ONE-PASS-THROW-AWAY LEARNING OF TEMPORAL CLASS-SHIFT BY MULTI-STRATUM NETWORK

Mr. Mongkhon Thakong

A Dissertation Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy Program in Computer Science and Information Technology
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2015

การเรียนรู้โดยอ่านข้อมูลครั้งเดียวแล้วทิ้งของการเปลี่ยนคลาสตามเวลาโดยเครือข่ายแบบหลายชั้น

นายมงคล ทะกอง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

Thesis Title	ONE-PASS-THROW-AWAY LEARNING OF TEMPORAL CLASS-SHIFT BY MULTI-STRATUM NETWORK
By	Mr. Mongkhon Thakong
Field of Study	Computer Science and Information Technology
Thesis Advisor	Assistant Professor Suphakant Phimoltares, Ph.D.
Thesis Co-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

(Associate Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

_______________________________Chairman

(Saichon Jaiyen, Ph.D.)

_______________________________Thesis Advisor

(Assistant Professor Suphakant Phimoltares, Ph.D.)

_______________________________Thesis Co-Advisor

(Professor Chidchanok Lursinsap, Ph.D.)

_______________________________Examiner

(Associate Professor Peraphon Sophatsathit, Ph.D.)

_______________________________Examiner

(Assistant Professor Saranya Maneeroj, Ph.D.)

_______________________________External Examiner

(Associate Professor Sartra Wongthanavasu, Ph.D.)

มงคล ทะกอง : การเรียนรู้โดยอ่านข้อมูลครั้งเดียวแล้วทิ้งของการเปลี่ยนคลาสตามเวลาโดยเครือข่ายแบบหลายชั้น (ONE-PASS-THROW-AWAY LEARNING OF TEMPORAL CLASS-SHIFT BY MULTI-STRATUM NETWORK) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร.ศุภกานต์ พิมลธเรศ, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม: ศ. ดร.ชิดชนก เหลือสินทรัพย์, 67 หน้า.

การศึกษาปัญหาที่เกิดขึ้นจากการประยุกต์ใช้ในธุรกิจหลายๆ ด้านของข้อมูลในลักษณะที่เป็นแบบสตรีมมิ่งนั้น ข้อมูลอาจจะหมดอายุชั่วคราวและเปลี่ยนแปลงคลาสได้ ถ้าเกิดเหตุการณ์ที่ไม่ทราบสาเหตุแน่ชัดในลักษณะที่คลาสของข้อมูลหมดอายุขึ้น โครงสร้างของการเรียนรู้ข้อมูลที่เกี่ยวข้องและการเชื่อมโยงควรจะต้องถูกลบออกไป ดังนั้นการเรียนรู้แบบใหม่โดยการใช้โครงสร้างแบบยืดหยุ่นซึ่งเรียกว่า การเรียนรู้เครือข่ายแบบหลายชั้น ได้ถูกเสนอเพื่อจัดการกับปัญหาที่ข้อมูลมีการเปลี่ยนแปลงคลาส การเรียนรู้ที่เป็นแนวคิดใหม่นี้จัดการกับปัญหาข้อมูลสตรีมมิ่งที่หมดอายุและเปลี่ยนแปลงคลาส โดยจะสามารถเรียนรู้ได้เร็วขึ้นและใช้หน่วยความจำน้อย รูปแบบการเรียนรู้โดยอ่านครั้งเดียวแล้วทิ้งในลักษณะรูปแบบของฟังก์ชันแบบเรียกซ้ำ ซึ่งผลการทดลองแสดงให้เห็นว่าอัลกอริทึมที่เสนอมีประสิทธิภาพที่ดีกว่าวิธีการอื่น ทั้งประสิทธิภาพของเวลาการเรียนรู้และการใช้พื้นที่หน่วยความจำ

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	
		ลายมือชื่อ อ.ที่ปรึกษาหลัก	
สาขาวิชา	วิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ	ลายมือชื่อ อ.ที่ปรึกษาร่วม	
ปีการศึกษา	2558		

5273922223 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS: DATA STREAM CLASSIFICATION / INCREMENTAL LEARNING / EXPIRED DATA / CONCEPT DRIFT / NONSTATIONARY ENVIRONMENTS

MONGKHON THAKONG: ONE-PASS-THROW-AWAY LEARNING OF TEMPORAL CLASS-SHIFT BY MULTI-STRATUM NETWORK. ADVISOR: ASST. PROF. SUPHAKANT PHIMOLTARES, Ph.D., CO-ADVISOR: PROF. CHIDCHANOK LURSINSAP, Ph.D., 67 pp.

The problem of learning streaming non-expired and temporally expired data occurring in various business applications was studied. If there exists a class whose all data are eventually expired with some unknown reasons, then the relevant neurons and their links must be entirely removed. A new learning based on dynamic network structure called Multi-Stratum Network Learning (MSNL) was proposed to cope with this problem of data life change. Furthermore, to speed up the learning time and to maintain a minimum space complexity for streaming data, the new concept of one-pass-throw-away learning in forms of recursive functions for handling the expiration class was introduced. The experimental results signified that the proposed algorithm outperformed the other incremental-like learning algorithms in terms of time and space complexities.

Department:	Mathematics and Computer Science	Student's Signature	....
		Advisor's Signature	....
Field of Study:	Computer Science and Information Technology	Co-Advisor's Signature	....
Academic Year:	2015		

ACKNOWLEDGEMENTS

I am deeply grateful to my advisors, Assistant Professor Dr. Suphakant Phimoltares and Professor Dr. Chidchanok Lursinsap, for their invaluable guidance throughout the study of this dissertation. Their insightful advice and knowledge greatly helped me to work the completion of this dissertation. Furthermore, I am also grateful to Dr. Saichon Jaiyen, who suggests me in all the development of Ph.D. research.

I am also like to extend my sincere thank to Associate Professor Dr. Peraphon Sophatsathit, Assistant Professor Dr. Saranya Maneeroj, and Associate Professor Dr. Sartra Wongthanavasu for their suggestions in this dissertation.

I would like to express my gratitude to Udon Thani Rajabhat University for their scholarship in supporting me during my education.

I would like to express my gratitude to Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University for supporting me during my education.

I am grateful to the Advanced Virtual and Intelligence Computing (AVIC) center for their material support in accommodating all the study of this dissertation.

I would also like to thank all my colleagues at AVIC center, who give me for their friendship and encouragement throughout the study.

I would like to thank my parents and family for their great love and encouragement. I would not accomplish the Ph.D. degree without their love.

Additionally, I would like to thank those whose names are not mentioned here but have greatly inspired and encouraged me until the completion of this dissertation.

CONTENTS

Page

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I
INTRODUCTION

1.1 Problem identification

Current learning algorithms were designed to cope only with timeless data or stationary-class data. This implies that the lifetime of the data is assumed no expiration and the data belong to a certain class forever. But in some situations or scenarios, this implication is not always true. For example, changes in student status may include suspension, withdrawal, lapsing and reinstatement, and change of program under conditions of university even though the features of this student have never been changed. In this study, I focused on the development of a new learning algorithm and network structure to cope with data with expiration and data whose class can be temporally changed. The amount of training data is variable. However, if the training data are not assumed to gradually flow into the network, the actual amount of total data will overwhelm the space complexity of the learning network. When the class of any datum is expired or changed, its features are still the same. The network structure can learn this datum in the training process. But the structure cannot be used to distinguish this datum whether its class is expired or changed to another class in the testing process. This is because the features are firstly classified according to its target by a neuron α. When its class is changed to a new class, another neuron β is used to secondly learn this datum. Hence when predicting the actual class of this datum after its class change, both neurons α and β will response to this datum. This creates a non-deterministic situation.

To overcome these problems, a new incremental learning algorithm in various environments based on versatile elliptic basis function (VEBF) called a multi-stratum network has been proposed. The proposed learning algorithm can learn expired and class-changed data without storing all the previous training set. In abrupt changing environments, the proposed algorithm can preserve a good balance in both stability and plasticity using to maintain some relevant information in the network and to learn a new class in such this situation during training process. For improving the performance of the proposed algorithm in training process, the training data can be learned with a training datum or a chunk of training data for adapting the network.

1.2 Problem formulation

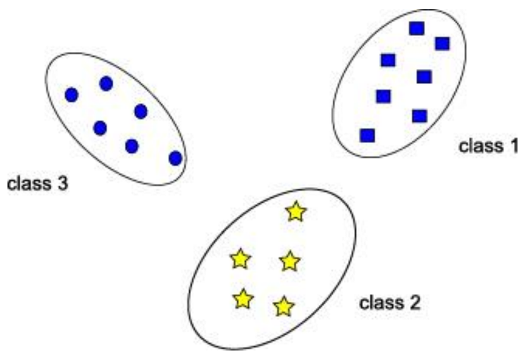
Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of training data. Each datum $\mathbf{x}_i$ consists of a d features, i.e., $\mathbf{x}_i = \{x_{i,1}, \ldots, x_{i,d}\}$ with a label of class target $\tau_i(k) \in \{0, 1, \ldots, m\}$ at time k. A datum is expired if $\tau_i(k) = 0$ otherwise it is in one of m classes. At the beginning all $\tau_i(1) \in \{1, \ldots, m\}$. After a period of time a, this datum $\mathbf{x}_i$ is expired which implies that $\tau_i(a) = 0$. However, datum $\mathbf{x}_i$ can be one of the queried data at any time and the classifier must be able to indicate that $\mathbf{x}_i$ is already expired. The following constraints are imposed on these studied problems.

1. For any datum $\mathbf{x}_i$, the feature set $\{x_{i,1}, \ldots, x_{i,d}\}$ is fixed regardless of time and class target.
2. For any datum $\mathbf{x}_i$, time k in $\tau_i(k)$ is randomly defined.
3. Once datum $\mathbf{x}_i$ is expired at time a, i.e. $\tau_i(a) = 0$, it is discarded forever from both training and testing processes for $k \geq a$.
4. Any datum $\mathbf{x}_i$ may or may not appear in testing set during the evaluation process.
5. Only streaming data are involved in this study. They are learned by one-pass-throw-away training process to maintain a linear learning time complexity with respect to the number of data.
6. Since streaming data are concerned, any datum $\mathbf{x}_i$ will be arbitrarily allowed to reenter or not to reenter the testing process.

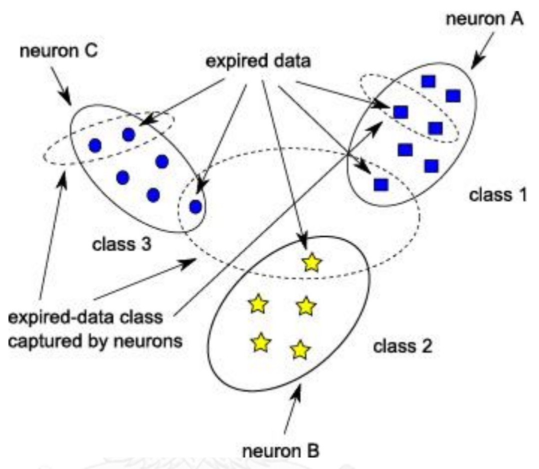
In this study, the problem of data expiration is transformed into the problem of learning class change by treating any datum whose life time is expired as a datum in class 0. This implies that the datum must be trained twice and captured by at least two sets of neurons of different classes. The first training process occurs when the datum is not in class 0. The second training process is when the datum changes its class when it is expired. It is possible that a non-deterministic situation may occur due to the capture of data by neurons from different classes. However, the network must be able to identify the exact class of any queried data. Although some data are expired, their class cannot be omitted from consideration to maintain the correctness of classification. Furthermore, if the number of expired data increases, then the size of the network may proportionally increase as well. This is an undesirable scenario. The increasing rate of network size should be slower than the increasing rate of the number of expired data. Therefore, the studied problem concerns the following issues.

1. What is the appropriate network structure to capture this datum before and after its class change?
2. How to perform one-pass-throw-away training in temporal class change situation for the data are streaming data?
3. How to identify the correct class of any datum whose class is temporally changed?
4. How to keep track of expired data with low space complexity?

Figure 1.1 shows an example of how expired data are handled. There are three classes denoted by *stars, thick dots, and squares* as shown in Figure 1.1(a). When some data in each class are expired, these expired data are captured by some neurons as shown in Figure 1.1(b). Note that any expired data are captured by two neurons, one from expiration class and another from non-expiration class. The proposed method must be able to handle this non-deterministic situation.



(a)



(b)

Figure 1.1 An example of how expired data are captured. There are three classes denoted by stars, thick dots, and squares. (a) Before some data are expired. (b) After the data are expired. All those expired data are captured by dashed ellipses.

1.3 Research objective

The main objective of this dissertation is to develop a new learning algorithm for classifying expired or class-changed data.

1.4 Scope of work

In this dissertation, the scope of work is constrained as follows:

1. The proposed algorithm is focused on the classification problems whose data are expired and changed class labels.
2. The benchmark data sets are taken from the University of California at Irvine (UCI) repository of machine learning database [1] to simulate the expiration or class change and popular data sets for the concept drift problems. Those data from the concept drift problems were given by U.S. National Oceanic and Atmospheric Administration collected from temperature, pressure, visibility, and other-related events of weather measurements as described in [2] and by New South Wales Electricity Market collected from time and demand fluctuations in Australia as described in [3].

1.5 Research advantage

The proposed learning algorithm can efficiently apply to various business applications whose data are expired and changed class labels, e.g., electricity market, weather prediction, spam and fraud detection. After learning, all trained data can be discarded forever.

1.6 Outline of the dissertation

The remainder of the dissertation is organized as follows. Chapter II reviews the related literatures and the relevant backgrounds. Chapter III describes the concept of my proposed methods. Chapter IV shows the experimental results. Chapter V concludes the study.

CHAPTER II
LITERATURE REVIEW

Incremental learning algorithms have been developed and widely applied for solving classification problems [4-8], such as data stream problems [9-12], large-scaled problems [6], and pattern recognition problems [13]. There are a lot of techniques that can be applied to solve these problems. For instance, the technique described in [11] built a decision tree incrementally to solve classification problem. Incremental kernel principal component analysis (IKPCA) [14], incremental principal component analysis (IPCA) [7], and incremental linear discriminant analysis (ILDA) [15] were proposed for online feature extraction in the classification system. The technique studied in [6] extended online incremental support vector machine (OI-SVM) to solve large-scaled problems consisting both in stationary environment and in non-stationary environment. For high dimensional data, [16] presented self-organizing incremental neural network by optimizing distance metric in learning process. The versatile elliptic basis function (VEBF) [4] can be used for one-pass-throw-away learning without storing all previous data. The adjusted self-organizing incremental neural network classifier (ASC) [8] automatically learns the number of prototypes needed to determine the decision boundary. In case of application to imaging field, incremental multiple-object recognition and localization (IMORL) [17] can automatically and adaptively learn from continuous video streams. These techniques can be applied to the real-world applications.

A similar problem of learning in non-stationary environment was studied. In this environment, data can change their classes over time. This leads to the dynamical change of class distribution in the data space. The learning in non-stationary environments is known as concept drift as described in [18, 19]. Types of changing in non-stationary environment consist of: gradual or trend changes, and abrupt (sudden) changes [3]. The recent learning algorithm under various environments has been developed and widely studied in several classification problems such as electricity market prediction [20], weather prediction [3], credit card fraud protection [21], spam detection [10], and surgery prediction [22]. For instance, the technique based on dynamic fuzzy pattern matching (DFPM) [23] was proposed to recognize pattern for the online monitoring of non-stationary environments. Linear discriminant analysis was applied by [24] to observe and model under gradual or abrupt changes in data distribution. Ensemble of subset online sequential extreme learning machine (ESOS-ELM) [25] was proposed a change detection technique to promptly detect concept drift situations. The trigger-based ensemble (TBE) [22] was designed to handle concept drift in surgery prediction that the guidelines of referral were changed due to scientific developments and clinical practices. The technique of multiple expert systems also called ensemble systems under concept drift situations [26] was proposed to adjust each expert based on its loss function and weighted majority vote (WMV) [27] provided to be used adaptation bounds on the loss function. The difficult challenge of learning in various environments is how to preserve all acquired knowledge, so that it must decide to whatever knowledge should replace or retain for improving of its performance [28]. However, learning should retain any previously acquired knowledge which is still essentially known as “stability-plasticity dilemma”, where “stability” means to maintain existing knowledge and “plasticity” describes the ability to learn new knowledge [3]. In recent machine learning survey under non-stationary environments, the developed techniques have been proposed for solving problems under various assumptions and the following issues:

Window-based technique: The earliest technique has been developed with underlying non-stationary environments by moving window containing block of the last training data. With this technique, the window size is considered. If a longer window is used for adapting, an environment of a system is slowly varied but stable and well trained classifier is obtained. On the other hand, a smaller window used for adjusting the system reacts quickly in fast changing environment but its performance may be low due to insufficient training data in the window. For this technique, selecting or adapting the window size is further studied [19]. The classifier proposed in [29] is based on windows of various sizes, which is more versatile and able to learn abrupt change of class concept than the classifier with a window of single size.

Ensemble of classifiers: Many recent techniques use multiple classifiers for solving problems in non-stationary environments. SEA [29] is the first ensemble of classifiers has been proposed for learning problems in non-stationary environments to each consecutive time window of training set. Learn++.NSE [3] can learn in these situations and can provide well-modeled knowledge of a good balance in both stability and plasticity. In addition, it still does not discard any of the classifiers which may contain relevant information for learning a new classifier in the future. They are composed of several classifiers which are combined to gain the final hypothesis. However, it seems that this solution consumes large time and space complexities.

Basically, the incremental learning algorithms not only append the training data when the learned data are fed into the network sequentially, but also can adjust the network during the learning process without storing all the previous data. The following actions are carried out [30]:

(1) Aggregating the new data into the existing knowledge without storing all the previous data.

(2) Retaining all previous knowledge.

Many researchers have proposed several techniques to maintain the above-mentioned criterion and to increase their performance. Although many incremental learning algorithms, such as VEBF [4], ASC [8], IPCA [7], ILDA [16], IMORL [17], the learn++ family of algorithms (e.g., learn++.NC [31], learn++.MF [32], and learn++.NSE [3]) in non-stationary environments were proposed, they still do not deal with the problem of data that expires over the time. The VEBF algorithm is interesting among all incremental learning algorithms, because it can learn by using only incoming datum and consumes less space and time complexities. However, the VEBF algorithm cannot be applied is situations where the class labels are changed over time or the data are expired in the testing process.

2.1 VEBF learning algorithm

This section provides some backgrounds related to the studied problem and proposed algorithm. The proposed algorithm adapted some partial concept of one-pass-throw-away to create hidden neurons in the learning process. The summary of one-pass-throw-away learning and VEBF learning algorithm in [4] is the following.

The concept of VEBF learning algorithm is based on the operation of capturing one incoming datum at a time. If the incoming datum does fall into the inside of any VEBF neuron, then no new VEBF neuron is introduced to capture this datum. Otherwise, a new small VEBF neuron is introduced into the network to capture this datum. Furthermore, if there are many VEBF neurons capturing data of the same class and locating close to each other, then these VEBF neurons are grouped and replaced by a larger VEBF neuron. Once a datum is captured, it is completely discarded from the training process. The brief VEBF training algorithm is given in Algorithm 1 and the summary of VEBF algorithm can be described by steps of flowchart as shown in Figure 2.1.

Algorithm 1: VEBF Learning Algorithm

```
1.  Let $A_0$ be the initial width vector.
2.  Present the training datum $\mathbf{x}_j$.
3.  **If** there exists the hidden neurons **then**
4.      Find a closest hidden neuron labeled with the same class as the class of $\mathbf{x}_j$
5.      Let $\mathbf{C}_k$ be the center vector of the closest VEBF neuron $\mathbf{\Omega}_k$ found in Step 4.
6.      Update the center vector $\mathbf{C}_k$ by including $\mathbf{x}_j$.
7.      **If** $\mathbf{x}_j$ lies inside $\Omega_k$ **then**
8.          Update parameters of $\mathbf{\Omega}_k$ based on the direction of data distribution.
9.      **Else**
10.         Introduce a new neuron.
11.     **EndIf**
12. **Else**
13.     Introduce a new neuron.
```

```
14. **EndIf**
15. **If** there exists any two close neurons of the same class **then**
16.     Merge the two neurons into one new neuron using Equations (5) – (9).
17. **EndIf**
18. Go to step 2 until the training set is empty.
```

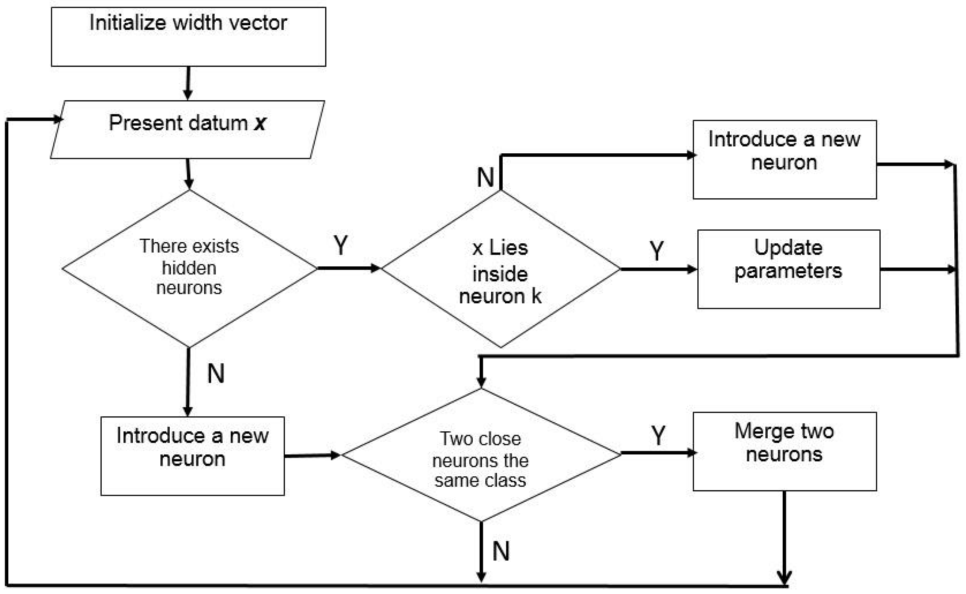


Figure 2.1 The summary of VEBF learning algorithm.

The output of the k^{th} neuron with respect to an input $\mathbf{x}$ is computed from a rotated elliptic function shown in Equation (1).

$$\Psi_k(\mathbf{x}) = \sum_{i=1}^{d} \frac{((\mathbf{x}-\mathbf{C}_k)^T \mathbf{u}_i)^2}{\sigma_i^2} - 1 \tag{1}$$

where $\mathbf{C}_k$ is the center vector of ellipse, $\mathbf{u}_i$ is the i^{th} eigenvector of data covariance matrix distribution, and σ_i^2 is the variance or eigenvalue of $\mathbf{u}_i$.

For any hidden neuron Ω_k, the relevant parameters which are the new center, new covariance matrix, and new variance must be computed. Since all trained data are discarded and only the new incoming datum is used for adjusting these parameters, therefore the computation of these parameters must be written in forms of recursive functions. The following attributes are defined and involved in the adjusting process.

$\mathbf{C}_k$ is the center of $\mathbf{\Omega}_k$. $\mathbf{S}_k$ is the covariance matrix of data captured by $\mathbf{\Omega}_k$. N_k is the number of data captured by $\mathbf{\Omega}_k$. A_k is the width vector of $\mathbf{\Omega}_k$. l_k is the class label of $\mathbf{\Omega}_k$. Suppose a new datum $\mathbf{x}_{N_k+1}$ arrives and falls inside the boundary of $\mathbf{\Omega}_k$. The value of $\Psi_k(\mathbf{x})$ must be less or equal to 0.

The new center $\mathbf{C}_k^{(new)}$ of $\mathbf{\Omega}_k$ can be computed from the old center by $\mathbf{C}_k^{(old)}$ the following recursive function.

$$\mathbf{C}_k^{(new)} = \frac{N_k}{N_k+1}\mathbf{C}_{(k)}^{(old)} + \frac{\mathbf{x}_{N_k+1}}{N_k+1} \tag{2}$$

The new covariance matrix $\mathbf{S}_k^{(new)}$ of $\mathbf{\Omega}_k$ can be formed by the following recursive computation with the old covariance matrix $\mathbf{S}_k^{(old)}$ as follows.

$$\begin{aligned}\mathbf{S}_k^{(new)} = {} & \frac{N_k}{N_k+1}\mathbf{S}_k^{(old)} + \frac{\mathbf{x}_{N_k+1}\mathbf{x}_{N_k+1}^T}{N_k+1} - \frac{\mathbf{C}_k^{(old)}(\mathbf{C}_k^{(old)})^T}{N_k+1} - \\ & \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T + \mathbf{C}_k^{(old)}(\mathbf{C}_k^{(old)})^T\end{aligned} \tag{3}$$

The total data captured by $\mathbf{\Omega}_k$ becomes N_k+1. The new variance $\sigma_i^{(new)}$ of each eigenvector $\mathbf{u}_i$ can be easily computed from the old variance $\sigma_i^{(old)}$ by the following equation.

$$\sigma_i^{(new)} = \sigma_i^{(old)} + \left|(\mathbf{C}_k^{(new)} - \mathbf{C}_k^{(old)})^T\mathbf{u}_i\right|, \quad 1 \le i \le d \tag{4}$$

Suppose any two hidden neurons $\mathbf{\Omega}_a$ and $\mathbf{\Omega}_b$ are merged into one new hidden neuron $\mathbf{\Omega}_c$. Let $\mathbf{\Omega}_a = (\mathbf{C}_a^{(old)}, \mathbf{S}_a^{(old)}, N_a, A_a, l_a)$ and $\mathbf{\Omega}_b = (\mathbf{C}_b^{(old)}, \mathbf{S}_b^{(old)}, N_b, A_b, l_b)$ be two hidden neurons. After merging, the new attributes can be computed by the following equations.

$$\mathbf{C}_c^{(new)} = \frac{1}{N_a+N_b}\left(N_a\mathbf{C}_a^{(old)} + N_b\mathbf{C}_b^{(old)}\right) \tag{5}$$

$$\begin{aligned}\mathbf{S}_c^{(new)} = {} & \frac{N_a}{N_a+N_b}\mathbf{S}_a^{(old)} + \frac{N_b}{N_a+N_b}\mathbf{S}_b^{(old)} + \\ & \frac{N_aN_b}{(N_a+N_b)^2}\left(\mathbf{C}_a^{(old)} - \mathbf{C}_b^{(old)}\right)\left(\mathbf{C}_a^{(old)} - \mathbf{C}_b^{(old)}\right)^T\end{aligned} \tag{6}$$

$$N_c = N_a + N_b \tag{7}$$

$$\sigma_i^{(new)} = \sqrt{2\pi|\lambda_i|}, \quad i = 1, 2, \ldots, d \tag{8}$$

$$d_c = d_a = d_b \tag{9}$$

λ_i is the i^{th} eigenvalue of the new covariance matrix $\mathbf{S}_c^{(new)}$. After merging, Ω_a and Ω_b are removed from the network.

2.2 Merging LDA operation

A concept of LDA is mapping the projection matrix Φ^* in order to maximize class separability of the data set. The projection matrix Φ^* is maximized as follows:

$$\Phi^* = \arg\max_{\Phi} \frac{|\Phi^T \mathbf{B} \Phi|}{|\Phi^T \mathbf{W} \Phi|} \tag{10}$$

where $\mathbf{B}$ is the between-class scatter matrix and $\mathbf{W}$ is the within-class scatter matrix. Those are given by

$$\mathbf{B} = \sum_{k=1}^{m} \mathbf{B}_k = \sum_{k=1}^{m} N_k (\mathbf{C}_k - \mathbf{C})(\mathbf{C}_k - \mathbf{C})^T \tag{11}$$

$$\mathbf{W} = \sum_{k=1}^{m} \mathbf{S}_k \tag{12}$$

$\mathbf{C}_k$, $\mathbf{S}_k$, N_k are the center vector, the covariance matrix, and the number of data of class k, respectively and $\mathbf{C}$ is the center vector of the data set.

The projection matrix Φ^* that is a $d \times d$ matrix whose columns correspond to the discriminant eigenvectors obtained by solving the following eigenvalue problem:

$$\mathbf{D}\Phi^* = \Phi^* \Lambda \tag{13}$$

where $\mathbf{D} = \mathbf{W}^{-1}\mathbf{B}$ and Λ is an eigenvalue matrix.

In the LDA merging operation discussed by [15], the models Ω_a and Ω_b can be denoted as the following 3-tuples:

$$\Omega_a = (\{\mathbf{S}_a\},\{\mathbf{C}_a\},\{N_a\}) \tag{14}$$

$$\Omega_b = (\{\mathbf{S}_b\},\{\mathbf{C}_b\},\{N_b\}) \tag{15}$$

where $\mathbf{S}_a$ and $\mathbf{S}_b$ are the class covariance matrices, $\mathbf{C}_a$ and $\mathbf{C}_b$ are the class center vectors, and N_a and N_b are the number of data captured by the model $\mathbf{\Omega}_a$ and $\mathbf{\Omega}_b$, respectively. Suppose there are m classes of $N^{(old)}$ data vectors and all-instances center vector $\mathbf{C}^{(old)}$ in the data set. Then, incremental LDA updates the all-instances center vector $\mathbf{C}^{(new)}$, the within-class scatter matrix $\mathbf{W}^{(new)}$, and between-class scatter matrix $\mathbf{B}^{(new)}$ given by the following.

$$\mathbf{C}^{(new)} = (N^{(old)}\mathbf{C}^{(old)} + \sum_{i}^{N_b} \mathbf{z}_i) / (N^{(old)} + N_b)\,, \tag{16}$$

$$\mathbf{B}^{(new)} = \sum_c N_c (\mathbf{C}_c^{(new)} - \mathbf{C}^{(new)})(\mathbf{C}_c^{(new)} - \mathbf{C}^{(new)})^T\,, \tag{17}$$

$$\mathbf{W}^{(new)} = \sum_c \mathbf{S}_c^{(new)}, \text{for } c = 1,2,\ldots,m \tag{18}$$

where $\mathbf{C}_c^{(new)}$ is the new center vector and $\mathbf{S}_c^{(new)}$ is the new covariance matrix of class c given by using Equations (5) and (6), respectively.

CHAPTER III
PROPOSED METHOD

This study concerns two types of classes. The first type is the class of all expired data. The second type is the classes of non-expired data. To distinguish these two classes in this discussion, the class of first type is called ***expiration class*** and any class of the second type is called ***live class***. When a datum in expiration class, it is treated as a datum in class 0. This datum must be learned by a sub-network of class 0. However, if all data in class i are expired, the sub-network of class i must be entirely removed from the learning system. This situation is obviously different from the scenario studied in [4] and the others' previously mentioned. Those studies concerned only the condition of increasing new data which imply that the network is expanded throughout the training period. But in the studied case, the structure of network can be expanded or shrunk according to the temporal status of the incoming data. Another different issue from the others' studies is the problem of how to reconcile the indeterminate situation as previously discussed and the expansion of expired data. Regardless of the class, any expired data must be included in class 0. This implies that if all data are expired and the structure of sub-network for handling expired data is not appropriately designed, then the space complexity of the structure will be very high. The difficulty of my scenario is the arbitrary class change of data. There are two important scenarios to be considered.

The first scenario concerns the arbitrary class change to any other live classes. Datum $\mathbf{x}_i$ may change its classes several times throughout its lifetime but its features never change. This implies that the location of datum $\mathbf{x}_i$ is fixed in the features space. When there is a class change, datum $\mathbf{x}_i$ must be covered by function $\Psi_k(\mathbf{x}_i)$ of the most recent class. In response to the class query of datum $\mathbf{x}_i$, the most recent class of $\mathbf{x}_i$ must be tested first. The problem is how to keep track of these temporal class changes. Figure 3.1 shows an example of class change. At the beginning, $\mathbf{x}_i$ is in class 1 and, later on, its class changes to classes 2, 3, and 1, respectively. Hence, the most recent class is class 1 and the function $\Psi_k(\mathbf{x}_i)$ of class 1 must be examined first if there is class query of $\mathbf{x}_i$. Furthermore, datum $\mathbf{x}_i$ must be deleted from those previous classes 2 and 3. The members in each live class can vary according to the class change of each datum.

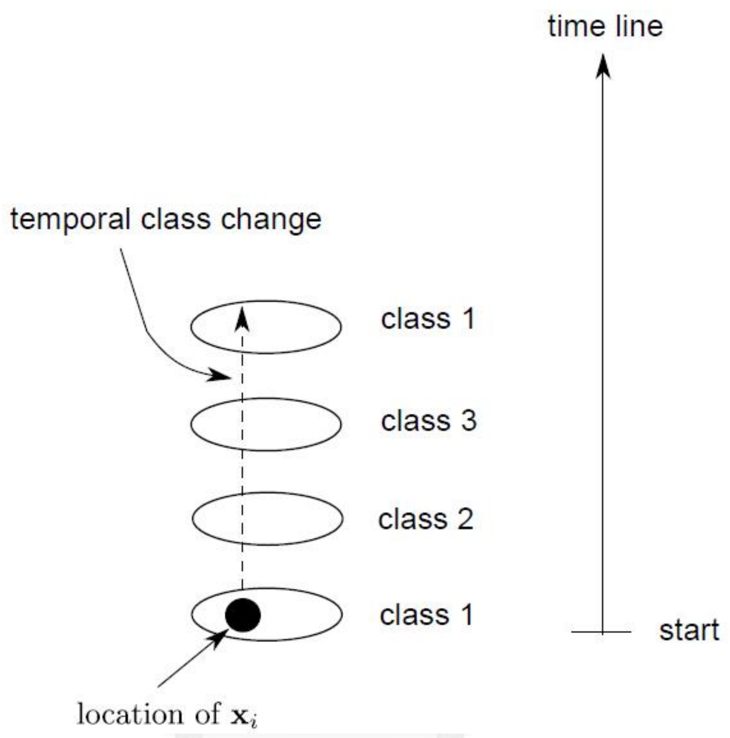


Figure 3.1 An example of temporal class change.

The second scenario is when datum $\mathbf{x}_i$ is expired and moved to class 0. Obviously, class 0 will be the permanent class of $\mathbf{x}_i$ after its expiration. Any class query concerning datum $\mathbf{x}_i$ after its expiration must receive the answer that $\mathbf{x}_i$ is in class 0. Furthermore, the number of members in class 0 keeps increasing. This implies that the class query of any datum $\mathbf{x}_j$ must be checked with the data in class 0 first. If $\mathbf{x}_j$ is not in class 0, then the existence of $\mathbf{x}_j$ in other classes will be checked next. Notice that the sub-network representing Class 0 must be retained throughout the training and testing periods due to the permanent class changes of all data in this class.

To resolve those discussed problems, the following issues will be concentrated in my proposed method. The first issue is created with the structure of the network of the non-expired and expired data. The structure must be expandable and shrinkable according to the situations. The second issue is the representation of expired data and deterministic testing method. Each issue is detailed in the following sections.

Figure 3.2 shows diagram of the overview of learning process consisting of proposed width vector initialization algorithm, algorithms for learning non-expired data, expired data, and class-changed data using VEBF and Hybrid LDA-PCA algorithms. To improve the performance of learning algorithm, VEBF algorithm has been applied to find axes with maximum variance whereas Hybrid LDA-PCA algorithm will be applied to seek the axes of the neuron for best class separability. In terms of non-deterministic situation that data are expired or changed classes, the updated parameters consisting of center vector and covariance matrix can be computed by using theorems 1 and 2. Furthermore, the updated parameters consisting of between-class scatter matrix and within-class scatter matrix can be computed by using proposition 1. To limit space complexity of increasing the expired data, a special structure of class 0 provided for storing all expired data in order to update vectors $\mathbf{u}$ and Δ using theorems 3 and 4 is proposed.

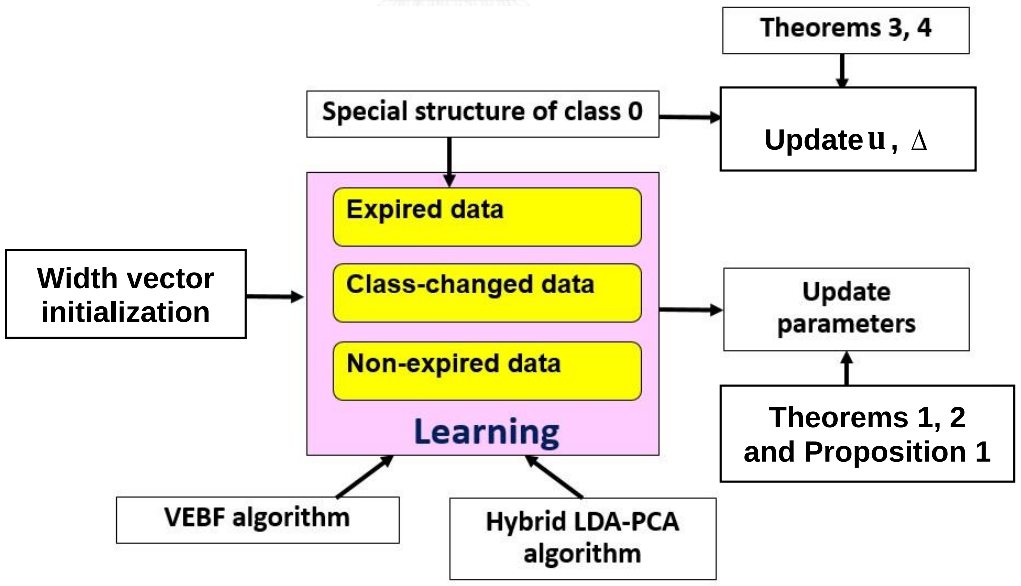


Figure 3.2 Overview of learning process.

3.1 Proposed width vector initialization algorithm

The initial width of a VEBF is very important to the learning speed and outcome accuracy. If it is too large, the VEBF may cover some data from the other classes. But if it is too small, it must be temporally adjusted to cover the new incoming data, which obviously increases the unnecessary computational time. As mentioned before, one of the limitations of VEBF algorithm is to find the appropriate initial width σ_k computed by the following equation.

$$\sigma_k = \delta D_{av};\ 1 \le k \le d \tag{19}$$

where

$$D_{av} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left\|\mathbf{x}_i - \mathbf{x}_j\right\| \tag{20}$$

As shown in Equation (19), the initial value of σ_k can be computed from the Euclidean distances among data points in a training set. The constant δ is used to adjust the value of D_{av} to make σ_k close to its actual value which is unknown. Different training sets require different δ values. To avoid this difficulty of determining the value of δ, the following steps were proposed to compute the initial σ_k for $1 \le k \le d$. No concept of using a constant δ to adjust D_{av} is deployed in my algorithm. The initial width should be derived from the density of distances among all pairs of training data as follows.

Algorithm 2: Initializing Width of VEBF

```
Input: Training set $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$.
Output: Initial values of $\sigma_1, \dots, \sigma_d$.
 1. For each $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$ and $i \ne j$ do
 2.     Compute $D_{i,j} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|$.
 3. EndFor
 4. Let $I$ be a number of intervals.
```

```
5.  Compute $\gamma = \frac{\max_{i,j}(D_{i,j}) - \min_{i,j}(D_{i,j})}{I}$ be the value of instance.
6.  Let $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_I\}$ be a set of bins for storing $D_{i,j}$.
7.  Let $q = \min_{i,j}(D_{i,j})$.
8.  For $\forall i, j;\ D_{i,j}$ do
9.      Put $D_{i,j}$ in bin $\mathbf{b}_k$ where $k = \left\lceil \frac{D_{i,j} - q}{\gamma} \right\rceil + 1$.
10. EndFor
11. If $k$ has maximum number of $D_{i,j}$, then
12.     Set initial $\sigma_1 = \sigma_2 = \ldots = \sigma_d = q + k - 1 + \frac{\gamma}{2}$.
13. EndIf
```

3.2 Updating center vector, covariance matrix, and scatter matrices

Unlike the scenario in [4], some data in any VEBF of every class except class 0 in my study can be removed due to their expiration or added when they are new incoming data. This implies that the structure of a VEBF must be expandable and shrinkable. Furthermore, the center vector and covariance matrix must be re-computed. When adding a new datum, the recursive functions for re-computing the center and covariance of any VEBF were already given in [4] and the within-class and between-class scatter matrices of LDA concept were discussed in [5] as well. But when removing a datum or a chunk of data from a network, the recursive functions for re-computing the center vector and covariance matrix have not been discussed before. Moreover, the within-class and between-class scatter matrices can be also computed in the recursive functions. The details of these recursive functions are the following.

Theorem 1. For Ω_k of any non-expiration class, let $\mathbf{X}$ in R^d be a set of current N data vectors. Suppose $\mathbf{C}_k^{(old)}$ is the center vector of set $\mathbf{X}$ and $\mathbf{S}_k^{(old)}$ is the covariance matrix. If $\mathbf{x}_j \in \Omega_k$ is expired and moved to class 0, then

$$\mathbf{C}_k^{(new)} = \frac{N}{N-1}\mathbf{C}_k^{(old)} - \frac{\mathbf{x}_j}{N-1} \tag{21}$$

and

$$\mathbf{S}_k^{(new)} = \frac{N}{N-1}\mathbf{S}_k^{(old)} + \frac{1}{N}(-\mathbf{C}_k^{(new)} + \mathbf{x}_j)(\mathbf{C}_k^{(new)} - \mathbf{x}_j)^T \ . \tag{22}$$

Proof The center vectors $\mathbf{C}_k^{(new)}$ and $\mathbf{C}_k^{(old)}$ are

$$\begin{aligned}\mathbf{C}_k^{(new)} &= \frac{1}{N-1}\left(\sum_{i=1}^{N}\mathbf{x}_i - \mathbf{x}_j\right) \\ &= \frac{N}{N-1}\mathbf{C}_k^{(old)} - \frac{\mathbf{x}_j}{N-1} \end{aligned} \tag{23}$$

$$\mathbf{C}_k^{(old)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \text{ and } N > 1. \tag{24}$$

The covariance matrices $\mathbf{S}_k^{(new)}$ and $\mathbf{S}_k^{(old)}$ are

$$\mathbf{S}_k^{(old)} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_k^{(old)}(\mathbf{C}_k^{(old)})^T \tag{25}$$

$$\mathbf{S}_k^{(new)} = \frac{1}{N-1}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \frac{\mathbf{x}_j\mathbf{x}_j^T}{N-1} - \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T \ . \tag{26}$$

Subtracting Equation (26) by (25), we have

$$\begin{aligned}\mathbf{S}_k^{(new)} - \mathbf{S}_k^{(old)} &= \frac{1}{N-1}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \frac{\mathbf{x}_j\mathbf{x}_j^T}{N-1} - \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T - \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}^{(old)}(\mathbf{C}_k^{old})^T \\ &= \frac{1}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_k^{(old)}(\mathbf{C}_k^{(old)})^T\right] - \frac{\mathbf{x}_j\mathbf{x}_j^T}{N-1} - \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T + \frac{N}{N-1}\mathbf{C}_k^{(old)}(\mathbf{C}_k^{(old)})^T \\ &= \frac{1}{N-1}\mathbf{S}_k^{(old)} - \frac{\mathbf{x}_j\mathbf{x}_j^T}{N-1} - \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T + \frac{N}{N-1}\mathbf{C}_k^{(old)}(\mathbf{C}_k^{(old)})^T \end{aligned} \tag{27}$$

Therefore,

$$\begin{aligned}\mathbf{S}_k^{(new)} &= \mathbf{S}_k^{(old)} + \frac{1}{N-1}\mathbf{S}_k^{(old)} - \frac{\mathbf{x}_j\mathbf{x}_j^T}{N-1} - \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T + \frac{N}{N-1}\mathbf{C}_k^{(old)}(\mathbf{C}_k^{old})^T \\ &= \frac{N}{N-1}\mathbf{S}_k^{(old)} + \frac{1}{N}\left(-\mathbf{C}_k^{(new)} + \mathbf{x}_j\right)\left(\mathbf{C}_k^{(new)} - \mathbf{x}_j\right)^T \end{aligned} \tag{28}$$

When a chunk of data is expired and moved to class 0, the recursive functions for computing the center vector and covariance matrix of Ω_k are summarized in the following theorem.

Theorem 2. For Ω_k of any non-expiration class, let $\mathbf{X}$ in R^d be a set of data chunks. Set $\mathbf{X}$ has N data points. Assume that a chunk of data $\mathbf{Z} \in \mathbf{X}$ whose $|\mathbf{Z}| = L$ is expired and moved to class 0. Let $\mathbf{C}_k^{(\mathbf{X})}$ be the center vector of set $\mathbf{X}$; $\mathbf{S}_k^{(\mathbf{X})}$ be the covariance matrix of $\mathbf{X}$; $\mathbf{C}_k^{(\mathbf{Z})}$ be the center vector of set $\mathbf{Z}$; and $\mathbf{S}_k^{(\mathbf{Z})}$ be the covariance matrix of $\mathbf{Z}$. The new center $\mathbf{C}_k^{(new)}$ and new covariance matrix $\mathbf{S}_k^{(new)}$ of set $\mathbf{X} - \mathbf{Z}$ can be computed as follows.

$$\mathbf{C}_k^{(new)} = \frac{N}{N-L}(N\mathbf{C}_k^{(\mathbf{X})} - L\mathbf{C}_k^{(\mathbf{Z})}) \tag{29}$$

and

$$\begin{aligned}\mathbf{S}_k^{(new)} = {} & \frac{N}{N-L}\mathbf{S}_k^{(\mathbf{X})} - \frac{L}{N-L}\mathbf{S}_k^{(\mathbf{Z})} - \\ & \frac{NL}{(N-L)^2}(\mathbf{C}_k^{(\mathbf{X})} - \mathbf{C}_k^{(\mathbf{Z})})(\mathbf{C}_k^{(\mathbf{X})} - \mathbf{C}_k^{(\mathbf{Z})})^T\end{aligned} \tag{30}$$

Proof Let

$$\mathbf{C}_k^{(\mathbf{X})} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{31}$$

and

$$\mathbf{C}_k^{(\mathbf{Z})} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{x}_l \ . \tag{32}$$

From Theorem 1, the new center vector can be written as

$$\mathbf{C}_k^{(new)} = \frac{N}{N-1}\mathbf{C}_k^{(old)} - \frac{\mathbf{x}_j}{N-1} \tag{33}$$

Similarly, the new center vector of a chunk of training samples is easily obtained as follows:

$$\mathbf{C}_k^{(new)} = \frac{1}{N-L}(N\mathbf{C}_k^{(\mathbf{X})} - L\mathbf{C}_k^{(\mathbf{Z})}) \tag{34}$$

For the covariance matrix, let

$$\mathbf{S}_k^{(\mathbf{X})} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \mathbf{C}_k^{(\mathbf{X})}(\mathbf{C}_k^{(\mathbf{X})})^T \tag{35}$$

$$\mathbf{S}_k^{(\mathbf{Z})} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{x}_l\mathbf{x}_l^T - \mathbf{C}_k^{(\mathbf{Z})}(\mathbf{C}_k^{(\mathbf{Z})})^T \tag{36}$$

Hence,

$$\begin{aligned}
\mathbf{S}_k^{(new)} &= \frac{1}{N-L}\left(\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T - \sum_{l=1}^{L}\mathbf{x}_l\mathbf{x}_l^T\right) - \mathbf{C}_k^{(new)}(\mathbf{C}_k^{(new)})^T \\
&= \frac{1}{N-L}\left(N\mathbf{S}_k^{(\mathbf{X})} + N\mathbf{C}_k^{(\mathbf{X})}(\mathbf{C}_k^{(\mathbf{X})})^T - L\mathbf{S}_k^{(\mathbf{Z})} - L\mathbf{C}_k^{(\mathbf{Z})}(\mathbf{C}_k^{(\mathbf{Z})})^T\right) - \\
&\quad \frac{1}{(N-L)^2}\left(N^2\mathbf{C}_k^{(\mathbf{X})}(\mathbf{C}_k^{(\mathbf{X})})^T + L^2\mathbf{C}_k^{(\mathbf{Z})}(\mathbf{C}_k^{(\mathbf{Z})})^T - NL\mathbf{C}_k^{(\mathbf{X})}(\mathbf{C}_k^{(\mathbf{Z})})^T - NL\mathbf{C}_k^{(\mathbf{Z})}(\mathbf{C}_k^{(\mathbf{X})})^T\right) \\
&= \frac{N}{N-L}\mathbf{S}_k^{(\mathbf{X})} - \frac{L}{N-L}\mathbf{S}_k^{(\mathbf{Z})} - \frac{NL}{(N-L)^2}\mathbf{C}_k^{(\mathbf{X})}(\mathbf{C}_k^{(\mathbf{X})})^T - \frac{NL}{(N-L)^2}\mathbf{C}_k^{(\mathbf{Z})}(\mathbf{C}_k^{(\mathbf{Z})})^T + \\
&\quad \frac{NL}{(N-L)^2}\mathbf{C}_k^{(\mathbf{X})}(\mathbf{C}_k^{(\mathbf{Z})})^T + \frac{NL}{(N-L)^2}\mathbf{C}_k^{(\mathbf{Z})}(\mathbf{C}_k^{(\mathbf{X})})^T \\
&= \frac{N}{N-L}\mathbf{S}_k^{(\mathbf{X})} - \frac{L}{N-L}\mathbf{S}_k^{(\mathbf{Z})} - \frac{NL}{(N-L)^2}(\mathbf{C}_k^{(\mathbf{X})} - \mathbf{C}_k^{(\mathbf{Z})})(\mathbf{C}_k^{(\mathbf{X})} - \mathbf{C}_k^{(\mathbf{Z})})^T
\end{aligned} \tag{37}$$

Proposition 1. For Ω_k, the following attributes $\{\mathbf{C}_k, \mathbf{S}_k, N_k, A_k, l_k\}$ of any non-expiration class, let $\mathbf{X}$ in R^d be a set of data chunks of any class k and $\mathbf{C}$ is the all-instance center vector. There are m classes of the data set. Assume that a chunk of data $\mathbf{Z} \in \mathbf{X}$ whose $|\mathbf{Z}| = L$ is expired and moved to class 0. The between-class and within-class scatter matrices can be updated by analogy to LDA merging, as follows:

$$\mathbf{B}^{(new)} = \sum_{k=1}^{m}(N_k - L)(\mathbf{C}_k^{(new)} - \mathbf{C}^{(new)})(\mathbf{C}_k^{(new)} - \mathbf{C}^{(new)})^T\ , \tag{38}$$

$$\mathbf{W}^{(new)} = \sum_{k=1}^{m}\mathbf{S}_k^{(new)} \tag{39}$$

where $\mathbf{C}^{(new)}$ is new all-instance center vector, and $\mathbf{C}_k^{(new)}$ and $\mathbf{S}_k^{(new)}$ are the new center vector and covariance matrix in class k computed by using Theorems 1 and 2.

3.3 Special structure of class 0

The network must be able to identify a testing datum whether it is expired or it belongs to any class $i > 0$, regardless of its temporal aspect. This implies that there must be a special structure to keep track of these expired data throughout the operational period of the network. However it is obvious that the process of keeping track of these data may increase the space complexity of the structure because all expired data must be captured. In this study, I limited the space complexity of this special structure to $\theta(d \times L)$, where d is the number of dimensions and L is the amount of new incoming data. The problem is how to limit this space complexity so that it has a minimum effect on the classification accuracy of the network. To resolve this problem, two relevant issues were considered.

The first issue concerns the representation of all expired data in terms of a column vector $\mathbf{u}$ in R^d such that for any expired datum $\mathbf{x}_i$, $\mathbf{u}^T\mathbf{x}_i = 0$. The value 0 is the class number. At the beginning, the value of $\mathbf{u}$ is computed from the first incoming chunk of data $\mathbf{Z}$ represented in forms of a matrix $\mathbf{V}_{[1,L]} = [\mathbf{x}_1, \dots, \mathbf{x}_L]$. The size of $\mathbf{V}_{[1,L]}$ is equal to $d \times L$. Without loss of generality of class number, the value of $\mathbf{u}$ is computed by this simple equations

$$\mathbf{u}^T\mathbf{V}_{[1,L]} = \mathbf{t}^T \qquad (40)$$

$$\mathbf{u}^T = \mathbf{t}^T\mathbf{V}_{[1,L]}^{+}. \qquad (41)$$

Vector $\mathbf{t}$ is the column vector representing class numbers or targets of all incoming data and $\mathbf{V}_{[1,L]}^{+}$ is the pseudoinverse matrix of $\mathbf{V}_{[1,L]}$. The process of pseudoinverse matrix is employed in this computation because it is possible that the amount of incoming data may not be equal to the number of dimensions, i.e., $L \neq d$. Since $\mathbf{u}$ is computed from $\mathbf{V}_{[1,L]}^{+}$, therefore $\mathbf{u}^T\mathbf{V}_{[1,L]} \approx \mathbf{t}^T$. To make $\mathbf{u}^T\mathbf{V}_{[1,L]}$ equal to $\mathbf{t}^T$, a vector Δ is computed by the following equation and added to $\mathbf{u}$.

$$\Delta^T = \mathbf{t}^T - \mathbf{u}^T\mathbf{V}_{[1,L]} \qquad (42)$$

The second issue focuses on how to recursively compute the values of $\mathbf{u}, \mathbf{V}_{[1,L]}^{+}$, and Δ with respect to only those new incoming data. In my approach, all learned data are thrown away. This makes the typical pseudoinverse process of $\mathbf{V}_L$ impossible. The concept proposed by Greville [33], Tapson and Van Schaik [34], Udwadia and Kalaba [35], and Predrag, Marko, Igor, Sladjana [36] were applied to formulate the recursive computations for $\mathbf{u}$, $\mathbf{V}_{[1,L]}^{+}$, and Δ. The following notations are defined in order to formulate the recursive computations:

$\mathbf{u}(i)$	: the values of $\mathbf{u}$ at time i.
$\Delta(i)$	: the values of Δ at time i.
$\mathbf{t}_{[i,j]}$	: column vector $\mathbf{t}$ containing the i^{th} to the j^{th} elements.
$\mathbf{V}_{[i,j]}$	: sub-matrix of $\mathbf{V}_{[1,L]}$ consisting of a set of column vectors from columns i to j.

Matrix $\mathbf{V}_{[1,L]}$ is decomposed into two sub-matrices, i.e., $\mathbf{V}_{[1,L]} = [\mathbf{V}_{[1,p]}\ \mathbf{V}_{[p+1,L]}]$ and $\mathbf{t}_{[1,L]}$ is also separated into two sub-column vectors denoted as $\mathbf{t}_{[1,L]}^{T} = [\mathbf{t}_{[1,p]}^{T}\ \mathbf{t}_{[p+1,L]}^{T}]$. We assume that data from $\mathbf{x}_1$ to $\mathbf{x}_p$ are the first chunk of incoming data and the rest $\mathbf{x}_{p+1}$ to $\mathbf{x}_L$ are new currently incoming data. From the results in [35, 36], I have

$$\mathbf{V}_{[1,L]}^{+} = \begin{bmatrix} \mathbf{V}_{[1,p]}^{+}(I - \mathbf{V}_{[p+1,L]}\mathbf{R}^{T}) \\ \mathbf{R}^{T} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{V}_{[1,p]}^{+} - \mathbf{Q}\mathbf{R}^{T} \\ \mathbf{R}^{T} \end{bmatrix} \tag{43}$$

Matrices $\mathbf{Q}$ and $\mathbf{R}$ are computed as follows.

$$\mathbf{Q} = \mathbf{V}_{[1,p]}^{+}\mathbf{V}_{[p+1,L]} \tag{44}$$

$$\mathbf{R} = ((\mathbf{V}_{[1,p]}^{+})^{T}\mathbf{Q})\left(I + \mathbf{Q}^{T}\mathbf{Q}\right)^{-1} \tag{45}$$

The value of $\mathbf{u}(i)$ and $\Delta(i)$ can be recursively formulated as stated in the following Theorems.

Theorem 3. Given $\mathbf{t}^T_{[p+1,L]}$, $\mathbf{R}^T = \left(((\mathbf{V}^+_{[1,p]})^T \mathbf{Q}) \left(I + \mathbf{Q}^T \mathbf{Q} \right)^{-1} \right)^T$, $\mathbf{Q} = \mathbf{V}^+_{[1,p]} \mathbf{V}_{[p+1,L]}$, $\mathbf{u}^T(i-1)$, and $\mathbf{V}_{[p+1,L]}$, then vector $\mathbf{u}^T(i)$ is equal to $\mathbf{u}^T(i-1) - \mathbf{u}^T(i-1)\mathbf{V}_{[p+1,L]}\mathbf{R}^T + \mathbf{t}^T_{[p+1,L]}\mathbf{R}^T$.

Proof Let

$$\begin{aligned}
\mathbf{u}^T(i) &= \mathbf{t}^T_{[1,L]} \mathbf{V}^+_{[1,L]} \\
&= [\mathbf{t}^T_{[1,p]} \;\; \mathbf{t}^T_{[p+1,L]}] \begin{bmatrix} \mathbf{V}^+_{[1,p]}(I - \mathbf{V}_{[p+1,L]}\mathbf{R}^T) \\ \mathbf{R}^T \end{bmatrix} \\
&= \mathbf{t}^T_{[1,p]}\mathbf{V}^+_{[1,p]} - \mathbf{t}^T_{[1,p]}\mathbf{V}^+_{[1,p]}\mathbf{V}_{[p+1,L]}\mathbf{R}^T + \mathbf{t}^T_{[p+1,L]}\mathbf{R}^T \\
&= \mathbf{u}^T(i-1) - \mathbf{u}^T(i-1)\mathbf{V}_{[p+1,L]}\mathbf{R}^T + \mathbf{t}^T_{[p+1,L]}\mathbf{R}^T \qquad (46)
\end{aligned}$$

Theorem 4. Given $\mathbf{t}^T_{[p+1,L]}$, $\mathbf{R}^T = \left(((\mathbf{V}^+_{[1,p]})^T \mathbf{Q}) \left(I + \mathbf{Q}^T \mathbf{Q} \right)^{-1} \right)^T$, $\mathbf{Q} = \mathbf{V}^+_{[1,p]} \mathbf{V}_{[p+1,L]}$, $\mathbf{u}^T(i-1)$, $\mathbf{V}^+_{[1,p]}$, and $\mathbf{V}_{[p+1,L]}$, then vector $\Delta^T(i)$ is equal to $[\Delta^T(i-1) + (\mathbf{u}^T(i-1)\mathbf{V}_{[p+1,L]} - \mathbf{t}^T_{[p+1,L]})\mathbf{R}^T(\mathbf{V}^+_{[1,p]})^+ \quad \mathbf{t}^T_{[p+1,L]} - \mathbf{u}^T(i)\mathbf{V}_{[p+1,L]}]$.

Proof Let

$$\begin{aligned}
\Delta^T(i) &= \mathbf{t}^T_{[1,L]} - \mathbf{u}^T(i)\mathbf{V}_{[1,L]} \\
&= [\mathbf{t}^T_{[1,p]} \quad \mathbf{t}^T_{[p+1,L]}] - [\mathbf{u}^T(i)\mathbf{V}_{[1,p]} \quad \mathbf{u}^T(i)\mathbf{V}_{[p+1,L]}] \\
&= [\mathbf{t}^T_{[1,p]} - \mathbf{u}^T(i)\mathbf{V}_{[1,p]} \quad \mathbf{t}^T_{[p+1,L]} - \mathbf{u}^T(i)\mathbf{V}_{[p+1,L]}] \\
&= [\mathbf{t}^T_{[1,p]} - (\mathbf{u}^T(i-1) - \mathbf{u}^T(i-1)\mathbf{V}_{[p+1,L]}\mathbf{R}^T + \mathbf{t}^T_{[p+1,L]}\mathbf{R}^T)\mathbf{V}_{[1,p]} \\
&\qquad \mathbf{t}^T_{[p+1,L]} - \mathbf{u}^T(i)\mathbf{V}_{[p+1,L]}] \\
&= [\Delta^T(i-1) + (\mathbf{u}^T(i-1)\mathbf{V}_{[p+1,L]} - \mathbf{t}^T_{[p+1,L]})\mathbf{R}^T(\mathbf{V}^+_{[1,p]})^+ \\
&\qquad \mathbf{t}^T_{[p+1,L]} - \mathbf{u}^T(i)\mathbf{V}_{[p+1,L]}] \qquad (47)
\end{aligned}$$

3.4 Hybrid LDA-PCA algorithm

To enhance the incremental learning algorithm, the new one-pass-throw-away learning algorithm based on LDA and PCA hybridization [5] can be used to compute the directions of the hyperellipsoid in the network by combining the orthonormal basis of the PCA discussed in [4] and the scatter matrices of the LDA. A detail of the algorithm is described in Hybrid LDA-PCA learning algorithm.

Algorithm 3: Hybrid LDA-PCA Algorithm

Inputs: Chunk of non-expired data $\mathbf{Z}$.

Output: Set of Ω_h.

```
1.  Initialize the width vector $\mathbf{A}_0$ based on Algorithm 2.
2.  For each $\mathbf{x}_i \in \mathbf{Z}$ do
3.      If there exists the hidden neurons then
4.          Find a closest hidden neuron labeling the same class of datum $\mathbf{x}_i$
            measured by:
                $t = \arg\min_h (\|\mathbf{x}_i - \mathbf{C}_h\|)$
            where $\mathbf{C}_h$ is the center vector of the $h^{th}$ neuron.
5.          Compute the new center vector $\mathbf{C}_t$ and the new covariance matrix
            $\mathbf{S}_t$ based on Equations (5) and (6), respectively.
6.          If $\exists \Psi_t(\mathbf{x}_i) \le 0$ then
7.              Compute the within-class scatter matrix $\mathbf{W}$.
8.              If $\mathbf{W}$ is a non-singular matrix then
9.                  Compute the eigenvectors and the eigenvalues by using
                    Equation (13) based on the LDA concept.
10.             Else
11.                 Compute the eigenvectors and the eigenvalues by
                    using the orthonormal basis of VEBF algorithm.
12.             EndIf
13.         EndIf
14.     Else
```

```
15.         Create a new neuron.
16.     EndIf
17. EndFor
```

3.5 Dynamical structure of proposed network for learning the expiration and live classes

Since obtaining the correct class of any datum concerns the most recent class change of datum, a new structure was proposed to handle this complexity in this research. The structure consists of two strata as shown in Figure 3.3. Assume that there are four classes which are 0, 1, 2, and 3. Without loss of generality, I assume that the network of each class i has only one output denoted by y_i. The upper stratum is for class 0. A 3-layer feed-forward network is employed to learn all data in this class. The lower stratum is for all *live* classes. All classes in the lower layer is learned by a 3-layer network. The hidden layer of this network consists of several groups of neurons, i.e. one group for one class.

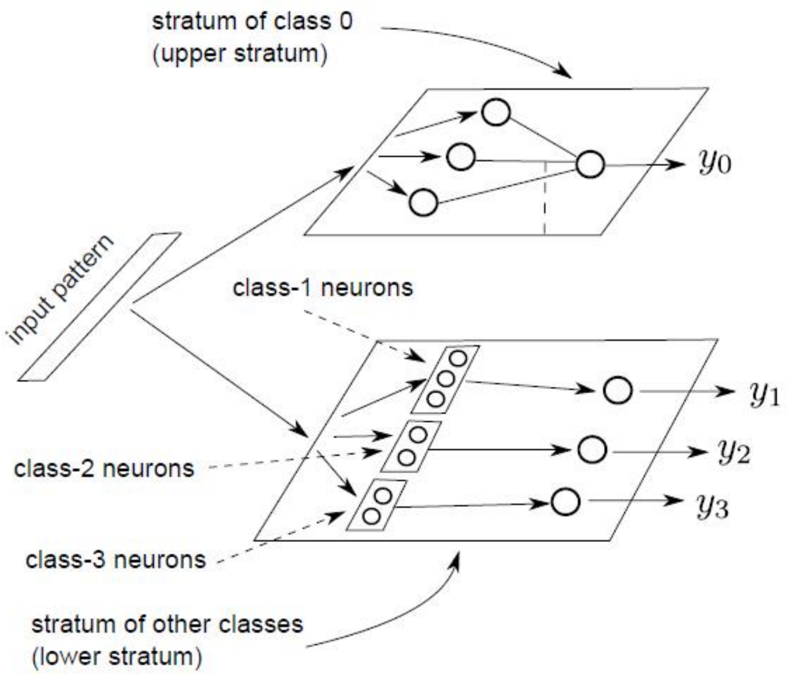
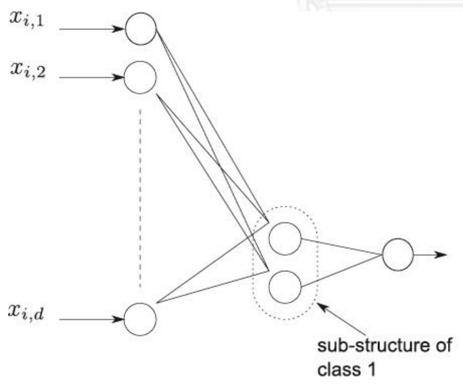
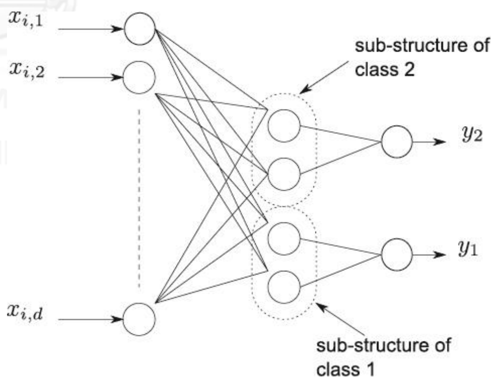


Figure 3.3 The proposed stratous structure for handing live and expired class changes. The upper stratum is for expired class and the lower stratum is for all live classes.

Figure 3.4 shows the concept of how the proposed structure is evolved during the learning process of streaming data. The output neuron of class i is denoted by y_i. Suppose during the first period all incoming data are in class 1 and they are learned by hidden neurons in class-1 group as shown in Figure 3.4(a). Then the data in class 2 flow in during the next period. All class-2 data are learned by the hidden neurons in class-2 group as shown in Figure 3.4(b). If some already learned data in class 2 are expired, these expired data will be assigned to class 0 in upper stratum as shown in Figure 3.4(c). Next all data in class 2 are expired and there are some new incoming class-3 data. This causes the entire structure of class 2 to be removed and all expired data are moved to class 0 in the upper stratum whilst the hidden neurons in class-3 group are created as shown in Figure 3.4(d). After that if some expired data are revived to class 2, then the sub-structure of class 2 will be recreated in lower stratum again as shown in Figure 3.4(e). Subsequently, other some expired data are changed to new class 4, the sub-structure of class 4 will be created as shown in Figure 3.4(f).



(a)



(b)

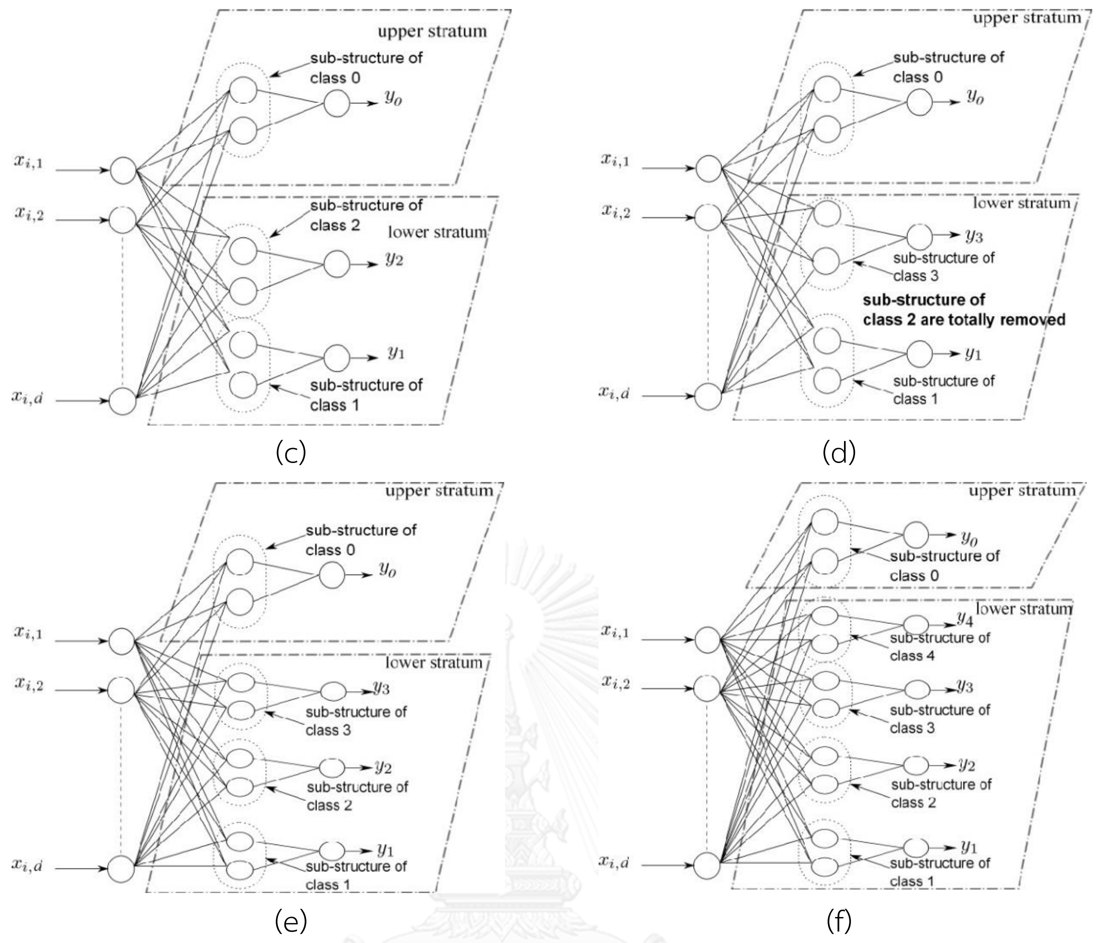
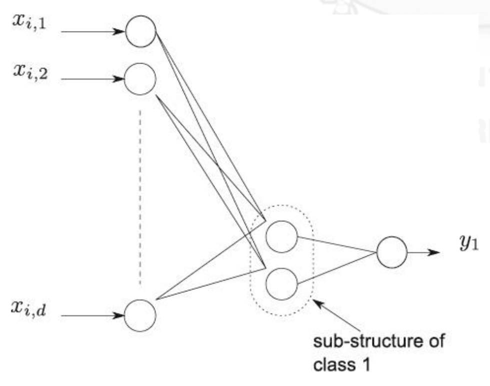


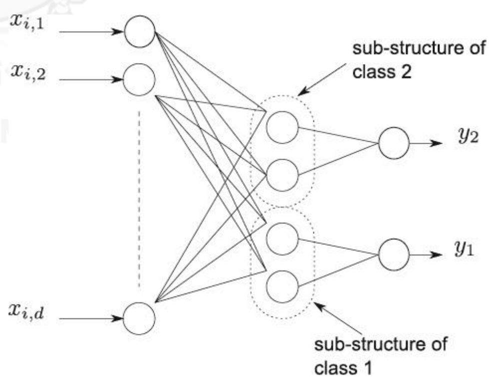
Figure 3.4 The dynamical structure of proposed network. (a) Suppose at the beginning all incoming data are in class 1 and learned by the sub-structure of class 1. (b) Some additional new incoming data in class 2 are learned by the sub-structure of class 2. (c) Some data in class 2 are expired and learned by sub-structure of class 0 in upper stratum. (d) All data in class 2 are expired and there are some new incoming data of class 3 learned by the sub-structure of class 3. The sub-structure of class 2 is entirely removed and the expired data of class 2 are moved to class 0 in the upper stratum. (e) Some expired data in class 0 are revived and recreated the sub-structure of class 2 in lower stratum. (f) Other expired data in class 0 are changed to new class 4 learned by the sub-structure of class 4 in the lower stratum.

3.6 Dynamical structure of proposed network for learning with special structure of class 0

The structure for class 0 is different from the other classes. The difference is the sub-structure for handling class 0. Figure 3.5 shows the concept of proposed structure. The output neuron of class i is denoted by y_i. Suppose during the first period all incoming data are in class 1 and they are learned by hidden neurons in class-1 group as shown in Figure 3.5(a). Then the data in class 2 flow in during the next period. All class-2 data are learned by the hidden neurons in class-2 group as shown in Figure 3.5(b). If some already learned data in class 2 are expired, then these expired data are assigned to class 0 as shown in Figure 3.5(c). To distinguish class-0 structure from the other class structures, a square is used to denote the special structure of class 0. Next we assume that all data in class 2 are expired and there are some new incoming data of class 3. This causes the entire structure of class 2 to be removed and all expired data are moved to class 0 as shown in Figure 3.5(d). Note that the size of class 0 keeps increasing. Furthermore, the sub-structure of class 3 is created to learn the new incoming data.



(a)



(b)

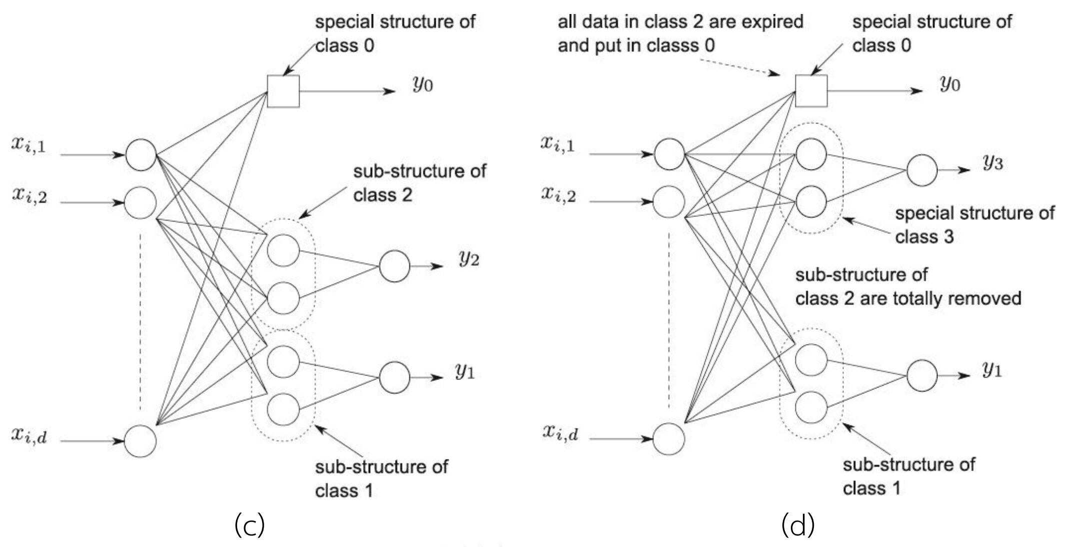

special structure of class 0
y_0
$x_{i,1}$
$x_{i,2}$
$x_{i,d}$
sub-structure of class 2
y_2
y_1
sub-structure of class 1
(c)
all data in class 2 are expired and put in classs 0
special structure of class 0
y_0
y_3
special structure of class 3
sub-structure of class 2 are totally removed
y_1
sub-structure of class 1
(d)

Figure 3.5 The dynamical structure of proposed network with special structure of class 0

3.7 Proposed learning and testing algorithms with special structure of class 0

To keep track of increasing expired data, the special structure of class 0 was proposed for storing all expired data. From the key idea of the proposed network mentioned, my proposed algorithm named ***Hybridization Multi-Stratum Network Learning (Hybrid-MSNL)*** is composed of the following three main steps.

The first step is to learn all data of different classes in the first incoming chunk by using the concept of Hybrid LDA-PCA algorithm. After learning, there exists a set of neurons with relevant parameters as discussed in this Section. We assume that there is no expired data in the first chunk of incoming data. Each incoming datum $\mathbf{x}_i$ is tagged with its class. Those already learned data are discarded from the learning process forever.

The second step is to learn other temporally incoming data chunks. Some data may be in class 0 or in any class $i \neq 0$. Those data whose class $i \neq 0$ are learned by the same algorithm as those data in the first data chunk. Some new neurons in different classes may be introduced to the network to learn these non-expired data. But for those expired data, all previously learned neurons from these data which are not formerly expired are considered. The number of data in these considered neurons is decreased. After decreasing, if the number of data in any one of these neurons is equal to zero, then the neuron and its links are entirely removed from the network. For any neuron Ω_a, its parameters $(\mathbf{C}_a^{(old)}, \mathbf{S}_a^{(old)}, N_a, A_a, l_a)$ are updated afterwards. The parameters $\mathbf{t}_{[p+1,L]}^T, \mathbf{R}^T, \mathbf{u}^T(i-1)$, $\Delta^T(i-1)$, and $\mathbf{V}_{[p+1,L]}$ of special structure of class 0 are updated as well. This step is continued until there is no more incoming data. The details of steps 1 and 2 are described in Algorithm 4.

The third step is to predict the class of any testing datum $\mathbf{x}_i$ denoted as $class(\mathbf{x}_i)$ by using the facts in Theorems 3 and 4 for those expired data and the following condition for non-expired data. Let $\mathbf{s}_h$ be a set of neurons with class label l_h.

$$class(\mathbf{x}_i) = \arg\min_h \left(\Psi_k(\mathbf{x}_i) \mid \Omega_k \in \mathbf{s}_h \right) \tag{48}$$

where $\Psi_k(\mathbf{x}_i)$ according to Equation (1) is the output of hidden neuron Ω_k and l_h is the class label of $\Psi_k(.)$. Algorithm 5 shows the detail of this predicting process.

Algorithm 4: Hybridization Multi-Stratum Network Learning (Hybrid-MSNL)

Input: Chunks of data.

Output: Set of Ω_h, $\mathbf{u}^T$, and Δ^T.

```
1.  Get the first data chunk and learn these data by using Hybrid LDA-PCA algorithm.
    Discard all data after learning.
2.  Get the second data chunk and separate the data into expired data set $\mathbf{E}$ and
    non-expired data set $\mathbf{Z}$.
3.  Learn data set $\mathbf{Z}$ by using Hybrid LDA-PCA algorithm and applying Theorems 1 and
    2 to adjust all parameters of each $\Omega_h$.
4.  Let $count_h$ be the number of data learned by $\Omega_h$ obtained from steps 1 and 3.
5.  Let $p$ be the amount of expired data in $\mathbf{E}$.
6.  Form $\mathbf{V}_{[1,p]}$ and compute $\mathbf{V}_{[1,p]}^+$, $\mathbf{u}^T(1)$, and $\Delta^T(1)$.
7.  For each $\mathbf{x}_j \in \mathbf{E}$ do
8.      If $\exists \Psi_h(\mathbf{x}_j) \le 0$ then
9.          Set $count_h = count_h - 1$.
10.         If $count_h$ is equal to 0 then
11.             Remove $\Omega_h$ and its links from the network.
```

```
12.         EndIf
13.     EndIf
14. EndFor
15. Discard sets $\mathbf{E}$, $\mathbf{Z}$, and all learned data.
16. Let $i=1$ denote the time step.
17. While there exists a new incoming data chunk do
18.     Set $i=i+1$.
19.     Separate the data into expired data set $\mathbf{E}$ and non-expired data set $\mathbf{Z}$.
20.     Learn data set $\mathbf{Z}$ by using Hybrid LDA-PCA algorithm and applying Theorems
    1 and 2 to recursively adjust all parameters of each $\Omega_h$.
21.     For each newly created $\Omega_h$ do
22.             Let $count_h$ be the number of data learned by each $\Omega_h$ obtained
                from step 20.
23.     EndFor
24.     Let $L$ be the amount of expired data in $\mathbf{E}$.
25.     Form $\mathbf{V}_{[1,L]}$ and recursively compute $\mathbf{V}^{+}_{[1,L]}$, $\mathbf{u}^T(i)$, and $\Delta^T(i)$ by using Theorems
    3 and 4 with $\mathbf{u}^T(i-1)$, and $\Delta^T(i-1)$.
26.     For each $\mathbf{x}_j \in \mathbf{E}$ do
27.         If $\exists \Psi_h(\mathbf{x}_j) \le 0$ then
28.             $count_h = count_h - 1$.
29.             If $count_h$ is equal to 0 then
30.                 Remove $\Omega_h$ and its links from the network.
31.             EndIf
32.     EndIf
33.  EndFor
34. Discard sets $\mathbf{E}$, $\mathbf{Z}$, and learned data.
35. EndWhile
```

Algorithm 5: Predicting Class of Testing Datum

Input: A testing datum $\mathbf{x}_j$, $\mathbf{u}^T$, Δ^T, and all Ω_k.

Output: Predicted $class(\mathbf{x}_j)$.

```
1.  If $\mathbf{u}^T\mathbf{x}_j + \Delta^T$ is equal to $0$ then
2.      Set $class(\mathbf{x}_j) = 0$.
3.  Else
4.      Let $\mathbf{s}_h$ be a set of neurons.
5.      Set $class(\mathbf{x}_j) = \arg\min_h \left(\Psi_k(\mathbf{x}_j) \mid \Omega_k \in \mathbf{s}_h\right)$.
6.  EndIf
```

3.8 Example of hybridization multi-stratum learning process

To illustrate how Hybrid-MSNL works, Figure 3.6 shows an example of the snapshot of events when some data are expired and some are new incoming. Suppose there are two neurons of classes $C1$ and $C2$ in the forms of two versatile elliptic functions. At the beginning shown in Figure 3.6(a), there are ten data denoted by ten "+" symbols in class $C1$ and four data denoted by four "✶" symbols in class $C2$. After that, two data in class $C1$ are expired. This causes the neuron in class $C1$ to shrink its size and the expired data are removed to class 0 as shown in Figure 3.6(b) - (c). In Figure 3.6(d), there are more new incoming data, i.e., three of class $C1$ and six of class $C2$. Those data of class $C2$ lie outside the versatile elliptic function but those of $C1$ lie inside the versatile elliptic function. Therefore only the versatile elliptic function of $C1$ must be expanded to cover the new incoming data. For the last event shown in Figure 3.6(e), three data in $C1$ are expired but none is expired in $C2$. To reduce the possible prediction error, the size of versatile elliptic function of $C1$, the expired data are moved to class 0 as shown in Figure 3.6(f).

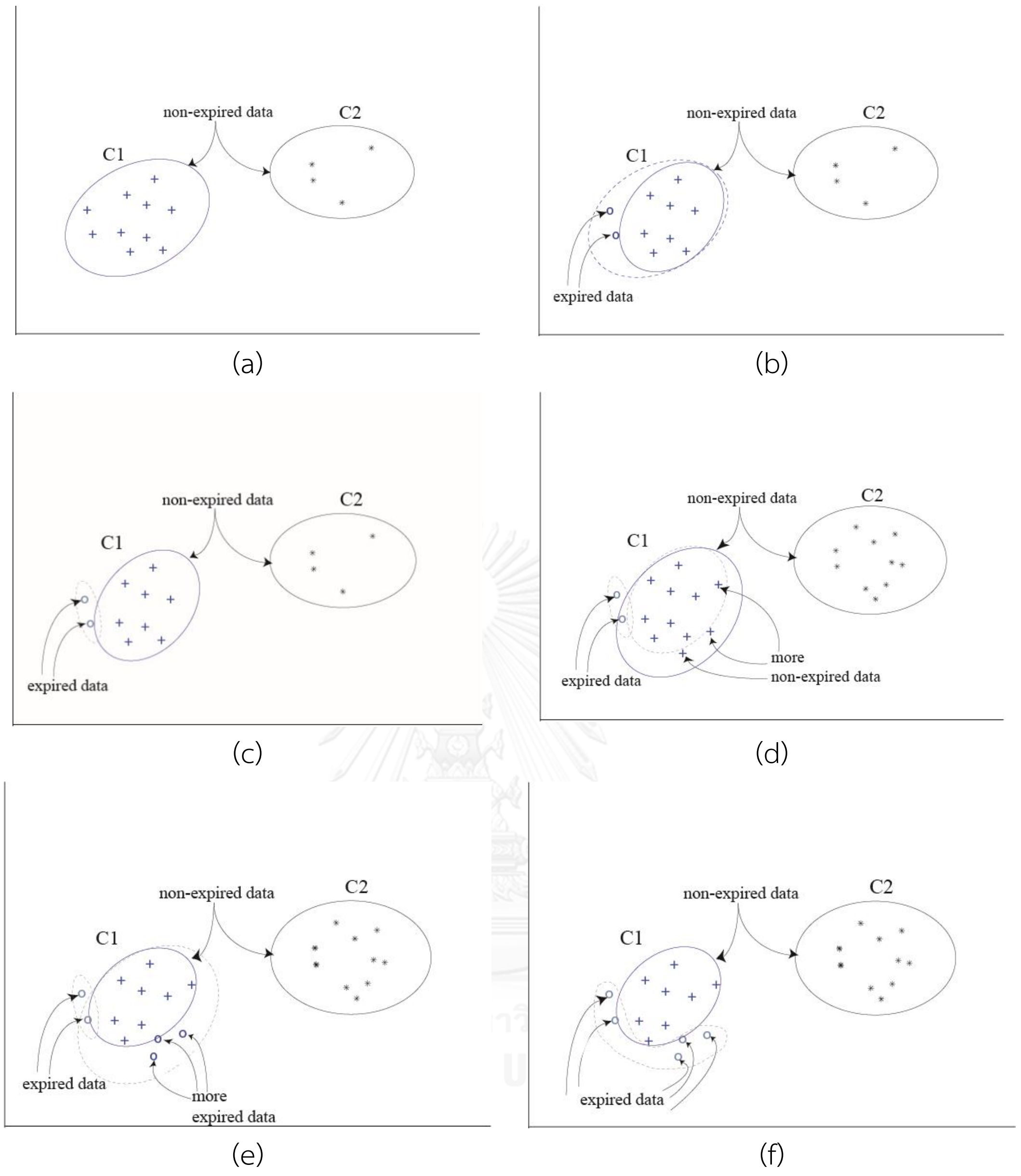


Figure 3.6 An example of how Hybrid-MSNL works. Assume that there two neurons of classes C1 and C2 in forms of versatile elliptic functions. (a) The beginning situation. There are ten data in C1 and four data in C2. (b) Two expired data in C1 but none in C2. The versatile elliptic function of C1 is shrunk. (c) The versatile elliptic function of C1 after being shrunk. (d) Expanding the versatile elliptic function of C1 to cover three new incoming data. (e) Three more new expired data of C1. The versatile elliptic function of C1 is shrunk. (f) The final situation.

3.9 Proposed learning and testing algorithms in case of expiration and live classes

Since any data can be temporally expired, therefore it is possible that any neuron and its links can be entirely removed from the network when all data learned by the neuron are expired. From this observation, my proposed algorithm named ***Multi-Stratum Network Learning (MSNL)*** is composed of the following three main steps.

The first step is to learn all data of different classes in the first incoming chunk by using the concept of VEBF algorithm. After learning, there exists a set of neurons with relevant parameters as discussed in previous Section. We assume that there is no expired data in the first chunk of incoming data. Each incoming datum $\mathbf{x}_i$ is tagged with its class. Those already learned data are discarded from the learning process forever.

The second step is to learn other temporally incoming data chunks. Some data may be in class 0 or in any class $i \neq 0$. Those data whose class $i \neq 0$ are learned by the same algorithm as those data in the first data chunk. Some new neurons in different classes may be introduced to the network defined in its lower stratum to learn these non-expired data. But for those expired data, all previously learned neurons from these data which are not formerly expired are considered. The number of data in these considered neurons is decreased. After decreasing, if the number of data in any one of these neurons is equal to zero, then the neuron and its links are entirely removed from the network. For any neuron Ω_a, its parameters $(\mathbf{C}_a^{(old)}, \mathbf{S}_a^{(old)}, N_a, A_a, l_a)$ are updated afterwards. Since each datum $\mathbf{x}_i$ can have different class changes during its lifetime, thus achieving the correct response of class query for $\mathbf{x}_i$ requires a time tracking process of the neuron used to learn $\mathbf{x}_i$. To distinguish the neurons in the upper and lower strata, notations $\Omega_a^{(lo)}$ and $\Omega_b^{(up)}$ denote neuron a in the lower stratum and b in the upper stratum, respectively. Let ω_a be the time stamp of neuron $\Omega_a^{(lo)}$ when it learns $\mathbf{x}_i$.

The value of learning time ω_a of the neuron $\Omega_a^{(lo)}$ in lower stratum can be computed as the following equation.

$$\omega_a^{(new)} = \omega_a^{(old)} + \left(\frac{\varepsilon N_a^{(old)} + (1-\varepsilon) N_a^{(new)}}{N_a^{(old)} + N_a^{(new)}} \right) \tag{49}$$

where $0 < \varepsilon < 0.5$. The structure of class 0 are updated in upper level as well. This step is continued until there is no more incoming data. The details of steps 1 and 2 are described in Algorithm 6.

The third step is to predict the class of any testing datum $\mathbf{x}_i$ denoted as $class(\mathbf{x}_i)$ by using the following condition. Let $\mathbf{s}_h$ be a set of neurons in lower level with class label h.

$$class(\mathbf{x}_i) = \arg\max_h (\omega_k \mid \Omega_k^{(lo)} \in \mathbf{s}_h \text{ and } \Psi_k^{(lo)}(\mathbf{x}_i) \le 0) \tag{50}$$

or

$$class(\mathbf{x}_i) = \arg\min_h \left(\Psi_k^{(lo)}(\mathbf{x}_i) \mid \Omega_k^{(lo)} \in \mathbf{s}_h \right) \tag{51}$$

where $\Psi_k^{(lo)}(\mathbf{x}_i)$ according to Equation (1) is the output of hidden neuron $\Omega_k^{(lo)}$ in lower stratum. Algorithm 7 shows the detail of this predicting process.

Algorithm 6: Multi-Stratum Network Learning (MSNL)

Input: Chunks of data.

Output: Sets of $\Omega_k^{(lo)}$, $\Omega_k^{(up)}$, and ω_k.

1. Get the first $N^{(old)}$ data chunk and learn these data by using VEBF algorithm. Discard all data after learning.
2. Let $N_k^{(old)}$ be the number of data learned by each neuron in any stratum.
3. Set initial time stamp $\omega_k = 0$ for each neuron k.
4. Get the second data chunk and separate the data into expired data set $\mathbf{E}$ and non-expired data set $\mathbf{Z}$.
5. Learn data set $\mathbf{Z}$ by using VEBF algorithm and adjust all parameters of each $\Omega_k^{(lo)}$ by applying Theorems 1 and 2.
6. Learn data set $\mathbf{E}$ by using VEBF algorithm and adjust all parameters of each $\Omega_k^{(up)}$ used by applying Theorems 1 and 2.
7. Let $N_k^{(new)}$ be the number of data learned by each neuron k in any stratum.

```
8.  Update each $\omega_k$ of each neuron $k$ in lower stratum by using $N_k^{(old)}$ and $N_k^{(new)}$
    with Equation (49).
9.  Reset $N_k^{(old)} = N_k^{(new)}$ for each neuron $k$.
10. For each neuron $\Omega_k^{(lo)}$ in the lower stratum do
11.     If $N_k^{(old)}$ of neuron $\Omega_k^{(lo)}$ is equal to 0 then
12.         Remove $\Omega_k^{(lo)}$ and its links from the network in the lower stratum.
13.     EndIf
14. EndFor
15. Discard sets $\mathbf{E}$, $\mathbf{Z}$, and all learned data.
16. While there exists a new incoming data chunk do
17.     Separate the data into expired data set $\mathbf{E}$ and non-expired data set $\mathbf{Z}$.
18.     Learn data set $\mathbf{Z}$ by using VEBF algorithm.
19.     For any datum $\mathrm{x}_i \in \mathbf{Z}$ and $\Psi_k^{(up)}(\mathrm{x}_i) \le 0$ do
20.         If $\mathrm{x}_i$ is learned by neuron $\Omega_k^{(up)}$ then
21.             Remove $\mathrm{x}_i$ from $\Omega_k^{(up)}$ and adjust all parameters of $\Omega_k^{(up)}$
                by using Theorems 1 and 2.
22.         EndIf
23.     EndFor
24.     Let $N_k^{(new)}$ be the number of data learned by each neuron $k$ in any stratum.
25.     Update each $\omega_k$ of each neuron $k$ in lower stratum by using $N_k^{(old)}$ and $N_k^{(new)}$
        with Equation (49).
26.     Reset $N_k^{(old)} = N_k^{(new)}$ for each neuron $k$.
27.     For each neuron $\Omega_k^{(up)}$ in the upper stratum do
28.         If $N_k^{(old)}$ of neuron $\Omega_k^{(up)}$ is equal to $0$ then
29.             Remove $\Omega_k^{(lo)}$ and its links from the network in
                the upper stratum.
30.         EndIf
31.     EndFor
32.     Learn data set $\mathbf{E}$ by using VEBF algorithm in upper stratum and
```

```
        adjust all parameters of $\Omega_k^{(up)}$ used by applying Theorems 1 and 2.
33.     For each neuron $\Omega_k^{(lo)}$ in the lower stratum do
34.         If $N_k^{(old)}$ of neuron $\Omega_k^{(lo)}$ is equal to $0$ then
35.             Remove $\Omega_k^{(lo)}$ and its links from the network in
                the lower stratum.
36.         EndIf
37.     EndFor
38.     Discard sets $\mathbf{E}$, $\mathbf{Z}$, and learned data.
39. EndWhile
```

Algorithm 7: Predicting Class of Testing Datum

Input: A testing datum $\mathbf{x}_j$, all of $\Omega_k^{(lo)}$ and $\Omega_k^{(up)}$, and ω_k.

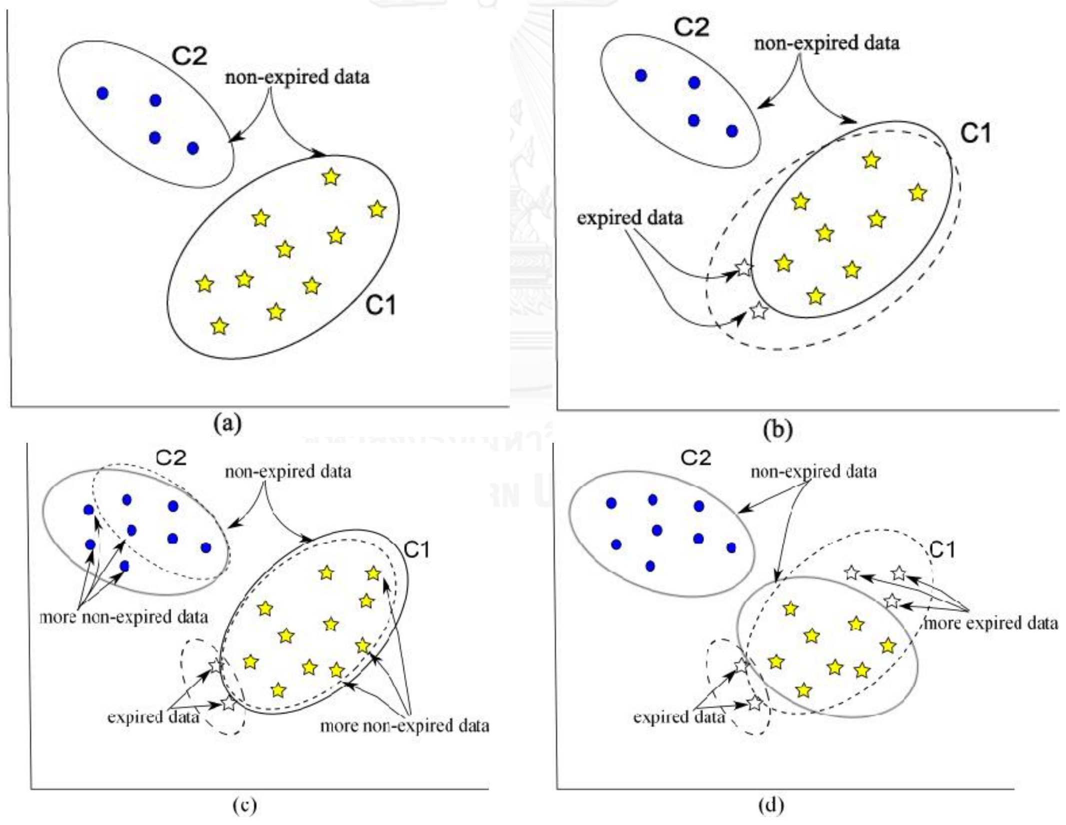
Output: Predicted $class(\mathbf{x}_j)$.

```
1. If $\Psi_k^{(up)}(\mathrm{x}_i) \le 0$ then
2.     Set $class(\mathbf{x}_j) = 0$.
3. ElseIf $\Psi_k^{(lo)} \le 0$ then
4.     Let $\mathbf{s}_h$ be a set of neurons in lower stratum such that $\Psi_k^{(lo)} \le 0$.
5.     Set $class(\mathbf{x}_i) = \arg\max_h(\omega_k \,|\, \Omega_k^{(lo)} \in \mathbf{s}_h \text{ and } \Psi_k^{(lo)}(\mathbf{x}_i) \le 0)$.
6. Else
7.     Let $\mathbf{s}_h$ be a set of neurons in lower stratum.
8.     Set $class(\mathbf{x}_i) = \arg\min_h\left(\Psi_k^{(lo)}(\mathbf{x}_i) \mid \Omega_k^{(lo)} \in \mathbf{s}_h\right)$.
9. EndIf
```

3.10 Example of multi-stratum learning process

To illustrate how MSNL works, Figure 3.7 shows an example of the snapshot of events when some data are expired and some are new incoming. Suppose there are two neurons of classes $C1$ and $C2$ in forms of two versatile elliptic functions. At the beginning shown in Figure 3.7(a), there are ten data denoted by ten stars in class $C1$ and four data denoted by four thick dots in class $C2$. After that, two data in class $C1$

are expired. This causes the neuron in class $C1$ to shrink its size and the expired data are removed to class 0 as shown in Figure 3.7(b). In Figure 3.7(c), there are more new incoming data, i.e., three of class $C1$ and four of class $C2$. Those data of class $C2$ lie outside the versatile elliptic function but those of $C1$ lie inside the versatile elliptic function. Therefore only the versatile elliptic function of $C2$ must be expanded to cover the new incoming data. In Figure 3.7(d), three data in $C1$ are expired but none is expired in $C2$. To reduce the possible prediction error, the size of versatile elliptic function of $C1$ is shrunk. After shrinking the versatile elliptic function of $C1$, the expired data are moved to class 0 as shown in Figure 3.7(e). For the last event shown in Figure 3.7(f), the class labels of two expired data are changed to new class $C3$.



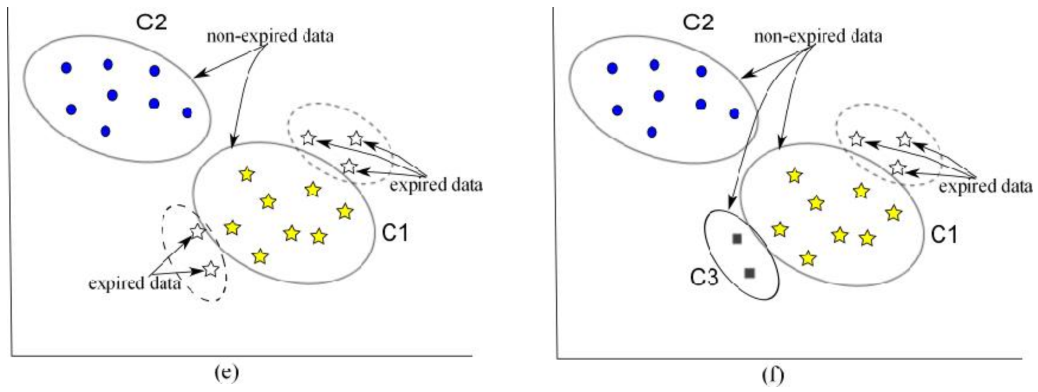


Figure 3.7 An example of how MSNL works. Assume that there two neurons of classes C1 and C2 in forms of versatile elliptic functions. (a) The beginning situation. There are 10 data in C1 and four data in C2. (b) Two expired data in C1 but none in C2. The versatile elliptic function of C1 is shrunk. (c) Expanding the versatile elliptic function of C1 and C2 to cover three new incoming data and four new incoming data, respectively. (d) Three more new expired data of C1. The versatile elliptic function of C1 is shrunk. (e) The versatile elliptic function of C1 after being shrunk. (f) There are two expired data changed to new class C3.

CHAPTER IV
EXPERIMENTAL RESULTS AND DISCUSSION

In case of non-expired and expired data, the performance of proposed algorithm was evaluated and compared with the existing methods. The percentage of the average accuracy and the computational time were concentrated as the evaluating measures. Total 10 benchmarked data sets (both 2-class and multi-class labels) shown in Table 4.1 were collected from the University of California at Irvine (UCI) repository of machine learning database [1] and popular data sets were taken from the concept drift problems. Those data from the concept drift problems were given by New South Wales Electricity Market collected from time and demand fluctuations in Australia as described in [2] and by U.S. National Oceanic and Atmospheric Administration taken as temperature, pressure, visibility, and other-related events of weather measurements as described in [3].

Table 4.1 The benchmark data sets in my experiments with special structure class 0

Data sets	Source	# Instances	# Attributes	# Classes
Derm	UCI	358	34	6
VehicleSilhouette	UCI	846	18	4
Leaf	UCI	340	15	36
EyeDetectioin	UCI	14980	14	2
Spambase	UCI	4601	57	2
PimaDiabetes	UCI	768	8	2
BanknoteAuthen	UCI	1372	4	2
Skin	UCI	245057	3	2
Weather	Concept drift	18159	8	2
Electric	Concept drift	45312	8	2

In case of expired and class-changed data, the performance of proposed MSNL algorithm was evaluated and compared with the existing methods. The percentage of the averaged accuracy and the computational time were concentrated as the evaluating measures. Total 12 benchmarked data sets (both 2-class and multi-class labels) as shown in Table 4.2 were collected from UCI database [1] given different from Table 4.1 and popular data sets for the concept drift problems the same as Table 4.1.

Table 4.2 The benchmark data sets in experiments with temporal class change

Data sets	Source	# Instances	# Attributes	# Classes
Balance	UCI	625	4	3
BreastCancer	UCI	683	9	2
Haberman	UCI	306	3	2
Sonar	UCI	208	60	2
Thyroid	UCI	215	5	3
Vertebral	UCI	310	6	3
Movement	UCI	360	90	15
Wine	UCI	178	13	3
Image	UCI	2310	19	7
Waveform	UCI	5000	21	3
Weather	Concept drift	18159	8	2
Electric	Concept drift	45312	8	2

Since the compared algorithms were not designed to handle the situation of data expiration, the following experimental set-up was conducted to fairly evaluate their performances. The data in both class 0 and the other classes were mixed into one training data chunk for VEBF algorithm, OI-SVM, ASC, learn++.NSE, and WMV methods. Algorithm 8 was used to generate the data in class 0 and changed class data. For my experimental set-up, the same set of data used by those compared algorithms was also used in my experiment. But the data were randomly partitioned into several chunks to test my concept of one-pass-throw-away learning approach.

4.1 Setting live and expired states to experimental data sets

Cross Validation (CV) is a commonly used technique for assessing the performance of a model. For state-of-the-art cross validation, ***distribution optimally balanced standard stratified cross validation*** **(DOB-SCV)** as presented in [37] is a technique to reduce the impact of partition-induced covariate shift with the reliability of classifier performance through cross validation. In processing steps of DOB-SCV, first of all, an unassigned example is randomly selected and then its $k-1$ nearest unassigned neighbors of the same class are considered. After that, it assigns each of those examples to a different fold. The process is repeated until all examples are assigned. The whole process is repeated for each class. From the concept of DOB-SCV, it can be applied for creating expired and class-changed data with optimal distribution in the entire data set. The detail of how to create expired and class-changed data based on DOB-SCV in each fold is summarized in Algorithm 8. Suppose that the training and testing sets are already formed and no expired data exist in these sets prior to the execution of Algorithm 8.

During the testing process, my proposed network must be able to determine the recent class of any datum $\mathbf{x}_a$ after its class has been previously changed several times. Therefore, some datum already learned are included in the testing set to evaluate this capability of the network. Let M be the original set of classes in a given data set before my class set-up process. In all experiments, without loss of generality, only 10 percent of training data were used for setting up live and expired states in training and testing processes. One of the following four events regarding the *live* and *expired* states is randomly set up for each training and testing datum $\mathbf{x}_a$.

Suppose datum $\mathbf{x}_a$ is originally in class $j \in M$.

1. Event 0: Set $\mathbf{x}_a$ to class 0.
2. Event 1: Set $\mathbf{x}_a$ to its original class j.
3. Event 2: Set $\mathbf{x}_a$ to any other class $k \in M$ and $k \neq j$.
4. Event 3: Set $\mathbf{x}_a$ to a new class $\gamma \notin M$.

Algorithm 8: Setting Expired and Class-Changed Data for Training and Testing

Input: A training set $T^{(Train)}$ and a testing set $T^{(test)}$ obtained by applying DOB-SCV concept.

Output: The training set $T^{(Train)}$ and testing set $T^{(test)}$ with some expired and class-changed data in both sets.

```
1.  Let $f$ be the number of folds and $e$ be the maximum number of expired data.
2.  **For** each class $j$ in $T^{(Train)}$ **do**
3.      Set $i=0$.
4.      **While** $i \le \frac{e}{m}$ **do**
5.          Randomly select a datum $\mathbf{x}_a$ and its closest neighbor $\mathbf{x}_b$ in
            the same class $j$.
6.          Let $\alpha$ be the duplicate of $\mathbf{x}_a$.
7.          Let $\beta$ be the duplicate of $\mathbf{x}_b$.
8.          Set target classes of $\alpha$ and $\beta$ to 0.
9.          Randomly select an event $\in \{0,1,2,3\}$.
10.         **Case** event **do**
11.             0: Set $T^{(train)} = T^{(train)} \cup \{\alpha,\beta\}$.
12.                Set $T^{(test)} = T^{(test)} \cup \{\alpha,\beta\}$.
13.             1: Set target classes of $\mathbf{x}_a$ and $\mathbf{x}_b$ to the same class $j$.
14.                Set $T^{(train)} = T^{(train)} \cup \{\alpha,\beta\} \cup \{\mathbf{x}_a,\mathbf{x}_b\}$.
15.                Set $T^{(test)} = T^{(test)} \cup \{\mathbf{x}_a,\mathbf{x}_b\}$.
16.             2: Set target classes of $\mathbf{x}_a$ and $\mathbf{x}_b$ to $k \in M$ and $k \ne j$.
17.                Set $T^{(train)} = T^{(train)} \cup \{\alpha,\beta\} \cup \{\mathbf{x}_a,\mathbf{x}_b\}$.
18.                Set $T^{(test)} = T^{(test)} \cup \{\mathbf{x}_a,\mathbf{x}_b\}$.
19.             3: Set target classes of $\mathbf{x}_a$ and $\mathbf{x}_b$ to a new class $\gamma \notin M$ .
20.                Set $T^{(train)} = T^{(train)} \cup \{\alpha,\beta\} \cup \{\mathbf{x}_a,\mathbf{x}_b\}$.
21.                Set $T^{(test)} = T^{(test)} \cup \{\mathbf{x}_a,\mathbf{x}_b\}$.
22.         **EndCase**
23.         Update $i = i + f$ .
24.     **EndWhile**
```

25. **EndFor**

4.2 Performance evaluation and comparison

The performance of the proposed algorithm was compared with families of incremental learning of VEBF neural network [4], OI-SVM neural network [6], ASC neural network [8], concept drift incremental learning of learn++.NSE [3], and WMV [27]. The same initial width vector set-up was adopted from VEBF algorithm given by Algorithm 2. Moreover, the value of this initial width was used in VEBF algorithm and also defined for tuning the optimal expansion of Gaussian kernel in the experiments of OI-SVM and ASC methods. Other user-defined parameters were optimized by using the technique of grid search reported in [6, 8] to obtain the optimal values in the experiments of OI-SVM and ASC methods. The sigmoid parameters for the learn++.NSE recommended in [3] were set to 0.5 and 10. The learn++.NSE and WMV applied classification and regression tree (CART) as their base classifier and the ensemble size was limited to the number of MSNL neurons according to each experiment. The parameters and their values used in MSNL experiment were given by Table 4.3. All results summarized in this section are averaged according to 10 runs. By considering the data sets, we randomly divided each data set into five disjoint subsets (a 5-fold cross validation strategy). Then, four subsets were used as the training data and the rest as the testing data. Each of five subsets was taken turns to be a testing subset and the rest were the training subsets. This taking turns was repeated five times in each run. Consequently, the average accuracy and computational time were used for evaluating the performance of each algorithm.

Table 4.3 Setting the parameters in experiments

Data sets	# Classes (m)	I	ε
Balance	m=3	10*m	0.3
BreastCancer	m=2	4*m	0.2
Haberman	m=2	50*m	0.3
Sonar	m=2	50*m	0.3
Thyroid	m=3	50*m	0.2
Vertebral	m=3	4*m	0.2
Movement	m=15	4*m	0.2
Wine	m=3	10*m	0.2
Image	m=7	50*m	0.2
Waveform	m=3	8*m	0.3
Weather	m=2	10*m	0.2
Electric	m=2	10*m	0.2

4.2.1 Experimental results with special structure of class 0

The names of data sets, number of instances, number of attributes (or dimensions), and number of classes used in my experiments are summarized in Table 4.1. The comparison results from different algorithms for all data sets are shown in Table 4.4 and Figure 4.1 displays these average accuracy in terms of expired and non-expired data in a form of graph. The comparison was focused on the learning time and the average accuracy of classification with standard deviation shown in parenthesis. There were five folds in each experiment. My Hybrid-MSNL achieved the almost highest average accuracy for all data sets because Hybrid-MSNL gradually and temporally adjusted the neural parameters according to the new incoming data chunk during the learning process but the other algorithms adjusted their neural parameters based on the whole data set. This may imply that local information of how the data are distributed in the space is rather crucial to speed up the learning time complexity and accuracy. For some data sets, my approach spent more learning time than learn++.NSE and WMV methods.

Table 4.4 The average performance of data sets with special structure of class 0

Data sets	Hybrid-MSNL		VEBF		OI-SVM		NSE	WMV		
	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation
Derm	2.94	**78.37(3)**	2.6	74.16(3)	16.7	65.97(2)	**1.19**	29.53(1)	1.293	54.01(1)
Vehicle-Silshouette	19.8	**75.47(2)**	22.9	59.81(1)	205	51.58(2)	15.68	42.1(1)	**15.57**	51.79(1)
Leaf	**0.98**	**76.78(3)**	1.57	69.56(4)	206.79	68.59(3)	1.38	57.09(2)	1.03	55.62(1)
EyeDetect	121.98	**78.23(2)**	129.42	59.47(4)	13591.59	67.34(5)	**30.05**	60.16(2)	34.53	70.42(2)
Spam	57.09	**81.28(3)**	69.89	71.84(8)	2754	61.5(4)	18.24	73.86(1)	**17.18**	73.37(2)
PimaDiabetes	2.61	**80.66(4)**	2.47	61.77(4)	11.25	68.2(9)	**1.487**	51.31(1)	1.584	63.47(1)
Banknote-Authen	2.47	**98.24(1)**	1.49	61.61(4)	5.96	71.65(3)	1.438	62.43(2)	**1.423**	55.43(3)
Skin	**317**	**82.99(3)**	1896	79.48(2)	30163	75.89(5)	1984	66.03(2)	1576	56.08(2)
Weather	18.12	**80.52(5)**	29.59	60.65(6)	154.89	55.89(7)	19.67	70.42(2)	**15.89**	65.49(3)
Electric	29.86	**75.29(3)**	41.61	59.15(3)	2497.14	60.16(5)	26.81	61.22(4)	**17.87**	64.19(3)

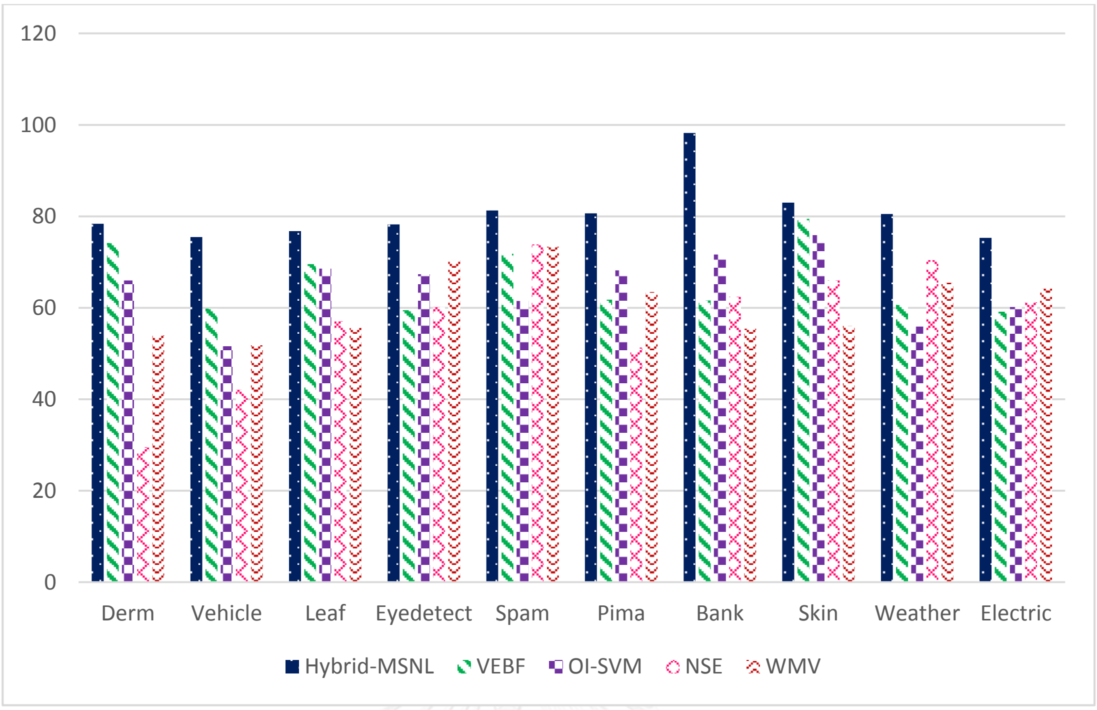


Figure 4.1 The comparison of overall average accuracy in terms of in terms of expired and non-expired data

To demonstrate the effect of percentage of expired data versus the accuracy, five experiments with different percentages 0%, 5%, 10%, 15%, and 20% of expired data were tested. The overall average accuracy of class 0 and the other classes with different percentage of expired data obtained from each algorithm for *BanknoteAuthen* data set is shown in Figure 4.2. Each experimental result with different percentage depicts in a form of graph shown in Figures 4.3 – 4.7. The results concerning this effect for the other data sets are not shown due to the sizes of Figures. Observe that the average accuracy of expiration class of Hybrid-MSNL and learn++.NSE is rather stable and almost independent from the percentage of expired data. But this is not true for the other algorithms. However, for the non-expiration classes, my approach produced slightly less accuracy than that of OI-SVM, but my approach produced more accuracy than learn++.NSE and others'.

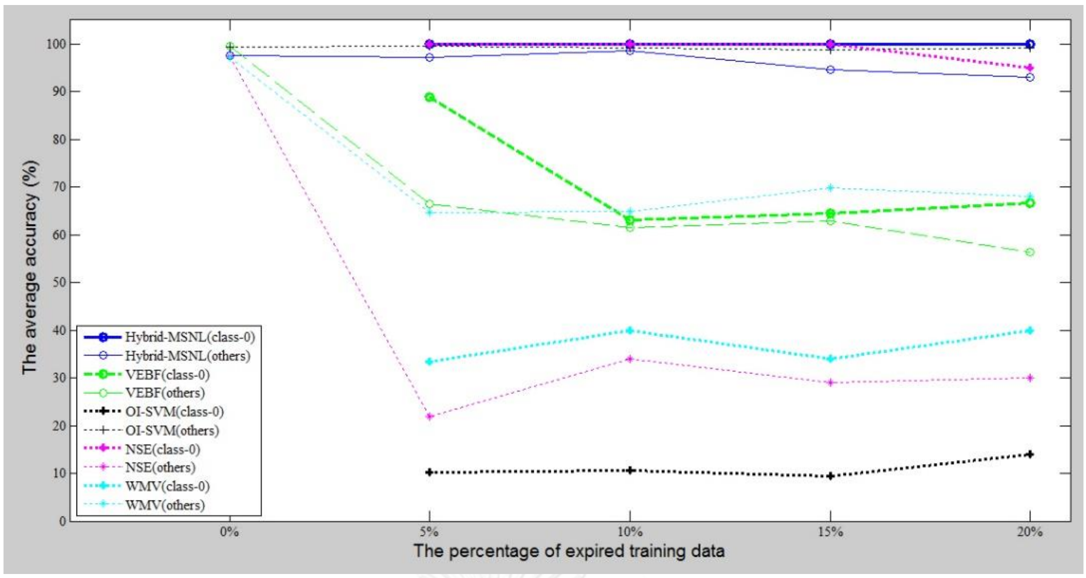


Figure 4.2 The average accuracy of *BanknoteAuthen* data set

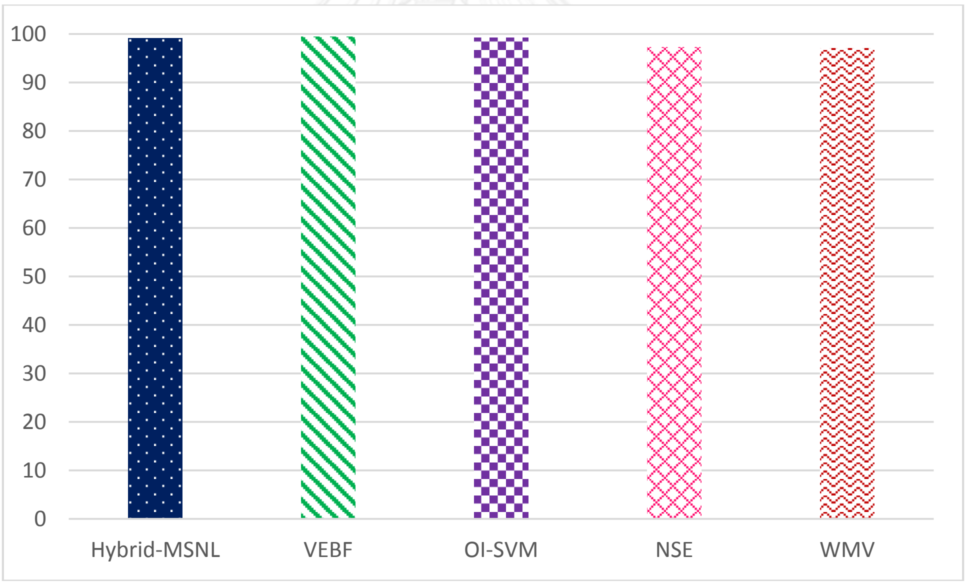


Figure 4.3 The average accuracy of *BanknoteAuthen* data set without expired data

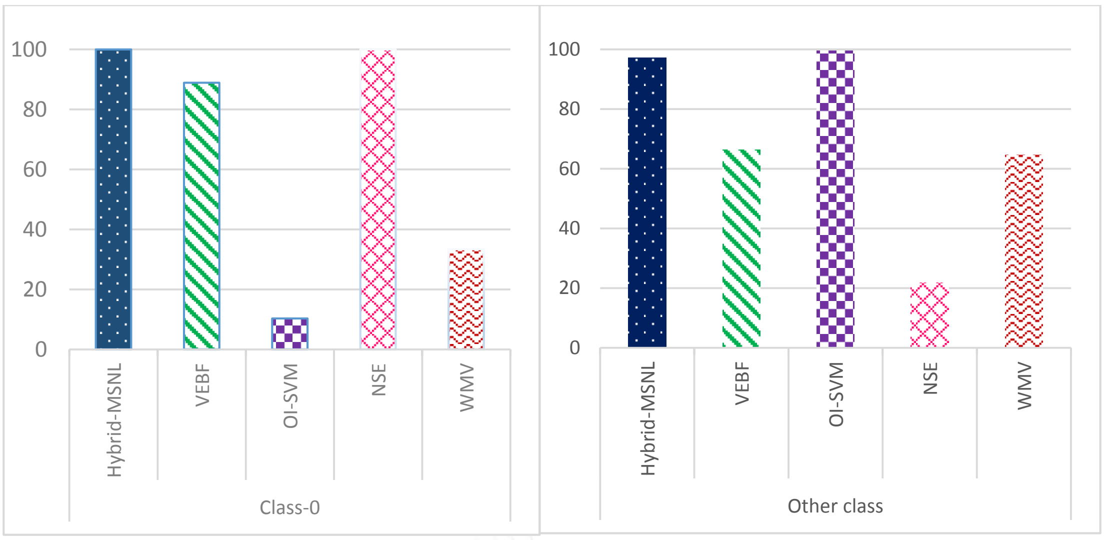


Figure 4.4 The average accuracy of *BanknoteAuthen* data set with percentage 5% of expired training data

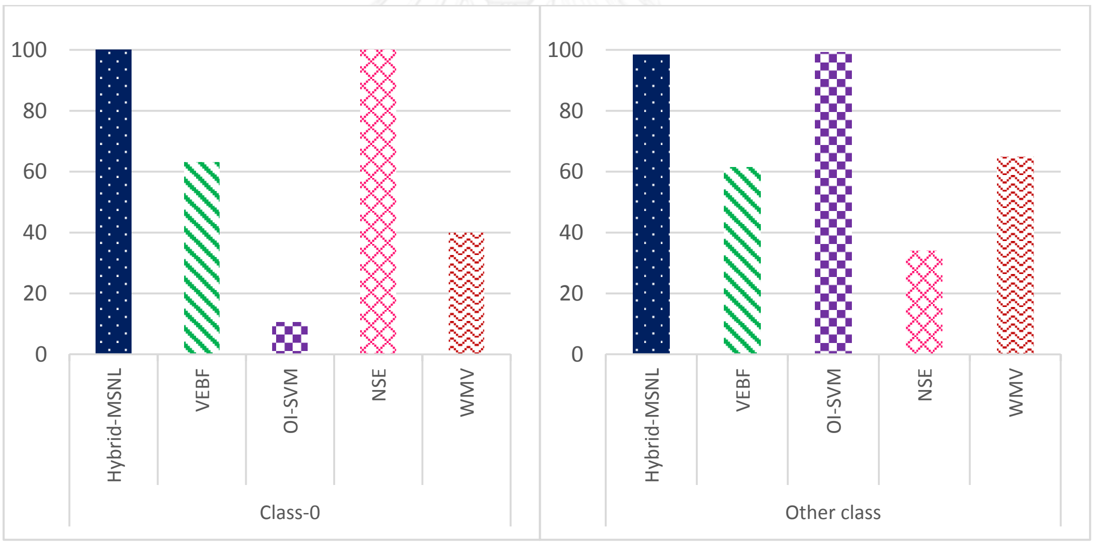


Figure 4.5 The average accuracy of *BanknoteAuthen* data set with percentage 10% of expired training data

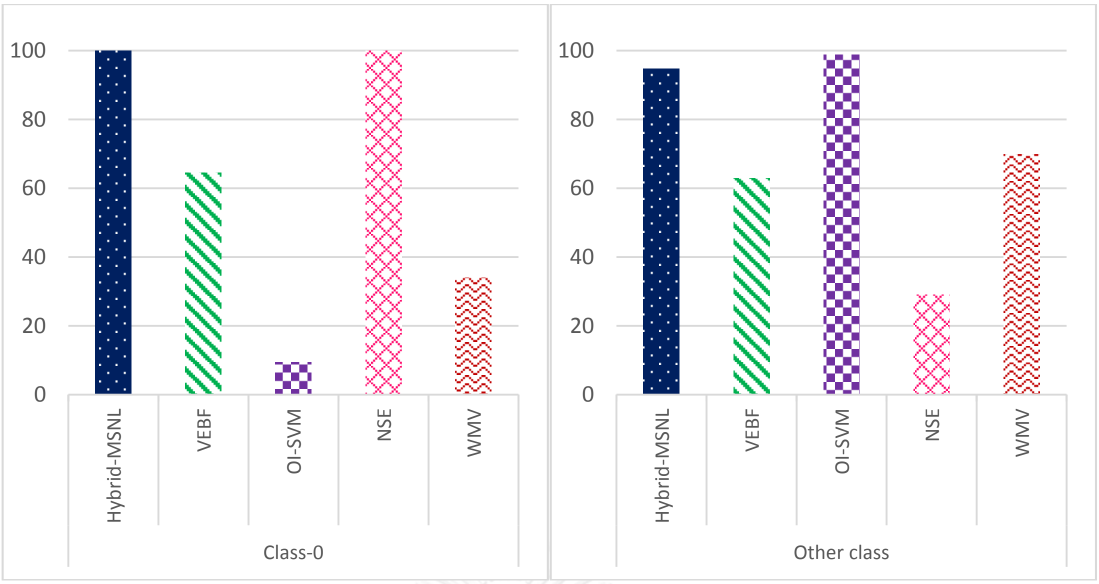


Figure 4.6 The average accuracy of *BanknoteAuthen* data set with percentage 15% of expired training data

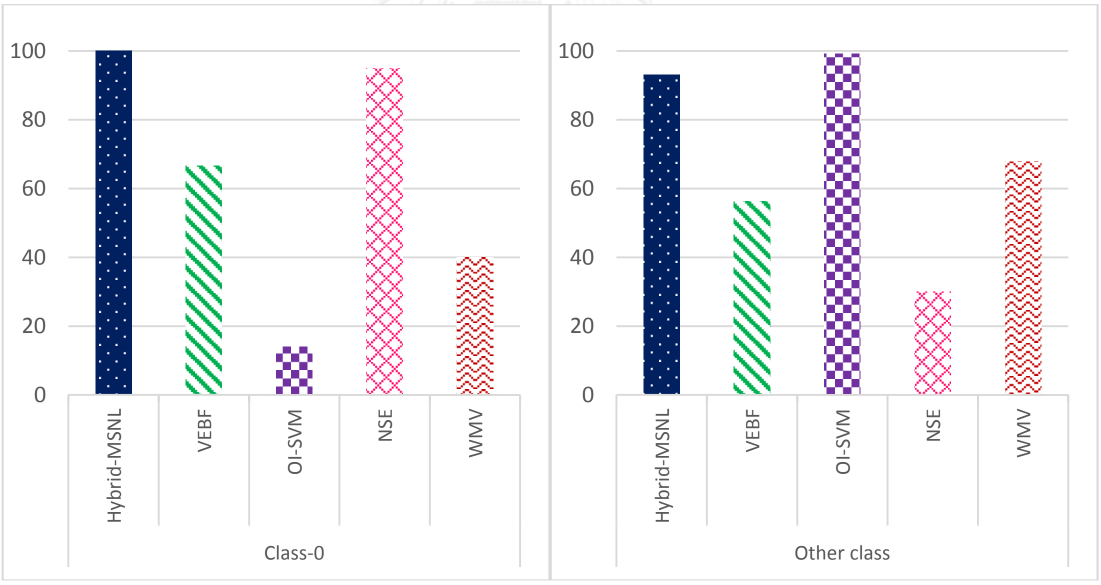


Figure 4.7 The average accuracy of *BanknoteAuthen* data set with percentage 20% of expired training data

4.2.2 Experimental results with temporal class change

The names of data sets, number of instances, number of attributes (or dimensions), and number of classes used in my experiments are summarized in Table 4.2. The comparison results from different algorithms for all data sets are shown in Table 4.5 and Figure 4.8 displays these average accuracy in terms of expired and class-changed data in a form of graph. The comparison was focused on the learning time and the accuracy of classification with standard deviation shown in parenthesis. There were five folds in each experiment. My MSNL achieved the highest average accuracy for all data sets because MSNL gradually and temporally adjusted the neural parameters according to the new incoming data chunk during the learning process but the other algorithms adjusted their neural parameters based on the whole data set. This may imply that local information of how the data are distributed in the space is rather crucial to speed up the learning time complexity and accuracy. For some data sets, my approach spent more learning time than VEBF, WMV, and ASC methods.

Table 4.5 The average performance of data sets with temporal class change

Data sets	MSNL		VEBF		OI-SVM		ASC		NSE		WMV	
	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation	Training time	Accuracy (%) with its standard deviation
Balance	0.75	**87.64(3)**	1.17	74.28(2)	16.7	76.85(3)	10.32	60.29(1)	1.26	69.97(5)	**0.68**	65.12(4)
Breast-Cancer	**1.37**	**88.36(6)**	2.26	66.74(6)	1.93	75.69(2)	1.92	74.57(6)	1.47	73.66(1)	1.41	69.12(3)
Haberman	1.36	**89.48(3)**	1.71	69.13(10)	8.77	68.2(10)	9.72	69.09(12)	2.15	68.41(3)	**1.29**	61.73(4)
Sonar	0.42	**86.63(3)**	**0.39**	79.85(7)	0.76	64.26(10)	0.67	58.79(13)	0.48	70.4(4)	0.46	65.09(4)
Thyroid	**0.71**	**90.36(6)**	0.86	73.56(5)	1.08	82.55(3)	0.121	68.55(10)	0.94	84.99(8)	0.82	76.37(5)
Vertebral	**0.15**	**88.04(4)**	**0.15**	75.43(9)	2.32	86.52(6)	0.18	76.09(6)	0.17	80.43(2)	0.16	73.44(4)
Movement	6.68	**90.59(7)**	8.64	76.98(10)	6.47	70.95(4)	**6.09**	61.11(3)	7.34	62.86(5)	6.26	62.7(3)
Wine	0.41	**81.17(4)**	0.47	72.87(3)	0.67	65.71(9)	0.97	68.43(6)	0.49	75.86(5)	**0.29**	72.76(2)
Image	**17.7**	**87.68(2)**	25.9	78.96(11)	44.43	65.57(12)	79.12	75.67(2)	68.78	68.11(3)	36.52	60.73(9)
Waveform	15.7	**75.82(6)**	16.6	65.26(3)	40.56	67.91(5)	29.58	49.77(3)	29.08	65.53(3)	**14.34**	59.09(6)
Weather	**24.12**	**82.85(5)**	25.59	61.69(6)	159.89	58.76(8)	120.23	53.66(7)	58.67	71.22(6)	37.89	66.39(7)
Electric	**34.99**	**78.46(8)**	35.12	67.47(7)	249.43	70.49(10)	213.87	65.97(8)	95.52	69.53(9)	59.11	68.69(6)

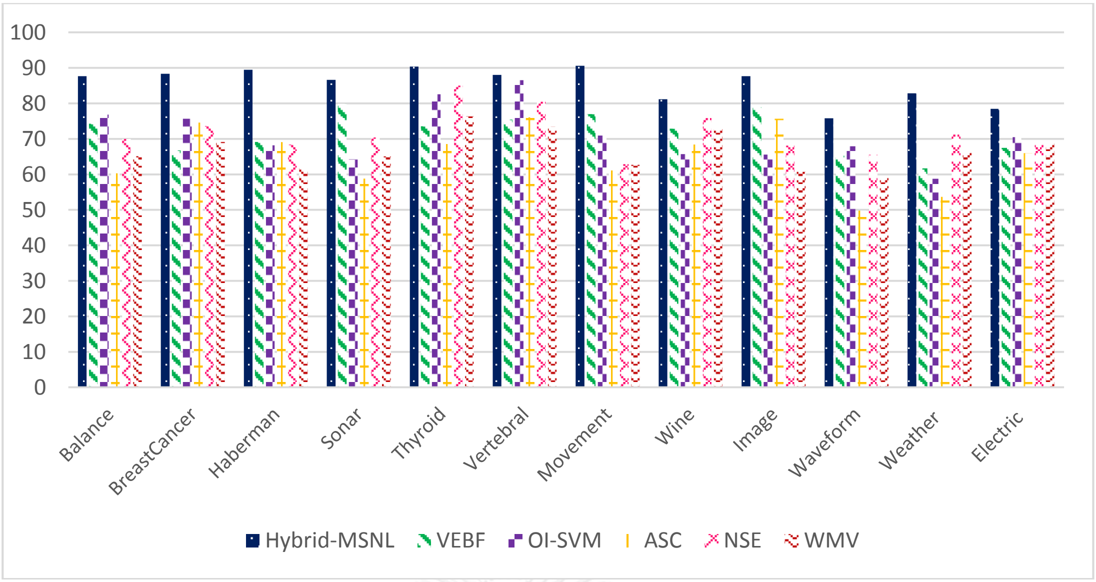


Figure 4.8 The comparison of overall average accuracy in terms of expired and class-changed data

To demonstrate the effect of percentage of data expired and changed class versus the accuracy, four experiments with different percentages 0%, 10%, 20%, and 30% of expired and class-changed data were tested. The overall average accuracy of non-expired, expired, and class-changed classes with different percentage of expired data obtained from each algorithm for *Thyroid* data set is shown in Figure 4.9. Each experimental result with different percentage depicts in a form of graph shown in Figures 4.10 - 4.13. The results concerning this effect for the other data sets are not shown due to the sizes of Figures. Observe that the average accuracy of MSNL is rather stable and almost independent of the percentage of expired data and data changed class. But this is not true for the other algorithms. However, for live classes, my approach produced slightly less accuracy than that of OI-SVM and learn++.NSE.

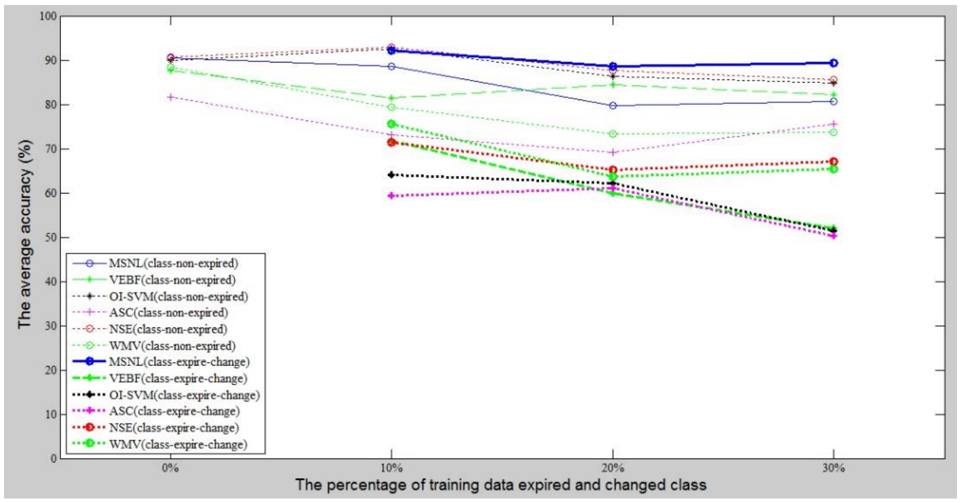


Figure 4.9 The average accuracy of *Thyroid* data set

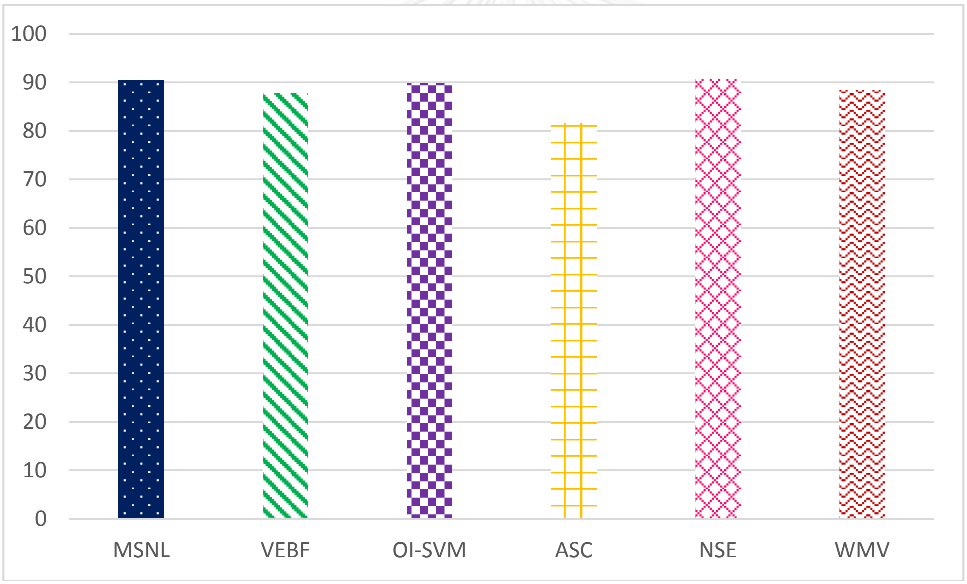


Figure 4.10 The average accuracy of *Thyroid* data set without expired data

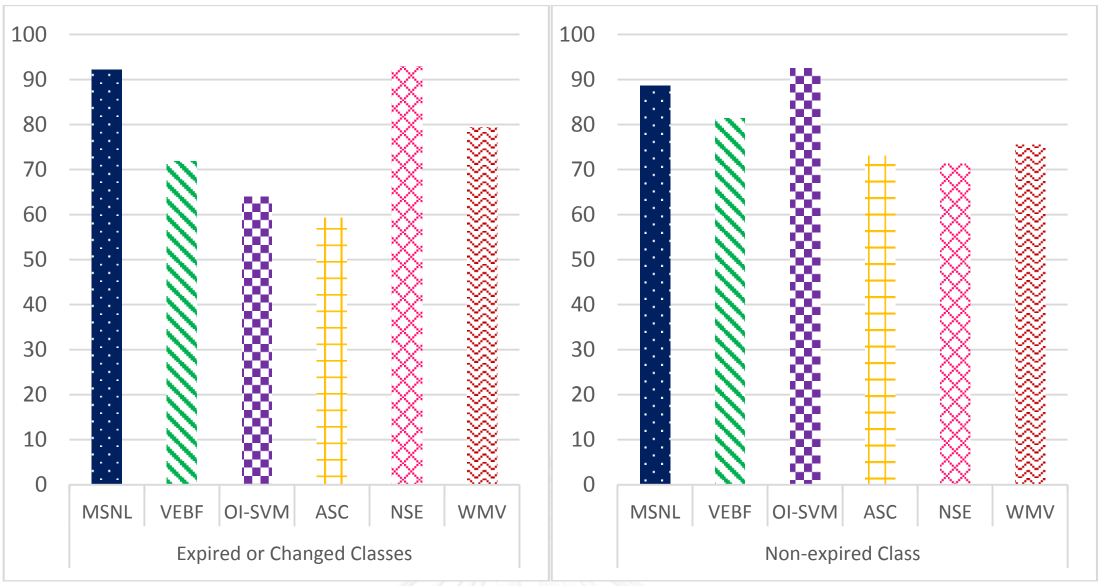


Figure 4.11 The average accuracy of *Thyroid* data set with percentage 10% of expired and class-changed training data

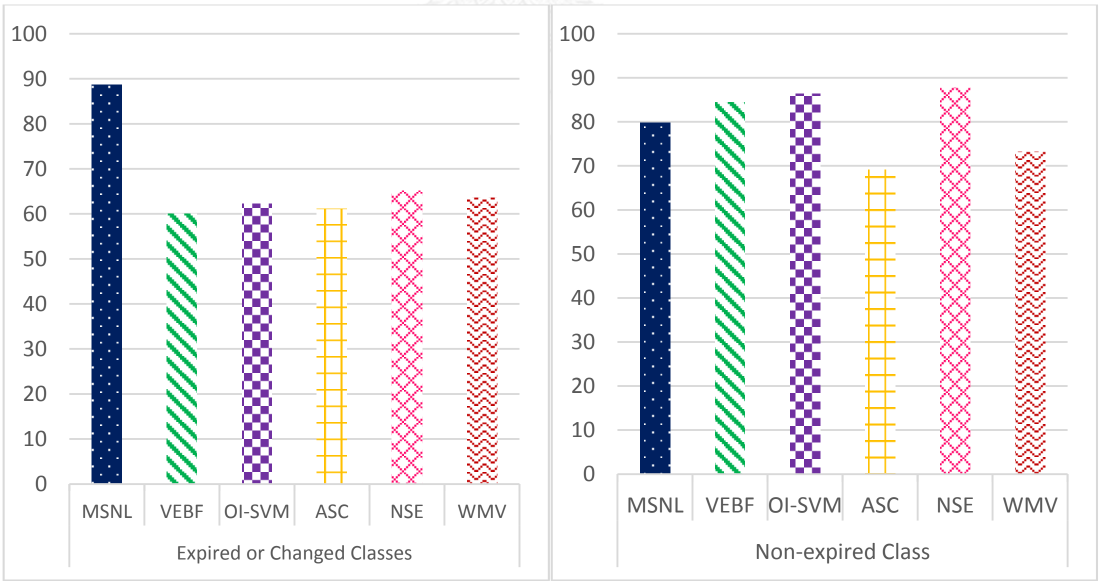


Figure 4.12 The average accuracy of *Thyroid* data set with percentage 20% of expired and class-changed training data

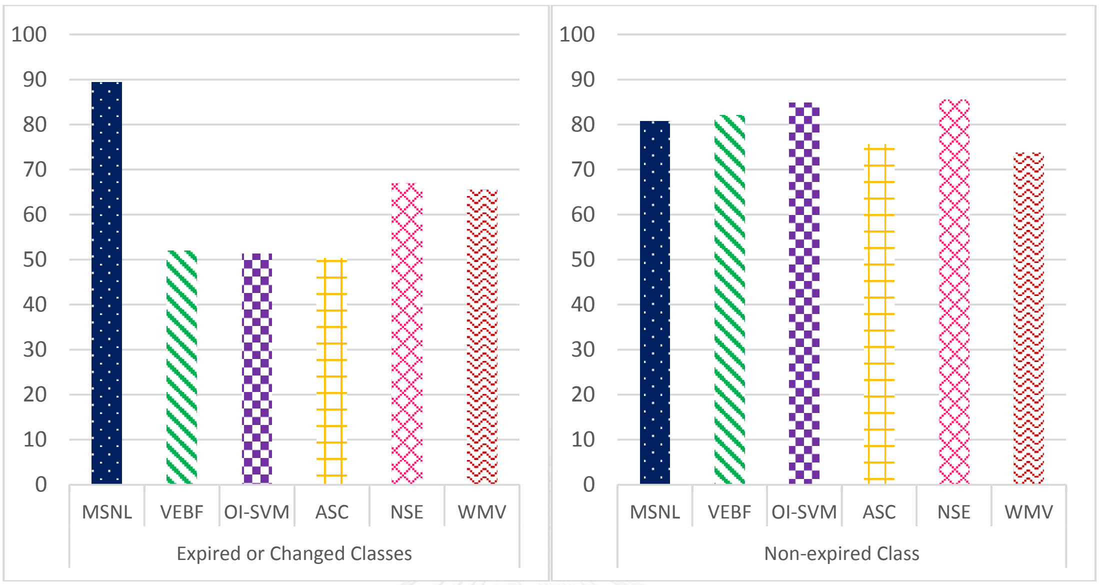


Figure 4.13 The average accuracy of *Thyroid* data set with percentage 30% of expired and class-changed training data

4.2.3 Experiments on concept drift problem

The SEA data stream of classical concept drift problem discussed in [29] was tested to evaluate the performance of the proposed algorithm and ensemble methods of classifiers for handling the concept drift scenario. Also, a real-world data set of non-stationary environments, weather prediction [3] is another data stream experimented in this problem. They were separated into chunks using test-then-train strategy. Figure 4.14 and Figure 4.15 show the experimental results in classification error of SEA and Weather data sets, respectively. From these figures, the proposed MSNL performed tracking the drifting distribution achieving accuracies nearly the same as Learn++.NSE and WMV. In some time steps of Weather real-world data set, MSNL performed better than Learn++.NSE and WMV.

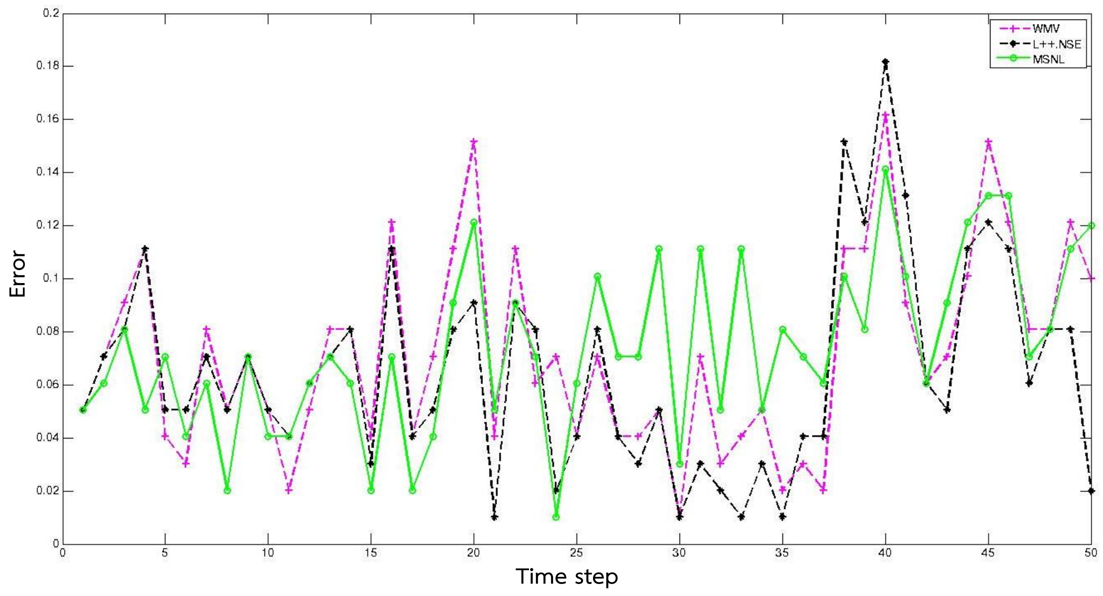


Figure 4.14 Classification error of algorithms on the classical SEA data stream

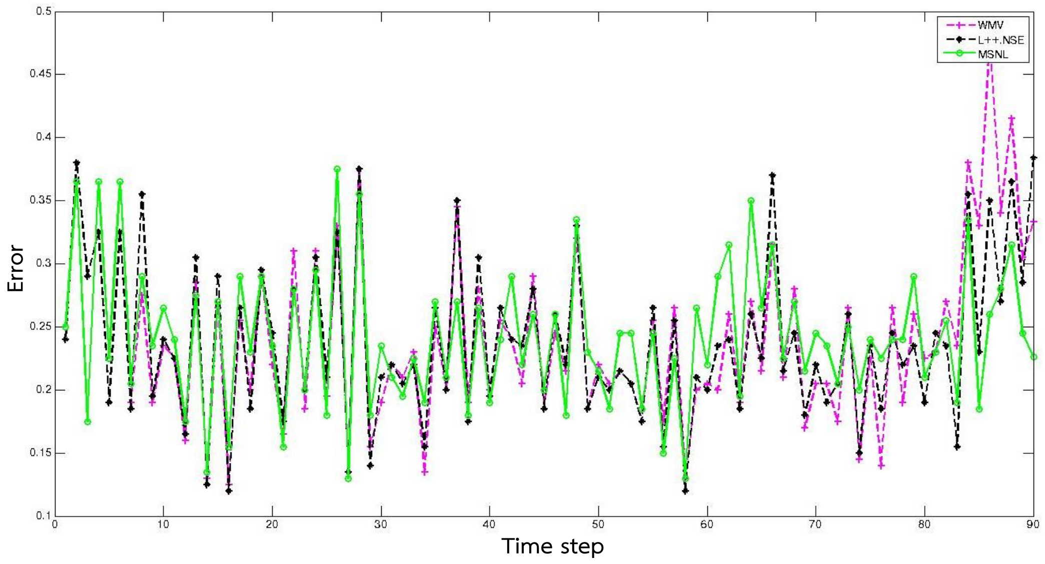


Figure 4.15 Classification error of algorithms on the Weather data set

4.3 Discussion

The proposed algorithm can be applied with randomly expiring or class-changing of data under non-stationary situation. This situation is similar to concept drift problem or non-stationary environment. The results in the experiments serve that the proposed MSNL is robust to changing on various environments. However, the updated parameters of each neuron of the proposed network are slightly shrunk or exceedingly expanded according to incoming data that are expired or even changed in classes during the learning process affecting the performance of proposed algorithm.

4.3.1 Analysis of complexity with special structure of class 0

In case of non-expired data, the learning time complexity of proposed algorithm is equal to that of VEBF algorithm with respect to the number of non-expired data. But in case of expired data, the learning time complexity involves the computational time of pseudoinverse matrix process [36] and time to compute $\mathbf{u}^T(i)$ and $\Delta^T(i)$. If there are L expired data vectors in R^d, then pseudoinverse time complexity is $O(d^2L)$. The time complexity to compute both $\mathbf{u}^T(i)$ and $\Delta^T(i)$ is $O(L)$.

Similar to the time complexity, the space complexity concerns two aspects. The first one is the space complexity for leaning non-expired data. The analysis of this complexity is rather complex because it must involve the number of expired data covered by each neuron. As previously discussed, a neuron with its links can be entirely removed from the network if all its learned data are expired data. Obviously, this situation effects the analysis of space complexity. Therefore, the analysis of space complexity for non-expired data will be the further study. The second aspect is the space complexity for expired data. Let p be the first data chunk. Since all expired data are captured in forms of vectors $\mathbf{u}^T(i)$ and $\Delta^T(i)$ which can be recursively computed from $\mathbf{V}^+_{[1,p]}, \mathbf{V}_{[p+1,L]}, \mathbf{u}^T(i-1)$, and $\Delta^T(i-1)$, hence the space complexity is equal to $O(pd)+O(dL)+O(d)+O(d)=O(\max(p,L))$.

4.3.2 Analysis of complexity with temporal class change

In case of live data, the learning time complexity of proposed algorithm is equal to that of VEBF algorithm with respect to the number of live data. However, in case of expired data, the learning time complexity is additionally computed as twice of the previous case because a neuron of expired data can be recreated after being removing from the network of live data in the lower stratum. For case of class change, the learning time complexity is also the same as that of the case previously discussed due to the removal of neuron from the network of expired data and revived to the network of live data again.

Similar to the time complexity, the space complexity concerns two aspects. The first one is the space complexity for learning live data. The analysis of this complexity is rather complex because it must involve the number of expired data covered by each neuron. As previously discussed, a neuron with its links can be entirely removed from the network if all its learned data are expired. Obviously, this situation effects the analysis of space complexity. Therefore, the analysis of space complexity for live data will be the further study. The second aspect of expired and class changed data is similar to in case of the non-expired data.

CHAPTER V
CONCLUSION

In various applications, the lifetime of data must be concerned to determine the classes. The problem of learning both live and expired data was studied. A new learning algorithm using the structure of multi-stratum network named ***Multi-Stratum Network Learning (MSNL)*** was proposed to learn these data. The main structure consists of two strata. The first stratum is similar to the conventional feed-forward structure but the proposed structure can be dynamically and temporally changed according to the status of incoming data, i.e. live and class change. A set of new recursive functions for computing mean, variance, and covariance matrix when some data are removed was proposed to achieve the minimum computational space complexities and to speed up the learning time. The second stratum is for storing only those expired data. The comparison of experimental results from MSNL and the other algorithms with several benchmarked data sets signified that MSNL achieved a fast speed as well as higher accuracy than the others'.

REFERENCES

[1] A. Asuncion and D. J. Newman. UCI Repository of Machine Learning, University of California, Irvine, School of Information and Computer Sciences [Online]. Available: http://archive.ics.uci.edu/ml/

[2] M. Harries, "SPLICE-2 comparative evaluation: Electricity pricing," *Tech. Rep.,* p. 9905, 1999.

[3] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Netw.,* vol. 22, pp. 1517-1531, 2011.

[4] S. Jaiyen, C. Lursinsap, and S. Phimoltares, "A very fast neural learning for classification using only new incoming datum," *IEEE Trans. Neural Netw.,* vol. 21, pp. 381-392, 2010.

[5] M. Thakong, S. Phimoltares, S. Jaiyen, and C. Lursinsap, "One-pass-throw-away learning algorithm based on hybridization of LDA and PCA," in *International Conference on Information Science and Applications*, Chonburi, Thailand, 2013, pp. 445-448.

[6] J. Zheng, H. Yu, F. Shen, and J. Zhao, "An online incremental learning support vector machine for large-scale data," *Neural Comput. Appl.,* vol. 22, pp. 1023-1035, 2013.

[7] S. Ozava, S. Pang, and N. Kasabov, "Incremental learning of chunk data for online pattern classification systems," *IEEE Trans. Neural Netw.,* vol. 16, pp. 1061-1074, 2008.

[8] S. Furao and O. Hasegawa, "A fast nearest neighbor classifier based on self-organizing incremental neural network," *Neural Netw.,* vol. 21, pp. 1537-1547, 2008.

[9] H. He, S. Chen, and X. X. K. Li, "Incremental learning from stream data," *IEEE Trans. Neural Netw.,* vol. 22, pp. 1901-1914, 2011.

[10] H. Abdulsalam, D. B. Skillicorn, and P. Martin, "Classification using streaming random forests," *IEEE Trans. Knowl. Data Eng.,* vol. 23, pp. 22-36, 2011.

[11] X. Wu, P. Li, and X. Hu, "Learning from concept drifting data streams with unlabeled data," *Neurocomputing,* vol. 92, pp. 145-155, 2012.

[12] N. Wattanakitrungroj and C. Lursinsap, "Memory-less unsupervised clustering for data streaming by versatile ellipsoidal function," in *the 20th ACM Conference on Information and Knowledge Management*, Glasgow, United Kingdom, 2011, pp. 967-972.

[13] S. Ozawa, A. Roy, and D. Roussinov, "A multitask learning model for online pattern recognition," *IEEE Trans. Neural Netw.,* vol. 20, pp. 430-445, 2009.

[14] T. Tokumoto and S. Ozawa, "A fast incremental kernel principal component analysis for learning stream of data chunks," in *the International Joint Conference on Neural Networks*, San Jose, California, USA, 2011, pp. 2881-2888.

[15] S. Pang, T. Ban, Y. Kadobayashi, and N. K. Kasabov, "LDA merging and splitting with applications to multi-agent cooperative learning and system alteration," *IEEE Trans. Syst. Man, Cybern. B, Cybern.,* vol. 42, pp. 552-563, 2012.

[16] S. Okada and T. Nishida, "Online incremental clustering with distance metric learning for high dimensional data," in *the International Joint Conference on Neural Networks*, San Jose, California, USA, 2011, pp. 2047-2053.

[17] H. He and S. Chen, "IMORL: Incremental multiple-object recognition and localization," *IEEE Trans. Neural Netw.,* vol. 19, pp. 1727-1738, 2008.

[18] R. Elwell and R. Polikar, "Incremental learning in nonstationary environments with controlled forgetting," in *the International Joint Conference on Neural Networks*, Atlanta, Georgia USA, 2009, pp. 771-778.

[19] L. Kuncheval and I. Zliobaite, "On window size change for classification in changing environments," *Intell. Data Anal.,* vol. 13, pp. 861-872, 2009.

[20] D. Martinex-Rego, B. Perez, O. Fontenla-Romero, and A. Alonso-Betanzos, "A robust incremental learning method for non-stationary environments," *Neurocomputing,* vol. 74, pp. 1800-1808, 2011.

[21] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *the Ninth ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining, Washington, DC, USA, 2003, pp. 226-235.

[22] A. A. Beyene, T. Welemariam, M. Persson, and N. Lavesson, "Improved concept drift handling in surgery prediction and other applications," *Knowl. Inf. Syst.,* vol. 43, pp. 389-416, 2014.

[23] L. Hartert and M. Sayed-Mouchaweh, "Dynamic supervised classification method for online monitoring in non-stationary environments," *Neurocomputing,* vol. 126, pp. 118-131, 2014.

[24] Y. Yeh and Y. F. Wang, "A rank-one update method for least squares linear discriminant analysis with concept drift," *Pattern Recognit.,* vol. 45, pp. 1267-1276, 2013.

[25] B. Mirza and N. L. Z. Lin, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing,* vol. 149, pp. 316-329, 2015.

[26] G. Ditzler, G. Rosen, and R. Polikar, "Discounted expert weighting for concept drift," in *IEEE Symposium of Computational Intelligence in Dynamic and Uncertain Environments*, Singapore, 2013, pp. 61-67.

[27] G. Ditzler, G. Rosen, and R. Polikar, "Domain adaptation bounds for multiple expert systems under concept drift," in *the International Joint Conference on Neural Networks*, Beijing, China, 2014, pp. 595-601.

[28] A. Dries and U. Ruckert, "Adaptive concept drift detection," *Statistical Anal. Data Mining,* vol. 2, pp. 311-327, 2009.

[29] W.N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Seventh ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, San Francisco, CA, USA, 2001, pp. 377-382.

[30] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst. Man, Cybern. C, Appl. Rev.,* vol. 31, pp. 497-508, 2001.

[31] M. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient

incremental learning of new classes," *IEEE Trans. Neural Netw.,* vol. 20, pp. 152-168, 2009.

[32] R. Polikar, J. DePasquale, H. S. Mohammed, G. Brown, and L. I. Kuncheva, "Learn++.MF: A random subspace approach for the missing feature problem," *Pattern Recognit.,* vol. 43, pp. 3817-3832, 2010.

[33] T. N. E. Greville, "Some Applications of the pseudoinverse of a matrix," *Siam Review,* vol. 2, pp. 15-22, 1960.

[34] J. Tapson and A. Van Schaik, "Learning the pseudoinverse solution to network weights," *Neural Netw.,* vol. 45, pp. 94-100, 2013.

[35] F. E. Udwadia and R. E. Kalaba, "General forms for the recursive determination of generalized inverses: unified approach," *Journal of Optimization Theory and Applications,* vol. 101, pp. 509-521, 1999.

[36] S. Predrag, M. Marko, S. Igor, and M. Sladjana, "Application of the partitioning method to specific toeplitz matrices," *Int. J. Appl. Math. Comput. Sci.,* vol. 23, pp. 809-821, 2013.

[37] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera, "Study on the impact of partition-induced data set shift on k-fold cross-validation," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 23, pp. 1304-1312, 2012.

APPENDIX

VITA

Name: Mr. Mongkhon Thakong

Date of Birth: May 12, 1977

Educations:

1999 Graduate Bachelor Degree of Mathematics, Department of Mathematics, Faculty of Science, Khon Kaen University

2002 Graduate Post-baccalaureate Certificate in Information Technology System Development, National Institute of Development Administration

2007 Graduate Master Degree of Computer Science, Department of Computer Science, Faculty of Science, Khon Kaen University

Publications:

M. Thakong, S. Phimoltares, S. Jaiyen, and C. Lursinsap, One-pass-throw-away learning algorithm based on hybridization of LDA and PCA, in: Proceedings of the International Conference on Information Science and Applications, 2013, Chonburi, Thailand, pp. 445-448.