IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 25, 2020, accepted May 14, 2020, date of publication May 27, 2020, date of current version June 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997944

Scalable Hyper-Ellipsoidal Function With Projection Ratio for Local Distributed Streaming Data Classification

PERASUT RUNGCHARASSANG AND CHIDCHANOK LURSINSAP, (Member, IEEE)
Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand
Corresponding author: Perasut Rungcharassang (perasut.r@gmail.com)

This work was supported by the Thailand Research Fund under Grant RTA6080013.

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

INDEX TERMS Streaming data classification, discriminant analysis, discard-after-learn, incremental learning, projection ratio.

I. INTRODUCTION

Classifying or learning streaming data has been an interesting topic and existed in many fields such as business (financial data [1], credit card fraud detection [2]), academia, and medical information (health care sensor [3], EEG signal [4], [5]). Streaming data refers to the data that are continuously generated without any time bound. The amount of generated data at any point in time may be varied according to their environment. The data flow into a learning process in forms of either one datum at a time or one chunk of unfixed size at a time. One of the obvious examples of streaming data is the data temporally generated throughout the internet in every unit time. It is remarkable that the speed of data generation goes beyond the speed of developing the technology for memory capacity per unit area. Due to this technology lag, the streaming data cannot be entirely stored inside the memory causing the problem of how to accurately learn these data in real time and online applications. Generally, this type of data has many names such as *streaming data*, *data streams*, *online data*, and *real-time data*. This situation leads to a very challenging development of new neural learning algorithms to cope with the problems of learning streaming data under the constraints of memory overflow, fast learning speed, as well as limited computing resources such as low energy consumption [6]–[8]. Moreover, the neural learning speed must be faster than the speed of incoming data in order to avoid any data loss and to achieve the classification accuracy with respect to the occasional testing data.

Streaming data can be in various forms but this study concerns only the numeric data in the form of continuous chunks of vectors representing a set of features extracted from some learning objects such as images, characters. The process of how to extract important features is not considered in this study. In addition, some specific characteristics of data and targets such as concept drift [9]–[12] and time series are not involved because these characteristics occur only in some applications.

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Cusano.

Traditional classification methods such as k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA) were built on the condition that the entire data set must be kept in the memory during the training process. These traditional methods can also be applied to streaming data but the results are not accurate because the entire training data set is not known during the training period. Only those incoming data are involved in the training process. Besides, the new incoming data must be combined with the previously trained data in order to retrain the network but this creates the consequence of memory overflow. Some new incoming data are lost after the memory overflow crisis. The retraining of augmented training data set makes the number of epochs uncontrollably increase. However, some attempts such as incremental learning were introduced to solve the memory overflow constraint by controlling the number of deployed neurons.

Incremental learning method gradually learns the data from small portions of a whole data set by adding one or more neurons at a time. It is suitable to learn big data or streaming data when memory overflow may occur. This approach has been applied and combined with various classification methods such as LDA [6], [13]–[15], Neural Network [16], ILDA [17], and SVM [18], [19]. Although incremental learning achieved some success, the constraint of memory overflow still exists and the learning time is still uncontrollable. There are several structures of mathematical functions behind the incremental learning based on the data distribution which can be either linear or non-linear.

Handling a nonlinear classification problem may be divided by three main approaches which are (1) a nonlinear kernel classifiers such as a Radial Basis Function kernel [20], (2) a local hyperplane [21], and (3) a mathematical structure based on local data distribution such as Hyper-ellipsoidal function: the family of Versatile Elliptic Basis Function (VEBF) [22], [23], k-Nearest Neighbors [24], and k-mean clustering [25]. Some approaches [24], [25] require parameters such as the center and the variance of data distribution of each class to be computed before the training process. This implies that the entire training data must be known in advance but it is impossible to do so in case of streaming data.

Although the VEBF-based structure is rather convenient and efficient for the new learning environment and constraints, the learning accuracy depends upon the initial values of VEBF parameters. The problem is that the incoming chunk of data is not the actual population of the whole training data. Thus, the initial values computed from only the incoming data chunk may be inaccurate. Furthermore, updating these parameters must be continuously done during the learning process which makes the time complexity unavoidably increases. Besides the problem of initializing parameters, the procedure to identify the classes of a queried testing datum is mainly based on the distance between the datum and the center of VEBF. This distance works well for some applications but it gives inaccurate classification results in most cases because the data density and distribution inside the VEBF is not involved. The data density and distribution can be viewed as the information stored inside the VEBF. Identifying the class of a queried datum must minimum interfere the data information of the corresponding VEBF. Another problem not studied in those VEBF-based structure is the situation when the number of feature dimensions is larger than the number of training data. This implies that the eigenvectors and eigenvalues of the covariance matrix of a VEBF cannot be uniquely computed. Many applications, especially for medical data, face this crisis.

A new classifier method named *Scalable Hyper-Ellipsoidal Function* (SHEF) method is proposed to solve the above mentioned problems in VEBF-based structure learning process. Our approach concerns the direction of data distribution inside any local cluster. The local characteristics imply the local importance of each dimension which can be used to compute the nearest distance between a VEBF and a queried datum in order to identify its correct class. Several new concepts are introduced to solve the problems such as projection ratio, checking overlap, being inside, and touching of two hyper-ellipsoids.

The rest of paper is organized as follows. Section II addresses the constraints and studied problems. Section III summarizes the relevant concept and background used in this study. Section IV discusses the new structure and the equations for updating the parameters of SHEF. Section V explains the proposed new learning method using scalable hyper-ellipsoidal function and illustrates our algorithm step-by-step. Section VI proposes our new distance measure and its concept in order to determine class of queried datum. Experimental data and set-up hyperparameters are mentioned in Section VII. Section VIII reports the experimental results and performance comparison with the other methods, also shows effects of our hyperparameters on classification accuracy and number of SHEFs in Grid Search diagram. Section IX discusses the rationale behind the results. Section X concludes the paper.

II. CONSTRAINTS AND STUDIED PROBLEMS

The proposed method in this study is based on the hyper-elliptical structure similar to those methods in *Versatile Elliptic Basis Function* (VEBF) [22], *Class-wise Incremental Learning* (CIL) [23], *Versatile Hyper-Elliptic Clustering* (VHEC) [26], and *Dynamic multi-Stratum* (Dstratum) [27]. These methods solve the problem of memory overflow occurring in streaming data environment by deploying the concept of *discard-after-learn*, where either one datum or one data chunk of several classes are learned at a time and completely discarded after that. To improve some inferior capabilities of these methods, few new concepts are introduced to speed up the computation of parameter updating of the structure, to relief the curse of dimensionality, and to measure class distance for testing data.

The structure of hyper-ellipsoid is simple and versatile enough for classification but the accuracy of classification of this structure depends upon the initial width of each created

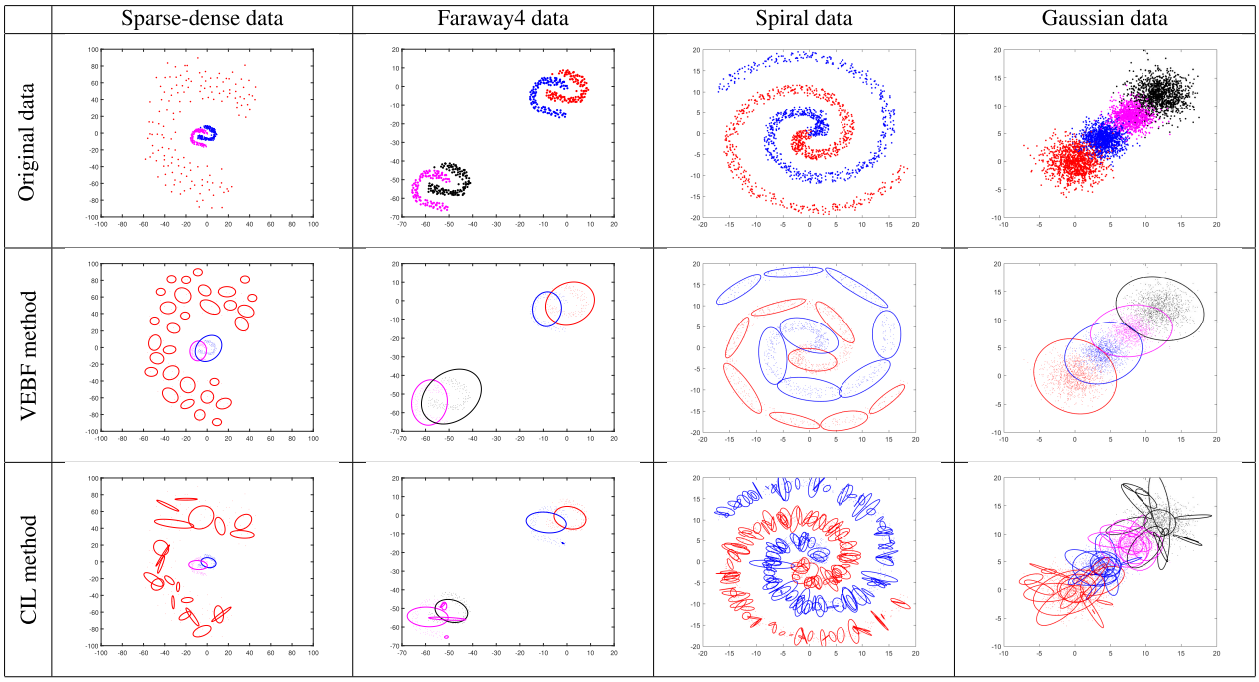


FIGURE 1. Examples of unfavorable results of VEBF and CIL due to initialization of parameters and some steps of learning process. (see online version for color figure).

hyper-ellipsoid. To elaborate how initial width effects the accuracy of classification, Fig. 1 illustrates an example of how VEBF and CIL methods capture four synthetic data sets in the second and the third row, i.e. Sparse-dense data, Faraway4 data, Spiral Data, and Gaussian data, in terms of the number of hyper-ellipsoids and their directions. Orginally, VEBF [22] was designed to learn one vector at a time in order to reach the lower bound of learning time complexity. It explores only the entire training set to compute the average pair distance and uses this distance as the initial width of each hyper-ellipsoid without touching any vectors in the testing set. But CIL [23] extended the concept of VEBF by deploying only the first 20% of the training data. Note that if the initial width is not properly initialized, then hyper-ellipsoids from different classes may overlap. Moreover, if the initial width is too small, then too many redundant hyper-ellipsoids are also created during the learning period.

A. CONSTRAINTS

This study focuses only on streaming data environment. Each datum consists of a feature vector and a target. The following constraints are considered in this study.

1) New incoming data gradually flow into the learning process as a chunk of data. The memory size is large enough to hold the incoming data chunk.
2) The probability of data distribution in each class is unknown in advance.
3) Each chunk may contain one or more classes. The number of incoming classes is unknown in advance.
4) The size of each class in each incoming chunk at any time is arbitrary.
5) Learned data are assumed to have no class drift or class change characteristic.
6) Incoming data are completely discarded from the learning process after being leaned (discard-after-learn concept).
7) The size of working memory is assumed to be fixed throughout the learning and testing processes.
8) The learning process is executed by one single processing unit.

B. STUDIED PROBLEMS

The following problems are studied to increase the classification accuracy, to reduce the number of neurons, and to solve the condition of curse of dimensionality due to insufficient amount of data which remains existent in those VEBF-based approaches.

1) How to handle the singular covariance matrix problem due to the condition of curse of dimensionality?
2) How to compute the appropriate initial width without using the whole training data in advance?
3) Is there any new distance measure to select the nearest hyper-ellipsoid with respect to the queried datum?

Some relevant backgrounds used in this study will be given in the next section.

III. RELEVANT BACKGROUND

Our proposed method is related to the structure of hyper-elliptic function introduced in [22], [23] and the

concept of linear discriminant analysis. The summary of each related issue is given in the following subsections.

A. BASIC CONCEPT OF STANDARD HYPER-ELLIPSOID FUNCTION

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

$$\mathbf{S} = \mathbf{E}[(\mathbf{X} - \mathbf{E}[\mathbf{X}])(\mathbf{X} - \mathbf{E}[\mathbf{X}])^T] \tag{1}$$

where $\mathbf{E}[\cdot]$ represents the expected value. To realize the concept of *discard-after-learn*, it would be better to compute matrix $\mathbf{S}$ in the form of summation as defined in (3).

$$\mathbf{c} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{2}$$

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{c})(\mathbf{x}_i - \mathbf{c})^T \tag{3}$$

where $\mathbf{c} \in \mathbb{R}^d$ is the mean or centroid of data vectors in $\mathbf{X}$.

The distribution directions of all data vectors in set $\mathbf{X}$ are the set of eigenvectors of $\mathbf{S}$, denoted by $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d\}$ such that each $||\mathbf{u}_i|| = 1$. The data variances of all eigenvectors are the set of corresponding eigenvalues, denoted by $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_d\}$. The covariance matrix S can be diagonalized in terms of its eigenvalues and eigenvectors as $\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^T$. The generic equation of hyper-ellipsoid can be written in terms of covariance matrix $\mathbf{S}$, center $\mathbf{c}$, and a constant ξ as follows. Constant ξ is for adjusting the size of the hyper-ellipsoid, usually set to 1.

$$(\mathbf{x} - \mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x} - \mathbf{c}) = \xi \tag{4}$$

B. CONCEPT OF LDA WITH MULTIPLE CLASSES AND DICHOTOMOUS CLASSES

Suppose $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is a set of N data vectors with K classes. Let $\mathcal{C}_k$ denote class k and n_k be $|\mathcal{C}_k|$,for $1 \leq k \leq K$. The covariance matrix of each class C_k is denoted by $\mathbf{S}_k$. The centroid of $\mathbf{X}$ is at $\mathbf{c}$ and the centroid of each class k is at $\mathbf{c}_k$. The traditional LDA aims to find $(K-1)$ discriminant vectors formed as a d-by-$(K-1)$ projection matrix $\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_{K-1}]$ in order to maximize the following Fisher criterion [28].

$$\begin{aligned} &\underset{\mathbf{W}}{\text{maximize}}\ J(\mathbf{W}) := \frac{|\mathbf{W}^T\mathbf{S}_B\mathbf{W}|}{|\mathbf{W}^T\mathbf{S}_W\mathbf{W}|} \\ &\text{subject to } ||\mathbf{w}_i|| = 1 \quad \text{where } i = 1, \ldots, K-1. \end{aligned} \tag{5}$$

Between-class scatter matrix $\mathbf{S}_B$ and within-class scatter matrix $\mathbf{S}_W$ are defined as follows.

$$\mathbf{S}_B = \sum_{k=1}^{K} n_k(\mathbf{c}_k - \mathbf{c})(\mathbf{c}_k - \mathbf{c})^T \tag{6}$$

$$\mathbf{S}_W = \sum_{k=1}^{K}\mathbf{S}_k. \tag{7}$$

For a special case, when LDA is only used for a dichotomous class problem (two classes, $K = 2$), the projection matrix $\mathbf{W}$ consists of only one discriminant vector $\mathbf{w}$ of size 1. The value of $\mathbf{w}$ can be computed by the following equation.

$$\mathbf{w} = \frac{\mathbf{S}_W^{-1}(\mathbf{c}_1 - \mathbf{c}_2)}{||\mathbf{S}_W^{-1}(\mathbf{c}_1 - \mathbf{c}_2)||}. \tag{8}$$

C. CHECKING OVERLAP OF TWO HYPER-ELLIPSOIDS

Checking the overlap of two hyper-ellipsoids is essential in order to merge two hyper-elliptic structures of the same class into a larger one. This paper modified the method of checking touch of two ellipsoids at a single point proposed by Alfano and Greer [29]. Their concept is as follows. Let $\mathbf{A}$ and $\mathbf{B}$ be the represented matrices of the first and second ellipsoids, respectively. Suppose $\mathbf{X}$ and $\mathbf{Y}$ are data vectors for the first and second ellipsoids, respectively. The equations of both ellipsoids can be written as follows.

$$\mathbf{X}\mathbf{A}\mathbf{X}^T = 0 \tag{9}$$

$$\mathbf{Y}\mathbf{B}\mathbf{Y}^T = 0. \tag{10}$$

Assume that $\mathbf{X}$ is in both ellipsoids when they touch each other. Thus, we have

$$\mathbf{X}\mathbf{A}\mathbf{X}^T = 0 \tag{11}$$

$$\mathbf{X}\mathbf{B}\mathbf{X}^T = 0. \tag{12}$$

Testing touch of both ellipsoids can be transformed into the process of formulating eigenvalue by these steps. A constant λ is multiplied to matrix $\mathbf{A}$ in (11) first.

$$\mathbf{X}(\lambda\mathbf{A})\mathbf{X}^T = 0. \tag{13}$$

Then subtract (12) from (13) to obtain these equations.

$$\mathbf{X}(\lambda\mathbf{A} - \mathbf{B})\mathbf{X}^T = 0 \tag{14}$$

$$\mathbf{X}\mathbf{A}(\lambda\mathbf{I} - \mathbf{A}^{-1}\mathbf{B})\mathbf{X}^T = 0. \tag{15}$$

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

IV. PROPOSED SCALABLE HYPER-ELLIPSOIDAL FUNCTION AND PARAMETER UPDATING

The previously introduced hyper-ellipsoidal functions [22], [23], [26], [27] were not designed to solve the problem of curse of dimensionality which occurs in various applications. To improve this inferior capability, a new function of hyper-ellipsoid called *scalable hyper-ellipsoidal function* is proposed as follows.

A. STRUCTURE OF SCALABLE HYPER-ELLIPSOIDAL FUNCTION

Expanding or shrinking the size of a hyper-elliptic function requires the computations of eigenvectors and eigenvalues

first. To reduce this prior computations, the following generic form of standard hyper-elliptic function was used in our approach. Instead of setting the right-hand side of standard hyper-elliptic function to a constant of one, this constant is replaced by a positive scalable constant r so that the width of hyper-ellipsoid shape in each dimension can be easily scaled by using only r. The equation of this new scalable hyper-ellipsoidal function (SHEF) is defined as follows.

$$(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c}) = r^2. \tag{16}$$

$\mathbf{x} \in \mathbf{X}$ is a data vector. $\mathbf{c}$ is the center of $\mathbf{X}$. $\mathbf{S}$ is the covariance matrix of $\mathbf{X}$. Let λ_i be the eigenvalue of the i^{th} dimension of SHEF when $r^2 = 1$. If $r^2 \neq 1$, then the new eigenvalue becomes $r^2\lambda_i$, which obviously implies that the λ_i is scaled by r^2. Fig. 2 illustrates the geometrical structure and its eigenvalues when the scalable constant $r > 0$ as defined in (16) with radius $r\sqrt{\lambda_i}$ at the i^{th} dimension. However, the directions of eigenvectors are not scaled or changed by r.

In several applications such medical data classification, the covariance matrix $\mathbf{S}$ can be singular due to less amount of data than the number of dimensions. To avoid this condition, the concept of regularization [30], [31] was adapted to SHEF by adding a small positive constant ϵ to the covariance matrix $\mathbf{S}$ as shown in the following equation.

$$(\mathbf{x}-\mathbf{c})^T(\mathbf{S}+\epsilon\mathbf{I})^{-1}(\mathbf{x}-\mathbf{c}) = r^2. \tag{17}$$

$\mathbf{I}$ is a d-by-d identity matrix. In this paper, a *regularization parameter* ϵ is set to 0.0001.

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

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

$$\mathbf{S} = \mathbf{U}\Lambda\mathbf{U}^T. \tag{18}$$

Substitute (18) into $\mathbf{S}^* = \mathbf{S} + \epsilon\mathbf{I}$, we obtain the following equation.

$$S^* = \mathbf{U}\Lambda\mathbf{U}^T + \epsilon\mathbf{I}. \tag{19}$$

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

$$\begin{aligned}\mathbf{S}^* &= \mathbf{U}\Lambda\mathbf{U}^T + (\epsilon\mathbf{I})\mathbf{U}\mathbf{U}^T \\ &= \mathbf{U}\Lambda\mathbf{U}^T + \mathbf{U}(\epsilon\mathbf{I})\mathbf{U}^T \\ &= \mathbf{U}(\Lambda + \epsilon\mathbf{I})\mathbf{U}^T, \end{aligned} \tag{20}$$

$$\begin{aligned}\Lambda^* = \Lambda + \epsilon\mathbf{I} &= \begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_d\end{bmatrix} + \epsilon\begin{bmatrix}1 & & \\ & \ddots & \\ & & 1\end{bmatrix} \\ &= \begin{bmatrix}\lambda_1+\epsilon & & \\ & \ddots & \\ & & \lambda_d+\epsilon\end{bmatrix}. \end{aligned} \tag{21}$$

□

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

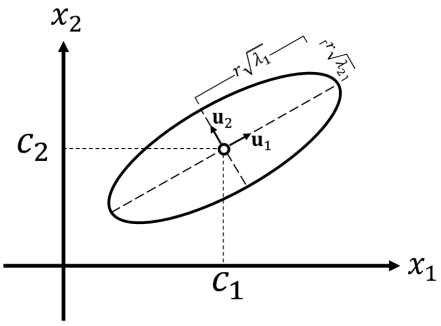


FIGURE 2. The effect of scalable constant r on the eigenvalues of hyper-ellipsoidal shape. The eigenvectors remain unchanged.

B. UPDATING PARAMETERS OF SHEF

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

Assume that incoming datum $\mathbf{x}^{new} \in \mathbb{R}^d$ is captured by the j^{th} SHEF of the same class. Let n_j^{old}, $\mathbf{c}_j^{old}$, $\mathbf{c}_j^{new}$, $\mathbf{S}_j^{old}$, and $\mathbf{S}_j^{new}$ be the current number of data vectors, the current centroid, the updated centroid, the current covariance matrix, and the updated covariance matrix, respectively. To cope with the possibility of data overflow and to preserve the time and space complexities when employing the concept of *discard-after-learn*, [22], [23], [26] suggested the following set of recursive functions for computing and updating the new centriod and new covariance matrix.

$$\mathbf{c}_j^{new} = \frac{n_j^{old}\mathbf{c}_j^{old} + \mathbf{x}^{new}}{n_j^{old}+1} \tag{22}$$

$$\begin{aligned}\mathbf{S}_j^{new} = &\frac{n_j^{old}\left(\mathbf{S}_j^{old} + \mathbf{c}_j^{old}(\mathbf{c}_j^{old})^T\right) + \mathbf{x}^{new}(\mathbf{x}^{new})^T}{n_j^{old}+1} \\ &- \mathbf{c}_j^{new}(\mathbf{c}_j^{new})^T. \end{aligned} \tag{23}$$

Although these recursive functions efficiently support the concept of *discard-after-learn*, it is possible to speed up the updating process of covariance matrix by rewriting (23) as stated in the following Theorem.

Theorem 1: A new covaraince matrix $\mathbf{S}_j^{new}$ can be computed by the following recursive function.

$$\mathbf{S}_j^{new} = \frac{n_j^{old}}{n_j^{old}+1}\left(\mathbf{S}_j^{old} + \frac{(\mathbf{c}_j^{old}-\mathbf{x}^{new})(\mathbf{c}_j^{old}-\mathbf{x}^{new})^T}{n_j^{old}+1}\right). \tag{24}$$

The proof of (24) is given in APPENDIX A. Note that the time spent on computing $\mathbf{c}_j^{old}(\mathbf{c}_j^{old})^T$, $\mathbf{x}^{new}(\mathbf{x}^{new})^T$, and $\mathbf{c}_j^{new}(\mathbf{c}_j^{new})^T$ in (23) is reduced by computing only $(\mathbf{c}_j^{old} - \mathbf{x}^{new})(\mathbf{c}_j^{old} - \mathbf{x}^{new})^T$ instead in (24). Although *Theorem 1* addresses only one incoming datum, (24) can be adapted to

an incoming data chunk by updating the covariance matrix with one datum at a time.

V. PROPOSED NEW LEARNING METHOD USING SCALABLE HYPER-ELLIPSOIDAL FUNCTION

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

1) Capturing an incoming data chunk by introducing a new SHEF or by expanding some existing SHEF of the same class k. The criteria for performing each operation depend upon: (1) the minimum distance and median distance among data within each class and (2) an adaptive threshold distance based on the number of SHEFs of the same class k and the amount of data in each SHEF of class k.
2) Merging two SHEFs of the same class k into one larger SHEF to reduce the number of SHEFs of class k. The merging criteria is based on degree of overlap between two nearest SHEFs of the same class k based on Euclidean distance.

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

A. INITIALIZING SHEF WIDTHS AND THRESHOLD DISTANCE FOR INTRODUCING NEW SHEF

At the starting step of learning process, the initial size of the first SHEF for capturing the first data chunk of a class, say class k, must be defined. If the class has only one datum, then a constant ϵ as introduced in *Lemma 1* is deployed as the initial width of SHEF in all dimensions. Otherwise the width of SHEF in each dimension is computed by the following equation. Suppose $\mathbf{x}_j^{(1)}$ is in class k and the amount of data in this class is n_k. Let $dist(\mathbf{x}_j^{(1)})$ be the Euclidean distance from $\mathbf{x}_j^{(1)}$ to its nearest neighbour of the same class in the first incoming chunk, where $j = 1, 2, \ldots, n_k$. The initial width, denoted as $dist\_init_k$, of SHEF in class k is set up as follows.

$$dist\_init_k = \begin{cases} median_{\mathbf{x}_j^{(1)} \in \text{ class } k}(dist(\mathbf{x}_j^{(1)}) & n_k > 1 \\ \sqrt{\epsilon} \text{ in Lemma 1} & n_k = 1 \end{cases} \quad (25)$$

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

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

Let M be the predefined minimum amount data allowed within each SHEF of class k. Suppose there are m_k SHEFs whose amount of data in each SHEF is less than M. The threshold distance of class k, denoted as $dist\_ths_k$, is defined as follows. For the first chunk, $dist\_ths_k$ is set to $dist\_init_k$. But for the other incoming chunks, the threshold distance is set by (26).

$$dist\_ths_k = \begin{cases} dist\_ths_k & \text{if } m_k \leq \text{half of} \\ & \text{existing SHEFs in class } k \\ 2 \times dist\_ths_k & \text{if } m_k > \text{half of} \\ & \text{existing SHEFs in class } k \end{cases} \quad (26)$$

From (26), its concept is that if there exist many inefficiently generated SHEFs (each SHEF captured data less than M), the threshold distance should be scaled up.

B. CONDITION OF INTERSECTION OF TWO SCALABLE HYPER-ELLIPSOIDS

The structure scalable hyper-ellipsoids in this paper is different from the structure studied by Alfano and Greer [29]. Their structure is based on the standard elliptic function, where the right-hand side of elliptic equation is set to zero but SHEF employs a scaling positive constant r^2 as defined in (16) instead. However, their technique of deriving the intersecting condition was adapted to our scenario.

Suppose two SHEFs of the same class, SHEF_α and SHEF_β intersect. The following theorem states the conditions of intersection of two scalable hyper-ellipsoids.

Theorem 2: Both of SHEF_α and SHEF_β do not overlap each other if all eigenvalues of the following matrix $\mathbf{P}$ are all distinct real numbers and some of them are negative. Otherwise they are in one these states: overlap, inside, or touch.

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{D} & -\mathbf{D}\mathbf{c}_\beta + \mathbf{c}_\alpha \\ \hline \mathbf{F} & -\mathbf{F}\mathbf{c}_\beta + 1 \end{array}\right], \quad (27)$$

where $\mathbf{F} = (-\mathbf{c}_\alpha^T + \mathbf{c}_\beta^T)\widetilde{\mathbf{S}}_\beta^{-1}$, $\mathbf{D} = \widetilde{\mathbf{S}}_\alpha\widetilde{\mathbf{S}}_\beta^{-1} + \mathbf{c}_\alpha\mathbf{F}$, $\widetilde{\mathbf{S}} = \mathbf{S}/r^2$, and $\widetilde{\mathbf{S}}^{-1} = \mathbf{S}^{-1}/r^2$. Centroids $\mathbf{c}_\alpha$ and $\mathbf{c}_\beta$ are of SHEF_α and SHEF_β, respectively. $\widetilde{\mathbf{S}}$ will be derived in Section VI-C.

The proof of *Theorem 2* is given in APPENDIX B. If two SHEFs of the same class satisfy the conditions in *Theorem 2*, then both of them are merged into a larger SHEF_γ and all relevant parameters are updated as follows.

$$n_\gamma = n_\alpha + n_\beta \quad (28)$$

$$\mathbf{c}_\gamma = \frac{n_\alpha \mathbf{c}_\alpha + n_\beta \mathbf{c}_\beta}{n_\gamma} \tag{29}$$

$$\mathbf{S}_\gamma = \frac{1}{n_\gamma}\left(n_\alpha \mathbf{S}_\alpha + n_\beta \mathbf{S}_\beta + \frac{n_\alpha n_\beta}{n_\gamma}(\mathbf{c}_\alpha - \mathbf{c}_\beta)(\mathbf{c}_\alpha - \mathbf{c}_\beta)^T\right). \tag{30}$$

C. PROPOSED LEARNING ALGORITHM OF SHEF

The learning process of SHEF consists of three main procedures. The first procedure (Steps 1-3) is initializing the width of the first SHEF based on the condition stated in Section V-A and (25). The second procedure (Steps 6-12) is checking the condition for introducing a new SHEF to capture new incoming data based on the threshold distance discussed in Section V-A and defined in (26). The last procedure (Steps 14-21) is merging two SHEFs of the same class according to the overlap constraints in Section V-B and *Theorem 2* by using (28), (29), and (30). The detail of learning algorithm is in **Algorithm 1**.

An example of how SHEF learning algorithm according to **Algorithm 1** works is illustrated in Fig. 3. There are two incoming chunks. The first chunk is shown in Fig. 3a. There are 11 streaming data belonging to two classes. Five circles are data in class 1 and six squares are data in class 2. For the first chunk, one datum at a time in each class is captured by a SHEF as shown in Fig. 3b - Fig. 3i. A dotted-line SHEF denotes the state of SHEF after capturing the data while a solid-line SHEF denotes the state of SHEF when its shape is expanded and rotated to capture new data. An opaque circle and an opaque square denote the data before being discarded from a SHEF. Four SHEFs are used to capture the first chunk. The second chunk is shown in Fig. 3j. There is only one datum of class 2 in this chunk. Fig. 3j - Fig. 3l illustrate how this datum is captured by SHEF_2. After capturing this datum, SHEF_2 and SHEF_3 are merged and replaced by SHEF_5.

Algorithm 1 Learning Procedure of SHEF for Current Incoming Data Chunk

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

Output: a set of SHEFs and their updated parameters.

1. **If** the first data chunk **then**
2. Initialize $dist\_ths_{y_j} = dist\_init_{y_j}$ using (25) for every class y_j in chunk $\mathbf{X}$.
3. **EndIf**
4. **For** each pair $(\mathbf{x}_i, y_i) \in \mathbf{X}$ **do**
5. **If** there exists a set of SHEFs of class y_i **then**
6. Let $\mathbf{c}_j$, n_j, and $\mathbf{S}_j$ be the centroid, amount data, and covariance matrix of captured data of SHEF_j of class y_i, respectively.
7. Let $\xi = \arg\min_j(||\mathbf{x}_i - \mathbf{c}_j||)$.
8. **If** $||\mathbf{x}_i - \mathbf{c}_\xi|| > dist\_ths_{y_i}$ **then**
9. Introduce a new SHEF of class y_i to capture $\mathbf{x}_i$ and update $dist\_ths_{y_i}$ using (26).
10. **else**
11. Put $\mathbf{x}_i$ in SHEF_ξ and update parameters by using (22) and (24). Update $n_j = n_j + 1$.
12. **EndIf**
13. Discard pair $(\mathbf{x}_i, y_i)$ from the training set.
14. Let SHEF_α be SHEF capturing $\mathbf{x}_i$.
15. **If** $n_\alpha \geq M$ **then**
16. Deploy the conditions in **Thm.2** to test overlap SHEF_α and the nearest SHEF_β of class y_i.
17. **If** SHEF_β overlaps SHEF_α **then**
18. Merge SHEF_α and SHEF_β into a larger SHEF_γ.
19. Update parameters of SHEF_γ using (28), (29), and (30).
20. **EndIf**
21. **EndIf**
22. **else**
23. Introduce a new SHEF of class y_i to capture $\mathbf{x}_i$ and update $dist\_ths_{y_i}$ using (26).
24. Discard pair $(\mathbf{x}_i, y_i)$ from the training set.
25. **EndIf**
26. **EndFor**

VI. IDENTIFYING CLASSES OF TESTING DATA

Determining the class of queried datum is also a very essential step to achieve the highest classification accuracy. Generally, the class of queried datum is decided by finding a cluster having the nearest distance measured from either the centroid (center) of the cluster or the boundary of the cluster to queried datum. Although this approach is very practical and rather efficient, the accuracy of classifying datum depends strongly upon the shape of data distribution of the cluster. A new improvement of measuring the nearest distance based on local LDA is proposed in this study. Some of the popular distance measures for class identification are briefly summarized as follows. Let $\mathbf{x}$ is a queried vector in d-dimensional space, $\mathbf{S}$ be a covariance matrix of the considered SHEF in d-dimensional space, and $\mathbf{c}$ be a centroid of the SHEF.

A. DISTANCE METHODS

1) EUCLIDEAN DISTANCE

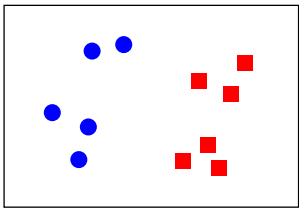
$$\text{ED}(\mathbf{x}, \mathbf{c}) := ||\mathbf{x} - \mathbf{c}|| := \sqrt{(\mathbf{x} - \mathbf{c})^T(\mathbf{x} - \mathbf{c})} \tag{31}$$

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

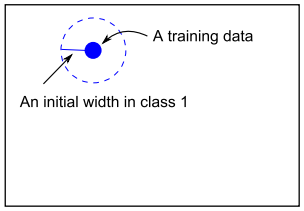
2) MAHALANOBIS DISTANCE

$$\text{MD}(\mathbf{x}, \mathbf{c}, \mathbf{S}) := \sqrt{(\mathbf{x} - \mathbf{c})^T \mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})} \tag{32}$$

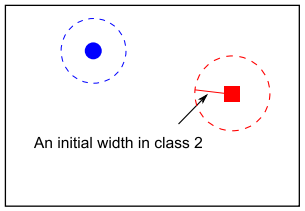
where $\mathbf{S}^{-1}$ represents the inverse of $\mathbf{S}$.



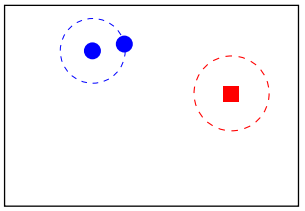
(a) Incoming data of the first chunk. The initial width of SHEF is computed from this chunk.



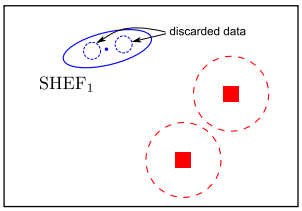
(b) Capturing a datum in class 1 denoted by a circle.



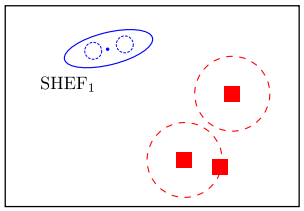
(c) Capturing a datum in class 2 denoted by a square.



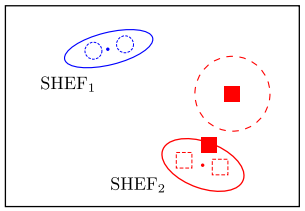
(d) Capturing another datum in class 1.



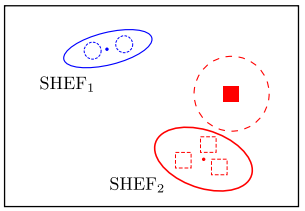
(e) Adjusting the shape of $SHEF_1$ and capturing another datum in class 2.



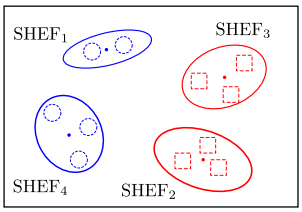
(f) Capturing a datum in class 2.



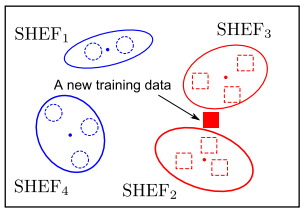
(g) Capturing a datum in class 2 by $SHEF_2$.



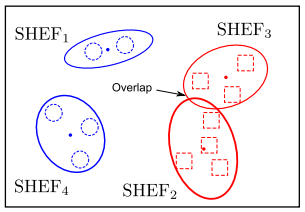
(h) Discarding the recently captured data by $SHEF_2$ and adjusting the shape of $SHEF_2$.



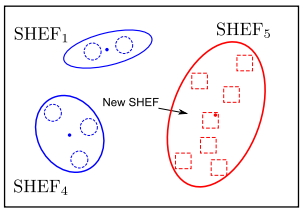
(i) Capturing the rest of data in the first chunk by $SHEF_3$ and $SHEF_4$.



(j) The second incoming chunk. There is only one datum in class 2.



(k) Capturing the datum class 2 by $SHEF_2$.



(l) Merging $SHEF_2$ and $SHEF_3$ and replacing them by $SHEF_5$.

FIGURE 3. An example of how Algorithm 1 works.

3) THE VERSATILE ELLIPTIC BASIS FUNCTION VALUE

VEBF [22] and CIL [23] use their shape-function value as a decision function to measure the closeness between a sample point $\mathbf{x}$ to SHEF. Their versatile elliptic basis function of the k^{th} neuron is defined as follows.

$$\psi_k(\mathbf{x}) = \sum_{i=1}^{d} \frac{((\mathbf{x}-\mathbf{c})^T\mathbf{u}_i)^2}{a_i^2} - 1 \tag{33}$$

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

4) APPROXIMATE BOUNDARY DISTANCE

Measuring the distance with respect to the boundary of any hyper-ellipsoids is rather complex, hence an approximate distance was proposed.

Zimmermann and Svoboda [32] proposed an approximate distance between a sample point $\mathbf{x}$ to the nearest boundary of an ellipse. It can also be applied to a high dimensional space of the ellipse, called a hyper-ellipsoid. This distance is measured on the line connecting the sample point and the centroid. The line intersects the boundary of the ellipse at a specific point. The actual distance is measured from the intersection point to the sample point. Instead of using the ellipse, they transformed the shape of the ellipse as a unit circle by matrix transformation which is much simpler as shown in Fig. 4.

The interesting concept of this strategy is described as follows. First, the original ellipse and the sample point are transformed by the inverse of the matrix $\mathbf{L}^T$ into a unit circle. This matrix is obtained by factorizing the covariance matrix representing an ellipse with a Cholesky factorization ($\mathbf{S} = \mathbf{L}\mathbf{L}^T$). The distance from a sample point to the unit circle can be easily computed by using Euclidean distance. After that, the distance value will be re-transformed to the original shape using the matrix $\mathbf{L}^T$. Approximate boundary distance according to the above concept is defined by the following equation.

$$\mathrm{BD}_1(\mathbf{x}, \mathbf{c}, \mathbf{S}) := ||(\mathbf{x}-\mathbf{c}) - \frac{(\mathbf{x}-\mathbf{c})}{||(\mathbf{L}^T)^{-1}(\mathbf{x}-\mathbf{c})||}|| \tag{34}$$

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

Wattanakitrungroj *et al.* [26] also proposed a method to compute the distance between the boundary of full micro-cluster and a data point by solving equations following the concept in Fig. 5. Although both methods [26], [32] deploy different definitions of ellipsoid, they end up with the same distance approximation. However, [26] takes less computation time and calculation steps than [32]. Approximate

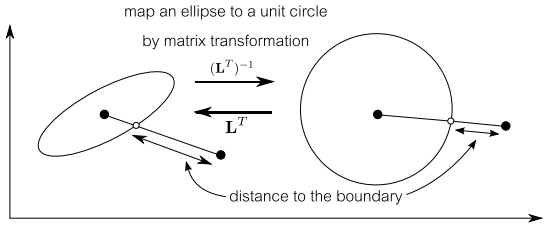


FIGURE 4. Approximate distance from a point to an ellipse by Zimmermann and Svoboda.

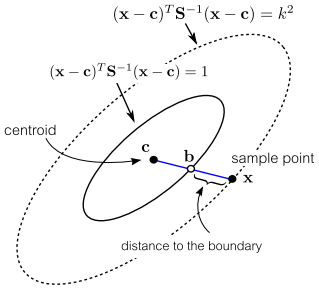


FIGURE 5. Approximate distance from a point to an ellipse by Wattanakitrungroj *et.al.*

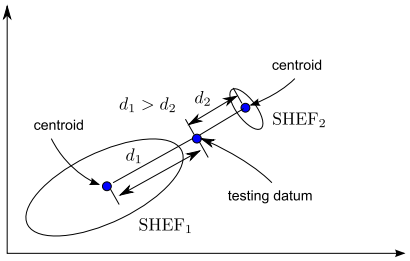


FIGURE 6. An example of wrong interpretation of closeness between testing datum and two SHEFs. The closeness is determined by measuring Euclidean distance from datum to the centroids of both SHEFs.

boundary distance according to [26] is defined by the following equation.

$$\mathrm{BD}_2(\mathbf{x},\mathbf{c},\mathbf{S}) := ||\mathbf{x}-\mathbf{c}||(1-\frac{1}{\sqrt{(\mathbf{x}-\mathbf{c})^T\mathbf{S}^{-1}(\mathbf{x}-\mathbf{c})}}). \quad (35)$$

B. LIMITATION OF EACH DISTANCE METHOD

Usually, the closeness between a point $\mathbf{x}$ and SHEFs can be easily determined in terms of Euclidean distance either from $\mathbf{x}$ to the centroid or from $\mathbf{x}$ to the boundary of SHEFs. However, the simplicity may lead to the wrong interpretation. For example, suppose there are two SHEFs, namely SHEF$_1$ and SHEF$_2$. Fig. 6 shows an example of the Euclidean distance, see (31), from a testing datum $\mathbf{x}$ to the centroids of SHEF$_1$ and SHEF$_2$. The distance from $\mathbf{x}$ to the centroid of SHEF$_1$ is longer the distance to the centroid of SHEF$_2$. Thus, $\mathbf{x}$ must be assigned to SHEF$_2$ instead of SHEF$_1$. But in fact, the correct closeness of the testing datum $\mathbf{x}$ in this example is SHEF$_1$ because it is closer to SHEF$_1$ than SHEF$_2$.

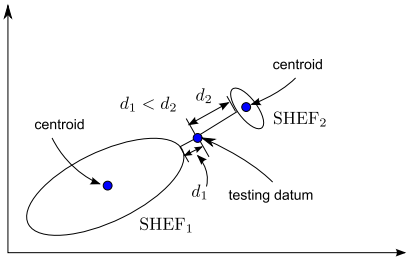


FIGURE 7. The closeness distance measured with respect to the boundary of SHEF$_1$ and SHEF$_2$. The datum is close to SHEF$_1$ instead of SHEF$_2$.

To achieve the correct identification, the closeness distance should be measured with respect to the boundaries of SHEF$_1$ and SHEF$_2$ as shown in Fig. 7. However, measuring the closest distance from the point $\mathbf{x}$ to the boundary of SHEFs in a high dimensional space is rather complex. Although there exists the approximate ways such as the methods of Zimmermann and Svoboda [32], see (34), and Wattanakitrungroj *et al* [26], see (35). They do not take the distribution of data into account.

Since Mahalanobis distance, see (32), is a distance measure between $\mathbf{x}$ and the considered SHEF at the same distribution, it needs to update a centroid and a covariance matrix with a new point. If the amount of data is extremely generated, it is expensive to calculate all updates.

Using the VEBF value to measure the closeness following (33) may lead to the wrong interpretation. Because of a size of each hyper-ellipsoid (a_i) is updated without depending on its distribution.

In this paper, a new similarity measure (distance measure) was designed based on the distribution of data inside each SHEF. The details of our new distance measure are described as below.

C. NEW DISTANCE MEASURE OF SHEF PROJECTION WIDTH

A new distance measure, namely *Projection Ratio*, was proposed. It relaxes the mentioned limitation of each distance. The Projection Ratio considers the tradeoff between the distance from the data point to the boundary of SHEF with the local data distribution of SHEF. Instead of using the direct distance between a testing datum to the boundary of SHEF, our closeness distance between any SHEFs and the testing datum is defined as the distance between the projected SHEF and the projected testing datum onto the linear-discriminant-analysis (LDA) vector. In this section, the projections of the SHEF boundary and its center onto the LDA vector (discriminant vector) are described as follows.

Let $\mathbf{w}$ be the discriminant vector. Suppose that $\mathbf{w}$ is already known. A considered SHEF is projected onto $\mathbf{w}$ as depicted in Fig. 8. Suppose that points A and B are the projected boundary points of SHEF onto $\mathbf{w}$. The centroid $\mathbf{c}$ is projected onto $\mathbf{w}$ at position $\mathbf{c}'$ using (36).

$$\mathbf{c}' = \mathbf{w}^T\mathbf{c}. \quad (36)$$

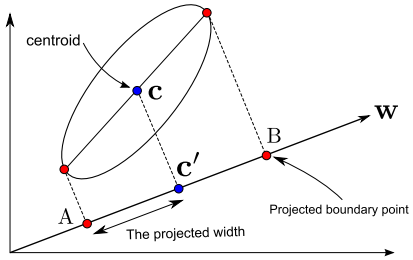


FIGURE 8. Projection of SHEF onto the discriminant vector w in a two-dimensional space.

Note that $||A - \mathbf{c}'||$ and $||B - \mathbf{c}'||$ are equal and they can be defined as the projected width of SHEF onto the discriminant vector $\mathbf{w}$. Actually, without knowing the locations of points A and B, the projected width can be computed by the square root of the projected covariance matrix of SHEF. The geometrical width of each eigenvector of SHEF directly correlates with the covariance matrix of captured data ($\mathbf{S}$) and the scaling positive constant (r). It is equal to $r\sqrt{\lambda_i}$, where λ_i is an eigenvalue of $\mathbf{S}$. This width value is actually the square root of eigenvalue along the eigenvector of the covariance matrix of SHEF. SHEF is just a shape hyper-ellipsoid without any data. All captured data of the SHEF are entirely discarded. Thus, the covariance matrix of SHEF for projection onto $\mathbf{w}$ must be derived from $\mathbf{S}$.

Let $\widetilde{\mathbf{S}}$ be the covariance matrix of SHEF with its eigenvalues $\tilde{\lambda}_i = r^2\lambda_i$ and $\mathbf{S}$ be the covariance matrix of captured data vectors with its eigenvalues λ_i. Since both $\widetilde{\mathbf{S}}$ and $\mathbf{S}$ are computed from the same data set, their eigenvectors $\mathbf{U}$ are the same. From (18) we have

$$\widetilde{\mathbf{S}} = \mathbf{U}\widetilde{\Lambda}\mathbf{U}^T = \mathbf{U}(r^2\Lambda)\mathbf{U}^T = r^2\mathbf{U}\Lambda\mathbf{U}^T = r^2\mathbf{S}. \quad (37)$$

The projected width of $||A - \mathbf{c}'||$ and $||B - \mathbf{c}'||$ can be computed by the following equation.

$$||A - \mathbf{c}'|| = ||B - \mathbf{c}'|| = \sqrt{\mathbf{w}^T\widetilde{\mathbf{S}}\mathbf{w}} = r\sqrt{\mathbf{w}^T\mathbf{S}\mathbf{w}} \quad (38)$$

where $\mathbf{w}^T\mathbf{S}\mathbf{w}$ is a non-negative scalar. $||\cdot||$ represents Euclidean norm. Note that, our projection method is similar to Pope's idea [33] using the different definition of hyper-ellipsoid function. Deriving the discriminant vector $\mathbf{w}$ in our case is different from that of LDA. LDA computes a discriminant vector from two classes of data but in our case there are only one datum and a SHEF which is representation of data. Therefore, the method of LDA cannot be directly applied to compute the discriminant vector in our case.

Let $\mathbf{S}$ and $\mathbf{c}$ be the covariance matrix and the centroid of captured data of SHEF, respectively. Suppose $\mathbf{x}$ is a single queried datum. In order to find the discriminant vector for both SHEF and $\mathbf{x}$, datum $\mathbf{x}$ is reconsidered as the centroid of another SHEF. By Using (8) with this circumstance, the discriminant vector can be derived as follows.

$$\mathbf{w}_p = \frac{\mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})}{||\mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})||}. \quad (39)$$

Fig. 9 shows an example of discriminant vector $\mathbf{w}_p$ and the projected width of SHEF as well as the projected location of

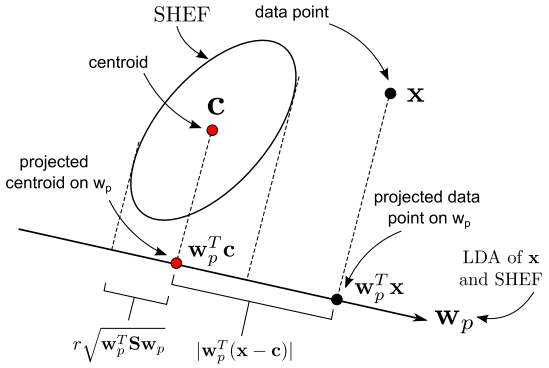


FIGURE 9. The concept of projection ratio distance for x and SHEF computed from $|\mathbf{w}_p^T(\mathbf{x} - \mathbf{c})|$ and $r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$ along the discriminant vector $\mathbf{w}_p$ defined in (39).

data point $\mathbf{x}$ onto $\mathbf{w}_p$. There are two significant distances to be used. The first distance $|\mathbf{w}_p^T(\mathbf{x} - \mathbf{c})|$ is the projected distance from the projected centroid to the projected location of $\mathbf{x}$. The second distance $r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}$ is the projected width of SHEF.

The Projection Ratio distance from data point $\mathbf{x}$ to SHEF is defined as follows.

$$D(\mathbf{x}, \mathbf{c}, \mathbf{S}) = \frac{|\mathbf{w}_p^T(\mathbf{x} - \mathbf{c})|}{r\sqrt{\mathbf{w}_p^T\mathbf{S}\mathbf{w}_p}}. \quad (40)$$

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

Theorem 3: Let $\mathbf{S}$ and $\mathbf{c}$ be a covariance matrix and a centroid of captured data of SHEF in a d-dimensional space. Suppose $\mathbf{x}$ is a single queried data point and a discriminant vector $\mathbf{w}_p$ is defined in (39).

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

The proof of *Theorem 3* is given in APPENDIX C.

D. DETERMINING CLASS OF QUERIED DATUM BASED ON PROJECTION RATIO DISTANCE

Datum $\mathbf{x}$ may encounter two possible scenarios when its Projection Ratio is deployed to determine its appropriate class. Suppose two SHEFs are close to $\mathbf{x}$. The projection ratios of both SHEF_1 and SHEF_2 are $D(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1)$ and $D(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$, respectively.

1) If both SHEFs represent the same class, then $\mathbf{x}$ is assigned to the class of either SHEF_1 or SHEF_2.
2) If both SHEFs represent the different classes, then the appropriate class of $\mathbf{x}$ is determined as follows.
   a) $D(\mathbf{x}, \mathbf{c}_1, \mathbf{S}_1) \leq 1 < D(\mathbf{x}, \mathbf{c}_2, \mathbf{S}_2)$. This implies that $\mathbf{x}$ is inside or on the boundary only SHEF_1. Thus, $\mathbf{x}$ should be assigned to the class of SHEF_1.
   b) Otherwise, *Projection Ratio* is deployed to both SHEFs onto a new discriminant vector $\mathbf{w}$ defined in (8) instead, as shown in Fig. 10. Denoted by

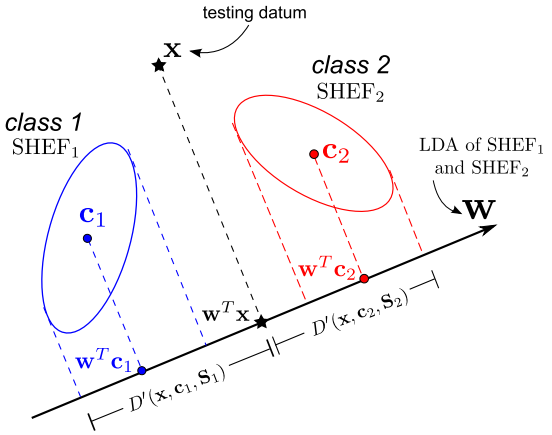


FIGURE 10. An example of projections of two SHEFs and x onto a discriminant vector w defined in (8). There are two *Projection Ratios*, one for each SHEF.

$D'(\mathbf{x}, \mathbf{c}, \mathbf{S}) = \dfrac{|\mathbf{w}^T(\mathbf{x}-\mathbf{c})|}{r\sqrt{\mathbf{w}^T\mathbf{S}\mathbf{w}}}$, the appropriate class of $\mathbf{x}$ is determined as follows.

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

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

VII. EXPERIMENTAL DATA AND SET-UP

All experiments were tested with two groups of data. The first group has four 2-dimensional synthetic data sets (see the illustration from the first row of Fig. 1): Sparse-dense data, Faraway4 data, Spiral data, and Gaussian distributed data. The second group contains seven real-world data sets from the University of California at Irvine (UCI) Repository of the machine learning database [34] which were selected by varying the number of features, the number of data, and the number of classes. For Letter, Shuttle_trn, and Kddcup99 data set, they were selected to imitate streaming data. For Kddcup99, four symbolic features which are more than two categories were removed. So there are 38 features in this data set. The description of all data sets are given in Table 1.

A 5-fold cross validation was used in all experiments for each method. Training data in each data set were divided into several chunks of randomly different sizes in order to simulate the environment of streaming data. These chunks were sequentially fed into the training process. The amount of data in each chunk was uniformly and randomly distributed and pre-defined by a variable called *chunk-size range*. Table 2 summarizes the chunk-size range of each data set and the number of training chunks in each fold. In order to study the effect on the order of the incoming data in each data set, 10 groups of different shuffled order of training chunks in each data set were created for each fold. Total experiments per one data set were 50 experiments.

Since the experiments are in a streaming environment in forms of sequential data chunks, the testing process in [6], [8], [35] was adopted. After learning one incoming chunk, the classification accuracy is evaluated by a testing set and a new chunk is learned next. The accuracy of each testing chunk is called *Chunk-wise Accuracy* (CA).

For streaming data evaluation, assume that training data was divided into T chunks. Let CA_t, for $1 \le t \le T$, be the evaluation accuracy of the $(t+1)^{th}$ chunk as a test set after training the t^{th} chunk as a train set. The following *Average Cumulative Accuracy* of chunk t (ACA_t) is measured after training the t^{th} chunk.

$$\text{ACA}_t = \frac{1}{t}\sum_{i=1}^{t}\text{CA}_i, \quad t = 1, \cdots, T-1. \tag{41}$$

The proposed method is also efficiently capable of learning non-streaming data, where there is only one train set and one test set. A 5-fold cross validation is used to evaluate the method. To distinguish the accuracy of non-streaming data classification from chunk-wise accuracy, this accuracy of non-streaming data is called *Population Accuracy* (PA).

The experimental results were compared with the results produced by the methods designed for learning streaming data with the concept of one-pass learning which are VEBF [22], LOL [21], and CIL [23] and with the concept of retained learning which are ILDA [17] and VLLDA [24]. However, these compared methods have similar and different characteristics as summarized in Table 3. The meaning of each characteristic is the following.

TABLE 1. Experimental data sets and their attributes.

Data set	# of features	# of data	# of classes
% Synthetic data sets			
Sparse-dense data	2	496	3
Faraway4 data	2	612	4
Spiral data	2	2000	2
Gaussian data	2	6000	4
% Real-world data sets			
Segment	19	2310	7
Spambase	57	4601	2
Waveform	21	5000	3
Satimage	36	6435	6
Letter	16	20000	26
Shuttle_ trn	9	43500	7
Kddcup99	38	494020	23

TABLE 2. Setting the number of training data in each fold for each streaming chunk.

Data set	Chunk-size range	# of training chunks in each fold				
		fold1	fold2	fold3	fold4	fold5
Sparse-dense	[20,50]	11	12	11	12	10
Faraway4	[20,50]	13	13	15	13	14
Spiral	[100,200]	11	11	10	10	10
Gaussian	[100,200]	31	30	33	32	31
Segment	[100,200]	12	13	12	12	13
Spambase	[100,200]	24	25	25	24	24
Waveform	[100,200]	26	26	28	27	25
Satimage	[100,200]	35	35	35	34	35
Letter	[100,200]	109	108	107	104	105
Shuttle_trn	[100,200]	230	231	232	236	232
Kddcup99	[1000,2000]	263	272	260	258	268

TABLE 3. Characteristics of the compared methods (✓Yes, - No).

Methods	Incremental		One-pass	Stream	FRM	Local	Tuned parameters (determined by user)
	Sequential	Chunk					
VEBF [22]	✓	-	✓	-	-	✓	δ
ILDA [17]	✓	-	-	✓	✓	-	No tuning
LOL [21]	✓	-	$\star$	✓	-	✓	k, λ, and C
CIL [23]	-	✓	✓	$\star\star$	-	✓	δ
VLLDA [24]	-	-	-	-	✓	✓	k
Proposed method (SHEF)	✓	-	✓	✓	✓	✓	No tuning

$\star$ LOL is the true one-pass learning for only dichotomous classification problem. When LOL is applied to classify multi-class data, it uses one-vs-all approach to classify one class at a time.
$\star\star$ The initial width of a hyper-ellipsoid in streaming version of CIL for any class is computed from the first 20% of total training data.

1) *Sequential* means learning one incoming datum at a time.
2) *Chunk* means learning one incoming chunk which may have several classes of data at a time.
3) *One-pass* means discarding all current training data after being learned, which may be a datum or a chunk of data.
4) *Stream* means processing without knowing any prior statistical information of data in advance and not using that data in advance to set any initial parameters.
5) *FRM* means using the feature reduction method for the classification problem.
6) *Local* means using only the information from the incoming data distribution to solve nonlinear separable problem.

The experiemntal results of proposed method were compared to those produced by VLLDA [24], ILDA [17], LOL [21], VEBF [22], and CIL [23]. All methods have different parameter settings. In this experiment, the parameter setting of each method is summarized in Table 4.

For VEBF and CIL, the constant δ was set to scale the initial width of VEBF shape function calculated from the average distance. Parameter settings for the data sets of Segment, Spambase, Waveform, and Letter were referred from [23]. VEBF used whole training data to calculate their initial average distance whereas CIL used the first 20% of total training data. Unfortunately, the number of training data in Shuttle_trn and Kddcup99 is too huge to be calculated in a short time. Hence, only first 10,000 training data of Shuttle_trn and Kddcup99 were calculated for VEBF and only first 10,000 training data of Kddcup99 were calculated for CIL. For LOL, the authors claimed that LOL is not very sensitive to the parameters and suggested the settings of three parameters, namely the number of prototypes k was set to 60; the balancing parameter λ was set to 1.0; and the aggressive parameter C was set to 1.0 for all experiments. For VLLDA, the parameter k must be set to be congruent with the k-nearest neighbour method. For ILDA and VLLDA which use LDA for classification, the number of selected discriminant vectors was set according to the number of non-zero eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$.

TABLE 4. Parameter settings of VLLDA, VEBF, and CIL.

Data set	VLLDA (k)	VEBF (δ)	CIL (δ)
Sparse-dense data	5	0.2	0.2
Faraway4	5	0.2	0.2
Spiral data	5	0.2	0.1
Gaussian data	10	0.4	0.5
Segment	10	1	0.7
Spambase	10	1	0.4
Waveform	5	1	0.3
Satimage	10	0.9	0.7
Letter	5	0.7	0.7
Shuttle_trn	10	20	1
Kddcup99	10	2	2

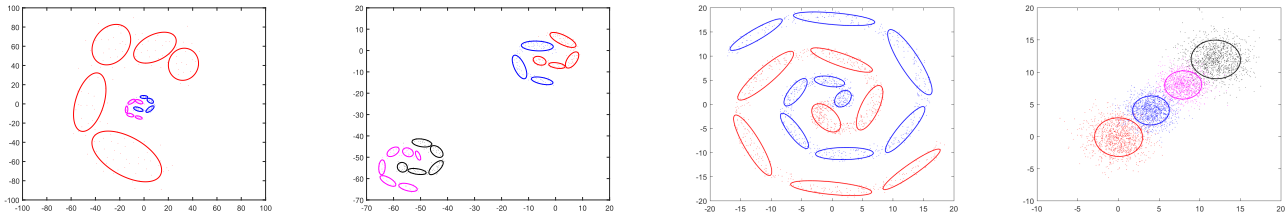
For our proposed method, the following hyperparameters were set: constant r was set to 1.5; regularization parameter ϵ in *Lemma 1* was set to 0.0001; minimum number of data M in SHEF was set to 3 for all experiments without tuning.

VIII. EXPERIMENTAL RESULTS AND PERFORMANCE COMPARISON

All experiments were conducted on a desktop PC with 8 GB RAM, Intel Core i7-4770, 3.4 GHz with licensed Matlab code. The dimensions of data sets vary from two dimensions to some higher dimensions. For two dimensions, Fig. 11 shows the results obtained from proposed SHEF. Note that these VEBF, CIL, SHEF approaches are based on the same concept of *discard-after-learn*. For other methods using different concepts and higher dimensional data sets, the results are compared and shown in Table 5 instead. The independent t-test was adopted to measure the statistically significant difference between the average value of proposed method and other methods. The value with asterisk (*) means that there is *no* statistically significant difference at the 5% significance level ($p > 0.05$). The experimental results concerned the following issues.

A. POPULATION ACCURACY (PA) AND AVERAGE CUMULATIVE ACCURACY (ACA)

The average population accuracy and the standard deviation of all methods in each data set are reported in Table 5. The compared methods used different approaches to learning data. The first approach is retaining all training data



(a) Sparse-dense data (b) Faraway4 data (c) Spiral data (d) Gaussian data

FIGURE 11. Results of four 2-dimensional synthetic data sets generated by proposed SHEF. (see online version for color figure).

TABLE 5. Comparison of average population accuracy (%) and standard deviation ($\bar{x}\pm sd$) in the five-fold validation with 10 replications.

Data set (# features, # classes)	Approach 1		Approach 2			
	VLLDA	ILDA	LOL	VEBF	CIL	SHEF
Sparse-dense data (2, 3)	99.798* ±0.408	99.798* ±0.408	91.930 ±9.280	86.592 ±4.024	89.291 ±2.863	**99.738** ±0.447
Faraway4 (2, 4)	99.672* ±0.662	99.672* ±0.662	93.447 ±8.028	81.131 ±6.079	80.461 ±5.971	**99.575** ±0.744
Spiral data (2, 2)	99.600 ±0.258	67.850 ±1.602	50.150 ±2.163	97.820 ±1.512	97.625 ±3.244	**98.750** ±1.680
Gaussian data (2, 4)	90.050 ±1.055	89.633 ±0.793	88.220 ±2.788	90.733 ±3.881	85.122 ±3.537	**93.345** ±0.401
Segment (19, 7)	95.065 ±1.511	95.714 ±1.479	79.320 ±2.938	82.498 ±2.935	88.563 ±2.305	**93.848** ±0.846
Spambase (57, 2)	85.773 ±0.588	87.433 ±0.324	60.533 ±1.391	71.902 ±8.121	90.239 ±1.847	**90.845** ±0.775
Waveform (21, 3)	81.920 ±0.679	82.019 ±1.217	80.926 ±2.142	84.116 ±4.201	81.388 ±1.139	**86.480** ±0.907
Satimage (36, 6)	88.163 ±1.037	86.869 ±0.585	80.634 ±2.950	83.081 ±4.454	58.176 ±2.809	**84.881** ±0.903
Letter (16, 26)	95.552 ±0.279	95.723 ±0.328	61.205 ±2.332	65.985 ±5.730	26.432 ±16.741	**84.841** ±0.370
Shuttle_trn (9, 7)	99.862 ±0.055	99.883 ±0.029	**97.753** ±0.677	38.343 ±30.677	97.280 ±0.324	96.331 ±1.425
Kddcup99 (38, 23)	N/A	N/A	99.011 ±0.2703	78.919 ±0.38	19.93 ±0.189	**99.348** ±0.055

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

throughout the training and testing processes as implemented in VLLDA and ILDA. But the second approach uses the concept of discard-after-learn. Each datum is learned in one pass and discarded afterwards. No need to retain any incoming datum throughout the training and testing processes as implemented in LOL, VEBF, CIL, and our proposed method. To obtain a clear comparison based on these two approaches, the testing was conducted into two categories according to each approach.

Without tuning any hyper-parameters, our method **SHEF** achieved better accuracy with statistically significance in 10 out of 11 data sets compared to the methods using the second approach and in 4 out of 9 data set when being compared to the methods using the first approach. The better accuracy is shown in bold numbers and underlined numbers. Notice that there is a statistically significant difference between the average population accuracy of SHEF and other methods in the second approach (LOL, VEBF, and CIL) on every data set. LOL has the lowest accuracy on Spiral data, Segment, Spambase, and Waveform. VEBF has the lowest accuracy with large standard deviation on Shuttle_trn. CIL has the lowest accuracy on Satimage, Letter, and Kddcup99. For Kddcup99 data set, VLLDA and ILDA could not finish the learning process within one hour. For the first approach, VLLDA and ILDA have quite good accuracy on all data sets, except ILDA method on Spiral data.

Fig. 12 shows the chunk-wise accuracy for six methods after training in each chunk on seven real-wolrd data sets. One of 50 experiments was picked up to show the accuracy at each chunk of streaming data. Since VLLDA and ILDA used and retained all incoming training chunks (the first chunk to the current chunk of streaming data) to classify the testing data, thus they were recomputed with cumulative training data from all previous training chunks. It is remarkable that LOL has a wide swinging range of accuracy on Waveform, Satimage, and Kddcup99. VEBF has a wide swinging range of the accuracy on Spambase and Shuttle_trn. There is a sudden change of the accuracy value in VEBF on Shuttle_trn data set to be discussed in the next section. CIL has a wide swinging range of accuracy on Waveform, and Letter. For Kddcup99, CIL has the low accuracy in every chunk of data. The average cumulative accuracy of the results in Fig. 12 calculated by using (41) is shown in Fig. 13. This type of accuracy indicated the trend of accuracy as the result of incremental learning the incoming chunk after chunk. For the proposed method SHEF, the average cumulative accuracy trended to increase in all data sets when SHEFs were incrementally trained.

B. AVERAGE NUMBER OF GENERATED NEURONS

Table 6 shows the average number of generated neurons (or prototypes). The compared methods can be grouped by information used to generate the neurons during the learning period. The first group consists of VLLDA and ILDA. Both VLLDA and ILDA generated neurons based on the number of training data in each data set. The second group consists of LOL, VEBF, CIL, and SHEF. LOL generated neurons based on the number of prototypes (set to 60). VEBF, CIL, and SHEF generated neurons based on hyper-ellipsoidal shape function to capture the incoming data. At the end of training

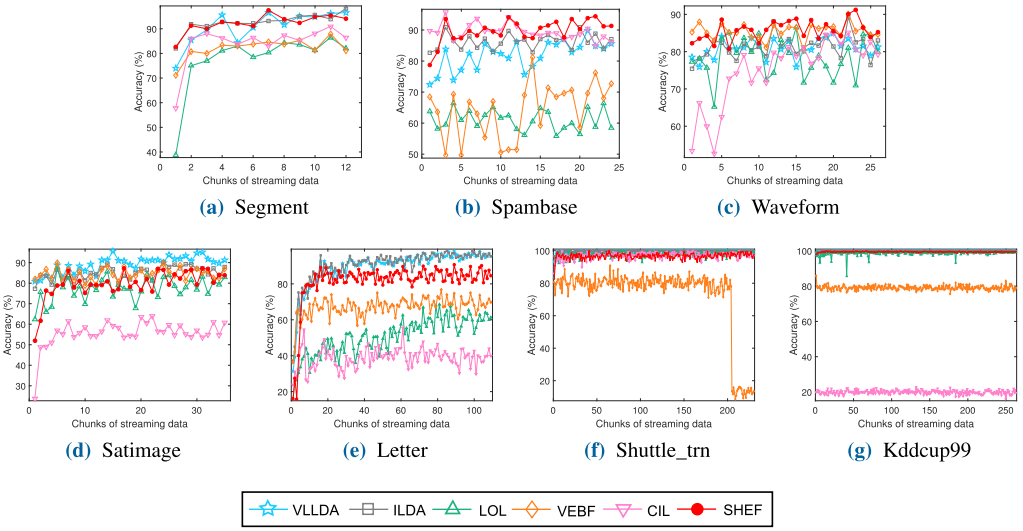


FIGURE 12. Chunk-wise accuracy (%) for six methods after training in each chunk on seven real-world data sets.

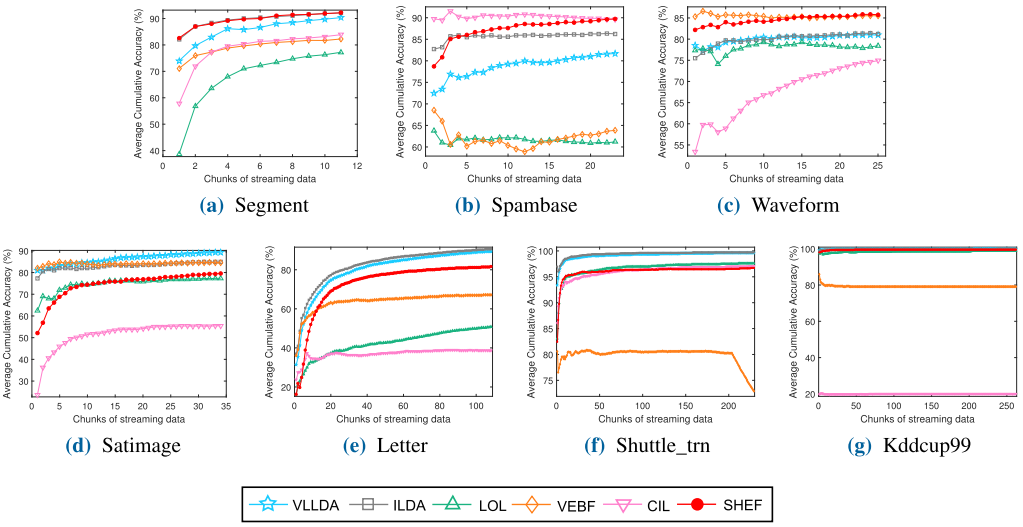


FIGURE 13. Average cumulative chunk-wise accuracy (%) for six methods in each chunk on seven real-world data sets.

all incoming chunks, the average number of generated SHEFs is less than or equal to those of the others in 10 out of 11 data sets. There are statistically significant difference between the average number of neurons in our proposed method (SHEF) and the other methods in every data set, except for VEBF on Waveform. There are six data sets, i.e. Gaussian data, Segment, Spambase, Waveform, Satimage, and Letter, where our method generated the average number of neurons close to the number of classes. However, there is one data set, Kddcup99 data set, where the number of generated neurons is less than the number of classes (23 classes). The detail will be discussed in the next section.

Fig. 14 shows the number of generated neurons for three methods (VEBF, CIL, and proposed SHEF) during training in each chunk on seven real-world data sets. VEBF and SHEF merge two overlapped hyper-ellipsoids. Therefore, the number of neurons during the training process might be increased or decreased according to their approaches. Since CIL does not have any merging strategy, the number of neurons is always increased.

C. AVERAGE COMPUTATIONAL TIME

Table 7 reports the computational time of training and testing processes. Training time is a total spending time after training all of streaming data chunks on each data set. Testing time is a spending time of the test set from the 5-fold cross validation. There is a statistically significant difference between the average of computational time (both training and testing) of the proposed method (SHEF) and the other five methods, except for the testing time of LOL on Sparse-dense data. VLLDA is processed without training so only the testing time is reported. The training time of VEBF, CIL, and SHEF included the computational time of calculating

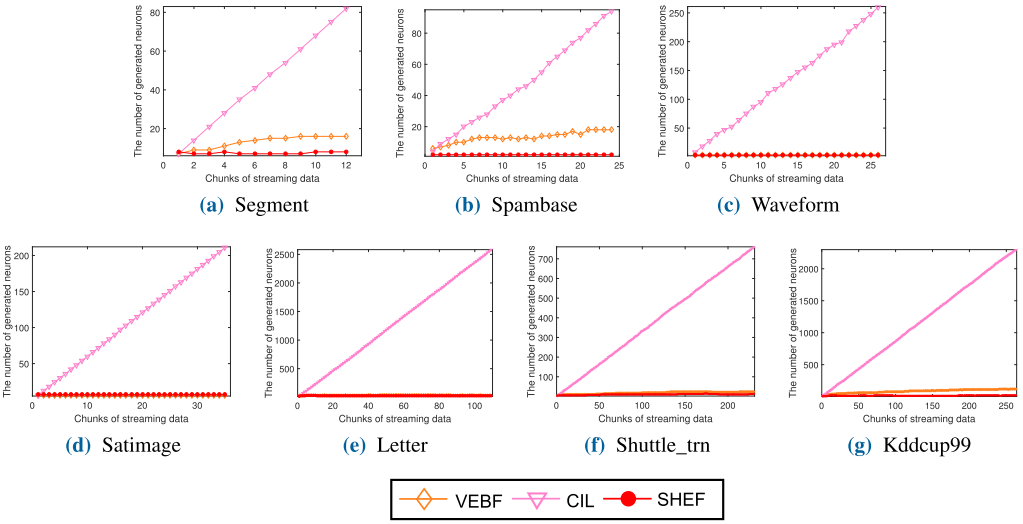


FIGURE 14. Number of generated neurons for three methods after training in each chunk on seven real-world data sets.

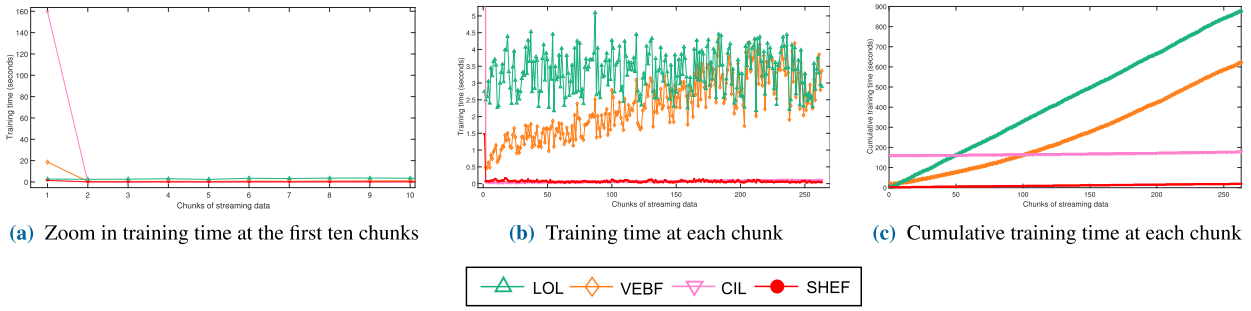


FIGURE 15. Training time of Kddcup99 during training each incoming chunk. (see online version for color figure).

the initial distance of hyper-ellipsoids. When the number of data is not huge, the computational time of all methods may not be different. However, the last three data sets (Letter, Shuttle_trn, and Kddcup99) are quite big. SHEF is obviously faster than others methods for both training and testing data. For example, SHEF spent about 14 seconds for 395,216 training data and about 94 seconds for 98,804 testing data in Kddcup99 data set whereas the other methods spent a much longer time for training and testing, especially for VLLDA and ILDA. They could not finish the process within one hour. Actually, VLLDA and ILDA spent about four hours and three days for Kddcup99 data set, respectively.

Fig. 15 shows the computational time of four methods (LOL, VEBF, CIL, and SHEF) for learning each incoming chunk on Kddcup99 data set (263 chunks). Fig. 15a shows the training time of the first ten chunks. For VEBF, CIL, and SHEF, the computational time to initialize the width of a hyper-ellipsoidal function was included in training time of the first chunk. Fig. 15b shows the training time of each chunk and Fig. 15c shows the cumulative training time from Fig. 15b of each chunk. Cumulative training time is the summation of the training time of each consecutive incoming chunk, starting from the first chunk to the current incoming chunk.

D. EFFECTS OF SHEF HYPER-PARAMETERS ON CLASSIFICATION ACCURACY AND NUMBER OF SHEFs

The values of two hyper-parameters r and ϵ in (17) were specially varied in this section to analyze how their values effect the classification accuracy and the number of generated SHEF neurons. Hyper-parameter ϵ was set to 10^{-6}, 10^{-4}, 10^{-2}, 10^{0}, and 10^{2} and positive hyper-parameter r was set to 0.5 to 2.5 with a step size of 0.5. Fig. 16 shows the results in the form of Grid Search diagram by varying ϵ and r of SHEF on 11 data sets. The x-axis denotes the range of ϵ and y-axis denotes the range of r. The average population accuracy of 50 experiments as the results of various values of ϵ and r is depicted by color levels varied from Blue to Red (50% to 100% of accuracy). The number in each box means the average number of generated neurons as the results of different values of ϵ and r. But the results from all experiments in the Sections previously discussed fixed the values of $\epsilon = 10^{-4}$ and $r = 1.5$. These results of fixed ϵ and

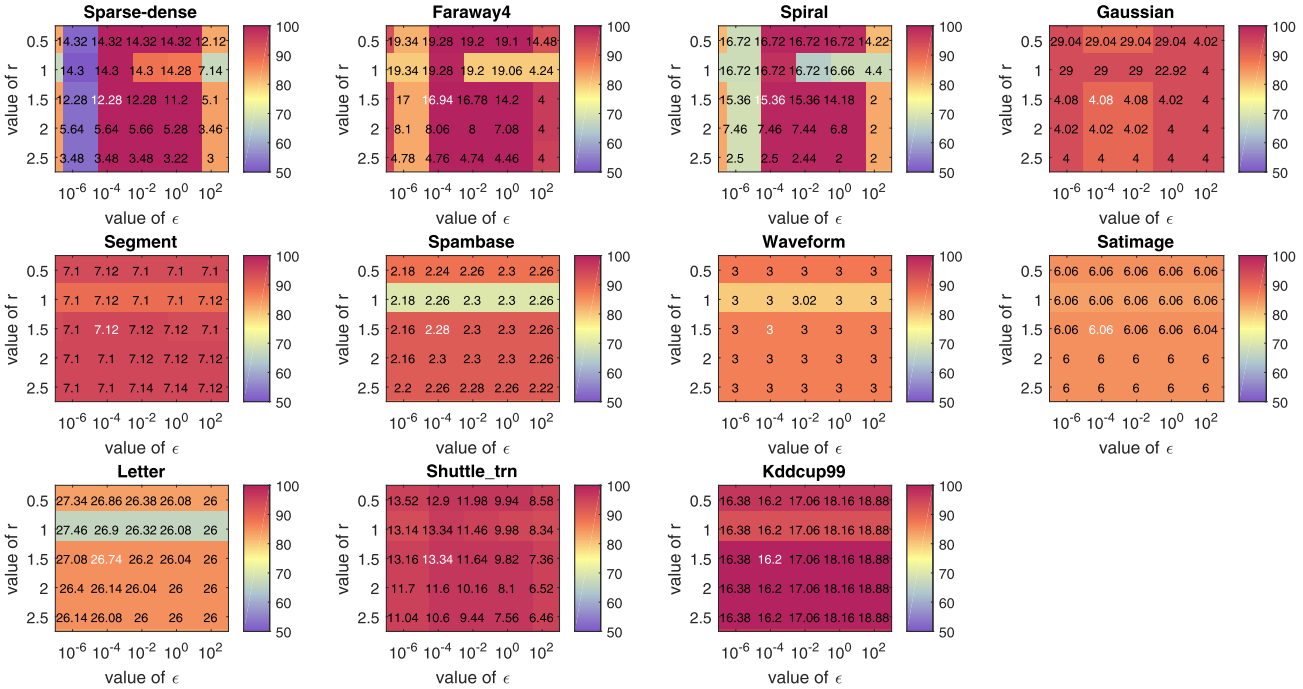


FIGURE 16. Average population accuracy and number of generated SHEF neurons in the form of Grid Search resulted by different values of ϵ vs r of SHEF on 11 data sets.

TABLE 6. Comparison of average and standard deviation ($\bar{x}\pm sd$) of the number of generated neurons at the end of training process.

Data set	Approach 1		Approach 2			
	VLLDA	ILDA	LOL	VEBF	CIL	SHEF
Sparse-dense	397	397	60	29.960 ±2.878	17.980 ±3.431	**12.280** ±1.485
Faraway4	490	490	60	**4.380** ±0.567	11.680 ±8.077	16.940 ±2.411
Spiral	1600	1600	60	16.640 ±1.120	186.640 ±11.374	**15.360** ±1.575
Gaussian	4800	4800	60	4.260 ±0.600	43.720 ±3.881	**4.08** ±0.444
Segment	1848	1848	60	14.320 ±0.978	86.040 ±3.812	**7.120** ±0.328
Spambase	3681	3681	60	20.420 ±1.500	108.140 ±6.958	**2.280** ±0.536
Waveform	4000	4000	60	**3.000*** ±0.000	80.300 ±2.845	**3.000** ±0.000
Satimage	5148	5148	60	6.500 ±0.909	211.780 ±3.247	**6.060** ±0.240
Letter	16000	16000	60	41.460 ±5.997	2532.380 ±30.438	**26.74** ±1.140
Shuttle_trn	34800	34800	60	24.680 ±1.253	751.460 ±9.232	**13.340** ±2.134
Kddcup99	395216	395216	60	117.34 ±18.92	2328.6 ±119.14	**16.2** ±1.917

1. **Bold** number indicates the minimum average number of neurons.
2. The value with asterisk ($\star$) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$).

r were depicted by the numbers in white color in the Grid Search diagram.

The results of each data set from Grid Search in Fig. 16 can be interpreted as follows. For example, the result of Sparse-dense set $\epsilon = 10^2$ and $r = 1.5$ was shown in the fifth column and the third row. The average population accuracy was represented by Blue color meaning the accuracy is about 55 %. The average number of generated neurons was 5.1. The same row means fixing r and varying ϵ. The same column means fixing ϵ and varying r.

IX. DISCUSSION

Definitely, VLLDA and ILDA are not suitable for a streaming scenario because they need to store the entire incoming data for classification. Besides, when data are big such as Shuttle_trn and Kddcup99, both VLLDA and ILDA methods spent too much training time because the time obviously depends upon the size of training data. However, VLLDA and ILDA still hold the advantages in classification accuracy because they employ the concept of k-NN and LDA. Both VLLDA and ILDA may be chosen as a traditional learning method in a streaming environment. Therefore, comparing the performances of VLLDA and ILDA to the performances of LOL, VEBF, CIL, and SHEF approaches is not appropriate due to the different concepts of handle the constraint of space complexity.

ILDA which is an incremental version of LDA also has the same limitation as LDA. When data are non-linearly separable, it will not work properly as seen in the results of Spiral data (accuracy 67.85%) in Table 5. The time complexity of updating equations for both $\mathbf{S}_B$ and $\mathbf{S}_W$ affects the training time when the training data are big. Hence, the more data are trained, the more time is obviously spent. VLLDA based on LDA with k-NN can improve LDA to handle the complex distribution of data as seen in the results of Spiral data in Table 5. But choosing k may affect the accuracy. Furthermore, VLLDA needs to compute Euclidean distances among all

TABLE 7. Average and standard deviation ($\bar{x}\pm sd$) of computational time (seconds) of each method.

Data set (# train : # test)	Process	Approach 1		Approach 2			
		VLLDA	ILDA	LOL	VEBF	CIL	**SHEF**
Sparse-dense data (397 : 99)	train	-	$0.149_{\pm 0.005}$	$0.057_{\pm 0.003}$	$0.124_{\pm 0.011}$	$\mathbf{0.034}_{\pm 0.016}$	$0.099_{\pm 0.028}$
	test	$0.121_{\pm 0.018}$	$0.121_{\pm 0.015}$	$0.014^{*}_{\pm 0.001}$	$0.039_{\pm 0.004}$	$\mathbf{0.005}_{\pm 0.002}$	$0.014_{\pm 0.004}$
Faraway4 (490 : 122)	train	-	$0.221_{\pm 0.006}$	$0.151_{\pm 0.013}$	$0.354_{\pm 0.030}$	$\mathbf{0.034}_{\pm 0.004}$	$0.095_{\pm 0.008}$
	test	$0.152_{\pm 0.004}$	$0.146_{\pm 0.003}$	$0.032_{\pm 0.003}$	$0.008_{\pm 0.001}$	$\mathbf{0.004}_{\pm 0.002}$	$0.020_{\pm 0.002}$
Spiral data (1600 : 400)	train	-	$2.263_{\pm 0.007}$	$\mathbf{0.074}_{\pm 0.001}$	$0.584_{\pm 0.022}$	$0.250_{\pm 0.019}$	$0.399_{\pm 0.039}$
	test	$0.770_{\pm 0.037}$	$0.614_{\pm 0.004}$	$\mathbf{0.016}_{\pm 0.001}$	$0.089_{\pm 0.006}$	$0.132_{\pm 0.009}$	$0.064_{\pm 0.005}$
Gaussian data (4800 : 1200)	train	-	$20.543_{\pm 0.072}$	$0.909_{\pm 0.007}$	$0.754_{\pm 0.019}$	$1.457_{\pm 0.016}$	$\mathbf{0.319}_{\pm 0.062}$
	test	$5.744_{\pm 0.189}$	$4.761_{\pm 0.035}$	$0.210_{\pm 0.001}$	$\mathbf{0.070}_{\pm 0.007}$	$0.105_{\pm 0.009}$	$0.086_{\pm 0.014}$
Segment (1848 : 462)	train	-	$4.238_{\pm 0.010}$	$0.714_{\pm 0.005}$	$0.399_{\pm 0.026}$	$0.258_{\pm 0.002}$	$\mathbf{0.220}_{\pm 0.027}$
	test	$0.766_{\pm 0.074}$	$0.522_{\pm 0.010}$	$0.179_{\pm 0.003}$	$0.334_{\pm 0.029}$	$0.093_{\pm 0.004}$	$\mathbf{0.084}_{\pm 0.005}$
Spambase (3681 : 920)	train	-	$78.007_{\pm 0.979}$	$\mathbf{0.387}_{\pm 0.042}$	$6.017_{\pm 0.237}$	$1.407_{\pm 0.048}$	$1.618_{\pm 0.537}$
	test	$4.440_{\pm 0.106}$	$2.740_{\pm 0.131}$	$\mathbf{0.056}_{\pm 0.001}$	$3.440_{\pm 0.264}$	$0.334_{\pm 0.022}$	$0.287_{\pm 0.058}$
Waveform (4000 : 1000)	train	-	$20.131_{\pm 0.069}$	$0.679_{\pm 0.003}$	$0.862_{\pm 0.060}$	$1.036_{\pm 0.020}$	$\mathbf{0.274}_{\pm 0.019}$
	test	$2.083_{\pm 0.027}$	$3.751_{\pm 0.027}$	$0.171_{\pm 0.003}$	$0.235_{\pm 0.014}$	$0.184_{\pm 0.008}$	$\mathbf{0.109}_{\pm 0.006}$
Satimage (5148 : 1287)	train	-	$48.688_{\pm 0.771}$	$1.982_{\pm 0.009}$	$1.887_{\pm 0.116}$	$1.999_{\pm 0.096}$	$\mathbf{0.802}_{\pm 0.249}$
	test	$4.782_{\pm 0.140}$	$6.411_{\pm 0.259}$	$0.471_{\pm 0.009}$	$1.459_{\pm 0.147}$	$0.718_{\pm 0.011}$	$\mathbf{0.383}_{\pm 0.019}$
Letter (16000 : 4000)	train	-	$304.808_{\pm 0.954}$	$22.495_{\pm 0.356}$	$12.323_{\pm 9.146}$	$18.609_{\pm 0.576}$	$\mathbf{0.706}_{\pm 0.156}$
	test	$14.724_{\pm 0.855}$	$8.673_{\pm 0.074}$	$5.014_{\pm 0.091}$	$7.691_{\pm 0.922}$	$24.378_{\pm 0.916}$	$\mathbf{2.210}_{\pm 0.138}$
Shuttle_trn (34800 : 8700)	train	-	$1207.920_{\pm 11.657}$	$12.469_{\pm 0.054}$	$12.419_{\pm 1.137}$	$76.331_{\pm 1.402}$	$\mathbf{5.836}_{\pm 0.996}$
	test	$458.069_{\pm 2.394}$	$359.521_{\pm 13.420}$	$2.851_{\pm 0.031}$	$4.337_{\pm 0.302}$	$13.261_{\pm 0.240}$	$\mathbf{1.985}_{\pm 0.384}$
Kddcup99 (395216 : 98804)	train	-	N/A	$854.86_{\pm 14.256}$	$515.48_{\pm 96.792}$	$176.92_{\pm 9.669}$	$\mathbf{14.202}_{\pm 1.150}$
	test	N/A	N/A	$170.03_{\pm 3.4624}$	$804.04_{\pm 119.51}$	$854.92_{\pm 48.558}$	$\mathbf{92.835}_{\pm 12.322}$

1. In VLLDA, it does not need to train the model.
2. N/A indicates that method could not finish the process within an hour.
3. **Bold** indicates the fastest computaional time on six methods.
4. The value with asterisk ($\star$) means that there is no statistically significant difference at the 5% significance level ($p > 0.05$).
5. For VEBF, CIL, and SHEF, computational time for the initial width of a hyper-ellipsoidal function is included to training time.

training data no matter how value of k is set. Hence, the more data are tested, the more time is obviously spent.

For LOL, the number of prototypes k set to 60 in all experiments may not be suitable for all data sets in the term of accuracy for the Spiral data, Spambase, and Letter. That means using only 60 hyperplanes may not be enough to classify those data sets. As the results, LOL is sensitive to the parameter settings in each data set and inconsistent with their conclusions in [21]. Moreover, the sequence of the incoming data has an effect on updating positions of prototypes and accuracy as seen in the results of Sparse-dense data and Faraway4 shown in Table 5 with a large standard deviation. Their computational time is directly proportional to the number of the training data and the number of classes, especially the number of classes. Since LOL adopts the one-vs-all strategy in the multi-class problem, LOL must iteratively learn one class at a time.

For VEBF, there is a large standard deviation of accuracy on Sparse-dense data, Faraway4, Gaussian data, Spambase, Waveform, Satimage, Letter, and Shuttle_trn. Consequently, the sequence of incoming data has an effect on the accuracy of this method that is consistent with the conclusion in [23]. For Shuttle_trn data set, VEBF has the unusual low accuracy at 38.343% with the unusual large standard deviation of 30.677 as shown in Table 5. To explain this situation, the accuracy characteristic displayed in Fig. 12f should be thoroughly analyzed. After considering all 50 experiments of VEBF on this data set, there are 31 out of 50 whose accuracy is immediately decreased at some incoming chunks and never increased again. This problem was found in the step of merging two neurons of VEBF algorithm when the initial width was set too large. After merging them, the width of the new neuron was set by assuming the Gaussian distribution of data in both neurons that may be smaller than the widths of two previous individual neurons. As the result, updating the parameters of a neuron based on the next training data may be dominated by a larger size of neuron. The training time is directly proportional to the number of the generated neurons during the training process. Whereas the testing time is directly proportional to the number of neurons in the final process. For example of Kddcup99 data set, there are about 117 generated neurons in the final state, see Table 6. They took about 515 seconds for 395,216 training data and about 804 seconds for 98804 testing data, see Table 7. Furthermore, if the number of neurons was unnecessarily generated, they would be stored in forms of d-by-d covariance matrices.

For CIL, the number of training data in each chunk is inversely proportional to the number of generated neurons. If the number of training data in each chunk is quite small, then the number of neurons may be redundantly generated (no merging strategy in this method). The number of generated neurons is directly proportional to the computational time. However, a large number of neurons may reduce the classification accuracy. Our findings show CIL is quite sensitive to the number of training data (chunk size) in each chunk on some data sets (Satimage and Letter). For examples, if the determined interval of Satimage was changed from [100,200] to [400,600], the average accuracy increased to 81.386% and the average number of used neurons decreased to 62.6. If the determined interval of Letter was changed from [100,200]

to [1000,2000], the average accuracy increased to 87.181% and the average number of used neurons decreased to 282.94. For Kddcup99, although CIL has the lowest average accuracy at 19.93%, see Table 5, it is not actually sensitive to the number of train data in each chunk due to the definition of the decision function for determining the class label of the testing data. This conclusion was confirmed by using the generated neurons from CIL in the first experiment of fold 1 and our proposed testing approach instead. The accuracy increased from 19.61% to 99.712%. with 2,292 neurons.

For our proposed SHEF, the performance depends on how SHEFs and the network of SHEFs are appropriately constructed to capture sequential streaming data. According to the 2-dimensional illustration in Fig. 11, it may be noticeable that no matter how complicated the patterns of data distribution are, SHEFs can capture them very well. The proposed method provided the high accuracy more than 80% on several data sets during the incremental learning in the training process at any time stamp of streaming data (see Fig. 12). Although there are more steps in our testing process than the testing process of other methods, our testing process is rather fast because fewer numbers of SHEFs were effectively generated on all experimental data sets. Moreover, our method also spent less training time than the others because the merging step helped reduce the number of generated SHEFs during the training process. For Kddcup99 data set, about 16 SHEFs were generated because there are only 15 classes out of 23 classes that have the number of data more than 20. Before testing the data set, any SHEFs containing the number of members fewer than M ($M = 3$) were eliminated. Therefore, some classes might be ignored as noisy data. The number of training data in each chunk did not affect massively to our accuracy. Even though the input data were fed as the chunk, our algorithm still sequentially learns each datum. The sequence of incoming data has a slight effect on the accuracy of SHEF compared with other methods by considering from our quite small standard deviation of all data sets in Table. 5.

Fig. 16 describes the effect of two hyper-parameters on the accuracy and the number of generated SHEF neurons. The preliminary experiments (see Fig. 16) found that when ϵ and r increased, the average number of generated neurons tended to decrease. The constants ϵ and r in the learning algorithm are directly proportional to the size of SHEF which can indirectly effect the merging strategy. Due to small values of ϵ and r, the size of SHEF was too small. As a result, any two SHEFs of the same class may not be merged, it may lead to many redundant SHEFs generated. Whereas the size of SHEF with large values of ϵ and r makes many SHEFs overlap. As a result, many SHEFs may be unnecessarily merged. Therefore, if we consider only the high accuracy, the value of ϵ and r should be chosen in the range of 10^{-6} to 10^{-2} and 0.5 to 1.5 respectively, such as the experiments on the 11 data sets.

Three synthetic data sets (except Gaussian) were generated with a complex distribution. The number of generated neurons has an effect on the classification accuracy. Hence, choosing ϵ and r has an important factor. Whereas Gaussian data set was generated with a normal distribution (a simple distribution), ϵ and r may have a slight effect on the accuracy. However, they have a major effect on the number of generated neurons and the computational time on Gaussian data. Consider on seven real-world data sets, it was found that when r was fixed, varying ϵ has a slight effect on accuracy but it has an effect on the number of generated neurons. Especially, on Kddcup99 data set, ϵ has an obvious effect on the number of generated neurons. On the other hand, when ϵ was fixed, varying r has an effect on the accuracy.

The upper bound of time complexity SHEF learning algorithm can be analyzed as follows. Assume that there is only one class in the first chunk of streaming data with N_1 vectors in d dimensions. Initializing the width takes $O(dN_1^2)$. In learning steps, the time complexity was analyzed for one datum at a time. Assume a datum is in class k and there are h SHEFs in this class. the time complexity of **Algorithm 1** from steps 7, 9, and 15-21 is $O(dN_1) + [O(dh) + O(h) + O(d^3)] = O(dN_1) + [O(dh) + O(d^3)]$. Suppose there are N training data points, $N_1 \ll N$, $d \ll N$, and $h \ll N$. So the time complexity becomes $O(dN_1) + \sum_{i=1}^{N}[O(dh) + O(d^3)] = O(dN_1) + O(dhN) + O(d^3N) = O(N)$.

X. CONCLUSION

This paper contributed the following issues to cope with learning streaming chunk data and determining the class of queried data in order to achieve higher classification accuracy and less number of neurons:

1) A generic structure of *Scalable Hyper-Ellipsoidal Function* (SHEF) with a regularization parameter ϵ and a scalable constant r to solve the problem of curse of dimensionality.
2) A new recursive function to update the covariance matrix of SHEF with less computational time than those of the other methods.
3) A fast and easy conditions to test the interacting states including overlap, inside, and touch of two SHEFs.
4) A new distance measure named *Projection Ratio* based on the projected distance on LDA discriminant vector.

The proposed concepts can achieve higher accuracy, less computational time, and less number of generated neurons. However, this approach has not been tested with more complex data characteristics such streaming data with dynamic class drift.

APPENDIX A
PROOF OF THEOREM 1

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

$$\mathbf{S}' = \frac{n(\mathbf{S} + \mathbf{c}\mathbf{c}^T) + \mathbf{x}'(\mathbf{x}')^T}{n'} - \mathbf{c}'(\mathbf{c}')^T \tag{42}$$

$$= \frac{n(\mathbf{S} + \mathbf{c}\mathbf{c}^T) + \mathbf{x}'(\mathbf{x}')^T}{n'} - (\frac{n\mathbf{c} + \mathbf{x}'}{n'})(\frac{n\mathbf{c} + \mathbf{x}'}{n'})^T \quad (43)$$

$$= \frac{n'n\mathbf{S} + n\mathbf{c}(\mathbf{c}^T - (\mathbf{x}')^T) + n\mathbf{x}'((\mathbf{x}')^T - \mathbf{c}^T)}{(n')^2} \quad (44)$$

$$= \frac{n'n\mathbf{S} + n(\mathbf{c} - \mathbf{x}')(\mathbf{c} - \mathbf{x}')^T}{(n')^2} \quad (45)$$

$$(46)$$

$$= \frac{n}{n'}(\mathbf{S} + \frac{(\mathbf{c} - \mathbf{x}')(\mathbf{c} - \mathbf{x}')^T}{n'}) \quad (47)$$

□

APPENDIX B
PROOF OF THEOREM 2

Let d be the dimensions of data; $\widetilde{\mathbf{S}}_i^{-1}$ of size $d \times d$ be the scaled covariance matrix $\mathbf{S}^{-1}/r^2$ of i^{th} SHEF; $\mathbf{R}_i$ of size $(n+1) \times (n+1)$ be the translation matrix of the centroid of the i^{th}SHEF; $\mathbf{x}$ be any data point, $\mathbf{c}_i$ be the center of data in the i^{th}SHEF. Then the representation of SHEF in [29] would be

$$\mathbf{x}^T \mathbf{R}_i^T \begin{bmatrix} \widetilde{\mathbf{S}}_i^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_i \mathbf{x} = 0 \quad (48)$$

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

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

From (48), two candidate SHEFs should be checked for overlap if they satisfy the following conditions.

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = 0 \quad (49)$$

$$\mathbf{x}^T \mathbf{B} \mathbf{x} = 0 \quad (50)$$

where

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

From (49) and (50), we get the following equations.

$$\mathbf{x}^T(\lambda \mathbf{A} - \mathbf{B})\mathbf{x} = 0 \quad (51)$$

$$\mathbf{x}^T \mathbf{A}(\lambda \mathbf{I} - \mathbf{A}^{-1}\mathbf{B})\mathbf{x} = 0 \quad (52)$$

where $\mathbf{I}$ is an identity matrix. Term $\lambda\mathbf{I} - \mathbf{A}^{-1}\mathbf{B}$ can be used to determine the states of touch, overlap, and inside. In [29], the intersection of SHEF_A and SHEF_B can be described with an eigenvalue of the matrix $\mathbf{A}^{-1}\mathbf{B}$ or $\mathbf{B}^{-1}\mathbf{A}$ where

$$\mathbf{A}^{-1}\mathbf{B} = \left(\mathbf{R}_A^T \begin{bmatrix} \widetilde{\mathbf{S}}_A^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_A\right)^{-1} \mathbf{R}_B^T \begin{bmatrix} \widetilde{\mathbf{S}}_B^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_B \quad (53)$$

for SHEF_A and SHEF_B, respectively.

$$\mathbf{A}^{-1}\mathbf{B} = \mathbf{R}_A^{-1} \begin{bmatrix} \widetilde{\mathbf{S}}_A & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} (\mathbf{R}_A^T)^{-1} \mathbf{R}_B^T \begin{bmatrix} \widetilde{\mathbf{S}}_B^{-1} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times d} & -1 \end{bmatrix} \mathbf{R}_B. \quad (54)$$

Since for any SHEF_i

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

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

$$\mathbf{A}^{-1}\mathbf{B} = \begin{bmatrix} \mathbf{D} & -\mathbf{D}\mathbf{c}_B + \mathbf{c}_A \\ \mathbf{F} & -\mathbf{F}\mathbf{c}_B + 1 \end{bmatrix}_{(d+1)\times(d+1)} \quad (55)$$

where

$$\begin{aligned} \mathbf{F}_{1\times d} &= (-\mathbf{c}_A + \mathbf{c}_B)^T \widetilde{\mathbf{S}}_B^{-1} \\ \mathbf{D}_{d\times d} &= \widetilde{\mathbf{S}}_A \widetilde{\mathbf{S}}_B^{-1} + \mathbf{c}_A \mathbf{F}. \end{aligned} \quad (56)$$

□

APPENDIX C
PROOF OF THEOREM 3

Proof: Suppose the projected $\mathbf{x}$ is inside the projected SHEF ($D(\mathbf{x}, \mathbf{c}, \mathbf{S}) < 1$) on the discriminant vector $\mathbf{w}_p$ defined in (39). The projection must satisfy the following inequality.

$$|\mathbf{w}_p^T(\mathbf{x} - \mathbf{c})| < r\sqrt{\mathbf{w}_p^T \mathbf{S} \mathbf{w}_p}. \quad (57)$$

Substitute $\mathbf{w}_p = \dfrac{\mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})}{||\mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})||}$ in (57) to obtain

$$|(\mathbf{x} - \mathbf{c})^T \mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})| < r\sqrt{(\mathbf{x} - \mathbf{c})^T \mathbf{S}^{-1}(\mathbf{x} - \mathbf{c})}. \quad (58)$$

It is obvious to see that

$$0 < (\mathbf{x} - \mathbf{c})^T \mathbf{S}^{-1}(\mathbf{x} - \mathbf{c}) < r^2. \quad (59)$$

Equation (59) implies that $\mathbf{x}$ is inside SHEF scaled by r in the original space following (16). □

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

ACKNOWLEDGMENT

The authors would like to thank all the anonymous reviewers for their valuable suggestions.

REFERENCES

[1] M. Zakrzewicz, M. Wojciechowski, and P. Gławiński, "Solution pattern for anomaly detection in financial data streams," in *New Trends in Databases and Information Systems*. Cham, Switzerland: Springer, 2019, pp. 77–84.

[2] A. Shen, R. Tong, and Y. Deng, "Application of classification models on credit card fraud detection," in *Proc. Int. Conf. Service Syst. Service Manage.*, Jun. 2007, pp. 1–4.

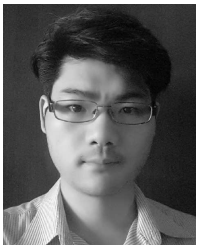
[3] K. Shameer, M. A. Badgeley, R. Miotto, B. S. Glicksberg, J. W. Morgan, and J. T. Dudley, "Translational bioinformatics in the era of real-time biomedical, health care and wellness data streams," *Briefings Bioinf.*, vol. 18, no. 1, pp. 105–124, Feb. 2016.

[4] E. Haselsteiner and G. Pfurtscheller, "Using time-dependent neural networks for EEG classification," *IEEE Trans. Rehabil. Eng.*, vol. 8, no. 4, pp. 457–463, Dec. 2000.

[5] N. Arunkumar, K. Ramkumar, V. Venkatraman, E. Abdulhay, S. L. Fernandes, S. Kadry, and S. Segal, "Classification of focal and non focal EEG using entropies," *Pattern Recognit. Lett.*, vol. 94, pp. 112–117, Jul. 2017.

[6] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Mach. Learn.*, vol. 90, no. 3, pp. 317–346, Mar. 2013.

[7] C. C. Aggarwal, *Data Streams: An Overview and Scientific Applications*. Berlin, Germany: Springer, 2010, pp. 377–397.
[8] H.-L. Nguyen, Y.-K. Woon, and W.-K. Ng, "A survey on data stream clustering and classification," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 535–569, Dec. 2014.
[9] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining Knowl. Discovery*, vol. 30, no. 4, pp. 964–994, Jul. 2016.
[10] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.
[11] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Gener. Comput. Syst.*, vol. 75, pp. 187–199, Oct. 2017.
[12] I. V. Z. E. liobait, M. Pechenizkiy, and J. Gama, *An Overview of Concept Drift Applications*. Cham, Switzerland: Springer, 2016, pp. 91–114.
[13] Y. Aliyari Ghassabeh, F. Rudzicz, and H. A. Moghaddam, "Fast incremental LDA feature extraction," *Pattern Recognit.*, vol. 48, no. 6, pp. 1999–2012, Jun. 2015.
[14] D. Chu, L.-Z. Liao, M. K.-P. Ng, and X. Wang, "Incremental linear discriminant analysis: A fast algorithm and comparisons," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2716–2735, Nov. 2015.
[15] G.-F. Lu, J. Zou, and Y. Wang, "A new and fast implementation of orthogonal LDA algorithm and its incremental extension," *Neural Process. Lett.*, vol. 43, no. 3, pp. 687–707, Jun. 2016.
[16] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.
[17] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 35, no. 5, pp. 905–914, Oct. 2005.
[18] I. A. Lawal, *Incremental SVM Learning: Review*. Cham, Switzerland: Springer, 2019, pp. 279–296.
[19] J. Xu, C. Xu, B. Zou, Y. Yan Tang, J. Peng, and X. You, "New incremental learning algorithm with support vector machines," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 11, pp. 2230–2241, Nov. 2019.
[20] B.-C. Kuo, H.-H. Ho, C.-H. Li, C.-C. Hung, and J.-S. Taur, "A kernel-based feature selection method for SVM with RBF kernel for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 1, pp. 317–326, Jan. 2014.
[21] Z. Zhou, W.-S. Zheng, J.-F. Hu, Y. Xu, and J. You, "One-pass online learning:A local approach," *Pattern Recognit.*, vol. 51, pp. 346–357, Mar. 2016.
[22] 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, no. 3, pp. 381–392, Mar. 2010.
[23] P. Junsawang, S. Phimoltares, and C. Lursinsap, "A fast learning method for streaming and randomly ordered multi-class data chunks by using one-pass-throw-away class-wise learning concept," *Expert Syst. Appl.*, vol. 63, pp. 249–266, Nov. 2016.
[24] Z. Fan, Y. Xu, and D. Zhang, "Local linear discriminant analysis framework using sample neighbors," *IEEE Trans. Neural Netw.*, vol. 22, no. 7, pp. 1119–1132, Jul. 2011.
[25] T.-K. Kim and J. Kittler, "Locally linear discriminant analysis for multimodally distributed classes for face recognition with a single model image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 318–327, Mar. 2005.
[26] N. Wattanakitrungroj, S. Maneeroj, and C. Lursinsap, "Versatile hyper-elliptic clustering approach for streaming data based on one-pass-thrown-away learning," *J. Classification*, vol. 34, no. 1, pp. 108–147, Apr. 2017.
[27] M. Thakong, S. Phimoltares, S. Jaiyen, and C. Lursinsap, "Fast learning and testing for imbalanced multi-class changes in streaming data by dynamic multi-stratum network," *IEEE Access*, vol. 5, pp. 10633–10648, 2017.
[28] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.
[29] S. Alfano and M. L. Greer, "Determining if two solid ellipsoids intersect," *J. Guid., Control, Dyn.*, vol. 26, no. 1, pp. 106–110, Jan. 2003.
[30] A. Sharma and K. K. Paliwal, "Linear discriminant analysis for the small sample size problem: An overview," *Int. J. Mach. Learn. Cybern.*, vol. 6, no. 3, pp. 443–454, Jun. 2015.
[31] J. H. Friedman, "Regularized discriminant analysis," *J. Amer. Statist. Assoc.*, vol. 84, no. 405, pp. 165–175, 1989.
[32] K. Zimmermann and T. Svoboda, "Approximation of Euclidean distance between point from ellipse," Center Mach. Perception, FEE Czech Tech. Univ., Prague, Czech Republic, Tech. Rep. CTU–CMP–2005–23, Aug. 2005.
[33] B. P. Stephen, "Algorithms for ellipsoids," Sibley School Mech. Aerosp. Eng., Cornell Univ., New York, NY, USA, Tech. Rep. FDA-08-01, Feb. 2008.
[34] D. Dheeru and E. K. Taniskidou. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[35] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2015, pp. 59–68.



PERASUT RUNGCHARASSANG received the B.Sc. degree in mathematics from Kasetsart University, Bangkok, Thailand, in 2008, and the M.Sc. degree in applied mathematics and computational science from Chulalongkorn University, Bangkok, in 2012, where he is currently pursuing the Ph.D. degree in applied mathematics and computational science.

His research interests include neural networks, streaming data learning, and classification problem.

CHIDCHANOK LURSINSAP (Member, IEEE) received the B.Eng. degree (Hons.) in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 1978, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana–Champaign, Urbana, in 1982 and 1986, respectively. He was a Lecturer with the Department of Computer Engineering, Chulalongkorn University, in 1979. In 1986, he was a Visiting Assistant Professor with the Department of Computer Science, University of Illinois at Urbana-Champaign. From 1987 to 1996, he has worked with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, as an Assistant and Associate Professor. After that, he came back to Thailand to establish Ph.D. Program in computer science at Chulalongkorn University, where he became a Full Professor. His major research interest includes neural learning and its applications to other science and engineering areas.