** ±0.328
Spambase	20.420 ±1.500	108.140 ±6.958	**2.280** ±0.536
Waveform	**3.000*** ±0.000	80.300 ±2.845	**3.000** ±0.000
Satimage	6.500 ±0.909	211.780 ±3.247	**6.060** ±0.240
Letter	41.460 ±5.997	2532.380 ±30.438	**27.840** ±2.235
Shuttle_trn	24.680 ±1.253	751.460 ±9.232	**13.160** ±2.394
Kddcup99	117.34 ±18.92	2328.6 ±119.14	**16.2** ±1.917

1. **Bold** number indicates the minimum average number of neurons.
2. The value with asterisk (*) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$).
3. Average number of neurons and standard deviation ($\bar{\mathbf{x}}$±sd) of 50 replications is reported.

Table 4.6: Computational time (seconds) for synthetic data

Data set	Process	Approach 1		Approach 2			
(# train : # test)		VLLDA	ILDA	LOL	VEBF	CIL	**SHEF**
Set1	train	-	0.149$_{\pm0.005}$	0.057$_{\pm0.003}$	0.124$_{\pm0.011}$	**0.034**$_{\pm0.016}$	0.099$_{\pm0.028}$
(397 : 99)	test	0.121$_{\pm0.018}$	0.121$_{\pm0.015}$	0.014$^{*}_{\pm0.001}$	0.039$_{\pm0.004}$	**0.005**$_{\pm0.002}$	0.014$_{\pm0.004}$
Set2	train	-	2.260$_{\pm0.017}$	**0.074**$_{\pm0.002}$	0.416$_{\pm0.026}$	0.218$_{\pm0.024}$	0.404$_{\pm0.034}$
(1600 : 400)	test	0.772$_{\pm0.083}$	0.621$_{\pm0.018}$	**0.016**$_{\pm0.001}$	0.045$_{\pm0.005}$	0.078$_{\pm0.009}$	0.053$_{\pm0.006}$
Set3	train	-	2.263$_{\pm0.007}$	**0.074**$_{\pm0.001}$	0.584$_{\pm0.022}$	0.250$_{\pm0.019}$	0.399$_{\pm0.039}$
(1600 : 400)	test	0.770$_{\pm0.037}$	0.614$_{\pm0.004}$	**0.016**$_{\pm0.001}$	0.089$_{\pm0.006}$	0.132$_{\pm0.009}$	0.064$_{\pm0.005}$
Set4	train	-	2.260$_{\pm0.013}$	**0.074**$_{\pm0.003}$	0.593$_{\pm0.032}$	0.224$_{\pm0.006}$	0.399$_{\pm0.043}$
(1600 : 400)	test	0.696$_{\pm0.006}$	0.618$_{\pm0.006}$	**0.016**$_{\pm0.000}$	0.076$_{\pm0.006}$	0.095$_{\pm0.008}$	0.058$_{\pm0.006}$
Set5	train	-	2.285$_{\pm0.014}$	0.301$_{\pm0.001}$	0.380$_{\pm0.015}$	**0.201**$_{\pm0.005}$	0.366$_{\pm0.028}$
(1600 : 400)	test	0.704$_{\pm0.009}$	0.687$_{\pm0.007}$	0.082$_{\pm0.002}$	0.083$_{\pm0.007}$	0.072$_{\pm0.012}$	**0.066**$_{\pm0.007}$
Set6	train	-	5.092$_{\pm0.022}$	**0.340**$_{\pm0.003}$	0.682$_{\pm0.042}$	0.428$_{\pm0.009}$	0.511$_{\pm0.009}$
(2400 : 600)	test	1.415$_{\pm0.010}$	1.396$_{\pm0.017}$	0.092$_{\pm0.002}$	0.089$_{\pm0.009}$	0.230$_{\pm0.020}$	**0.078**$_{\pm0.007}$
Set7	train	-	20.543$_{\pm0.072}$	0.909$_{\pm0.007}$	0.754$_{\pm0.019}$	1.457$_{\pm0.016}$	**0.411**$_{\pm0.063}$
(4800 : 1200)	test	5.744$_{\pm0.189}$	4.761$_{\pm0.035}$	0.210$_{\pm0.001}$	**0.070**$_{\pm0.007}$	0.105$_{\pm0.009}$	0.083$_{\pm0.020}$

1. In VLLDA, it does not need to train the model.
2. N/A indicates that method could not finish the process within an hour.
3. **Bold** indicates the fastest computaional time on six methods.
4. The value with asterisk (*) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$).
5. Average computational time and standard deviation ($\bar{\mathbf{x}}$±sd) of 50 replications is reported.

Table 4.7: Computational time (seconds) for real-world data

Data set	Process	Approach 1		Approach 2			
(# train : # test)		VLLDA	ILDA	LOL	VEBF	CIL	**SHEF**
Segment	train	-	4.238$_{\pm 0.010}$	0.714$_{\pm 0.005}$	0.399$_{\pm 0.026}$	0.258$_{\pm 0.002}$	**0.220**$_{\pm 0.027}$
(1848 : 462)	test	0.766$_{\pm 0.074}$	0.522$_{\pm 0.010}$	0.179$_{\pm 0.003}$	0.334$_{\pm 0.029}$	0.093$_{\pm 0.004}$	**0.084**$_{\pm 0.005}$
Spambase	train	-	78.007$_{\pm 0.979}$	**0.387**$_{\pm 0.042}$	6.017$_{\pm 0.237}$	1.407$_{\pm 0.048}$	1.618$_{\pm 0.537}$
(3681 : 920)	test	4.440$_{\pm 0.106}$	2.740$_{\pm 0.131}$	**0.056**$_{\pm 0.001}$	3.440$_{\pm 0.264}$	0.334$_{\pm 0.022}$	0.287$_{\pm 0.058}$
Waveform	train	-	20.131$_{\pm 0.069}$	0.679$_{\pm 0.003}$	0.862$_{\pm 0.060}$	1.036$_{\pm 0.020}$	**0.274**$_{\pm 0.019}$
(4000 : 1000)	test	2.083$_{\pm 0.027}$	3.751$_{\pm 0.027}$	0.171$_{\pm 0.003}$	0.235$_{\pm 0.014}$	0.184$_{\pm 0.008}$	**0.109**$_{\pm 0.006}$
Satimage	train	-	48.688$_{\pm 0.771}$	1.982$_{\pm 0.009}$	1.887$_{\pm 0.116}$	1.999$_{\pm 0.096}$	**0.802**$_{\pm 0.249}$
(5148 : 1287)	test	4.782$_{\pm 0.140}$	6.411$_{\pm 0.259}$	0.471$_{\pm 0.009}$	1.459$_{\pm 0.147}$	0.718$_{\pm 0.011}$	**0.383**$_{\pm 0.019}$
Letter	train	-	304.808$_{\pm 0.954}$	22.495$_{\pm 0.356}$	12.323$_{\pm 9.146}$	18.609$_{\pm 0.576}$	**0.997**$_{\pm 0.245}$
(16000 : 4000)	test	14.724$_{\pm 0.855}$	8.673$_{\pm 0.074}$	5.014$_{\pm 0.091}$	7.691$_{\pm 0.922}$	24.378$_{\pm 0.916}$	**2.132**$_{\pm 0.173}$
Shuttle_trn	train	-	1207.920$_{\pm 11.657}$	12.469$_{\pm 0.054}$	12.419$_{\pm 1.137}$	76.331$_{\pm 1.402}$	**5.836**$_{\pm 0.996}$
(34800 : 8700)	test	458.069$_{\pm 2.394}$	359.521$_{\pm 13.420}$	2.851$_{\pm 0.031}$	4.337$_{\pm 0.302}$	13.261$_{\pm 0.240}$	**1.985**$_{\pm 0.384}$
Kddcup99	train	-	N/A	553.41$_{\pm 24.448}$	515.48$_{\pm 96.792}$	176.92$_{\pm 9.669}$	**18.365**$_{\pm 1.208}$
(395216 : 98804)	test	N/A	N/A	115.66$_{\pm 4.9891}$	804.04$_{\pm 119.51}$	854.92$_{\pm 48.558}$	**93.878**$_{\pm 13.266}$

1. In VLLDA, it does not need to train the model.
2. N/A indicates that method could not finish the process within an hour.
3. **Bold** indicates the fastest computaional time on six methods.
4. Average computational time and standard deviation ($\bar{\mathbf{x}}$±sd) of 50 replications is reported.

4.5 Improve Test Accuracy on VEBF and CIL using Projection Ratio in Testing

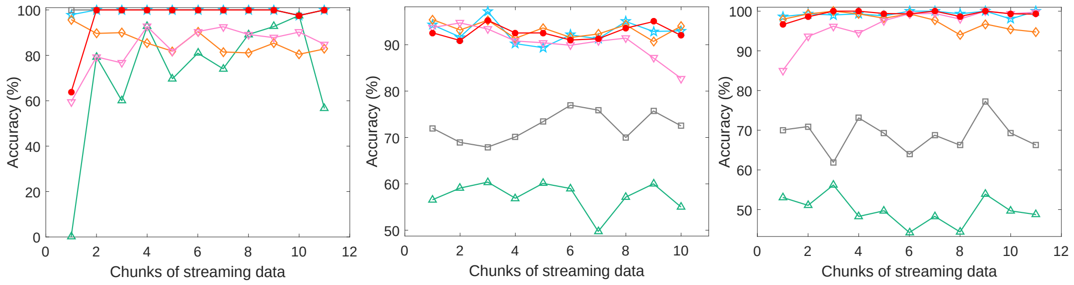
To measure the performance of projection ratio on other hyper-ellipsoidal shape functions, Models from the learning process of VEBF and CIL were used but the decision function for assigning the class label of testing data would be replaced with our proposed testing approach in Section 3.3.3 based on projection ratio. The average test accuracy and the standard deviation of VEBF and CIL were reported in Table 4.8. VEBF and CIL using our testing approach are denoted by VEBF+ and CIL+, respectively.

VEBF+ achieved better average accuracy in 12 out of 14 data sets compared to VEBF. The average accuracy of VEBF+ is statistically and significantly difference to VEBF on 9 data sets, namely Set2, Set7, Segment, Spambase, Waveform, Satimage, Letter, Shuttle_trn, and Kddcup99. CIL+ achieved better average accuracy in 12 out of 14 data sets compared to CIL. The average accuracy of CIL+ is statistically and significantly difference to CIL on every data set, except for Set7 and Letter.

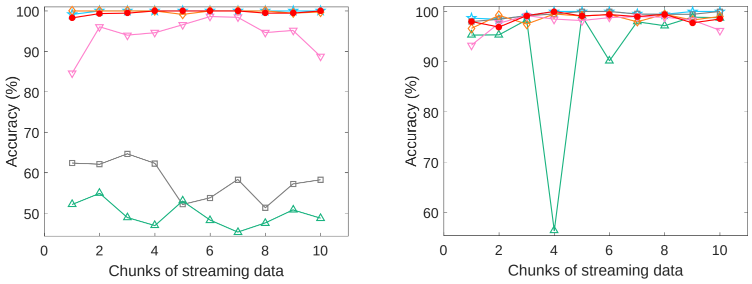
Table 4.8: Comparison of accuracy (%) of VEBF, CIL, VEBF+, and CIL+

Data set	VEBF	**VEBF+**	CIL	**CIL+**
Set1	86.592* ±4.024	**87.455** ±2.832	**89.291** ±2.863	87.579 ±2.984
Set2	**93.325** ±1.640	92.155 ±2.054	89.445 ±5.128	**93.080** ±2.322
Set3	**97.820*** ±1.512	97.545 ±1.666	97.625 ±3.244	**98.810** ±1.437
Set4	99.730* ±0.307	**99.745** ±0.321	93.255 ±4.840	**99.150** ±0.821
Set5	98.865* ±0.839	**99.140** ±0.722	92.430 ±4.830	**98.290** ±1.125
Set6	98.897* ±0.582	**99.073** ±0.593	92.710 ±3.928	**99.173** ±1.324
Set7	90.733 ±3.881	**93.310** ±0.411	85.122* ±3.537	**86.167** ±1.948
Segment	82.498 ±2.935	**93.913** ±1.359	88.563 ±2.305	**94.138** ±0.913
Spambase	71.902 ±8.121	**90.628** ±1.034	90.239 ±1.847	**92.885** ±0.979
Waveform	84.116 ±4.201	**86.480** ±0.907	81.388 ±1.139	**83.004** ±1.126
Satimage	83.081 ±4.454	**85.041** ±0.857	58.176 ±2.809	**61.768** ±2.251
Letter	65.985 ±5.730	**85.311** ±0.621	**26.432*** ±16.741	26.150 ±5.21
Shuttle_trn	38.343 ±30.677	**96.817** ±0.192	97.280 ±0.324	**97.879** ±0.227
Kddcup99	78.919 ±0.38	**99.738** ±0.062	19.93 ±0.189	**99.74** ±0.02

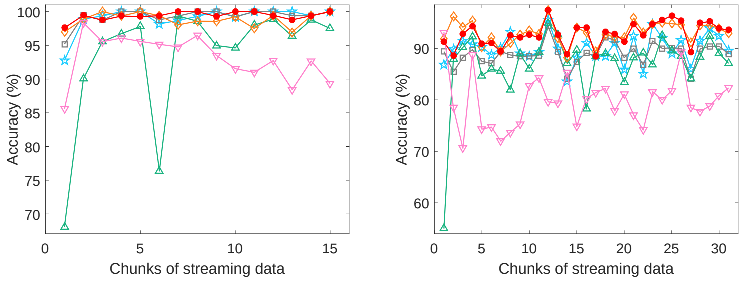
1. **Bold** number indicates the minimum average number of neurons.
2. The value with asterisk (*) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$).
3. VEBF+ and CIL+ refer VEBF and CIL using our criteria based on projection ratio, respectively.
4. Average test accuracy (%) and standard deviation ($\bar{\mathbf{x}}$±sd) of 50 replications is reported.



(a) Set1 **(b)** Set2 **(c)** Set3



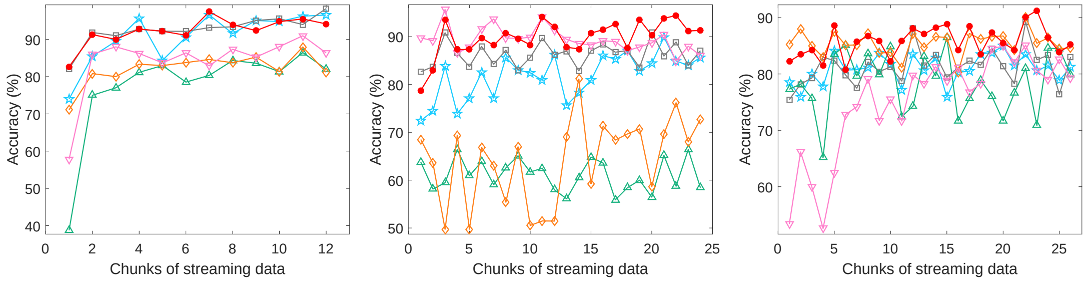
(d) Set4 **(e)** Set5



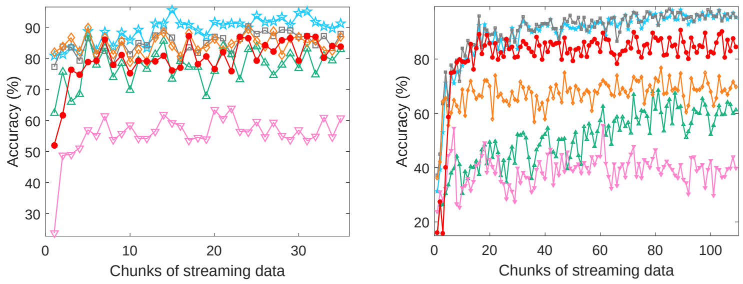
(f) Set6 **(g)** Set7

☆ VLLDA □ ILDA △ LOL ◇ VEBF ▽ CIL ● SHEF

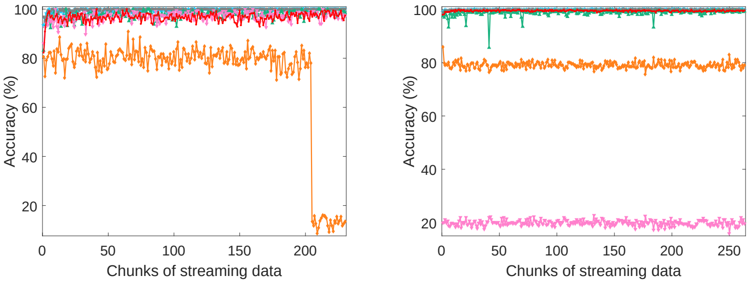
Figure 4.3: The chunk-wise accuracy (%) for six methods after training in each chunk on seven synthetic data sets



(a) Segment **(b)** Spambase **(c)** Waveform



(d) Satimage **(e)** Letter



(f) Shuttle_trn **(g)** Kddcup99

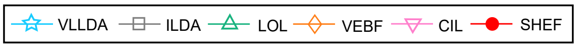
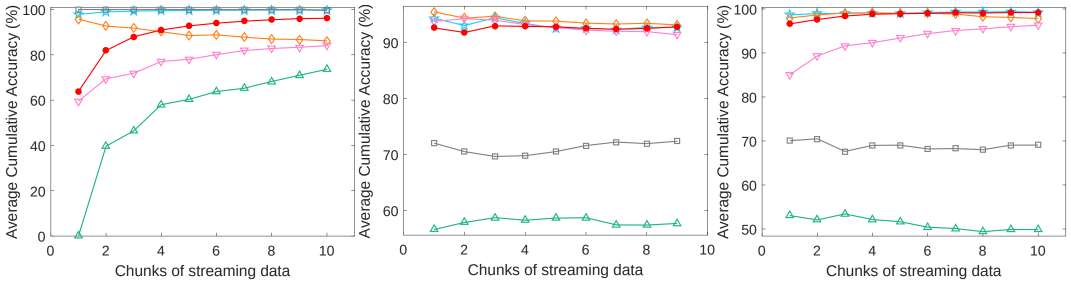
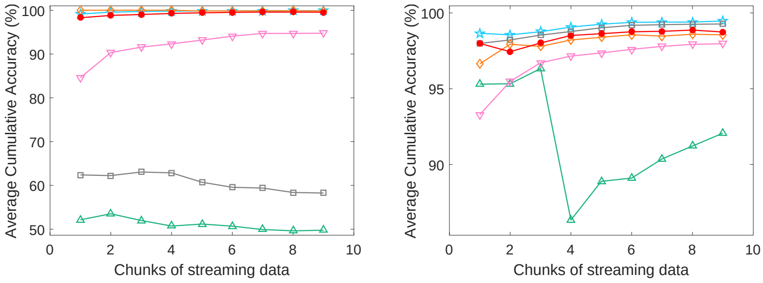


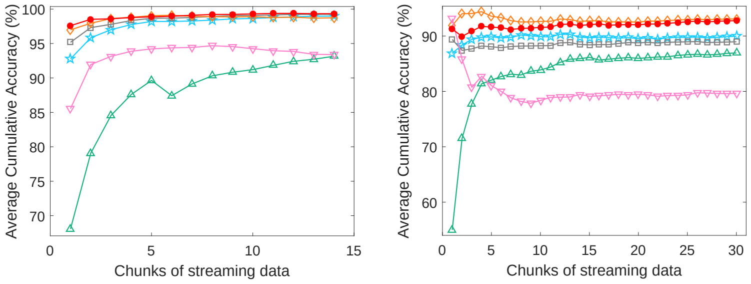
Figure 4.4: The chunk-wise accuracy (%) for six methods after training in each chunk on seven real-world data sets



(a) Set1 **(b)** Set2 **(c)** Set3



(d) Set4 **(e)** Set5



(f) Set6 **(g)** Set7

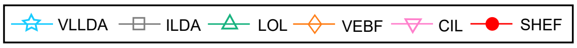
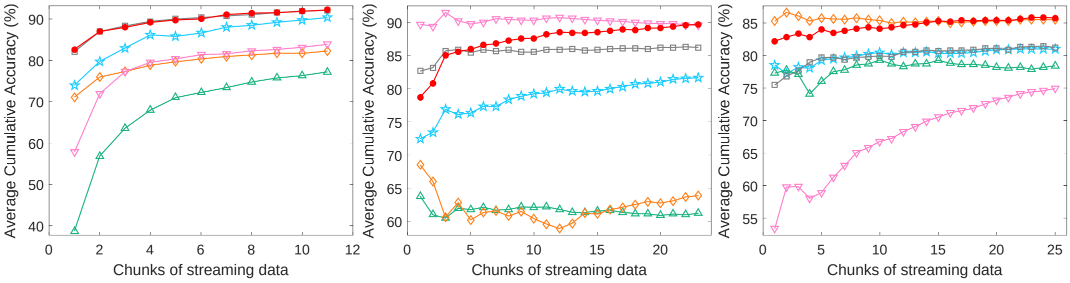
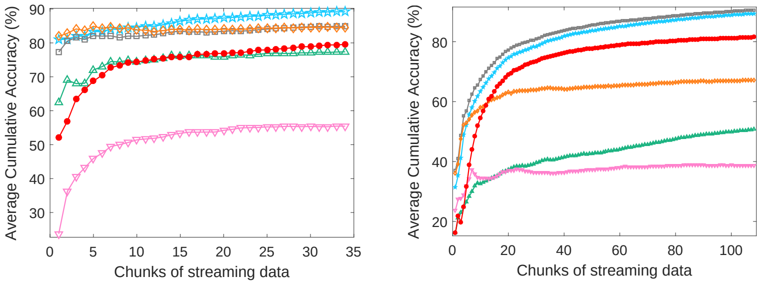


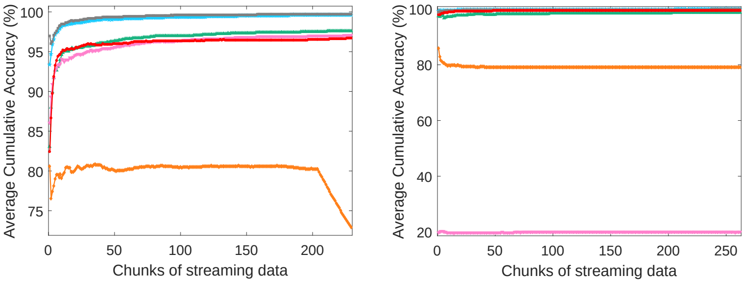
Figure 4.5: The average cumulative chunk-wise accuracy (%) for six methods after training in each chunk on seven synthetic data sets



(a) Segment **(b)** Spambase **(c)** Waveform



(d) Satimage **(e)** Letter



(f) Shuttle_trn **(g)** Kddcup99

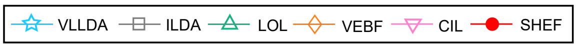
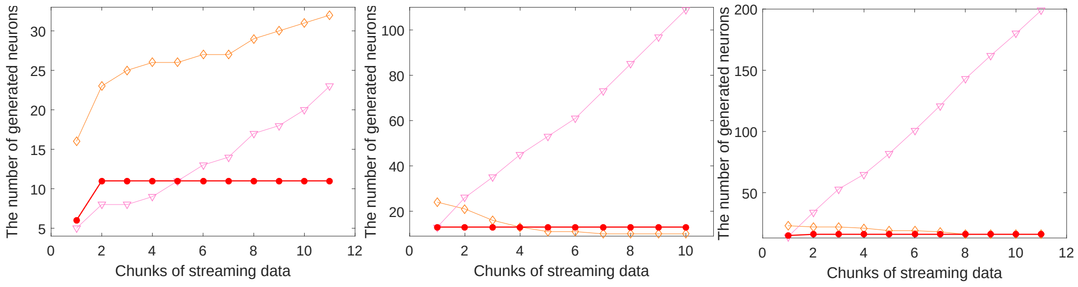
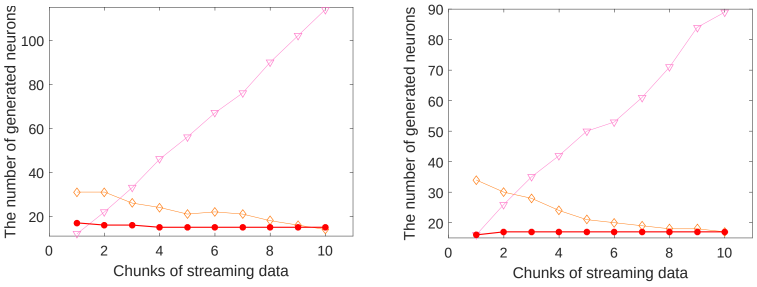


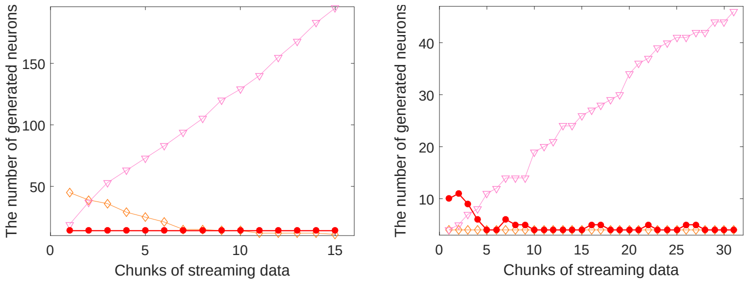
Figure 4.6: The average cumulative chunk-wise accuracy (%) for six methods after training in each chunk on seven real-world data sets



(a) Set1 **(b)** Set2 **(c)** Set3



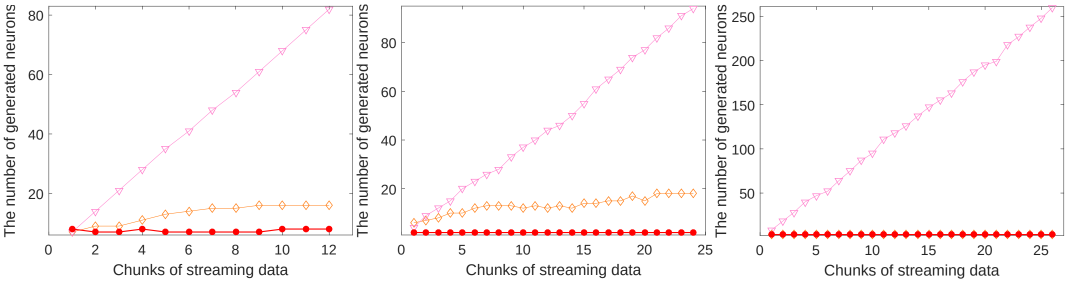
(d) Set4 **(e)** Set5



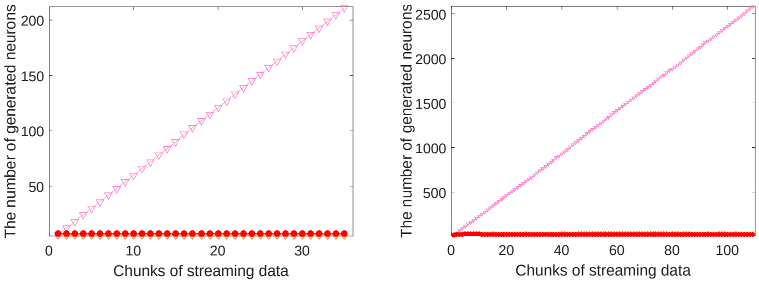
(f) Set6 **(g)** Set7

◇ VEBF ▽ CIL ● SHEF

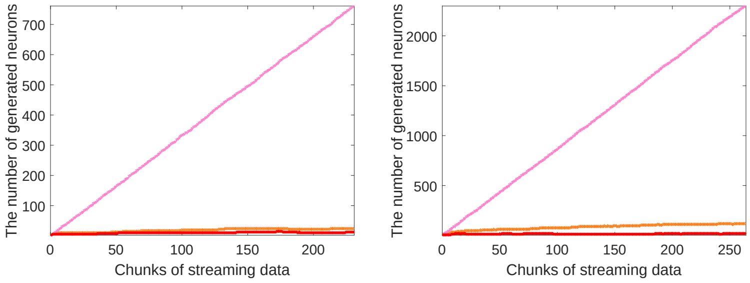
Figure 4.7: The number of generated neurons for three methods after training in each chunk on seven synthetic data sets



(a) Segment **(b)** Spambase **(c)** Waveform



(d) Satimage **(e)** Letter



(f) Shuttle_trn **(g)** Kddcup99

◇ VEBF ▽ CIL ● SHEF

Figure 4.8: The number of generated neurons for three methods after training in each chunk on seven real-world data sets

CHAPTER V

DISCUSSION AND CONCLUSION

The results in the previous section would be discussed in the term of the reasonable methods for applying to streaming data by focusing on the acceptable accuracy during the training process in the appropriately computational time.

Definitely, VLLDA and ILDA are not suitable for an streaming scenario because they need to store the whole incoming training data for classification. Besides when the amount of data is big such as Shuttle_trn and Kddcup99, both VLLDA and ILDA methods spent too much training time because they must hold all training data in the process. This makes them unsuitable for learning streaming data. However, VLLDA and ILDA still hold the advantages in classification accuracy because they employ the concept of k-NN and LDA. Comparing the performances of VLLDA and ILDA to the performances of LOL, VEBF, CIL, and SHEF approaches where no trained data are retained throughout the classification is not appropriate.

For ILDA, they have also the same limitation as LDA as seen in the results of Set2, Set3, and Set4 in Table 4.4. The time complexity of updating equations for both $\mathbf{S}_B$ and $\mathbf{S}_W$ affects the training time when training data are big. VLLDA based on LDA with k-NN can improve LDA to handle the complex distribution of data. But choosing k may affect the accuracy. Furthermore, VLLDA needs to compute Euclidean distances among all training data no matter how value of k was set. Hence, the more data are trained, the more time is obviously spent.

For LOL, the number of prototypes k that was set to 60 in all experiments may not be suitable for all data sets in the term of the accuracy such as the resulted accuracy in these data sets, i.e. Set2, Set3, Set4, Spambase, and Letter. Hence, using 60 hyperplanes may not be enough to classify those data sets. As the results, LOL is sensitive to the parameter settings in each data set inconsistent with their conclusions in [10]. Moreover, the sequence of the incoming data has an effect on updating positions of prototypes and the accuracy such as seen in the results of Set1 and Set5 shown in Table 4.4 with a large standard deviation. The computational time is directly proportional to the number of training data and the number of classes, especially the number of classes. Since LOL adopts the one-vs-all strategy in the multi-class problem, LOL must iteratively learn one class at a time.

For VEBF, there is a large standard deviation of accuracy on Set1, Set7, Spambase, Waveform, Satimage, Letter, and Shuttle_trn. Consequently, the sequence of incoming data quite affects on the accuracy of this method that is consistent with the conclusion in [12]. For Shuttle_trn data set, VEBF has the unusual low accuracy at 38.343% with the unusual large standard deviation at 30.677. To answer this situation, the accuracy characteristic displayed in Figure 4.4(f) should be thoroughly analyzed. After considering all 50 experiments of VEBF on this data set, there are 31 out of 50 whose accuracy was immediately decreased at some incoming chunks and never increased again. This problem may occur in the step of merging two neurons of VEBF algorithm when the initial width was set too large. After merging them, the width of the new neuron was set by assuming the Gaussian distribution of data in both neurons that may be smaller than the widths of two previous individual neurons. As the result, updating the parameters of a neuron based on the next training data may be dominated by the larger size of neuron. The training time is directly proportional to the number of the generated neurons during the training process. While the testing time is directly proportional to the number of neurons in the final process. For example, there are about 117 generated neurons in the final state on Kddcup99 data set. Taking about 515 seconds for 395,216 training data and about 804 seconds for 98,804 testing data. Furthermore, if the number of neurons was unnecessarily generated, they would be stored in forms of d-by-d covariance matrices.

For CIL, the number of training data in each chunk is inversely proportional to the number of generated neurons. If the number of training data in each chunk is quite small, the number of neurons may be redundantly generated (no merging strategy in this method). The number of generated neurons is directly proportional to the computational time. However, the large number of neurons may reduce the classification accuracy. CIL is sensitive to the number of training data in each chunk on some data sets (Satimage and Letter). For examples, if the chunk-size range of Satimage is changed from [100,200] to [400,600], the average accuracy will increase to 81.386% and the average number of used neurons will decrease to 62.6. If the chunk-size range of Letter is changed from [100,200] to [1000,2000], the average accuracy will increase to 87.181% and the average number of used neurons will decrease to 282.94. For Kddcup99, although CIL has the lowest average accuracy at 19.93% but the reason is not the sensitivity of the number of train data in each chunk. It is because of the definition of the decision function for determining the class label of the testing data. This conclusion was confirmed using the generated neurons from CIL and our proposed testing approach. The average test accuracy increases from 19.93% to 99.74% as shown in Table 4.8.

For our proposed method, the performance depends on how SHEFs and the network of SHEFs are appropriately constructed to capture the sequential streaming data. According to the 2-dimensional illustration in Figure 4.2, it is noticeable that no matter how complicated the patterns of data distribution are, SHEFs can capture them very well. The proposed method provided the high accuracy more than 80% on several data sets during the incremental learning in the training process at any time stamp of streaming data (see Figure 4.3 and Figure 4.4). Although there are more steps in our discriminant distance testing process than other testing process, our testing process is rather fast because fewer numbers of SHEFs were effectively generated. Moreover, our method also spent less training time than the others because the merging step helps reducing the number of generated SHEFs during the training process. For Kddcup99 data set, 16 SHEFs were generated because there are 15 classes out of 23 classes that have the number of data points more than 20. Before testing the data, any SHEFs containing the number of members fewer than M ($M = 3$) were eliminated. Therefore, some classes might be ignored as noisy data. The number of training data in each chunk does not affect massively to our accuracy. Even though the input data were fed as the chunk, our algorithm still sequentially learns each datum.

5.1 Suggestion on Parameter Settings for the Proposed Method

The positive constant r in the learning algorithm is directly proportional to the size of SHEF which can indirectly effect the merging strategy. The preliminary experiments found that when $r = 1$, the sizes of two SHEF are too small to make two SHEFs of the same class overlapped. As the result, it may lead to many redundant SHEFs generated. While $r = 2$, the size of SHEF is too large to make many SHEFs overlap. As the result, many SHEFs may be unnecessarily merged. Hence, $r = 1.5$ is suggested to appropriately scale the size of SHEF for only 14 data sets used in this dissertation.

The small positive constant ϵ was applied in order to find the inverse of a singular covariance matrix. It was used on two steps such as the step of checking overlap of two SHEFs in the learning process and the step of finding the discriminant vector in the testing process. Without the regularization parameter ϵ, the proposed method may not continue the process that it should be. In this dissertation, ϵ is set to 0.0001 for 14 data sets; however, this value can be changed according to the data set. ϵ should be suitably set without affecting the size of SHEF.

The predefined minimum amount data M in the learning algorithm is directly proportional to the number of SHEF used for determining class of testing data. Besides, it may be indirectly proportional to accuracy of the imbalanced data set; for example, some classes of Kddcup99 data set might be ignored when M was set to 3. Therefore, M can be changed following the data set.

In this dissertation contributed the following issues to cope with learning streaming chunk data and determining the class of queried data.

1. A more generic structure of hyper-ellipsoidal function called *Scalable Hyper-Ellipsoidal Function* (SHEF) was introduced to learn streaming data. This structure is capable of handling the problem of curse of dimensionality by introducing a regularization parameter ϵ into the covariance matrix of SHEF.
2. Initializing SHEFs width with taking distribution of an individual class into account makes SHEFs cope with the very different distributed data of each class.
3. A new recursive function to update the covariance matrix of SHEF based on only the incoming data chunk was proposed. This new function takes less computational than those previously introduced in VEBF, CIL.
4. Fast and easy conditions to test the states of being exactly overlapped, inside, or touching of two SHEFs were intoduced.

5. A new distance measure for determining the class of a queried datum based on the projected distance on LDA discriminant vector was proposed to achieve our concept of integrating distribution to the distance measure. This distance can significantly improve the classification accuracy when compared with others' distances.

The experimental results significantly supported the merit of our proposed concepts in terms of higher accuracy, less computational time, and less number of generated neurons. However, this approach has not been tested with more complex data characteristics such streaming data with dynamic class drift or streaming data with class imbalance.

REFERENCES

[1] J. Gama, *Knowledge Discovery from Data Streams.* Chapman & Hall/CRC, 1st ed., 2010.

[2] E. Keogh and A. Mueen, *Curse of Dimensionality*, pp. 314–315. Boston, MA: Springer US, 2017.

[3] K.-L. Du and M. N. S. Swamy, *Principal Component Analysis*, pp. 355–405. Springer London, 2014.

[4] K.-L. Du and M. N. S. Swamy, *Discriminant Analysis*, pp. 451–468. London: Springer London, 2014.

[5] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, pp. 905–914, Oct 2005.

[6] G.-F. Lu, J. Zou, and Y. Wang, "A new and fast implementation of orthogonal LDA algorithm and its incremental extension," *Neural Processing Letters*, vol. 43, pp. 687–707, Jun 2016.

[7] D. Chu, L. Z. Liao, M. K. P. Ng, and X. Wang, "Incremental linear discriminant analysis: A fast algorithm and comparisons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 2716–2735, Nov 2015.

[8] Y. A. Ghassabeh, F. Rudzicz, and H. A. Moghaddam, "Fast incremental lda feature extraction," *Pattern Recognition*, vol. 48, no. 6, pp. 1999 – 2012, 2015.

[9] B. C. Kuo, H. H. Ho, C. H. Li, C. C. Hung, and J. S. Taur, "A kernel-based feature selection method for svm with rbf kernel for hyperspectral image

classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, pp. 317–326, Jan 2014.

[10] Z. Zhou, W.-S. Zheng, J.-F. Hu, Y. Xu, and J. You, "One-pass online learning: A local approach," *Pattern Recognition*, vol. 51, pp. 346 – 357, 2016.

[11] S. Jaiyen, C. Lursinsap, and S. Phimoltares, "A very fast neural learning for classification using only new incoming datum," *IEEE Transactions on Neural Networks*, vol. 21, pp. 381–392, March 2010.

[12] P. Junsawang, S. Phimoltares, and C. Lursinsap, "A fast learning method for streaming and randomly ordered multi-class data chunks by using one-pass-throw-away class-wise learning concept," *Expert Systems with Applications*, vol. 63, no. Supplement C, pp. 249 – 266, 2016.

[13] Z. Fan, Y. Xu, and D. Zhang, "Local linear discriminant analysis framework using sample neighbors," *IEEE Transactions on Neural Networks*, vol. 22, pp. 1119–1132, July 2011.

[14] T.-K. Kim and J. Kittler, "Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 318–327, March 2005.

[15] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Dec. 2006.

[16] N. Wattanakitrungroj, S. Maneeroj, and C. Lursinsap, "Versatile hyper-elliptic clustering approach for streaming data based on one-pass-thrown-away learning," *Journal of Classification*, vol. 34, pp. 108–147, Apr 2017.

[17] M. Sugiyama, “Local fisher discriminant analysis for supervised dimensionality reduction,” in *Proceedings of the 23rd International Conference on Machine Learning*, ICML ’06, (New York, NY, USA), pp. 905–912, ACM, 2006.

[18] A. Sharma and K. K. Paliwal, “Linear discriminant analysis for the small sample size problem: an overview,” *International Journal of Machine Learning and Cybernetics*, vol. 6, pp. 443–454, Jun 2015.

[19] R. A. FISHER, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[20] S. Alfano and M. Greer, “Determining if two solid ellipsoids intersect,” *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM*, vol. 26, pp. 106–110, Jan 2003.

[21] K. Zimmermann and T. Svoboda, “Approximation of euclidean distance between point from ellipse,” Research Report CTU–CMP–2005–23, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, August 2005.

[22] J. H. Friedman, “Regularized discriminant analysis,” *Journal of the American Statistical Association*, vol. 84, no. 405, pp. 165–175, 1989.

[23] B. P. Stephen, “Algorithms for ellipsoids,” Tech. Rep. FDA-08-01, Sibley School of Mechanical & Aerospace Engineering Cornell University, Ithaca, New York 14853, February 2008.

[24] D. Dheeru and E. Karra Taniskidou, “UCI machine learning repository,” 2017.

APPENDIX

APPENDIX A

Proof of modifying in Theorem 3.1.1: To ease the complication of notations, the following simplified notations are used in the proof. Let x', n', $\mathbf{c}'$, and $\mathbf{S}'$ represent x^{new}, $n_j^{old}+1$, $\mathbf{c}_j^{new}$, and $\mathbf{S}_j^{new}$, respectively. Let n, $\mathbf{c}$, and $\mathbf{S}$ represent n_j^{old}, $\mathbf{c}_j^{old}$, and $\mathbf{S}_j^{old}$, respectively.

$$\begin{aligned}
\mathbf{S}' &= \frac{n(\mathbf{S}+\mathbf{c}\mathbf{c}^T)+\mathbf{x}'(\mathbf{x}')^T}{n'} - \mathbf{c}'(\mathbf{c}')^T \\
&= \frac{n(\mathbf{S}+\mathbf{c}\mathbf{c}^T)+\mathbf{x}'(\mathbf{x}')^T}{n'} - (\frac{n\mathbf{c}+\mathbf{x}'}{n'})(\frac{n\mathbf{c}+\mathbf{x}'}{n'})^T \\
&= \frac{n(\mathbf{S}+\mathbf{c}\mathbf{c}^T)+\mathbf{x}'(\mathbf{x}')^T}{n'} - \frac{n^2\mathbf{c}\mathbf{c}^T+n\mathbf{c}(\mathbf{x}')^T+n\mathbf{x}'\mathbf{c}^T+\mathbf{x}'(\mathbf{x}')^T}{(n')^2} \\
&= \frac{n'n\mathbf{S}+n'n\mathbf{c}\mathbf{c}^T+n'\mathbf{x}'(\mathbf{x}')^T}{(n')^2} - \frac{n^2\mathbf{c}\mathbf{c}^T+n\mathbf{c}(\mathbf{x}')^T+n\mathbf{x}'\mathbf{c}^T+\mathbf{x}'(\mathbf{x}')^T}{(n')^2} \\
&= \frac{n'n\mathbf{S}+(n'-n)n\mathbf{c}\mathbf{c}^T+(n'-1)\mathbf{x}'(\mathbf{x}')^T-n\mathbf{c}(\mathbf{x}')^T-n\mathbf{x}'\mathbf{c}^T}{(n')^2} \\
&= \frac{n'n\mathbf{S}+n\mathbf{c}\mathbf{c}^T+n\mathbf{x}'(\mathbf{x}')^T-n\mathbf{c}(\mathbf{x}')^T-n\mathbf{x}'\mathbf{c}^T}{(n')^2} \\
&= \frac{n'n\mathbf{S}+n\mathbf{c}(\mathbf{c}^T-(\mathbf{x}')^T)+n\mathbf{x}'((\mathbf{x}')^T-\mathbf{c}^T)}{(n')^2} \\
&= \frac{n'n\mathbf{S}+n(\mathbf{c}-\mathbf{x}')(\mathbf{c}-\mathbf{x}')^T}{(n')^2} \\
&= \frac{n}{n'}(\mathbf{S}+\frac{(\mathbf{c}-\mathbf{x}')(\mathbf{c}-\mathbf{x}')^T}{n'}) \quad \square
\end{aligned}$$

APPENDIX B

Proof of modifying in Theorem 3.2.1: Let d be the dimensions of data; $\widetilde{\mathbf{S}}_i$ of size d-by-d be a covariance matrix of a i^{th}SHEF; $\mathbf{R}_i$ of size $(d+1)$-by-$(d+1)$ be the translation matrix of the centroid of the i^{th}SHEF; $\mathbf{x}$ be any data point, $\mathbf{c}_i$ be the center of data in the i^{th}SHEF. Then the representation of the SHEF in [20] would be

$$\mathbf{x}^T\mathbf{R}_i^T \begin{bmatrix} \widetilde{\mathbf{S}}_i^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_i\mathbf{x} = 0 \tag{1}$$

where $\widetilde{\mathbf{S}}_i^{-1}$ is the inverse matrix of $\widetilde{\mathbf{S}}_i$, $\mathbf{0}_{d\times 1}$ is a zero vector, $\mathbf{0}_{1\times d}$ is the transpose of $\mathbf{0}_{d\times 1}$, and $\mathbf{I}_{d\times d}$ is an identity matrix,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ 1 \end{bmatrix}, \quad \mathbf{R}_i = \begin{bmatrix} \mathbf{I}_{d\times d} & -\mathbf{c}_i \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix}, \quad \mathbf{c}_i = \begin{bmatrix} c_{i1} \\ c_{i2} \\ \vdots \\ c_{id} \end{bmatrix}.$$

From (1), two candidate SHEFs should be checked for overlap if they satisfy the following conditions.

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = 0 \quad \text{and} \quad \mathbf{x}^T\mathbf{B}\mathbf{x} = 0,$$

where

$$\mathbf{A} = \mathbf{R}_A^T \begin{bmatrix} \widetilde{\mathbf{S}}_A^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_A, \quad \mathbf{B} = \mathbf{R}_B^T \begin{bmatrix} \widetilde{\mathbf{S}}_B^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_B.$$

In [20], the intersection of SHEF$_A$ and SHEF$_B$ can be described with an eigenvalue of the matrix $\mathbf{A}^{-1}\mathbf{B}$ or $\mathbf{B}^{-1}\mathbf{A}$ where

$$\mathbf{A}^{-1}\mathbf{B} = \left(\mathbf{R}_A^T \begin{bmatrix} \widetilde{\mathbf{S}}_A^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_A\right)^{-1} \mathbf{R}_B^T \begin{bmatrix} \widetilde{\mathbf{S}}_B^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_B \tag{2}$$

for SHEF$_A$ and SHEF$_B$, respectively.

$$\mathbf{A}^{-1}\mathbf{B} = \mathbf{R}_A^{-1} \begin{bmatrix} \widetilde{\mathbf{S}}_A & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} (\mathbf{R}_A^T)^{-1} \mathbf{R}_B^T \begin{bmatrix} \widetilde{\mathbf{S}}_B^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_B \tag{3}$$

Since for any SHEF$_i$

$$\mathbf{R}_i^{-1} = \begin{bmatrix} \mathbf{I}_{d\times d} & \mathbf{c}_i \\ \mathbf{0}_{1\times d} & 1 \end{bmatrix},$$

hence, the matrix $\mathbf{A}^{-1}\mathbf{B}$ can be simplified to our block matrix as

$$\mathbf{A}^{-1}\mathbf{B} = \begin{bmatrix} \mathbf{D} & -\mathbf{D}\mathbf{c}_B + \mathbf{c}_A \\ \mathbf{F} & -\mathbf{F}\mathbf{c}_B + 1 \end{bmatrix}_{(d+1)\times(d+1)} \tag{4}$$

where

$$\begin{aligned} \mathbf{F}_{1\times d} &= (-\mathbf{c}_A + \mathbf{c}_B)^T \widetilde{\mathbf{S}}_B^{-1} \qquad (5) \\ \mathbf{D}_{d\times d} &= \widetilde{\mathbf{S}}_A \widetilde{\mathbf{S}}_B^{-1} + \mathbf{c}_A \mathbf{F} \qquad \square \end{aligned}$$

APPENDIX C

Proof of Theorem 3.3.1: Suppose the projected $\mathbf{x}$ is inside the projected SHEF ($D(\mathbf{x},\mathbf{c},\mathbf{S}) < 1$) on the discriminant vector $\mathbf{w}_p$ defined in (3.24). The projection must satisfy the following inequality.

$$|\mathbf{w}_p^T(\mathbf{x}-\mathbf{c})| \; < \; r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p} \; . \tag{6}$$

Substitute $\mathbf{w}_p = \dfrac{\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})}{||\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})||}$ in (6) to obtain

$$|(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})| \quad < \quad r\sqrt{(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})} \; . \tag{7}$$

It is obvious to see that

$$0 < (\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c}) < r^2. \tag{8}$$

Equation (8) implies that $\mathbf{x}$ is inside the SHEF scaled by r^2 in the original space following (2.4). $\square$

The conditions of *outside* and *boundary* can be proved by the similar argument of these inequalities $|\mathbf{w}_p^T(\mathbf{x}-\mathbf{c})| \; > \; r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$ and $|\mathbf{w}_p^T(\mathbf{x}-\mathbf{c})| \; = \; r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$, respectively.

BIOGRAPHY

Name	Mr. Perasut Rungcharassang
Date of Birth	30 April 1987
Place of Birth	Bangkok, Thailand
Education	B.S. (Mathematics), Kasetsart University, 2008
	M.Sc., Chulalongkorn University, 2012