Knowledge and Information Systems
https://doi.org/10.1007/s10115-021-01582-4

SURVEY PAPER

Data stream classification with novel class detection: a review, comparison and challenges

Salah Ud Din[1,2,3] · Junming Shao[1,2] · Jay Kumar[1,2] · Cobbinah Bernard Mawuli[1,2] · S. M. Hasan Mahmud[4] · Wei Zhang[5] · Qinli Yang[1,2]

Received: 22 December 2020 / Revised: 27 May 2021 / Accepted: 30 May 2021

Abstract
Developing effective and efficient data stream classifiers is challenging for the machine learning community because of the dynamic nature of data streams. As a result, many data stream learning algorithms have been proposed during the past decades and achieve great success in various fields. This paper aims to explore a specific type of challenge in learning evolving data streams, called concept evolution (emergence of novel classes). Concept evolution indicates that the underlying patterns evolve over time, and new patterns (classes) may emerge at any time in streaming data. Therefore, data stream classifiers with emerging class detection have received increasing attention in recent years due to the practical values in many real-world applications. In this article, we provide a comprehensive overview of the existing works in this line of research. We discuss and analyze various aspects of the proposed algorithms for data stream classification with concept evolution detection and adaptation. Additionally, we discuss the potential application areas in which these techniques can be used. We also provide a detailed overview of evaluation measures and datasets used in these studies. Finally, we describe the current research challenges and future directions for data stream classification with novel class detection.

Keywords Novel class detection · Data stream classification · Concept drift · Clustering · Concept evolution

✉ Junming Shao
junmshao@uestc.edu.cn

1 School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

2 Yangtze Delta Region Institute (Huzhou), University of Electronic Science and Technology of China, Huzhou 313001, China

3 COMSATS University Islamabad, Abbottabad Campus, Islamabad, Pakistan

4 Computational Intelligence Lab, School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

5 Science and Technology on Electronic Information Control Laboratory, Chengdu, China

Published online: 08 July 2021

1 Introduction

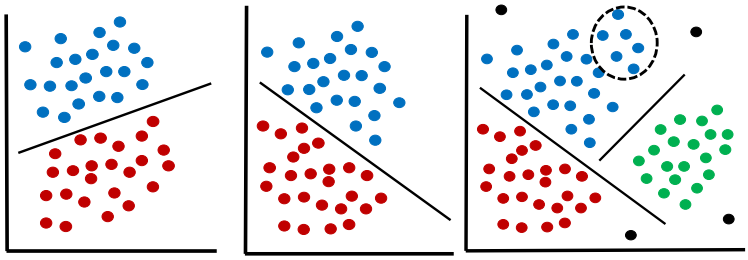
Data stream mining covers many tasks, like frequency counting, clustering, time-series analysis and classification. Among all these tasks, the data stream classification has been the most focused and significant research area over the past decade [17,89,122,165]. As a result, several algorithms have been proposed and gained considerable attention from the industry as well [20,37,46,56,74,83,100,131]. Data stream classification slightly differs from traditional static data classification task. Data stream classifiers continuously receive data sequentially at a very high speed, while in a static setting, all the data is available at once to train a classifier. Additionally, while dealing with the massive amount of online data, the data stream classifiers must work with limited time and memory.

Moreover, the nature of the data stream is non-stationary, which means that the data distribution with time evolves. This phenomenon is known as the concept drift [55,68,73,88]. Due to concept drift, the classifier trained from previous data can become obsolete/ ineffective for incoming data. Therefore, concept-drifting learning algorithms must be adaptive to evolving concepts. A variety of concept-drift learning methods have been proposed over the past decades [18,23,66,72,136,137].

Another important challenge for learning algorithms in data streams is the concept evolution (emergence of novel classes) [1,95,114,145,153,168]. For traditional classification tasks, the number of classes is predefined, and a classifier is constructed to classify the test data in these known classes. In many real-world data stream applications, however, the concept of underlying problems can evolve over time. Specifically, new patterns (classes) can emerge at any time during the streaming data. For illustration, consider the intrusion detection system as an example. Firstly, we train a classifier from certain types of known attacks (classes). This classifier only recognizes the attacks on which it was trained previously and cannot identify a new kind of attack (class) when it appears. Novel class identification is also crucial in the case where an important class is underrepresented in the initial phase of training [41,43,51,123,124,141,166,170]. Figure 1 provides a simple example for illustration. At the start (see Fig. 1a), we have initial training data with two classes, and a classifier is first trained based on the initial data. Over time, we can see that a concept drift happens (Fig. 1b). Additionally, some data points associated with the novel class are evolving (see Fig. 1c). Here, the black filled points are outliers, data points in the dashed circle are drifting points, and green filled data points indicate the data points with a new class (i.e., concept evolution). Thus, a significant challenge is to distinguish between the actual emerging class from drift in the existing classes or a noise.

Data stream classification with emerging class identification refers to learning algorithms that can detect and learn new classes. This becomes an important research challenge because of its practical significance in many applications such as text mining [80,148], intrusion detection [128], fraud detection [151], power distribution networks [93], activity recognition [1,115], bioinformatics [141], fault diagnosis [166] and runoff simulation [157,163]. The novel class detection system allows us to significantly reduce human efforts that would otherwise be needed to fix incorrect classifications due to unidentified emerging classes manually. Additionally, the system can learn or discover novel patterns that we did not initially anticipate.

In academic literature, concept evolution is considered from two viewpoints, i.e., concept evolution adaptation and novel class identification and adaptation. The former task is concerned with how to adapt new classes from the data stream to maintain an efficient classification model without explicitly detecting the novel class [52,53,81,92,101,114,119,145,



(a) Initial Data (b) Concept Drift (c) Concept Drift, Evolution and Outliers

Fig. 1 The challenges of data stream classification in the presence of concept drift, concept evolution and outliers. Blue and red dots are existing classes, green dots are novel class, and black dots are outliers

168,172]. But recent research is more concerned with how to detect possible new classes and provide automatic assistance in data labeling.

Learning evolving data stream is a hot research topic, and many survey papers [68,74,100, 122] have been published in the literature over the past years. These survey papers provide theoretical foundations and insights into both the established and the state-of-the-art learning algorithms. However, these surveys do not cover the concept evolution learning algorithms. In 2015, a survey paper related to novelty detection [61] was published. However, this survey paper provides an overview of only 12 research papers. Since then, several new state-of-the-art algorithms have been proposed. Therefore, it is necessary to integrate and evaluate these algorithms to provide the most recent research trends on data stream classification with novel class detection, which is one of the primary focus of this research.

The main contributions of the paper are summarized as follows.

- Provide a comprehensive review of different state-of-the-art algorithms for data stream classification with novel class detection in the literature.
- Provide deep insight into the existing works and analyzes the strength and weaknesses of each algorithm.
- Provide a taxonomy of literature on novel class detection with respect to different aspects.
- Provide an experimental comparison between different algorithms.
- Provide a discussion on the potential application areas in which these techniques can be used.
- Provide a list and description of available benchmark datasets and provide a discussion on evaluation measures that are used for performance comparison.
- Provide a summary of the available software tools for data stream mining, especially the link of source codes of different novel class detecting algorithms, is also provided.
- We also highlight the significant research challenges that need to be addressed in the coming years. We hope that this survey will provide inspiration and a roadmap for the development of new methods in this research area.

The remaining paper is organized as follows. Section 2 provides some fundamental concepts related to data stream. Section 3 presents the categorization of proposed algorithms from different aspects. The review of the proposed algorithms is presented in Sect. 4. Section 5 discusses the limitations of the reviewed algorithms. Section 6 presents evaluation measures and used in different algorithms. Section 7 presents the empirical study. Section 8 presents the application areas, datasets used in different algorithms, and some open-source

tools for mining streaming data. The paper ends with Sect. 9 with concluding remarks and future research directions.

2 Preliminaries

2.1 Notion and notations

Definition 1 (*Data stream*) A data stream is a potential infinite sequence of data items, which are continuously arrive at a rapid speed. Mathematically, let us define a data stream as $D = \{(x_t, y_t)\}_1^{\infty}$. Here, pair (x_t, y_t) is a data item arrived at a time stamp t, $x_t \in R^n$ is a n-dimensional feature vector and $y_t \in Y = \{c_1, c_2, \ldots, c_k\}$ is the ground truth, i.e., a class label (if available) associated with the data item.

Definition 2 (*Concept drift*) Concept drift occurs when a source that generates data changes over time. Formally, let's take S_t (data distribution) as sources generating instance x_t at the time point t. A concept drift occurs when two instances x_0 and x_1, with the timestamps t_0 and t_1, are generated from different sources, i.e., $S_{t_0} \neq S_{t_1}$. According to [68], this change can take different types of forms; that is, it may be due to a change in the characteristics of the input data (i.e., $P(x)$) or a change in the relationship between the input data and the target labels (i.e., $P(y|x)$). Mathematically, concept drift between two timestamps t_0 and t_1 can be defined as follows.

$$\exists X : P_{t_0}(x, y) \neq P_{t_1}(x, y). \tag{1}$$

Here, P_t represents the joint distribution between the feature vector x and class label y at time t. According to [149], the changes in the data may be due to a change in the posterior probabilities of the classes $P(y|x)$ or to a change in the distribution of the class $P(y)$ or change in feature space $P(x)$. Change in the feature space $P(x)$ is known as virtual drift, which may or may not affect the decision boundary. Also, it important to note that a change of $P(y|x)$ (with or without a change of $P(x)$) is considered a real concept drift, because a change in $P(y|x)$ affects the decision boundary [68].

Definition 3 (*Anomaly, outlier and novelty detection*) An anomaly is usually defined as an irregularity, or a noisy event in the normal data [5,40,129]. The task of anomaly detection is to find unexpected or abnormal behaviors (patterns) in normal data. In the literature, other names given to these patterns are contaminants, peculiarities, surprises, exceptions, outliers, anomalies [28,38,40]. Aggarwal et al. [3] describe outlier as a data item that can be regarded as an irregularity, a noise, or an abnormality. In contrast, an anomaly is considered as a particular type of outlier that needs the attention of a specialist. According to [39], outliers can be contenders for abnormal data, which can severely affect systems. Examples of outliers are a human fault, changes in the environment, instrumentation malfunction and malicious events.

Anomaly and outlier detection is often used in the literature as a synonym of novelty detection [21,42]. Novelty detection referred to the task of detecting data items that significantly different from the data that is available in the training set [102,103]. For [67], novelty or novel concepts (patterns) are cohesive groups of data items whose characteristics largely differ from normal data, while outliers are simply sparse independent data items without representing any new concepts. Mostly, novelty detection problem is taken as a one-class

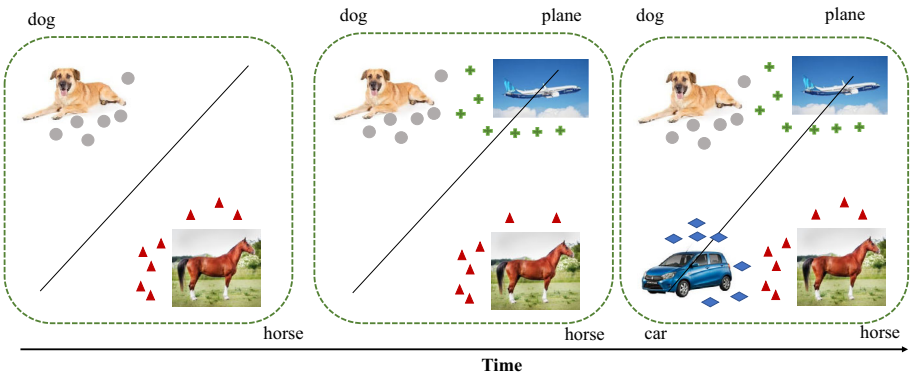


Fig. 2 Illustration of concept evolution. Initially, we have two classes and over time two more classes emerge during the stream

classification [21,130]. In this, a classifier is trained from only one class data representing the normal or positive class, and the task is to distinguish between normal and abnormal data (i.e., binary classification).

Definition 4 (*Concept evolution*) Let $Y = \{c_1, c_2, \ldots, c_k\}$ be a set of classes from a training set on which a classifier is trained. This dataset represents the known concepts about the underlying problem and the classes in the set Y are called known or existing classes. During the streaming data, a new class appears that is not present in the set Y. That is, a class c appears at the time t if $c \notin Y$. Such a class is called an emerging class. This phenomenon is called concept evolution [107,109]. It is also important to note that the class c remains a new class until examples belonging to the class c are labeled and incorporated in the classifier. As an example, Fig. 2 depicts the concept evolution in image classification in data stream.

2.2 Problem formalization

Data stream classifiers with emerging class detection usually work in two phases: warm-up and online phase [61].

2.2.1 Warm-up phase

In this phase, a model ($f : X \rightarrow Y$) is first learned with some initial training dataset, i.e., $X = D_{\text{init}} = \{(x_i, y_i)\}_1^m$. Here, $x_i \in R^n$ is an n-dimensional feature vector, $y_i \in Y = \{c_1, c_2, \ldots, c_k\}$ is associated class labels.

2.2.2 Online phase

In this phase, model f learned in the warm-up phase is used both for classification and emerging class identification simultaneously. In the online phase, the learned model has to perform three tasks, namely: (i) novel class identification, (ii) classification of the existing or known classes and (iii) updating the model with the latest data and incorporate newly detected classes into the model [116].

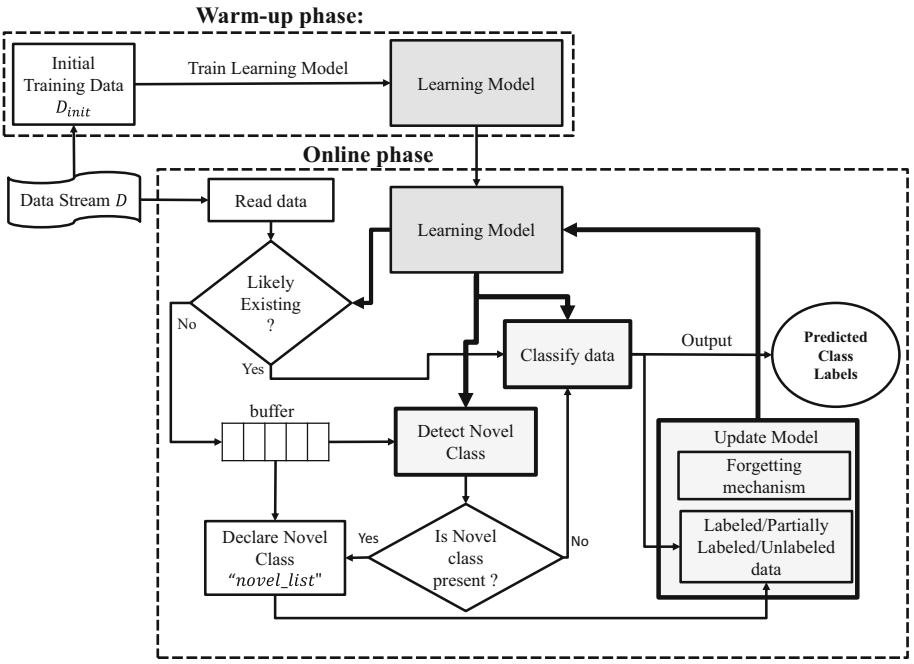


Fig. 3 The workflow of data stream classification with emerging class detection. Initially, in the warm-up, a model is learned from training data. In the online phase, the learned model perform three tasks, namely: (i) novel class identification, (ii) classification of the existing or known classes and (iii) updating the model with the latest data and incorporate newly detected classes into the model

In general, for incoming instances in the streaming data, the current decision model first filters the instances to determine whether they are likely to belong to the existing classes or potential emerging pattern. If the instances are likely to belong to the existing classes, their class labels are predicted by the model. Otherwise, the instances are marked as potential emerging class instances and stored in a buffer. Afterward, the buffer is periodically examined for a novel class. If a novel class is found it is incorporated in the model by expanding the label set $Y = \{c_1, c_2, \ldots, c_k, c_{k+1}\}$. So that it can correctly classify the future instances belonging to this new class. In evolving data streams, an important aspect is to keep the learning model up-to-date with the latest data to cope with concept drift and implement forgetting mechanisms to remove the outdated concepts from the model. The overall workflow of stream classifiers with novel class detection is depicted in Fig. 3.

3 Taxonomy of proposed approaches

Taxonomy to categorize novel class detection approaches for data streams can be based on several criteria [61]. The focus of this study is to analyze different aspects of data stream classifiers with emerging class identification. Therefore, grouping or categorization can be made according to the following criteria. Figure 4 shows the categorization of proposed algorithms from different aspects.

- Number of classes considered: One-class/Multi-class
- Number of classifiers used: Single/Ensemble
- Type of learning used in online phase: Unsupervised/ Semi-supervised/ Supervised
- Classifier construction: Cluster-based/Model-based
- Number of labels associated with data: Single/Multi-label
- Other relevant aspects
  - Handling outliers
  - Handling recurring contexts
  - Handling feature evolution
  - Number of novel classes detected: One/Multiple
  - Forgetting mechanism.

3.1 One or multi-class classification

Several proposed approaches work within the framework of one-class classification [10,79,90,133,142–144,146]. In these approaches, a classifier is trained from only one class data representing normal or positive class and the task is to distinguish between normal and abnormal data. The output of these classifiers is binary, i.e., all the test instances are classified as normal by the model if they represent the normal class. Otherwise, the instances are marked as unknown or abnormal. A group of cohesive unknown instances is then declared as a novelty or new class, while other approaches proposed in [1,2,7,7–9,16,29,44,48,49,58,64,65,70,71,75–77,84,85,104–111,117,120,125,126,138,155,156,160–162,167] are multi-class classifiers. In these approaches, a classifier is trained with a dataset that has multiple classes. The learned classifiers are then able to distinguish between multiple classes of data and detect the appearance of new classes.

3.2 Single or ensemble classifiers

The second aspect of categorization is number of classifiers used to build a learning model. Several approaches [7,30,44,49,60,63,71,79,85,116,131,142–144,155,156,161] build a single learning model which is incrementally update during the stream progress, while techniques proposed in [2,8,9,33,58,64,65,70,75–77,84,104–110,120,125,126,138,160] use ensemble of classifiers to build the learning model.

3.3 Type of learning used in online phase

Another important aspect of categorization to be considered is the type of learning used in the online phase. Therefore, based on this, approaches in this area can be categorized as unsupervised, semi-supervised and supervised. Unsupervised methods proposed in [7,30,33,44,60,63,70,71,79,85,116,118,131,142–144,155,156,171,173] are based on the assumption that class labels for training data are available only in initial model construction in the warm-up phase and no external feedback (class labels) is available after that. These methods update the model using unlabeled instances. In semi-supervised methods [1,29,75–77,110,120,167], the classification model is updated with partially labeled data, while supervised methods [2,7–9,16,48,49,58,64,65,84,104–109,111,117,125,126,138,155,156,160–162] assume that the class labels for all incoming streaming data will be accessible after a delay and classification model is updated with the true class labels.

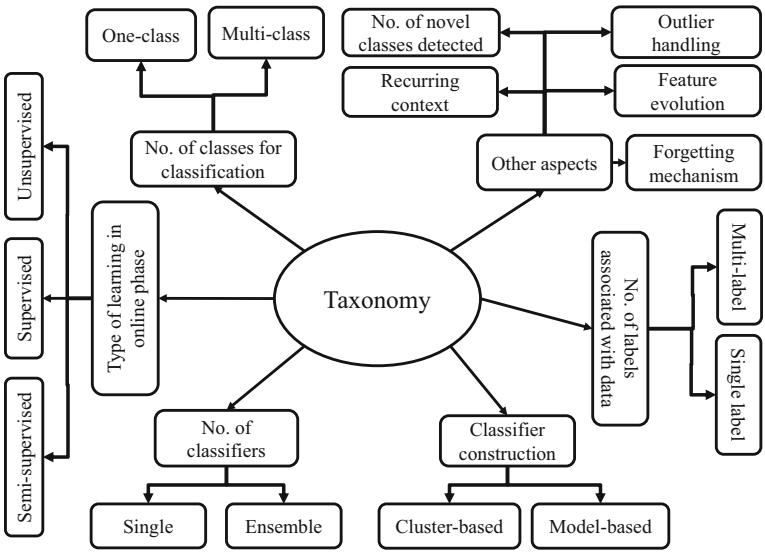


Fig. 4 Taxonomy of novel class detection algorithms from different aspects

3.4 Classifier construction

In the literature, different approaches use different types of techniques to construct the learning model. Therefore, based on the classifier construction, these approaches can be divided into two types: clustering-based [2,7,9,44,58,60,63,65,71,79,85,104–109,116,125,131,142–144,155,156] and model-based [30,64,65,90,116,118,125–127,133,146,167,171,173] learning algorithms.

3.5 Number of labels associated with data

Traditional supervised learning tasks consist of creating a classifier from a set of data, with each example of the dataset being associated with a single class (label) from a set of classes. In learning environments with multiple labels, an example can be associated with different labels [32,164]. In recent years, research into multi-label learning has received increasing attention due to its applied significance in many applications such as tag recommendation [86], bioinformatics [112], text mining [150], information retrieval [158]. Given its importance, some algorithms [85,118,171,173,174] are also proposed to identify emerging labels in the field of multi-label learning.

3.6 Other relevant aspects

Similarly, some other important aspects are also needed to consider when developing new algorithms. These aspects are not addressed by all approaches. These include handling outliers, recurring context identification, feature evolution and detecting one or more-than-one novel class(es).

3.6.1 Outliers

Outliers are sparse and isolated data items that have significantly different characteristics from normal data [40]. The learning algorithm needs to handle outliers because the presence of outliers can severely degrade the classifier performance as these outliers can be viewed as novel class or change in the existing classes due to concept drift.

3.6.2 Recurring contexts

Recurring contexts are the concepts that were learned in the past by the learning algorithm and may reappear in the future [6,87]. It is a special type of concept drift. Therefore, if a concept recurs, it is imperative to identify it as an old concept instead of considering it as a new concept. Because learning an old concept again for each reappearance would be a waste of time and effort for already learned concepts. To solve this problem, some techniques [8,9,60,63,105], save obsolete concepts instead of discarding them. The saved concepts are then analyzed for possible recurring context identification to ensure good classification performance in a cost-effective manner.

3.6.3 Feature evolution

Feature evolution is also a major problem in dynamic data streams. Traditional learning algorithms work with the fixed-size feature set, where the classifiers and test instances have similar feature space, however, new features may appear in data streams (such as text streams), and some old features may disappear during the streaming of data. Emerging class identification in feature evolving data streams is not a trivial task. Few works [2,106,125–127] have been proposed to solve this problem.

3.6.4 Forgetting mechanism

In dynamic data streams, the distribution of data can change over time. Learning algorithms must adapt to changing concepts with the latest data. Some algorithms also implement a forget mechanism to remove outdated concepts from the model that have been learned previously to make the model inline with the recent trends in the data.

3.6.5 Number of novel classes detected

Several techniques from the literature identify emerging pattern(s) as one class, i.e., they assume that only one class may appear at a time in the stream, while other approaches [60,63,107,108] consider that multiple novel classes can emerge simultaneously over time. Thus they can identify multiple class at the same time.

4 Review of proposed algorithms

The taxonomy we follow here, which categorizes the stream classifiers with emerging class identification, is based on the construction of learning models. In the literature, different approaches use different types of techniques to construct the learning model. Therefore, based

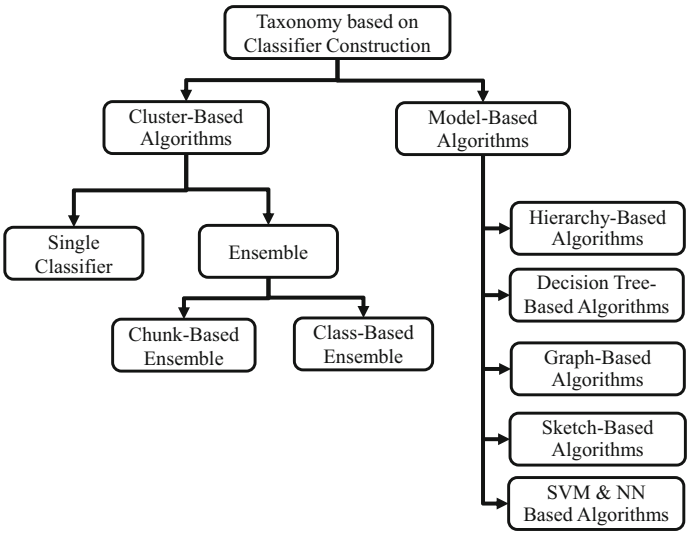


Fig. 5 Categorization of proposed approaches based on the classifier construction

on the classifier construction, these approaches can be divided into two types: clustering-based and model-based learning algorithms. In the following sections, we briefly discuss each algorithm. Figure 5 shows the categorization of proposed approaches based on classifier construction.

4.1 Clustering-based learning algorithms

The clustering-based learning algorithms are well-known methods of novel class recognition. The main idea of these algorithms is to create decision boundaries for learning models by applying some clustering algorithms (such as k-means or DBSCAN) to represent normal or known concepts (classes). A hypersphere represents every cluster of the model in the feature space by its center and radius. All test instances located within the hypersphere are classified as belonging to the normal concept. In comparison, instances residing outside of hyperspheres are marked as outliers (probable new class instances) and temporarily stored in a buffer for further analysis. Due to concept drift, the buffer may contain instances belonging to normal concepts or actual outliers. As a result, different algorithms apply different strategies to separate novel class instances from known class examples and outliers. Finally, a new class is declared if a group of cohesive outliers (with similar characteristics) is found. Figure 6 depicts the working of cluster-based algorithms. Approaches in this field can be further categorized into single or ensemble classifiers.

4.1.1 Clustering-based single classifiers

In these classifiers, a single cluster-based learning model is built which is incrementally update during the stream progress.

Early algorithms like [79,142–144] work under the framework of one-class classification. Spinosa et al. [142–144] proposed a technique called OLINDDA. Initially, OLINDDA created a learning model by clustering the training data of a single class into k clusters. Afterward,

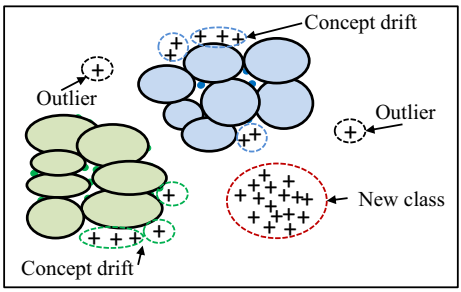


Fig. 6 Working diagram of cluster-based algorithms. Here, we have two classes (blue and green) and their corresponding cluster boundaries. Test instances (+) that fall inside the cluster boundaries are classified as an existing class instance. Similarly, other instances falling outside are further classified as outliers, drift in the existing classes and novel pattern

incoming stream instance which is outside the cluster boundaries is stored in a buffer. For novel class detection, the instances in the buffer are first divided into k clusters. Any new cluster that meets the conditions for cohesiveness and representativeness is considered a valid cluster. Cohesion (like cluster density) is used to measure the degree of likeness between instances of a cluster, while representativeness is a minimum of instances that belong to a cluster. Once the valid clusters are identified, OLINDDA then creates a macro-hypersphere that is defined by the center and radius. The center of the macro-hypersphere is the center of all clusters of the normal model, and the radius is determined by the distance between the center of the macro-hypersphere and the farthest cluster centroid. For all validated clusters, if their centers are in a macro-hypersphere, they are considered an extension of the normal model. Otherwise, they are considered novelties. All valid clusters are then added to the learning model.

Hayat et al. [79] proposed a technique called DETECTNOD. It is also a single class classification model. In this technique, however, after creating clusters, each cluster is subdivided into sub-clusters. For every sub-cluster, total instances are set to the same number by using interpolation. To further reduce memory usage, the DCT is applied to each sub-cluster and only a few DCT coefficients approximating the original data are retained. For each incoming stream instance, the closest sub-cluster closest in the model is selected and an unknown score is calculated. Which is the difference between DCT coefficients before and after merging the instance with the closest sub-cluster. Instances for which the score is greater than a threshold are marked as unknown and moved to a buffer. For emerging class identification, instances in the buffer are clustered, the number of instances is set to the same value by interpolation, and the DCT is applied to each cluster. The similarity between the candidate clusters and their closest sub-clusters in the model is then calculated. If the similarity is lesser than a threshold, the cluster is identified as a novel, otherwise, it is considered as an extension in the normal concepts. To avoid increasing the memory usage, sub-clusters that are not used for a long time are replaced by new clusters.

Faria et al. [60,63] proposed a technique called MINAS for classifying multiple classes. In this scheme, the initial set of training dataset is first divided into subsets, each subset containing the instances of a single class. Then, the data in each subset are clustered and a summary of each cluster, such as the center point, the radius and the label, indicating each class to which it belongs is stored in micro-cluster. Each instance that falls within the micro-cluster boundary receives the label of the nearest cluster, while instances falling outside are clustered and clusters with fewer instances than the given threshold are discarded. New clusters that are near (based on distance) to the existing micro-clusters considered concept drift in known classes and receives the labels of their closest micro-clusters in the model. Otherwise, the new clusters are declared as a novel class. Each valid cluster is then added to

the current model. After a specified period, the micro-clusters that do not receive a sample are removed from the model and stored in a sleeping memory. Similarly, the work presented in [85] is an extension of MINAS algorithm and proposed a multi-label learning algorithm for streaming data. Another technique similar to MINAS is also proposed in [71].

An extension of the MINAS algorithm [63] is proposed in [44], called FuzzND. This method uses the Fuzzy C-Means (FCM) clustering algorithm to create a set of supervised fuzzy micro-clusters (SFMiC) defined in [45]. For each incoming instance, the membership value is calculated for all micro-clusters in the current model. Membership values for each class micro-clusters are added and stored as a class compatibility value. Instances with maximum class compatibility higher than a threshold are classified as existing class instances, while others are stored in a buffer. To discover novel class, the buffered instances are first clustered with FCM. A fuzzy silhouette coefficient [35] is then calculated for each new cluster. New clusters whose silhouette coefficient is less than zero or whose number of samples is smaller than a certain threshold value are discarded, but their samples remain in memory. For all new valid clusters, a fuzzy cluster similarity [154] between new clusters and all existing clusters in the model is calculated. If the similarity is greater than a predefined threshold, the new clusters are tagged with the label of maximum similarity cluster. Otherwise, the clusters are declared a new class. FuzzND deletes the clusters that have not been updated for some time and removes old instances from the buffer.

CluMC [155] and McStream [156] algorithms are based on DenStream [36] clustering algorithm and maintain the summary of the data stream in two types of micro-clusters: outlier and potential micro-clusters. Initially, several p-micro-clusters are created according to DenStream and the class label to each p-micro-cluster is assigned based on the majority class of this cluster while outlier-buffer is empty. Test instances falling inside labeled p-micro-clusters get their labels. During the online data maintenance, if the test instance is added into an o-micro-cluster and its weight becomes greater than or equal to a given threshold value, then the cluster is inserted in the potential buffer. Afterward, the DBSCAN [59] algorithm is applied to the entire potential-buffer. If the new p-micro-cluster is density-reachable from other labeled micro-clusters, it is considered an extension of the known concept and gets the label from the nearest micro-cluster. Otherwise, it is considered to belong to a new class. Finally, the weights of all micro-clusters will gradually decrease if no example is added. After a specific period, each micro-cluster, if its weight is below the given threshold, is removed from the buffer.

The framework proposed in [131] maintains two windows containing the instances of the previous batch and the new batch. First, a change detection algorithm E-CUSUM [11], is applied to combined data from two windows. If a significant change is detected, the data of the current batch are divided into k and $k + 1$ number of clusters by the k-means clustering algorithm. Here, k is the number of known classes of the last batch. The resulting clustering outputs are then inputted to a silhouette function defined in [159]. The silhouette function computes a coefficient value for each instance by means of the within and between cluster dissimilarity. Then, the sum of the coefficients for all the instances achieved by each partition is compared, and the cluster partition for which the sum of the silhouette coefficients is larger is considered as the actual number of clusters. All the instances of current batch centroids receive the label of the closest pair of centroids in the previous batch. If the number of clusters in the previous batch is less than the current number of clusters, this means that one or more clusters are left unpaired. Unpaired clusters are then assigned an unknown label. On the other hand, if no substantial change is detected, instances of the current batch are classified.

The technique proposed in [162] called SVSCLASS keeps the decision boundaries in the kernel space for each class in the stream. For each class, a set of spheres is created and

managed using SVDD (description of the supporting vector domain) [147]. For each sphere, information like support vectors, bounded support vectors, a center, a radius and a maximum distance between the bound support vectors and the center of the sphere are stored. The instances which lie inside the spheres of the existing classes are given the class labels of that spheres. All other instances outside the spheres are first clustered by SVC (support vector cluster) [22], and then a neighborhood graph is created to recognize new class instances, as described in [160]. Finally, after obtaining the labeled block, the current classifier is updated by shrinking, enlarging, or creating new spheres.

Similar to SVSCLASS [162] algorithm, the work proposed in [161] called ONCLAD keeps a set of support vectors (boundary list) for each class observed. These support vectors are extracted by using the General Support Vector Representation Machine (GSVRM) [34]. Whenever a new test block arrives, some instances are generated for each existing class using their boundaries. Then, an extended version of the agglomerative fuzzy k-means algorithm [94] is used to divide the generated instances and newly arrived instances into clusters. Clusters with most instances generated from a single class are termed as known clusters and unlabeled instances in those clusters are assigned the label with the majority class. For clusters where most instances are unlabeled, they are declared as new if the number of instances in this cluster is higher than a particular threshold value. Clusters containing very few instances and semi-known clusters, the outcomes of the agglomerative method are examined in the reverse direction, where the cluster is merged with other clusters. The current decision model is updated upon receipt of the actual labels by modifying the support vector lists.

Similarly, another micro-clusters-based k-NN method is proposed in [49], called EMC. In this technique, after creating a set of clusters, summary information such as the linear sum, the sum squared, the covariance matrix of each cluster is stored in micro-clusters. EMC uses an error-based method described in [137], and the decay function to learn the importance of each micro-cluster over time. This allows EMC to dynamically maintain micro-clusters by inserting new ones or delete obsolete micro-clusters. This technique uses the local density method with the Local Outlier Factor (LOF) [31] algorithm to discover a novel class. For this, some instances for each class in the current model are generated using the mean and covariance matrix stored in each micro-cluster. Outlier instances with a local density similar to their neighbors in known classes are declared as the existing class instances, while remaining outlier instances are declared as new class instances if the total number of instances is greater than the given threshold.

4.1.2 Clustering-based ensemble classifiers

Ensembles classifiers are naturally suitable and can easily be used in a streaming data environment. Because the construction of the ensemble is modular, a new component can easily be added to the ensemble [74]. They also have the capability to modify an existing member of the classifiers or change the weights of each classifier during the voting combination. The cluster-based ensemble classifier is divided into the chunk-based ensemble and a class-based ensemble.

Chunk-based ensemble In these approaches, the data stream is processed chunk by chunk. During the warm-up phase, the first L labeled chunks are used to create an ensemble of classifiers, that is, $E = \{C_1, \ldots, C_L\}$. Here, E is an ensemble and C_i are individual cluster-based classifiers. Only one classifier C_i is trained on each chunk. In the online phase, the ensemble is continuously updated by adding new classifiers and removing outdated ones. Figure 7 depicts the working of chunk-based ensemble classifiers.

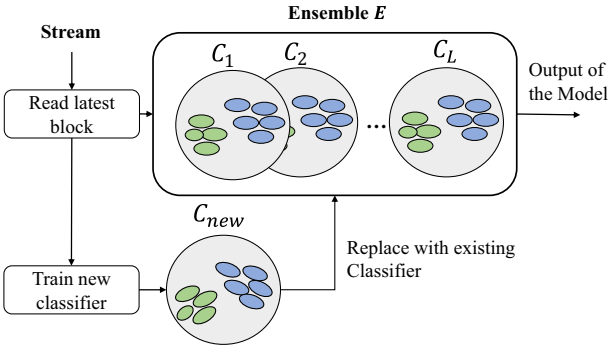


Fig. 7 Chunk-based ensemble training and update procedure

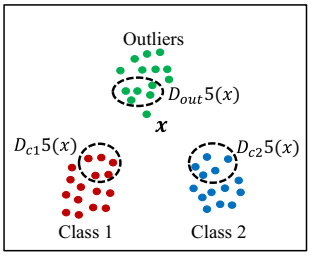


Fig. 8 Illustration of q-NSC for $q = 5$. Here, $D_{C_{\min}}, q(x) = \min(D_{C_1}, q(x), D_{C_2}, q(x))$

Masud et al. proposed MineClass [109] and ECSMiner [104] ensemble classifiers where each classifier C_i is a cluster-based k-NN classifier. After creating clusters with the k-means clustering algorithm, summary information (called pseudo-points) for each cluster is stored. Test instances falling inside the decision boundary of the ensemble are classified by taking the majority of the votes in the ensemble. For all outliers instances, a measure defined in Eq. (2) (called q-silhouette coefficient or q-NSC) is calculated.

$$q\text{-NSC}(x) = \frac{D_{C_{\min},q(x)} - D_{C_{\text{out}},q(x)}}{\max\{D_{C_{\min},q(x)}, D_{C_{\text{out}},q(x)}\}}. \tag{2}$$

Here, $D_{C_{\text{out}},q(x)}$, $D_{C_{\min},q(x)}$ is the average distance between an outlier x and its q-closest outliers and existing class instances, respectively (see Fig. 8 for illustration). According to the definition, an instance with a positive value of q-NSC means that x is closer to its outliers and away from an existing class. The q-NSC measure is calculated for each outlier and for each classifier in the ensemble. A novel class is declared if the number of outliers for which the value of q-NSC is positive is greater than a given threshold value. The work presented in [16], called ENCD, is based on MineClass [109] and uses an extra condition for creating a new classifier. In the MineClass, a new model is trained only after the most recent data block is completed. But in [16] whenever a new class is identified during the test chunk and a flag is set, a new classifier is trained and added in the ensemble. This algorithm does not wait for the test chunk to be completed to create a new classifier.

The techniques proposed in DECSMiner [106] and SNODSOC [2] also use the same strategy to build a cluster-based ensemble classifier as described in MineClass [109]. Additionally, these strategies provide a mechanism to handle feature evolution in the data streams. These techniques use two methods for feature extraction and selection: predictive and informative feature selection. In predictive feature selection, the feature set for incoming examples

is predicted in a supervised way (e.g., using information gain) from the previously labeled examples, while in informative feature selection, a test chunk is used to select the feature set in an unsupervised way (e.g., highest frequency features). Finally, the authors proposed a homogeneous feature space conversion method (called "lossless homogenization") in which the classifier and the test data convert their features sets to the union of their sets before classification.

Similar, another family of MineClass [109] algorithm is presented in [107,108] under the name MCM. In these techniques, various improvements have been proposed with respect to the existing algorithm. For example, in MineClass, the decision boundary of each classifier is fixed, while in MCM, a dynamic slack space outside the decision boundary of each classifier is defined to improve the outlier detection. For novel class identification, MCM proposes a new technique based on discrete Gini Coefficient. In [107], for all instances that are outside the slack space, the measure q-NSC defined in Eq. (2) is computed. If the value of q-NSC for an instance is negative, this instance is considered an existing class instance and is removed from the buffer. Then, an another measure called N-score(x) defined in Eq. (3) is computed for each instance with a positive score for q-NSC.

$$N\text{-score}(x) = \frac{1 - W(x)}{1 - \min W} q\text{-NSC}(x) \tag{3}$$

where $W(x) = e^{r-d}$ and r is the radius of the cluster closest to the instance x, d is the distance between x and the center of the nearest cluster, and $\min W$ is the minimum weight between all outliers with q-NSC positive value. To identify new class instances, the values of N-score(x) are discretized into n intervals. The cumulative distribution function (CDF) of N-score(x) and the discrete Gini coefficient are then calculated. A threshold is then determined to separate the new class instances. If the value of the Gini coefficient is greater than $\frac{n-1}{3n}$, all instances of the buffer are declared as new class instances. On the other hand, if the Gini coefficient is zero, all instances classified as the existing class instances. Finally, if the Gini coefficient is between [0, $\frac{n-1}{3n}$], the instances of the first interval are deleted, and the remaining instances are declared as belonging to a new class. MCM can identify multiple new classes at the same time by creating a graph of instances marked as new, and then searching for connected components in the graph, where each connected component represents a new, separate class, while in [108], after detecting and saving outlier in the buffer, the instances of the buffer are clustered using k-means to speed up the calculation of the q-NSC value. Subsequently, summary information (called O-pseudo point h) such as center point ($h.u$), Cluster size ($W(h)$) and radius is stored for each cluster. The q-NSC for an O-pseudo-point is calculated as follows.

$$q\text{-NSC}(h) = \frac{D_{C_{\min},q(h)} - D_{C_{\text{out}},q(h)}}{\max\{D_{C_{\min},q(h)}, D_{C_{\text{out}},q(h)}\}}. \tag{4}$$

Here, $D_{C_{\min},q(h)}$ is the minimum weighted average distance from O-pseudopoint h to q nearest pseudopoint of the existing class e. Which is given by:

$$D_{C,q(h)} = \frac{\sum_1^m W(e_i) D(h, e_i)}{\sum_1^m W(e_i)} \tag{5}$$

where $D(h, e_i)$ is centroid to centroid distance. Similarly, the mean distance $D_{C_{\text{out}},q(h)}$ from h to other O-pseudopoint is given by:

$$D_{C_{\text{cout}},q(h)} = \frac{W(h)h.u + \sum_1^{r-1} W(h_i) D(h, h_i)}{W(h) + \sum_1^{r-1} W(h_i)}. \tag{6}$$

Finally, the N-score defined in Eq. (3) is computed for all O-pseudo-points having a positive value. Then using the Gini coefficient and constructing a graph, multiple novel classes are identified. In addition, this algorithm also adopted the technique proposed in DECSMiner [106] to handle feature evolution in the data streams.

An extension of MCM [107] algorithm is proposed in [58] which introduces a technique to reduces the risk of false alarms from [107]. To further purify, the declared novel class instances are clustered using k-means, and the value of the number of clusters is chosen in a way to overestimate the actual number of clusters. After creating clusters, the density (ratio between the number of instances in the cluster and the average distance between them) of each cluster is measured. The cluster with low density or whose instances are below the given threshold value is eliminated. Similarly, another version of MineClass [109] and MCM [107] is also presented in DTNC [111]. In this paper, the authors proposed that using Very Fast Decision Tree (VFDT) [54] as base learners and k-prototypes++ as the clustering algorithm significantly improves the performance of MineClass. k-prototypes++ is a combination of two methods: K-means++ [15] and K-prototypes [82].

The work proposed in SCANR [105] consists of two ensembles, a primary and an auxiliary ensemble. The primary ensemble is created in the same way as described in MineClass [109], while the auxiliary ensemble is created for each class by creating a set of clusters and storing the clusters summaries. The auxiliary ensemble is maintained to remember each class in the data stream. Test instance falling inside the primary ensemble boundaries are classified by taking the majority voting among the classifiers. Test instances outside the primary ensemble boundaries are sent to the auxiliary ensemble for outlier detection. If the auxiliary ensemble declares the instances as outliers, then the technique of MineClass [109] algorithm is adapted to identify the novel class. Otherwise, the auxiliary ensemble considers them as the existing class instances.

In addition to the supervised algorithms, some techniques such as [69,75–77,110,169] use semi-supervised learning algorithms to build the classification model.

The technique presented in [110] called ActMiner extends MineClass [109] and proposes an active learning-based semi-supervised algorithm. During the classification, ActMiner selects the most uncertain (weakly classified) instances and requests the user to provide actual labels for them. The measurement of uncertainty is based on two factors. A test instance is considered the most uncertain if it is declared an outlier or if the ratio between the majority votes and the total votes is lower than a certain threshold value called the minimum majority threshold.

Other techniques SCDMiner [75], SAND [76] and ECHO [77] use a similar method as described in ECSMiner [104]. ECSMiner maintains an ensemble of classifiers on data blocks with a fixed size, while these approaches maintain a dynamic window, which stores the classifier's confidence score behind predicting the label of a test instance. This window, with the confidence scores, is then used to detect a change in the confidence of the classifier over time. If a significant change in the classifier's confidence is detected, a change point is determined in the current window. Data after the change point is kept while old data is removed and the size of the window is reduced. The true labels for the instances of the current window where the classifier showed weak confidence in the prediction are requested from the user. For the rest of the instances, predicted labels are used as final labels and include them in the training data. If the change is not significant, the current ensemble is kept unchanged, and the window size continues to grow.

The work proposed by Mustafa et at. [120] trains a Denoising Autoencoder (DAE) by computing the weights (W and b) to extract deep abstract features before training a classification model. The learned weights are kept unchanged and used to transform the original

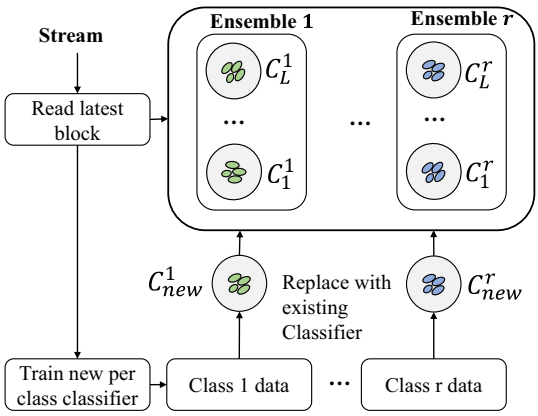


Fig. 9 Class-based ensemble training and update procedure

input features of incoming instances to abstract features. Afterward, a classification model similar to ECSMiner [104] is trained for emerging class identification. A change detection method is also proposed to determine the chunk boundaries dynamically.

The work presented in [70] creates two clustering-based ensembles and integrates them into the MINAS algorithm [63]. Two clustering ensembles are referred to as homogeneous clustering (HoCluS) and heterogeneous clustering (HeCluS) for data Streams. The former ensemble is created by varying the Clustream [4] algorithm parameters to generate different cluster partitions for training data. The later ensemble is created using different clustering algorithms to generate multiple cluster partitions.

Class-based ensemble In the class-based ensemble algorithms, the training data is split into disjoint sets so that each set contains the instances of a single class. A classifier for each class is then created by clustering the data belonging to a single class and saving the summaries of each cluster. For each class, an ensemble of such classifiers is maintained and regularly updated with new data by training new per class classifiers and replacing them in the corresponding ensemble. Figure 9 illustrates the training and update procedure of class-based ensemble classifiers.

The technique proposed by Alkateeb et al. called CLAM [8,9] is a per class ensemble algorithm and uses a k-means clustering algorithm to create a set of clusters for each class. Test instances outside the cluster boundaries for each class ensemble are analyzed for novel classes as described in ECSMiner [104]. To classify an instance, the distance between the instance and clusters of each per class ensemble is measured. The ensemble of class with minimum distance is declared as the class of the instance. Finally, the ensemble is updated upon receipt of a new labeled block by creating new per class classifiers and replacing them with the existing classifiers in the ensemble. In the CLAM algorithm, a per class ensemble is created on different data blocks while in another technique [84], each class ensemble is created on the same data by varying the seed parameters of the k-means algorithm which diversifies each classifier in the ensemble. In this algorithm, after receiving the labels of the most recent block, per class clusters are produced on instances that have been wrongly predicted by the classifier. For each newly created cluster, if the cluster center lies inside the existing cluster, they are merged. Otherwise, it is replaced with the cluster in the corresponding class ensemble that has the worst prediction performance on the latest block.

Similarly, LOCE [160] also maintains a set of cluster-based classifiers (called local classifiers) for each class. The output of these local classifiers is aggregated by a global classifier. Finally, all the outputs of the global classifiers are then aggregated to make the final prediction. Test instances for which the final prediction value is greater than a certain threshold are classified with the class label of the ensemble that generates the maximum value. For the detection of novel classes, a group of instances with low prediction values are first divided into clusters. A novelty score is then calculated for each cluster and clusters with positive scores are isolated from the other clusters. A neighborhood graph is then constructed with cluster centers (non-isolated and isolated) and their nearest centers in the existing ensemble. If an isolated cluster center is liked to a non-isolated cluster center or existing cluster center in the ensemble is removed. A novel class is declared if the instances in the isolated cluster are higher than a threshold.

The technique proposed by Siahroudi et al. [138] creates and maintains an ensemble of combined kernels for each class. Each combined kernel consists of a static and a dynamic kernel. Only one static kernel is created for each class and is shared with the other combined kernels in the ensemble of the same class but can have a different weight, while for every upcoming block, a new dynamic kernel is created and combined with the static kernel. This newly created combined kernel is then replaced by an existing combined kernel that has not been used for a long time. Each combined kernel stores two values, namely the center of the corresponding class and a confidence distance. For each instance of the upcoming block, a new feature vector is first extracted, then a distance between the instance and all other classes is calculated. If the distance is lower than the reliability distance of the closest class, the instance receives the label from the nearest class. All other instances whose distance is higher than the class reliability are first clustered. Clusters whose size is larger than a given threshold value are identified as new class instances. Finally, the remaining instances are marked as outliers of the existing classes and classified using the minimum distance between the instance and all existing classes.

Another class-based ensemble approach is proposed by Abdallah et al. [1]. In this method, clusters are formed in two steps. The initial training data is first divided into separate classes, and then clusters are created for each class. In the second step, the clusters created in the first step are divided into smaller sub-clusters by applying the EM algorithm. Each sub-cluster represents a different pattern within the cluster. All new concepts (clusters) that are outside of all existing clusters and continuously move away from all existing clusters are declared as new class clusters. A forgetting mechanism is also implemented to remove old clusters (concepts) that are no longer compatible with recent concepts. To reduce labeling costs, the authors also proposed an active learning method. Finally, the summary of cluster-based algorithms is given in Table 1.

Similarly, the work presented in [33] extends iNNE [19] algorithm and creates a per class instance-based ensemble classifier. Each instance in the ensemble represents an isolation hypersphere defined by its radius. The radius of the hypersphere is determined by the distance between the instance and its nearest neighbor of the same class. For every test instance, if the distance between its nearest neighbor in each class ensemble is greater than a certain threshold it is declared new: otherwise, the nearest class is assigned as its label.

4.2 Model-based learning algorithms

Unlike clustering-based techniques, model-based learning techniques focus on finding a model for classification and detecting new classes. Here, different traditional learning models

Table 1 Clustering-based learning algorithms

Algorithm	(CLF)	(LT)	(CL)	(DH)	(RCH)	(OTH)	(FM)	(FEH)	(NCD)	Evaluation measures
OLINDDA [142–144]	S	UN	OC	P	N	Y	N	N	OC	F_{new}, M_{new}, E
DETECTNOD [79]	S	UN	OC	P	N	N	Y	N	OC	F_{new}, M_{new}, E
MINAS [60,63]	S	UN	MC	P	Y	Y	Y	N	MC	CM
MINAS-BR [85]	S	UN	MC	P	Y	Y	Y	N	MC	CM, F1
FuzzND [44]	S	UN	MC	P	N	Y	Y	N	MC	CM
CluMC [155]	S	UN	MC	P	N	Y	Y	N	MC	P, R, A
McStream [156]	S	UN	MC	P	N	Y	Y	N	MC	P, R, A
DoubleStageDetector [131]	S	UN	MC	P	N	N	Y	N	OC	A
Higia [71]	S	UN	MC	P	N	N	N	N	MC	F_{new}, M_{new}, E
SVSCLASS [162]	S	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
ONCLAD [161]	S	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
EMC [49]	S	SP	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
MineClass [109]	E	SP	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
ECSMiner [104]	E	SP	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
ENCD [16]	E	SP	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
DECSMiner [106]	E	SP	MC	F	N	Y	Y	Y	OC	F_{new}, M_{new}, E
SNODSOC [2]	E	SP	MC	F	N	Y	Y	Y	OC	F_{new}, M_{new}, E
MCM [107]	E	SP	MC	F	N	Y	Y	N	MC	F_{new}, M_{new}, E, AUC, F1, P, R
MCM2 [108]	E	SP	MC	F	N	Y	Y	Y	MC	F_{new}, M_{new}, E, AUC, F1, P, R
IMCM [58]	E	SP	MC	F	N	Y	Y	N	MC	AUC, P, R
DTNC [111]	E	SP	MC	F	N	Y	Y	N	MC	F_{new}, M_{new}, E

Table 1 continued

Algorithm	(CLF)	(LT)	(CL)	(DH)	(RCH)	(OTH)	(FM)	(FEH)	(NCD)	Evaluation measures
SCANR [105]	E	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
ActMiner [110]	E	SM	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
SCDMiner [75]	E	SM	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E, F1
SAND [76]	E	SM	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
ECHO [77]	E	SM	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E, F1
NovelDetector [120]	E	SM	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
AnyNovel [1]	E	SM	MC	F	N	Y	Y	N	OC	F_{new}, M_{new}, E
HeCluS, HoCluS [70]	E	UN	MC	P	Y	Y	Y	N	MC	CM, F1
CLAM [8,9]	E	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
RNCDE [120]	E	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
LOCE [160]	E	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
Mukers [138]	E	SP	MC	F	Y	Y	Y	N	OC	F_{new}, M_{new}, E
SENNE [33]	E	UN	MC	P	N	N	N	N	OC	F_{new}, M_{new}, A

(CLF), Number of classifiers; E, Ensemble; S, Single; (LT), Learning task (online phase); SP, Supervised; UN, Unsupervised; SM, Semi-supervised; (CL), Number of classes; MC, Multi-class, SC, One Class; (DH), Drift Handling; F, Full-support; P, Partial-support; (RCH), Recurring contexts handling; (OTH), Outliers handling; (FM), forgetting mechanism; (FEH), Feature evolution handling; N, No; Y, Yes; (NCD), Number of novel classes detected; OC, One-class; MC, Multiple classes; E, Error; CM, Confusion matrix; F1, F-measure; P, Precision; R, Recall; A, Accuracy

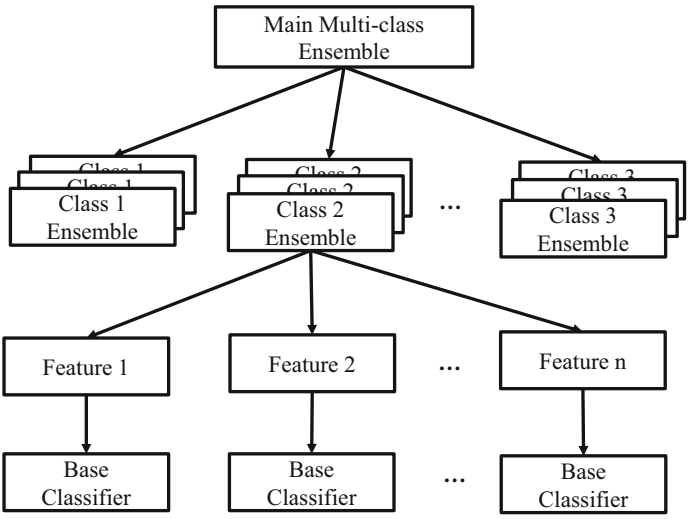


Fig. 10 Hierarchy-based ensemble classifier architecture

are adopted by different techniques and these models can be divided into the following categories.

- Hierarchy-based ensemble algorithms
- Decision tree-based algorithms
- Graph-based algorithms
- Sketch-based algorithms
- Support Vector Machine (SVM) and Neural Network (NN) based algorithms.

4.2.1 Hierarchy-based ensemble algorithms

Parker et al. [125,126] proposed a hierarchy of ensemble classifiers called HSMiner and SluiceBox, respectively. These methods built a hierarchy of ensemble classifiers. At the lowest level of the hierarchy, there is a base classifier for each attribute, and at a higher level, a per class ensemble is assigned. To classify an instance, it is passed down the hierarchy of ensemble and each base classifier at the bottom of the hierarchy cast a weighted vote in a continuous range $[-1, +1]$. The top tier ensemble then aggregates the votes into a final score. In HSMiner, two methods are proposed for emerging class identification. The first method is naïve and considers each instance as an emerging class whose final score is lower than a given threshold, whereas in the second method, a new single-class classifier is trained using instances with a "new" tag. Now, this new classifier is used to vote for every instance of the block. Instances whose final vote is above the threshold value are marked as "new." This process is repeated several times, and ultimately, instances marked as "new" are considered as new class instances and the remaining instances from the existing classes, while SluiceBox uses clustering to detect new classes described in [109]. Similarly, another technique based on the SluiceBox algorithm is presented in [127], called SluiceBoxAM (AnyMeans). SluiceBoxAM can discover clusters with non-spherical shapes and can also be used with other classifiers, for example, with leveraging bagging [26]. The architecture of the hierarchy-based ensemble classifiers is shown in Fig. 10.

4.2.2 Decision tree-based algorithms

The technique presented by Farid et al. [64] is based on a decision tree classifier. After creating the decision tree from the training dataset, the instances of each leaf node are clustered. A threshold value is then calculated, which is a ratio between the total number of instances in the leaf node and the total number of instances in the training set. For each test instance, if its addition to the leaf node increases the threshold value and outside all clusters of the leaf node, it is considered an emerging class instance. An extension of this work is proposed in [65], which creates an ensemble of three decision trees (DT). In this algorithm, first, a naive Bayes classifier is used to weight the initial training instances with the highest posterior probability. After weighing the instances, the first DT is built using selection and replacement techniques in the weighted training set, while the other two DTs are built using the highest weighted instances, which is calculated based on their prediction accuracy of the first DT. Weight is also associated with each DT based on the accuracy of its predictions on the training set. If a test instance is outside of all clusters corresponding to the leaf nodes of all the trees it reaches, it is considered an outlier and placed in the buffer. In addition, a new flag is set to 1. Otherwise, weighted majority voting is used by the ensemble to classify the instance. When the value of the new flag is 1 and the ratio of the incoming instances classified by a leaf node increases or decreases relative to the previously calculated threshold value, the instances are considered as new class instances. When a labeled data block is available, a new DT classifier is formed and replaced by the low-precision DT.

The method proposed in [116], called SENCForest, is based on an unsupervised anomaly detector iForest (isolationForest) [97]. An iForest is an ensemble of randomly generated decision trees using a subsample randomly selected in the training dataset. After building an iForest, a threshold called path length is used to divide the data space into two regions, normal and anomaly regions. The path length is the average number of edges traversed by the training examples from the root node to a leaf node. Training instances where the path length is less than the threshold define the normal region. The anomaly region is again subdivided into two regions, the anomaly of known classes and the outlying region (see Fig. 11 for illustration). Test instances that fall inside the outlying region are considered instances of new classes and stored in a buffer. After the construction of the new class detector, the class labels of the known classes are recorded in the normal and anomaly sub-region of known classes. Each test instance that falls inside the normal or anomaly sub-region of known classes is classified by taking majority class labels in that region. In the proposed technique, the model is updated in two ways. First, the current model is updated by growing a new subtree in an existing tree if the number of instances in the leaf node exceeds a certain limit. Otherwise, the statistics of the leaf node are updated; that is, the frequency of each class is updated to include the new class. In the second method, if the number of classes that a SENCForest can handle reaches a pre-defined threshold, a new SENCForest is trained and added to the model.

The technique proposed in [171,173] called MuENL is a multi-label classification algorithm with emerging label detection. For classification, MuENL trains a linear classifier for each label. To provide a better performance, they use label correlations among multi-labels where a pairwise label ranking loss is minimized. For novel label detection, MuENL extends iForest [97] algorithm with clustering to identify new labels. Instances that are outside the clusters in leaf nodes are placed in a buffer. When the buffer reaches a pre-defined limit, both classifier and detector are updated.

Similarly, another multi-label classification algorithm is proposed in [118] called NL-Forest. The framework of NL-Forest consists of two models: an instance-based model (I-F)

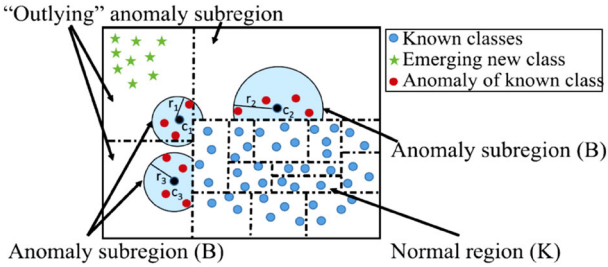


Fig. 11 Illustration of normal and anomaly regions defined in [116]

and a label-based model (L-F). The former is created for the entire training set, while the latter is created for each label that is present in the data. Both models are created using random decision trees like iForest [97]. This algorithm can detect two types of emerging labels, namely an absolutely new label or a partially new label. For absolutely new label detection, the same technique is used as described in [116], while for partially new label detection, the test instance is first passed to the I-F, where the probabilities of the known labels are generated as a label vector. The vector is then arranged in descending order of known labels. Depending on the order of the labels, the test instance is then passed on to L-F, where the height of the test instance is measured from each sub-forest in L-F. If the average height of the instance is less than a threshold value, it is considered new. Finally, both models are updated when a group of novel labels is found.

Ting et al. [146] proposed an ensemble of half-space trees (HS) for one-class classification. An HS tree is a complete binary tree in which all leaf nodes are at the same depth. It works by maintaining two consecutive windows of data; one is called the reference window, and the other containing the most recent instances. During the initial learning, an ensemble of HS trees is induced by learning the mass profile (i.e., the number of instances reaching the tree nodes) of the data in the reference window (called mass r). For each instance of the incoming block, it is passed down to the hierarchy of each HS tree of the ensemble, and an accumulated score called anomaly score is computed. If this anomaly score is higher than a threshold value, the instance is considered anomalous. Otherwise, it is considered as belonging to the normal class. From every new recent window, a new mass profile is computed that replaces the old profile in the reference window.

4.2.3 Graph-based algorithms

In graph-based models [29,30], a graph topology G (see Fig. 12) is built with instances as nodes and edges serving as a connection between different nodes. This graph is regularly updated by creating new nodes and connections between nodes as new data arrives. Each node of the graph represents the center of a hypersphere defined by its radius. All these hyperspheres cover the feature space and used to identify emerging classes as well as classify incoming instances as the existing classes. In [29], after receiving a block of instances, first, the instances are separated into two sets. The ones locating inside the hyperspheres as insiders and the ones which fall outside as outsiders. Meanwhile, a graph $\hat{G}$ is also constructed from the recent block. Then, for each instance, x in the new block, an average distance of x from its nearest nodes in G, and $\hat{G}$ is computed. Based on these values, a probability score of each

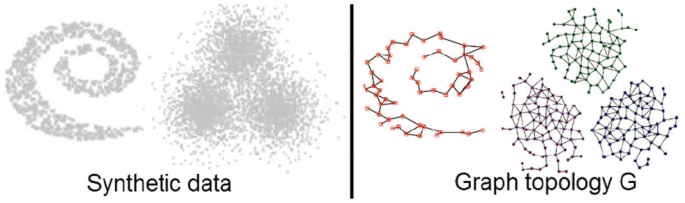


Fig. 12 Left: synthetic data. Right: graph topology G [29]

instance x is calculated as follows.

$$P(N|x) = \frac{d_x^E}{d_x^E + d_x^N} \tag{7}$$

$$P(E|x) = \frac{d_x^E}{d_x^E + d_x^N}. \tag{8}$$

Here, d_x^E, d_x^N are the means distances from G and $\hat{G}$. If x belongs to a novel class then $P(N|x)$ will be greater than $P(E|x)$). Afterward, an iterative procedure is adapted to refine further the set of insiders and outsiders, which may contain false positive or false negative instances. Insiders are then classified by nearest label nodes. It also requests for true class labels of weakly classified instances, while the technique proposed in [30] also provides an adaptation mechanism to cope with local changes in the feature space where the data distribution is changed. Additionally, an adaptive forgetting mechanism is also implemented to identify and remove irrelevant or obsolete nodes from the graph. Finally, a probabilistic evolutionary mechanism creates new neurons when it is necessary to represent data in new areas of the feature space. A test instance is considered a novel instance if the distance between the instance and its nearest node in the graph is greater than the average length of the current set of edges in the graph.

4.2.4 Sketch-based algorithms

The techniques presented in [117,167] are based on the matrix sketching technique described in [96]. A sketch of a matrix is a small matrix that approximates the very large original matrix. The calculations on the sketch matrix can be performed without much loss of precision. These algorithms exploit the matrix sketching method and create two low-dimensional matrices (called Global and Local) sketches for streaming data. The "Global Sketch" (GS) is created on the entire training set, while "The Local Sketch" (LS) is created for each class in the data. Figure 13 illustrates the process of building both sketches.

For each incoming instance x, a similarity between x and each row of matrix GS is measured by taking the indoor product. If the similarity exceeds a threshold value, the instance is flagged as a potential emerging class instance and placed in a buffer. Otherwise, LS is used to classify the instance into one of the known classes. For this, the inner product between x and each local matrix sketching LS_i for each class is calculated. When the buffer reaches a predefined limit, first, the instances in the buffer are sorted in ascending order based on the similarity values. Afterward, the list is subdivided into two sub-lists by searching a split point. The best split point between the two sub-lists is searched according to the following criteria.

$$\tau = \text{argmin}_\tau |\sigma(V_{left}) - \sigma(V_{right})|. \tag{9}$$

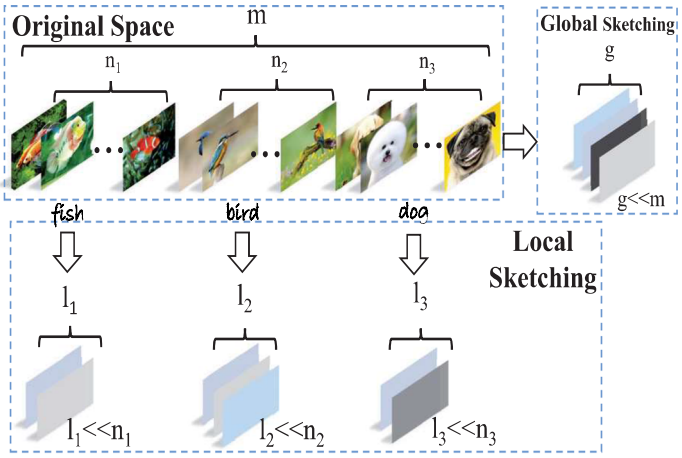


Fig. 13 Global and local matrix sketching to approximate very large original matrix [117]

Here, is $\sigma(.)$ is a standard deviation and τ is the best point. τ is the point that minimizes the standard deviation between the two sub-lists. After that, the GS and the LS are updated, and the buffer is reset. However, in [167], before building the sketches, a neural network is trained only with the initial training set for extracting features for each instance. Then with the extract features, the local and global sketches are built, while during online learning, the weights of the neural network are fixed. All the incoming data are first processed through the neural network, which extracts the features for each instance and then fed into the next module. It also used CFS (clustering by fast search) [132] algorithm to find multiple novel classes.

4.2.5 SVM and NN based algorithms

Krawczyk et al. [90] proposed a one-class classification algorithm based on WOCSVM (One-class weighted support vector machine) [24]. WOCSVM is based on OCSVM [134] and assigns a weight to each instance in the training data. These weights are used to minimize hypersphere volume, but still contain all the training data. After creating an initial WOCSVM classifier, incoming instances are classified as normal if they fall into the hypersphere. Otherwise, the instances are declared as a novel pattern. In this algorithm, two approaches are proposed to implement the forgetting mechanism. In the first approach, instance weights are decreased gradually over time, and instances with a weight of 0 are discarded. In the second approach, all instances of the old block are deleted when a new data block is received without considering the initial importance of the weights.

The work proposed by Rusiecki et al. [133] is also a one-class classification algorithm. This technique is based on a neural network (NN) and trains two autoregressive feedforward NN. To train the first NN, a robust learning technique is used that tries to minimize the error by reducing the influence of outliers on the training set. The second NN is trained using the traditional back-propagation technique that minimizes the least-squares error. These NNs are trained on a window with a predefined length that contains the most recent instances and moves in discrete periods. According to the authors, updating the NN for every incoming instance is computationally expressive. Therefore, a network is trained only once for a given period. When data of a predefined length is collected, it is then used to update the weights

Table 2 Model-based learning algorithms

Algorithm	(CLF)	Learning model	(LT)	(CL)	(DH)	(RCH)	(OTH)	(FM)	(FEH)	(NCD)	Evaluation measures
HSMiner [125]	E	Hierarchical	SP	MC	F	Y	Y	Y	Y	OC	E
SluiceBox [126]	E	Hierarchical	SP	MC	F	Y	Y	Y	Y	OC	A, AUC
SluiceBoxAM [127]	E	Hierarchical	SM	MC	F	Y	Y	Y	Y	OC	M_{new}, F_{new}, A
DT Novel Class [64]	S	Decision Tree (DT)	SP	MC	P	N	N	N	N	OC	M_{new}, F_{new}, E
DTE Novel Class [65]	E	DTs	SP	MC	F	N	N	Y	N	OC	M_{new}, F_{new}, E
SENCForest [116]	E	Random DTs	UN	MC	P	N	Y	Y	N	OC	A, F1
HS-Trees [146]	E	Half-Space Tree	UN	SC	P	N	N	Y	N	OC	AUC
A2INGN [29]	S	Graph-based	SM	MC	P	N	N	N	N	OC	A, F1
GNG-A [30]	S	Graph-based	UN	MC	P	N	N	Y	N	OC	A
SENC-MaS [117]	S	Sketch-based	SM	MC	P	Y	N	N	N	OC	A, F1
AMaSC [167]	S	Sketch-based	SM	MC	P	Y	N	N	N	MC	A, F1
Adaptive WOCSVM [90]	S	SVM	SP	SC	F	N	N	Y	N	OC	A
NN for NC [133]	S	Neural Network	SP	SC	P	N	N	N	N	OC	2D-plot*
CPE [152]	S	Neural Network	SM	MC	F	N	Y	Y	N	MC	M_{new}, F_{new}, A
MuENL [171,173]	E	Random DTs and Linear classifier	UN	MC	P	N	N	Y	N	MC	A, F1
NL-Forest [118]	E	Random DTs	UN	MC	P	N	N	Y	N	OC	P, RL, F1

(CLF), Number of classifiers; E, Ensemble; S, Single; (LT), Learning task (online phase); SP, Supervised; UN, Unsupervised; SM, Semi-supervised; (CL), Number of classes; MC, Multi-class; SC, One Class; (DH), Drift Handling; F, Full-support; P, Partial-support; (RCH), Recurring contexts handling; (OTH), Outliers handling; (FM), forgetting mechanism; (FEH), Feature evolution handling; N, No; Y, Yes; (NCD), Number of novel classes detected; OC, One-class; MC, Multiple classes; *(dataset x new class identification result); RL, Ranking Loss; E, Error, CM, Confusion matrix; F1, F-measure; P, Precision; R, Recall; A, Accuracy

of the network. Because not all instances are used to train the NN, another parameter is also used that defines the distance between two windows. For each test instance, a response is calculated for both NNs. If the difference between the output of these two networks is below a certain threshold value, the instance is considered a normal class instance. Otherwise, the instance is marked as a novelty. The threshold used to separate novelties from normal data is calculated as follows.

$$Tr = k * std(|y_{mse}(x_i) - y_{lmls}(x_i)|). \tag{10}$$

Here, y_{mse} is the output of traditional NN and y_{lmls} is the output of robust NN, std is the standard difference, and k is a user-defined constant.

The work presented in [152] is a CNN-based Prototype Ensemble (CPE). In this technique, a deep neural network is used to extract a set of prototypes (latent class distribution) for each class in the training set. For each incoming instance, the original input features are transformed into a representation of learned features via the network. The prototype closest to the test instance is then searched. If the instance is close to the existing prototype, it is labeled with the nearest prototype. Otherwise, it is placed in a buffer. When the buffer size reaches a predefined limit, the true class labels of the buffer instances are requested from the user. After obtaining the actual class labels, the prototypes of the new class were extracted and added to the model. The network parameters are also updated. At last, the summary of model-based algorithms is presented in Table 2.

5 Discussion

Although many exciting algorithms for the identification of novel classes in data streams have been proposed, the use of these algorithms in many real-world applications is not feasible due to various limitations. Many problems and challenges are still present and need to be adequately addressed for the development of efficient and effective learning algorithms.

The main limitation of cluster-based approaches is that they use a distance-based heuristic to identify new classes and drift in the existing classes. Defining an optimal distance threshold is a difficult task and depends on the data dynamics. Similarly, the majority of the techniques [8,9,49,58,60,63,71,75–77,79,84,104–109,120,125,131,142–144,160] presented use the k-means clustering algorithm to create the model that requires predefined value of the input parameter (number of clusters). The main assumption of these algorithms is that the underlying data distribution can be represented by hyperspheres. The main drawback is that a fixed number of clusters cannot accurately capture the underlying data distribution, especially data with a non-Gaussian distribution [91]. Another potential problem with cluster-based techniques is that they usually do not produce good results with complex real-world high-dimensional data.

Most approaches [2,7,7–9,16,30,44,48,49,58,60,63–65,71,75–77,79,84,104–109,111,116–118,120,125,126,131,138,142–144,155,155,156,156,160–162,171,173] assume that the emerging pattern(s) are geometrically far from the existing classes in the feature space. This assumption is not realistic in many applications, where the class boundaries are very close to each other. Thus, these approaches cannot handle the classes with complex distribution that can lead to a high degradation in classification performance.

An important problem is related to ensemble-based approaches [8,9,49,104,107–109] that process the data stream in fixed-size chunks. Determining the size of the chunk is a non-trivial task and the performance of these algorithms mainly depends on chunk size. Few techniques

use a change detection algorithm to dynamically determine the size of chunk. However, change detection using limited labeled data is also a non-trivial task [175].

Another limitation is related to supervised algorithms [2,7–9,16,48,49,58,64,65,84,104–109,111,117,125,126,138,155,156,160–162] which need a complete set of labeled data to update / refine their models to deal with concept drift. However, labeling fast and continuous streaming data is impossible in many real-world applications [50]. In contrast, several other algorithms [7,30,33,44,60,63,70,71,79,116,118,131,142–144,155,156,171,173] assume a full set of labels only in the offline phase to induce a model. Afterward, in the online phase, they constantly update their models with unlabeled data. Thus, unsupervised algorithms are not suitable for real concept drift where the target label given the input data ($P(y|x)$) changes without change in the distribution of the data ($P(x)$). Similarly, to reduce dependence on fully labeled data, some techniques [75–77,120] are based on an active learning framework and ask the user to provide labels for limited unlabeled data. These techniques then use limited labeled data along with classifiers own predicted labeled data to update the model. However, the cumulative error can increase if the data is wrongly predicted.

In a dynamic streaming environment, the evolution of features (appearance and disappearance of features) is also a critical problem. Most existing approaches can work with a fixed set of features and apply a new class detection procedure. Few algorithms address this problem, for example, [125–127] has proposed a set of hierarchical classifiers and created a separate classifier for each entity. However, such decomposition increases complexity and may not be suitable in scenarios where the attributes are correlated or irrelevant.

Some algorithms, such as [33,64,65,90,116,133,146], consider any change to be an evolution of the concept in the data. In these techniques, any instance, if not explained by the classifier, is considered a change without checking the cohesiveness between the examples of the new pattern(s). In dynamic streaming data, the existing concepts are constantly changing; therefore, it is essential to distinguish between the actual novel pattern from drift in the existing patterns or the presence of outlier or noise. Similarly, another important problem is determining when to execute the new class detection procedure. In some techniques [49,138,162], a new class detection procedure is performed each time a new block of data is received, while other techniques [29,60,63,70,79,104,142–144] define a fixed time interval or a fixed-size buffer to perform the detection procedure. Here, the main problem is the static values defined by the user for the execution of the detection procedure, the automatic definition of these parameters is still an open problem.

6 Evaluation measures

In static data learning algorithms, several well-known methods (such as leave-one-out, cross-over validation) are available to evaluate the performance of different algorithms. However, these standard methodologies are not suitable/applicable in data streams due to the dynamic or evolving nature of the data stream. For data stream classification algorithms, two methods for performance evaluation are often used: prequential and hold-out. The prequential method is also known as the test-then-train method, where each instance is first used to test and then update the classifier. Finally, the accumulated accuracy is calculated for all instances. In the hold-out method, the data stream is divided into training and test sets. Here, the learning model created on the training sets is tested at regular intervals with the test set.

However, little consideration has been given to the performance evaluation methods for stream classifiers with emerging class detection. Some algorithms have used different static

data evaluation methods, while in others, new ad hoc methods have been developed to evaluate performance.

Algorithms presented in [7–9,16,49,64,65,71,75–77,84,104–111,120,125–127,138,142–144,160–162] use the following metrics to evaluate their performance.

$$M_{\text{new}} = \frac{F_n}{N_c} \times 100 \tag{11}$$

$$F_{\text{new}} = \frac{F_p}{N - N_c} \times 100 \tag{12}$$

$$\text{Error} = \frac{F_n + F_p + F_e}{N} \times 100. \tag{13}$$

Here, F_p indicates the total number of the existing class examples that were wrongly identified as a new class (false positive). F_n represents the total number of examples of emerging classes that are wrongly classified in the existing classes (false negative). F_e is the total number of the existing class examples that have been incorrectly classified, except for F_p. N_c is the total number of examples of emerging classes, and N is the total number of examples in the stream. The measures M_{new} represent the percentage of examples of emerging classes that are misclassified as the existing classes. F_{new} is the percentage of the existing class examples that are misclassified as an emerging class, and $Error$ is the total misclassification percentage.

Similarly, some other methods [10,10,29,30,33,44,58,70,75,77,90,106–108,116,117,126,131,146,167] also use AUC (area under the curves), ROC curves, Precision, Recall, Accuracy and F-measure [139] to evaluate their performance. Accuracy and F-measure is given by the following equations.

$$\text{Accuracy} = \frac{A_n + A_o}{N} \times 100 \tag{14}$$

where A_n is total number of emerging class examples identified correctly, A_o is total number of known class examples classified correctly and N is the total examples.

$$F\text{-measure} = \frac{2 \times P \times R}{P + R} \times 100. \tag{15}$$

Here, P is precision and R is recall.

The evaluation strategy adopted by [44,60,63,70] is based on the incremental confusion matrix as described in [62], which grows incrementally whenever a new example is classified. The confusion matrix is composed of rows and columns representing unknown examples, existing classes and novel patterns. In this method, novel patterns may be considered as an extension of known class or have no association with the known class, i.e., representing a completely new class. It is important to indicate that a new class can be represented by several novelty patterns, or a complete novelty pattern represents a class. In addition, a separate measure is used to evaluate unknown examples, which are given by the following equation.

$$UnkR = \frac{1}{\#C} \left(\sum_{i=1}^{\#C} \frac{\#Unk_i}{\#Exc_i} \right). \tag{16}$$

Here, $\#Exc_i$ is the total number of class C_i examples, $\#Unk_i$ indicates total the number of class C_i examples identified as unknown, and $\#C$ is the total number of classes.

Table 3 False positive (F_{new}) and false negative (M_{new}) rates (%) of all algorithms on different datasets

Dataset	EM	EMC	ECSM	CLAM	ECHO	SAND	SENCF	MINAS
SynC	F_{new}	0.00	0.00	0.00	0.00	0.00	0.46	1.28
	M_{new}	–	–	–	–	–	–	–
SynRBF	F_{new}	0.00	0.08	0.12	0.00	0.50	0.00	2.98
	M_{new}	0.00	54.04	50.64	0.00	0.00	3.34	0.00
FCT	F_{new}	6.57	3.19	5.39	4.95	4.58	17.25	7.23
	M_{new}	10.08	30.35	28.71	23.54	24.91	66.23	86.14
kddcup	F_{new}	2.94	5.41	4.68	3.67	4.08	9.98	4.39
	M_{new}	28.01	40.37	35.54	15.55	20.45	21.28	27.55
IoT	F_{new}	1.35	1.32	2.54	0.00	1.24	24.25	3.09
	M_{new}	20.61	21.93	25.49	46.17	42.54	0.48	3.48
Shuttle	F_{new}	0.11	1.97	0.71	0.00	0.15	2.13	2.98
	M_{new}	6.18	45.91	37.25	1.48	4.54	31.56	1.68
HAR	F_{new}	3.54	0.00	0.54	0.00	0.00	7.66	3.00
	M_{new}	62.37	85.97	80.25	94.69	90.67	65.85	86.01

EM, evaluation measures

7 Empirical study

In this section, we present the empirical study conducted on six algorithms, namely EMC [49], MINAS [63] ECSMiser [104], CLAM [9], ECHO [77], SAND [76] and SENCForest [116]. EMC and MINAS are cluster-based single classifiers. ECSMiser, ECHO and SAND are cluster-based ensemble classifiers, while CLAM is a class-based ensemble and SENCForest is a decision tree-based ensemble classifier. For all algorithms, the default parameters setting given in the corresponding papers are followed. We have selected two synthetic and five real-world datasets to evaluate the performance of these algorithms. Datasets include SynC, SynRBF, KddCup, ForestCoverType (FCT), IoTbotnet attack (IoT), HAR and Shuttle. The detail of each dataset is given in Table 4. SynC contains two classes with concept drift but no novel class. All the other datasets are arranged to have new classes appear over time as described in [49]. For the KddCup dataset, ten largest classes are selected. For the initial training phase, three classes are used, and the remaining seven classes appear randomly with an extended period of three classes until all classes are finished. For the FCT dataset, we normalize the dataset and arrange it in the manner that at least two and three classes appear in any block and new class appears randomly in some blocks as described in [104,160]. Similarly, for IoT, Shuttle, HAR and RBF, initial training data contains 4, 2, 3 and 2 classes, respectively, and the remaining classes appear during the stream with known classes data. Shuttle and HAR have 58,000, 10,299 instances, respectively, while for all the other datasets, we have used 100K instances.

For performance evaluation, we the measures Accuracy, M_{new}, and F_{new} defined in Eqs. (14), (11), and (12), respectively. Table 3 and Fig. 14 report the results achieved by all algorithms. For Table 3, we can see that for simple datasets such as Shuttle and SynRBG, all the algorithms achieve good classification performance. In contrast, for more complex datasets, the obtained results by all algorithms are not satisfactory. Especially , for a high-dimensional dataset like HAR, novel class identification by all algorithms is very poor. For the results, we can see that the performance of unsupervised algorithms like MINAS and SENC-

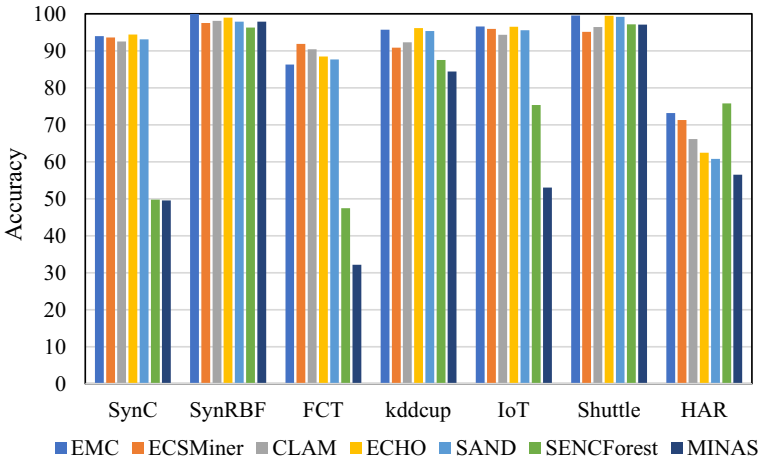
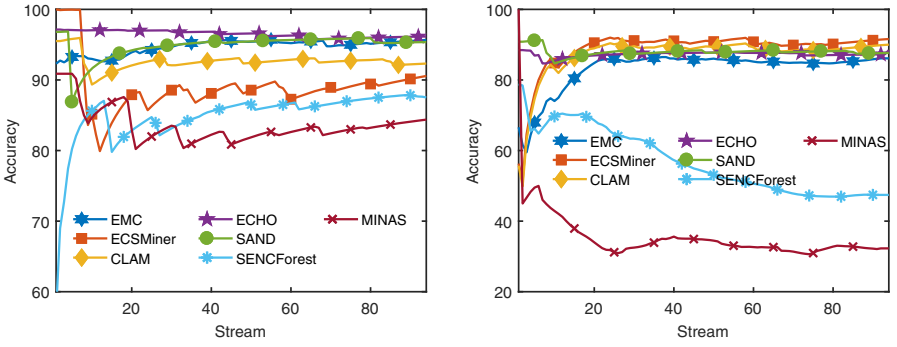
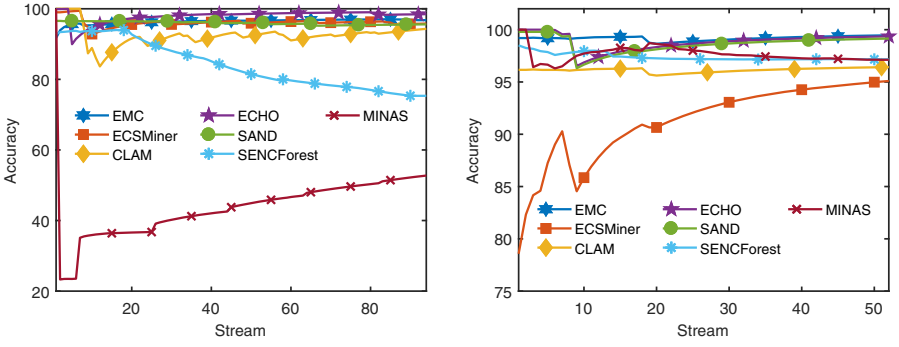


Fig. 14 Accuracy achieved by all algorithms on different datasets



(a) KddCup set (b) FCT dataset



(c) IoT set (d) Shuttle dataset

Fig. 15 Accumulative average accuracy over the streams of different datasets

Forest are significantly degraded on the SynC dataset. This dataset simulates the evolving concepts by changing the position and orientation of a hyperplane smoothly or suddenly in two-dimensional space. Thus simulate a gradual and abrupt concept drift. Because these methods update their predictive model using unlabeled data (without external feedback), they are unable to cope with the concept drift problem.

8 Applications, datasets and open-source software

8.1 Applications

Modern real-world applications and devices (such as sensors) generate an enormous amount of data every day. Various data mining algorithms have been developed to extract useful information from this data continuously. Because streaming data is dynamic, it is necessary that the learning algorithm is adoptive (retort to changes) in order to handle evolving concepts and to be able to learn new concepts. Among many applications in the real world, some of the significant applications in the literature include video surveillance, text extraction, intrusion detection, fraud detection, power distribution networks, activity recognition, fault diagnosis in machines. Algorithms proposed for classification and novel class detection in the literature have used several real applications data to compare the performance of their algorithms.

8.1.1 Intrusion detection

Most of the algorithms presented in the literature used network intrusion detection datasets (such as KDDcup99 and packages) for comparison purposes. The task of intrusion detection is to recognize unauthorized network traffic from normal traffic data. In this application, a learning model is first formed based on a set of initial learning data with normal and attacked data. In online learning, the model then classifies known attacks and identifies new types of emerging attacks. This dataset is used in algorithms such as [8,9,16,16,49,58,63–65,84,104–107,109–111,116,117,125–127,138,143,144,160–162,167,167].

8.1.2 Text mining

It is another application, and different algorithms used a set of real-world text dataset (like twitter stream) for comparison purposes. Stream classifiers such as [106–108,125] are first created with predefined topics, after which these classifiers can discover emerging topics during the text streaming and incorporate them in the classifier.

8.1.3 Forest cover type detection

The forest cover dataset is a well-known dataset. This dataset is often used to assess algorithm performance. This dataset contains a set of geospatial features that describe a type of forest. Algorithms such as [8,9,30,49,63,71,75–77,84,104–111,116,117,120,125–127,138,146,160–162,167,167] are initially trained with a set of known forest cover types and later these classifiers can be used to detect new forest cover types.

8.1.4 Human activity recognition

Similarly, human activity recognition is another important application. Activity recognition refers to the analysis of human behavior (normal or abnormal) in real-time. In this application, the main task of the learning algorithms is recognizing and learning evolving activities (concepts) in the stream. For example, identify new normal activities like "Driving" that are not available during the training phase of the classifier or discover abnormal activities like "Sudden fall." Algorithms such as [1,49,75–77,81,120,127] have used real-world datasets to recognize and identify new activities.

8.2 Datasets and open-source software

Several real-world and synthetic datasets are selected by various algorithms to evaluate their performance. A brief description and characteristics of all datasets that are used in different algorithms are presented in Table 4. These datasets are available on the following websites/repositories and can be downloaded.

- UCI Machine Learning Repository:[1] is a large collection of datasets that can be used for different types of machine learning tasks, like pattern recognition, classification, clustering, in a wide range of application domains. Datasets like KDDcup99, Forest cover type, Shuttle, PAMAP2 and HAR are available at this repository.

Similarly, some other concept drift datasets are available at the following websites. SEA and Weather datasets,[2] Forest cover type and Electricity datasets,[3] Spam,[4] Power Supply.[5]

In addition to these dataset collection repositories, some authors have developed codes to generate synthetic datasets that can be used to evaluate the performance of the proposed algorithms.

- Kuncheva et al. [121] have developed a framework based on Matlab (Concept Drift Generator)[6] to generate drifting synthetic data streams.
- Similarly, the framework[7] of Minku et al. [113] can also generate synthetic data streams.

Other useful data and software can be downloaded from the following links.[8,9,10,11]

Some open-source codes and software frameworks are also available to the research community for the development of new algorithms. Moreover, many authors have provided the software implementations and data of their algorithm used in the publication. Different algorithm's code and data publicly available are given in the following links.

[1] https://archive.ics.uci.edu/ml/datasets.php.

[2] http://users.rowan.edu/~polikar/nse.html.

[3] https://moa.cms.waikato.ac.nz/datasets/.

[4] http://mlkd.csd.auth.gr/concept_drift.html.

[5] http://www.cse.fau.edu/xqzhu/stream.html.

[6] https://lucykuncheva.co.uk/EPSRC_simulation_framework/changing_environments_stage1a.htm.

[7] http://www.cs.bham.ac.uk/+minkull/opensource.html.

[8] https://github.com/vlosing/driftDatasets.

[9] http://github.com/gditzler/ConceptDriftResources.

[10] http://roveri.faculty.polimi.it/software-and-datasets.

[11] http://dbpedia.org/page/Concept_drift.

Table 4 Description and characteristics of datasets used in different algorithms

Dataset	Brief description	Type	Size	#f	#c	Used in
SynC	Synthetic data with only concept drift. Generate using moving hyperplane available in MOA [25]	S	1,000,000	10	2	[9,49,76,77,104,105,107,109,110]
SynCN	Synthetic data with concept drift and novel class. Generate using Gaussian distribution [25]	S	1,000,000	20	10	[8,9,44,58,63,75,84,104,105,107,109,110,138,160–162]
SynRBF	Synthetic data generated using RandomRBFGeneratorDrift of MOA [25]	S	100,000	70	7	[44,63,70,71,75–77,120]
SEA	Two-class decision boundary defined by $a_1 + a_2 = b$. Here, a_1, a_2 are the two relevant features and b is a predefined threshold	S	600000	3	2	[30,49]
KDDcup99	Information about TCP connection of LAN of MIT Lincoln Labs	R	490,000	42	23	[8,9,16,16,49,58,63–65,70,84,104–107,109–111,116,117,125–127,138,143,144,160–162,167,167]
Forest cover type	Describes seven types of forest cover from different geographical areas was introduces in [27]	R	581,000	54	7	[8,9,30,33,49,63,70,71,75–77,84,104–111,116,117,120,125–127,138,146,160–162,167,167]
Twitter	It is tweets (messages) collection on multiple trends (classes)	R	Vary	Vary	Vary	[106–108,125]
ASRS	NASA Aviation Safety Reporting System text document collection containing flight anomalies	R	135,000	Vary	Vary	[106,107,125]
Shuttle	Statlog dataset	R	49,097	9	7	[49,146]
(RCV1-v2)	Collection of news stories provided by Reuters	R	222,000	Vary	Vary	[108]

Table 4 continued

Dataset	Brief description	Type	Size	#f	#c	Used in
PAMAP2	Records of different physical activities [13]	R	3,850,505	52	19	[49,75–77,120,126,127]
Electricity	Records about electricity prices market whose prices are affected by supply and demand as described in [78]	R	45,312	8	2	[30,75]
Spam	Introduced in [87]. Contains Spam Assassin collection of email messages. Feature is extracted using Boolean bag-of-words approach to represent each email	R	9324	500	2	[30,49,90]
Power supply	Contains hourly power supply information of an Italian electricity company	R	29,927	2	24	[76,77]
IMDB62	Collection of reviews on internet movie database [135]	R	62,000	Vary	62	[8,120]
Packets	Collection of network attacks data targeting a web server [12]	R	5,600	6000	28	[120]
SystemCalls	Collection of network attacks data targeting a web server [14]	R	5,600	6000	28	[120]
HAR	Records of daily living activities	R	10299	561	6	[33,49,116]
MNIST	Handwritten digits dataset	R	64,000	784	10	[33,116,117,152,167]
Outdood	Introduced in [98]. Contains images recorded by a mobile in a garden environment	R	4000	21	40	[30]

Table 4 continued

Dataset	Brief description	Type	Size	#f	#c	Used in
Rialto	Introduced in [99]. Contains images which are obtained from time-lapse videos captured by a webcam with fixed position	R	82,250	27	10	[30]
Weather	Introduced and used in [57]. Contains daily weather condition recording for 50 year	R	18,159	8	2	[30,49]
Keystroke	Contains keystroke dynamics for recognizing users by their typing rhythm as described in [140]	R	1600	10	4	[30]
Usenet	Consists of a simulation of news filtering with concept drift inspired from [87]	R	5931	658	2	[30]
Optdigits	Handwritten digits dataset	R	3829	64	10	[29,30]

#c: number of classes, #f: number of features, Type: (S: Synthetic and R: Real dataset)

- Massive Online Analysis (MOA)[12] [25]: is a Java-based open-source framework, that provides the implementation of many state-of-the-art algorithms for data stream mining. It includes clustering, classification, active learning and concept drifting stream generating models.
- Scalable Advanced Massive Online Analysis[13] [47]: is a Java-based distributed algorithm collection for mining big data streams.

In addition to these open-source frameworks, some authors made their code available to the public.

- SAND [76] https://github.com/ahaque-utd/SAND
- ECHO [77] https://github.com/ahaque-utd/ECHO
- MINAS [60,63] http://www.facom.ufu.br/~elaine/MINAS
- SENCForest [116] http://www.lamda.nju.edu.cn/files/SENCForest.zip
- EMC [49] https://github.com/SalahuddinSwati/EMC
- MuENL [171,173] http://www.lamda.nju.edu.cn/code_MuENL.ashx
- CPE [152] https://github.com/Vitvicky/

9 Summary and future research directions

In this study, we provide a review of data stream classification algorithms with emerging class detection. We offer a deep insight into the various state-of-the-art algorithms for novel class detection in the literature and analyzes the strength and weaknesses of each algorithm. Although several practical algorithms have been proposed, there are still several challenges preset in the existing algorithms that limit their applicability in many applications. Many problems remain open research questions that need to be resolved. Here, we provide a brief overview of possible research directions that seem useful and require further investigation.

9.1 Novel class detection with multiple types of features

Currently, most techniques developed to identify novel classes use only numeric attributes. Therefore, there is a need for algorithms that can work with other types of attributes, such as categorical, ordinal, or other data structures.

9.2 Novel class detection in high-dimensional data

High-dimensional data is a crucial issue that needs to address. Most of the existing algorithms are cluster-based methods and use distance-based measures to identify novel patterns, which is not a proven method for high-dimensional data streams.

9.3 Novel class detection and label scarcity

The availability of labeled data is also the biggest and most important issue that needs to be adequately considered. Most algorithms in the literature make an unrealistic assumption about the availability of labeled data. They assume that the ground truth for all data is

[12] http://moa.cms.waikato.ac.nz/.

[13] http://jmlr.org/papers/v16/morales15a.html.

available after some delay, which can be used to update learning models regularly. However, this assumption is far from the truth. It is impossible and unrealistic to provide labels for continuously arriving data streams. It is, therefore, necessary to have algorithms that can create and update their learning models with limited labeled data to reduce labeling costs and still provides an acceptable classification and a new class identification performance.

9.4 Novel class detection in feature evolving data stream

Only a few algorithms considered the problem of feature evolution. Most of the algorithms work with a fixed set of features. But in dynamic data streams, features may also evolve over time, where new features may be augmented while some old features vanished. Therefore, how to detect emerging patterns in feature evolving stream is an interesting research problem.

9.5 Multi-label learning with emerging new labels

In traditional learning tasks, one instance is associated with only one class (label), while in multi-label learning environments, an instance may be associated with multiple concepts (labels). It is also an exciting area, and only a few algorithms are developed for multi-label data streams learning with emerging new labels.

9.6 New evaluation measures for emerging class identification

To evaluate the performance of the algorithms, different algorithms have adopted different static data evaluation methods. However, these evaluation measures are not sufficient for streaming data due to the large volume of data, sequential arrival of data and dynamic data distribution. Therefore, the development of new measures to evaluate algorithm performance remains an open issue.

9.7 Recurring context with novel class identification

Various works presented in the literature implement forgetting mechanisms to eliminate outdated concepts. In these algorithms, every time a new pattern appears, it is considered a new concept. However, only a few techniques provide a mechanism to retain old concepts. When these concepts reappear, they are considered old concepts instead of declaring them as new concepts. It also a potential research area that can be further explored.

9.8 Parameter-free or less parameters algorithms

The majority of algorithms rely on various user-defined parameters, such as the number of clusters, the number of classifiers, the size of the block or window, threshold for emerging pattern identification and so on. The performance of these algorithms depends largely on these parameters and must be turned according to the dataset. Therefore, it is necessary to develop parameters free algorithms or algorithms that use less parameter for better prediction performance across diverse datasets.

9.9 Outlier handling with emerging patterns identification

Because data streams are dynamic, and existing concepts can change gradually or abruptly. In these scenarios, the handling of noise or outliers is a significant problem, and new robust algorithms are needed to identify emerging patterns from noisy data.

Acknowledgements This work is supported by the Fundamental Research Funds for the Central Universities (ZYGX 2019Z014), National Natural Science Foundation of China (61976044, 52079026), Fok Ying-Tong Education Foundation (161062) and Sichuan Science and Technology Program (2020 YFH0037).

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abdallah ZS, Gaber MM, Srinivasan B, Krishnaswamy S (2016) Anynovel: detection of novel concepts in evolving data streams. Evol Syst 7(2):73–93
2. Abrol S, Khan L, Khadilkar V, Thuraisingham B, Cadenhead T (2012) Design and implementation of snodsoc: Novel class detection for social network analysis. In: Proceedings of international conference on intelligence and security informatics, pp 215–220
3. Aggarwal CC (2015) Outlier analysis. In: Proceedings of data mining. Springer, pp 237–263
4. Aggarwal CC, Han J, Wang J, Yu PS (2003) A framework for clustering evolving data streams. In: Proceedings of 29th international conference on very large data bases, pp 81–92
5. Ahmad S, Lavin A, Purdy S, Agha Z (2017) Unsupervised real-time anomaly detection for streaming data. Neurocomputing 262:134–147 (**Online Real-Time Learning Strategies for Data Streams**)
6. Ahmadi Z, Kramer S (2018) Modeling recurring concepts in data streams: a graph-based framework. Knowl Inf Syst 55(1):15–44
7. Al-Behadili H, Grumpe A, Dopp C, Wöhler C (2015) Proc. incremental learning and novelty detection of gestures using extreme value theory. In: IEEE International conference on computer graphics, vision and information security, pp 169–174
8. Al-Khateeb T, Masud MM, Al-Naami KM, Seker SE, Mustafa AM, Khan L, Trabelsi Z, Aggarwal C, Han J (2016) Recurring and novel class detection using class-based ensemble for evolving data stream. IEEE Trans Knowl Data Eng 28(10):2752–2764
9. Al-Khateeb T, Masud MM, Khan L, Aggarwal C, Han J, Thuraisingham B (2012) Stream classification with recurring and novel class detection using class-based ensemble. In: Proceedings of IEEE 12th international conference on data mining, pp 31–40
10. Albertini MK, de Mello RF (2007) A self-organizing neural network for detecting novelties. In: Proceedings of ACM symposium on applied computing, pp 462–466
11. Alippi C, Roveri M (2008) Just-in-time adaptive classifiers—Part i: detecting nonstationary changes. IEEE Trans Neural Netw 19(7):1145–1153
12. Alnaami K, Ayoade G, Siddiqui A, Ruozzi N, Khan L, Thuraisingham B (2015) P2v: Effective website fingerprinting using vector space representations. In: Proceedings of IEEE symposium series on computational intelligence, pp 59–66
13. Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz JL (2012) Human activity recognition on smartphones using a multiclass hardware friendly support vector machine. In: Proceedings of 4th international workshop on ambient assisted living and home care, pp 216 – 223
14. Araujo F, Hamlen KW, Biedermann S, Katzenbeisser S (2014) From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In: Proceedings of ACM SIGSAC conference on computer and communications security, pp 942–953
15. Arthur D, Vassilvitskii S (2007) K-means++: The advantages of careful seeding. In: Proceedings of 18th annual ACM-SIAM symposium on discrete algorithms, pp 1027–1035
16. Attar V, Pingale G (2014) Novel class detection in data streams. In: Proceedings of 2nd international conference on soft computing for problem solving, pp 683–690
17. Bahri M, Bifet A, Gama J, Gomes HM, Maniu S (2021) Data stream analysis: foundations, major tasks and tools. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, p e1405

18. Bahri M, Gomes HM, Bifet A, Maniu S (2020) CS-ARF: compressed adaptive random forests for evolving data stream classification. In: 2020 international joint conference on neural networks, IJCNN, pp 1–8
19. Bandaragoda TR, Ting KM, Albrecht D, Liu FT, Zhu Y, Wells JR (2018) Isolation-based anomaly detection using nearest-neighbor ensembles. Comput Intell 34(4):968–998
20. Barddal JP, Loezer L, Enembreck F, Lanzuolo R (2020) Lessons learned from data stream classification applied to credit scoring. Expert Syst Appl 162:113899
21. Bartkowiak AM (2011) Anomaly, novelty, one-class classification: a comprehensive introduction. Int J Comput Inf Syst Ind Manag Appl 3(1):61–71
22. Ben-Hur A (2008) Support vector clustering. Scholarpedia 3(6):5187
23. Beyene AA, Welemariam T, Persson M, Lavesson N (2015) Improved concept drift handling in surgery prediction and other applications. Knowl Inf Syst 44(1):177–196
24. Bicego M, Figueiredo MA (2009) Soft clustering using weighted one-class support vector machines. Pattern Recogn 42(1):27–32
25. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) Moa: Massive online analysis. J Mach Learn Res 11:1601–1604
26. Bifet A, Holmes G, Pfahringer B (2010) Leveraging bagging for evolving data streams. In: Proceedings of European conference on machine learning and knowledge discovery in databases, pp 135–150
27. Blackard JA, Dean DJ (1999) Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Comput Electron Agric 24(3):131–151
28. Boldt M, Borg A, Ickin S, Gustafsson J (2020) Anomaly detection of event sequences using multiple temporal resolutions and markov chains. Knowl Inf Syst 62(2):669–686
29. Bouguelia M, Belaid Y, Belaid A (2014) Efficient active novel class detection for data stream classification. In: Proceedings of 22nd international conference on pattern recognition, pp 2826–2831
30. Bouguelia MR, Nowaczyk S, Payberah AH (2018) An adaptive algorithm for anomaly and novelty detection in evolving data streams. Data Min Knowl Disc 32(6):1597–1633
31. Breunig MM, Kriegel HP, Ng RT, Sander J (2000) Lof: Identifying density-based local outliers. In: Proceedings of ACM SIGMOD international conference on management of data, pp 93–104
32. Burkhardt S, Kramer S (2019) Multi-label classification using stacked hierarchical Dirichlet processes with reduced sampling complexity. Knowl Inf Syst 59(1):93–115
33. Cai X, Zhao P, Ting K, Mu X, Jiang Y (2019) Nearest neighbor ensembles: An effective method for difficult problems in streaming classification with emerging new classes. In: Proceedings of IEEE international conference on data mining, pp 970–975
34. Camci F, Chinnam RB (2008) General support vector representation machine for one-class classification of non-stationary classes. Pattern Recogn 41(10):3021–3034
35. Campello R, Hruschka E (2006) A fuzzy extension of the silhouette width criterion for cluster analysis. Fuzzy Sets Syst 157(21):2858–2875
36. Cao F, Ester M, Qian W, Zhou A (2006) Density-based clustering over an evolving data stream with noise. In: Proceedings of SIAM conference on data mining, pp 328–339
37. Castro-Cabrera P, Castellanos-Dominguez G, Mera C, Franco-Marín L, Orozco-Alzate M (2021) Adaptive classification using incremental learning for seismic-volcanic signals with concept drift. J Volcanol Geoth Res 413:107211
38. Cejnek M, Bukovsky I (2018) Concept drift robust adaptive novelty detection for data streams. Neurocomputing 309:46–53
39. Chandola V, Banerjee A, Kumar V (2007) Outlier detection: a survey. ACM Comput Surv 14:15
40. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):15
41. Coletta LF, Ponti M, Hruschka ER, Acharya A, Ghosh J (2019) Combining clustering and active learning for the detection and learning of new image classes. Neurocomputing 358:150–165
42. Cristiani AL, da Silva TP, de Arruda Camargo H (2020) A fuzzy approach for classification and novelty detection in data streams under intermediate latency. In: Cerri R, Prati RC (eds) Intelligent systems–9th Brazilian conference, BRACIS, Lecture Notes in Computer Science, vol 12320, pp 171–186
43. Da Q, Yu Y, Zhou ZH (2014) Learning with augmented class by exploiting unlabeled data. In: Proceedings of 28th AAAI conference on artificial intelligence, pp 1760–1766
44. da Silva TP, Schick L, de Abreu Lopes P, de Arruda Camargo H (2018) A fuzzy multiclass novelty detector for data streams. In: Proceedings of IEEE international conference on fuzzy systems, pp 1–8
45. da Silva TP, Urban GA, d. A. Lopes P, d. A. Camargo H (2017) A fuzzy variant for on-demand data stream classification. In: Proceedings of Brazilian conference on intelligent systems, pp 67–72
46. Dal Pozzolo A, Boracchi G, Caelen O, Alippi C, Bontempi G (2018) Credit card fraud detection: a realistic modeling and a novel learning strategy. IEEE Trans Neural Netw Learn Syst 29(8):3784–3797

47. De Francisci Morales G, Bifet A (2015) Samoa: Scalable advanced massive online analysis. J Mach Learn Res 16(1):149–153
48. Deng C, Yuan W, Tao Z, Cao J (2016) Detecting novel class for sensor-based activity recognition using reject rule. In: Proceedings of 9th international conference on internet and distributed computing systems, pp 34–44
49. Din SU, Shao J (2020) Exploiting evolving micro-clusters for data stream classification with emerging class detection. Inf Sci 507:404–420
50. Din SU, Shao J, Kumar J, Ali W, Liu J, Ye Y (2020) Online reliable semi-supervised learning on evolving data streams. Inf Sci 525:153–171
51. Ding S, Liu X, Zhang M (2018) Imbalanced augmented class learning with unlabeled data by label confidence propagation. In: Proceedings of IEEE international conference on data mining, pp 79–88
52. Ditzler G, Muhlbaier MD, Polikar R (2010) Incremental learning of new classes in unbalanced datasets: Learn++.udnc. In: Proceedings of 9th international workshop on multiple classifier systems, pp 33–42
53. Ditzler G, Rosen G, Polikar R (2013) Incremental learning of new classes from unbalanced data. In: Proceedings of international joint conference on neural networks, pp 1–8
54. Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of 6th ACM SIGKDD international conference on knowledge discovery and data mining, pp 71–80
55. Dries A, Rückert U (2009) Adaptive concept drift detection. Stat Anal Data Min 2(56):311–327
56. Ducange P, Pecori R, Mezzina P (2018) A glimpse on big data analytics in the framework of marketing strategies. Soft Comput 22(1):325–342
57. Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw 22(10):1517–1531
58. Erfani SM, Rajasegarar S, Leckie C (2011) An efficient approach to detecting concept-evolution in network data streams. In: Proceedings of Australasian telecommunication networks and applications conference, pp 1–7
59. Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings 2nd international conference on knowledge discovery and data mining, pp 226–231
60. Faria ER, Gama Ja, Carvalho ACPLF (2013) Novelty detection algorithm for data streams multi-class problems. In: Proceedings of 28th annual ACM symposium on applied computing, pp 795–800
61. Faria ER, Gonçalves IJCR, de Carvalho ACPLF, Gama J (2016) Novelty detection in data streams. Artif Intell Rev 45(2):235–269
62. de Faria ER, Goncalves IR, Gama J, de Leon Ferreira ACP et al (2015) Evaluation of multiclass novelty detection algorithms for data streams. IEEE Trans Knowl Data Eng 27(11):2961–2973
63. de Faria ER, Ponce de Leon Ferreira Carvalho AC, Gama J (2016) Minas: multiclass learning algorithm for novelty detection in data streams. Data Min Knowl Discov 30(3):640–680
64. Farid DM, Rahman CM (2012) Novel class detection in concept-drifting data stream mining employing decision tree. In: Proceedings of 7th international conference on electrical and computer engineering, pp 630–633
65. Farid DM, Zhang L, Hossain A, Rahman CM, Strachan R, Sexton G, Dahal K (2013) An adaptive ensemble classifier for mining concept drifting data streams. Expert Syst Appl 40(15):5895–5906
66. Folino G, Pisani FS, Pontieri L (2020) A gp-based ensemble classification framework for time-changing streams of intrusion detection data. Soft Comput 24(23):17541–17560
67. Gama J (2010) Knowledge discovery from data streams. Chapman and Hall/CRC, London
68. Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Comput Surv 46(4):44:1-44:37
69. Gao Y, Chandra S, Li Y, Khan L, Thuraisingham BM (2020) Saccos: A semi-supervised framework for emerging class detection and concept drift adaption over data streams. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2020.2993193
70. Garcia KD, de Faria ER, de Sá CR, Mendes-Moreira J, Aggarwal CC, de Carvalho AC, Kok JN (2019) Ensemble clustering for novelty detection in data streams. In: Proceedings of international conference on discovery science. Springer, pp 460–470
71. Garcia KD, Poel M, Kok JN, de Carvalho ACPLF (2019) Online clustering for novelty detection and concept drift in data streams. In: Proceedings of 19th conference on artificial intelligence, pp 448–459
72. Ghomeshi H, Gaber MM, Kovalchuk Y (2020) A non-canonical hybrid metaheuristic approach to adaptive data stream classification. Future Gener Comput Syst 102:127–139
73. Goldenberg I, Webb GI (2019) Survey of distance measures for quantifying concept drift and shift in numeric data. Knowl Inf Syst 60(2):591–615
74. Gomes HM, Barddal JP, Enembreck F, Bifet A (2017) A survey on ensemble learning for data stream classification. ACM Comput Surv 50(2):23:1-23:36

75. Haque A, Khan L, Baron M (2015) Semi supervised adaptive framework for classifying evolving data stream. In: Proceedings of 19th Pacific-Asia conference on advances in knowledge discovery and data mining, pp 383–394
76. Haque A, Khan L, Baron M (2016) Sand: Semi-supervised adaptive novel class detection and classification over data stream. In: Proceedings of 30th AAAI conference on artificial intelligence, pp 1652–1658
77. Haque A, Khan L, Baron M, Thuraisingham B, Aggarwal C (2016) Efficient handling of concept drift and concept evolution over stream data. In: Proceedings of IEEE 32nd international conference on data engineering, pp 481–492
78. Harries M, cse tr, UN, Wales NS (1999) Splice-2 comparative evaluation: electricity pricing. Technical report
79. Hayat MZ, Hashemi MR (2010) A dct based approach for detecting novelty and concept drift in data streams. In: Proceedings of international conference on soft computing and pattern recognition, pp 373–378
80. Hosseini MJ, Gholipour A, Beigy H (2016) An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. Knowl Inf Syst 46(3):567–597
81. Hu C, Chen Y, Hu L, Peng X (2018) A novel random forests based class incremental learning method for activity recognition. Pattern Recogn 78:277–290
82. Huang Z (1998) Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min Knowl Disccov 2(3):283–304
83. Iosifidis V, Ntoutsi E (2020) Sentiment analysis on big sparse data streams with limited labels. Knowl Inf Syst 62(4):1393–1432
84. Islam MR (2014) Recurring and novel class detection in concept-drifting data streams using class-based ensemble. In: Proceedings of 18th Pacific-Asia conference on advances in knowledge discovery and data mining, pp 425–436
85. Júnior JC, Faria E, Silva J, Gama J, Cerri R (2019) Novelty detection for multi-label stream classification. In: Proceedings of 8th IEEE Brazilian conference on intelligent systems, pp 144–149
86. Katakis I, Tsoumakas G, Vlahavas I (2008) Multilabel text classification for automated tag suggestion. In: Proceedings of ECML/PKDD workshop on discovery challenge
87. Katakis I, Tsoumakas G, Vlahavas I (2010) Tracking recurring contexts using ensemble classifiers: an application to email filtering. Knowl Inf Syst 22(3):371–391
88. Khezri S, Tanha J, Ahmadi A, Sharifi A (2021) A novel semi-supervised ensemble algorithm using a performance-based selection metric to non-stationary data streams. Neurocomputing 442:125–145
89. Krawczyk B, Stefanowski J, Wozniak M (2015) Data stream classification and big data analytics. Neurocomputing 150:238–239
90. Krawczyk B, Woźniak M (2013) Incremental learning and forgetting in one-class classifiers for data streams. In: Proceedings of 8th international conference on computer recognition systems, pp 319–328
91. Kumar J, Shao J, Uddin S, Ali W (2020) An online semantic-enhanced Dirichlet model for short text stream clustering. In: Jurafsky D, Chai J, Schluter N, Tetreault JR (eds) Proceedings of the 58th annual meeting of the association for computational linguistics. Association for Computational Linguistics, pp 766–776
92. Kuzborskij I, Orabona F, Caputo B (2013) From n to n+1: multiclass transfer incremental learning. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 3358–3365
93. Lazzaretti AE, Tax DMJ, Neto HV, Ferreira VH (2016) Novelty detection and multi-class classification in power distribution voltage waveforms. Expert Syst Appl 45:322–330
94. Li MJ, Ng MK, Cheung Y, Huang JZ (2008) Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters. IEEE Trans Knowl Data Eng 20(11):1519–1534
95. Li X, Zhou Y, Jin Z, Yu P, Zhou S (2020) A classification and novel class detection algorithm for concept drift data stream based on the cohesiveness and separation index of mahalanobis distance. J Electr Comput Eng 2020:4027423:1-4027423:8
96. Liberty E (2013) Simple and deterministic matrix sketching. In: Proc. 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp 581–588
97. Liu FT, Ting KM, hua Zhou Z (2008) Isolation forest. In: Proceedings of 8th IEEE international conference on data mining, pp 413–422
98. Losing V, Hammer B, Wersing H (2015) Interactive online learning for obstacle classification on a mobile robot. In: Proceedings of international joint conference on neural networks, pp 1–8
99. Losing V, Hammer B, Wersing H (2018) Tackling heterogeneous concept drift with the self-adjusting memory (SAM). Knowl Inf Syst 54(1):171–201
100. Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2019) Learning under concept drift: a review. IEEE Trans Knowl Data Eng 31(12):2346–2363. https://doi.org/10.1109/TKDE.2018.2876857

101. Lughofer E, Weigl E, Heidl W, Eitzinger C, Radauer T (2015) Integrating new classes on the fly in evolving fuzzy classifier designs and their application in visual inspection. Appl Soft Comput 35(C):558–582
102. Markou M, Singh S (2003) Novelty detection: a review—part 1: statistical approaches. Signal Process 83(12):2481–2497
103. Markou M, Singh S (2003) Novelty detection: a review—part 2: neural network based approaches. Signal Process 83(12):2499–2521
104. Masud M, Gao J, Khan L, Han J, Thuraisingham BM (2011) Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans Knowl Data Eng 23(6):859–874
105. Masud MM, Al-Khateeb TM, Khan L, Aggarwal C, Gao J, Han J, Thuraisingham B (2011) Detecting recurring and novel classes in concept-drifting data streams. In: Proceedings of IEEE 11th international conference on data mining, pp 1176–1181
106. Masud MM, Chen Q, Gao J, Khan L, Han J, Thuraisingham B (2010) Classification and novel class detection of data streams in a dynamic feature space. In: Proceedings of machine learning and knowledge discovery in databases. Springer, Berlin, Heidelberg, pp 337–352
107. Masud MM, Chen Q, Khan L, Aggarwal C, Gao J, Han J, Thuraisingham B (2010) Addressing concept-evolution in concept-drifting data streams. In: Proceedings of IEEE international conference on data mining, pp 929–934
108. Masud MM, Chen Q, Khan L, Aggarwal CC, Gao J, Han J, Srivastava A, Oza NC (2013) Classification and adaptive novel class detection of feature-evolving data streams. IEEE Trans Knowl Data Eng 25(7):1484–1497
109. Masud MM, Gao J, Khan L, Han J, Thuraisingham B (2009) Integrating novel class detection with classification for concept-drifting data streams. In: Proceedings of joint European conference on machine learning and knowledge discovery in databases, pp 79–94
110. Masud MM, Gao J, Khan L, Han J, Thuraisingham B (2010) Classification and novel class detection in data streams with active mining. In: Proceedings of 14th Pacific-Asia conference on advances in knowledge discovery and data mining, pp 311–324
111. Miao Y, Qiu L, Chen H, Zhang J, Wen Y (2013) Novel class detection within classification for data streams. In: Proceedings of 10th international symposium on neural networks, pp 413–420
112. Zhang M-L, Zhou Z-H (2006) Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans Knowl Data Eng 18(10):1338–1351
113. Minku LL, White AP, Yao X (2010) The impact of diversity on online ensemble learning in the presence of concept drift. IEEE Trans Knowl Data Eng 22(5):730–742
114. Mohamad S, Sayed-Mouchaweh M, Bouchachia A (2018) Active learning for classifying data streams with unknown number of classes. Neural Netw 98:1–15
115. Mohamad S, Sayed-Mouchaweh M, Bouchachia A (2020) Online active learning for human activity recognition from sensory data streams. Neurocomputing 390:341–358
116. Mu X, Ting KM, Zhou Z (2017) Classification under streaming emerging new classes: a solution using completely-random trees. IEEE Trans Knowl Data Eng 29(8):1605–1618
117. Mu X, Zhu F, Du J, Lim EP, Zhou ZH (2017) Streaming classification with emerging new class by class matrix sketching. In: Proceedings of 31st AAAI conference on artificial intelligence
118. Mu X, Zhu F, Liu Y, Lim EP, Zhou ZH (2018) Social stream classification with emerging new labels. In: Proceedings of 22nd Pacific-Asia conference on advances in knowledge discovery and data mining, pp 16–28
119. Muhlbaier MD, Topalis A, Polikar R (2009) Learn^{++}.nc: combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. IEEE Trans Neural Netw 20(1):152–168
120. Mustafa AM, Ayoade G, Al-Naami K, Khan L, Hamlen KW, Thuraisingham B, Araujo F (2017) Unsupervised deep embedding for novel class detection over data stream. In: Proceedings of IEEE international conference on big data, pp 1830–1839
121. Narasimhamurthy A, Kuncheva LI (2007) A framework for generating data to simulate changing environments. In: Proceedings of 25th international multi-conference: artificial intelligence and applications, pp 384–389
122. Nguyen H, Woon Y, Ng WK (2015) A survey on data stream clustering and classification. Knowl Inf Syst 45(3):535–569
123. Park CH, Shim H (2007) On detecting an emerging class. In: Proceedings of IEEE international conference on granular computing, pp 265–265
124. Park CH, Shim H (2010) Detection of an emerging new class using statistical hypothesis testing and density estimation. Int J Pattern Recogn Artif Intell 24:1–14

125. Parker B, Mustafa AM, Khan L (2012) Novel class detection and feature via a tiered ensemble approach for stream mining. In: Proceedings of IEEE 24th international conference on tools with artificial intelligence, vol 1, pp 1171–1178
126. Parker BS, Khan L (2013) Rapidly labeling and tracking dynamically evolving concepts in data streams. In: Proceedings of IEEE 13th international conference on data mining workshops, pp 1161–1164
127. Parker BS, Khan L (2015) Detecting and tracking concept class drift and emergence in non-stationary fast data streams. In: Proceedings of 29th AAAI conference on artificial intelligence, pp 2908–2913
128. Parveen P, McDaniel N, Hariharan VS, Thuraisingham B, Khan L (2012) Unsupervised ensemble based learning for insider threat detection. In: Proceedings of international conference on privacy, security, risk and trust and international conference on social computing, pp 718–727
129. Patcha A, Park JM (2007) An overview of anomaly detection techniques: existing solutions and latest technological trends. Comput Netw 51(12):3448–3470
130. Pimentel MA, Clifton DA, Clifton L, Tarassenko L (2014) A review of novelty detection. Signal Process 99:215–249
131. Razavi-Far R, Hallaji E, Saif M, Ditzler G (2019) A novelty detector and extreme verification latency model for nonstationary environments. IEEE Trans Industr Electron 66(1):561–570
132. Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496
133. Rusiecki A (2012) Robust neural network for novelty detection on data streams. In: Proceedings of 11th international conference on artificial intelligence and soft computing, pp 178–186
134. Scholkopf B, Smola AJ (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge, MA
135. Seroussi Y, Bohnert F, Zukerman I (2011) Personalised rating prediction for new users using latent factor models. In: Proceedings of 22nd ACM conference on hypertext and hypermedia, pp 47–56
136. Shao J, Ahmadi Z, Kramer S (2014) Prototype-based learning on concept-drifting data streams. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 412–421
137. Shao J, Huang F, Yang Q, Luo G (2018) Robust prototype-based learning on data streams. IEEE Trans Knowl Data Eng 30(5):978–991
138. Siahroudi SK, Moodi PZ, Beigy H (2018) Detection of evolving concepts in non-stationary data streams: a multiple kernel learning approach. Expert Syst Appl 91:187–197
139. Sokolova M, Lapalme G (2009) A systematic analysis of performance measures for classification tasks. Inf Process Manag 45(4):427–437
140. Souza VM, Silva DF, Gama J, Batista GE (2015) Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: Proceedings of SIAM international conference on data mining, pp 873–881
141. Spinosa EJ, Carvalho ACPLF (2005) Support vector machines for novel class detection in bioinformatics. Genet Mol Res 4(3):608–615
142. Spinosa EJ, de Leon F. de Carvalho AP, Gama Ja (2007) Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In: Proceedings of ACM symposium on applied computing, pp 448–452
143. Spinosa EJ, de Leon F. de Carvalho AP, Gama Ja (2008) Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks. In: Proceedings of ACM symposium on applied computing, pp 976–980
144. Spinosa EJ, de Leon F, de Carvalho AP, Gama J (2009) Novelty detection with application to data streams. Intell Data Anal 13(3):405–422
145. Sun Y, Tang K, Minku LL, Wang S, Yao X (2016) Online ensemble learning of data streams with gradually evolved classes. IEEE Trans Knowl Data Eng 28(6):1532–1545
146. Tan SC, Ting KM, Liu TF (2011) Fast anomaly detection for streaming data. In: Proceedings of 22nd international joint conference on artificial intelligence, pp 1511–1516
147. Tax DM, Duin RP (1999) Support vector domain description. Pattern Recogn Lett 20(11):1191–1199
148. Tian G, Huang J, Peng M, Zhu J, Zhang Y (2017) Dynamic sampling of text streams and its application in text analysis. Knowl Inf Syst 53(2):507–531
149. Tsymbal A (2004) The problem of concept drift: definitions and related work. Technical rep
150. Ueda N, Saito K (2002) Parametric mixture models for multi-labeled text. In: Proceedings of 15th international conference on neural information processing systems, pp 737–744
151. Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of 9th ACM SIGKDD international conference on knowledge discovery and data mining, pp 226–235

152. Wang Z, Kong Z, Changra S, Tao H, Khan L (2019) Robust high dimensional stream classification with novel class detection. In: Proceedings of IEEE 35th international conference on data engineering, pp 1418–1429
153. Wang Z, Tao H, Kong Z, Chandra S, Khan L (2019) Metric learning based framework for streaming classification with concept evolution. In: 2019 international joint conference on neural networks (IJCNN), pp 1–8
154. Xiong X, Chan KL, Tan KL (2004) Similarity-driven cluster merging method for unsupervised fuzzy clustering. In: Proceedings of 20th conference on uncertainty in artificial intelligence, pp 611–618
155. Yan G, Ai M (2013) A framework for concept drifting p2p traffic identification. TELKOMNIKA: Indones J Electr Eng 11(8):4317–4326
156. Yan GH, Ai MH (2013) A micro-cluster-based data stream clustering method for p2p traffic classification. Proc Appl Mech Mater 263:1121–1126
157. Yang Q, Zhang H, Wang G, Luo S, Chen D, Peng W, Shao J (2019) Dynamic runoff simulation in a changing environment: a data stream approach. Environ Model Softw 112:157–165
158. Yang Y, Gopal S (2012) Multilabel classification with meta-level features in a learning-to-rank framework. Mach Learn 88(1):47–68
159. Yesilbudak M (2016) Clustering analysis of multidimensional wind speed data using k-means approach. In: Proceedings of IEEE international conference on renewable energy research and applications, pp 961–965
160. ZareMoodi P, Beigy H, Siahroudi SK (2015) Novel class detection in data streams using local patterns and neighborhood graph. Neurocomputing 158:234–245
161. ZareMoodi P, Kamali Siahroudi S, Beigy H (2019) Concept-evolution detection in non-stationary data streams: a fuzzy clustering approach. Knowl Inf Syst 60(3):1329–1352
162. ZareMoodi P, Siahroudi SK, Beigy H (2016) A support vector based approach for classification beyond the learned label space in data streams. In: Proceeding of 31st annual ACM symposium on applied computing, pp 910–915
163. Zhang H, Yang Q, Shao J, Wang G (2019) Dynamic streamflow simulation via online gradient-boosted regression tree. J Hydrol Eng 24(10):04019041
164. Zhang M, Zhou Z (2014) A review on multi-label learning algorithms. IEEE Trans Knowl Data Eng 26(8):1819–1837
165. si Zhang S, Wei Liu J, Zuo X (2021) Adaptive online incremental learning for evolving data streams. Appl Soft Comput 105:107255
166. Zhang S, Wang M, Li W, Luo J, Lin Z (2019) Deep learning with emerging new labels for fault diagnosis. IEEE Access 7:6279–6287
167. Zhang Z, Li Y, Zhang Z, Jin C, Gao M (2018) Adaptive matrix sketching and clustering for semisupervised incremental learning. IEEE Signal Process Lett 25(7):1069–1073
168. Zhang Z, Zhou J (2010) Transfer estimation of evolving class priors in data stream classification. Pattern Recogn 43(9):3151–3161
169. Zheng X, Li P, Hu X, Yu K (2021) Semi-supervised classification on data streams with recurring concept drift and concept evolution. Knowl-Based Syst 215:106749
170. Zhou QF, Zhou H, Ning YP, Yang F, Li T (2015) Two approaches for novelty detection using random forest. Expert Syst Appl 42(10):4840–4850
171. Zhu Y, Ting K, Zhou Z (2016) Multi-label learning with emerging new labels. In: Proceedings of IEEE 16th international conference on data mining, pp 1371–1376
172. Zhu Y, Ting KM, Zhou Z (2017) New class adaptation via instance generation in one-pass class incremental learning. In: Proceedings of IEEE international conference on data mining, pp 1207–1212
173. Zhu Y, Ting KM, Zhou Z (2018) Multi-label learning with emerging new labels. IEEE Trans Knowl Data Eng 30(10):1901–1914
174. Zhu Y, Ting KM, Zhou ZH (2017) Discover multiple novel labels in multi-instance multi-label learning. In: Proceedings of thirty-first AAAI conference on artificial intelligence
175. Żliobaite I (2010) Change with delayed labeling: when is it detectable? In: 2010 IEEE international conference on data mining workshops. IEEE, pp 843–850

Salah Ud Din received the PhD degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2020. He is currently serving as an Assistant Professor with the Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan. His research interests focus on data stream mining, especially on data stream classification, novel class detection, and semi-supervised learning. Other related interest includes machine learning, data science, big data, text mining, pattern recognition, image processing, and document image analysis.

Junming Shao received his PhD degree with the highest honor ("Summa Cum Laude") at the University of Munich, Germany, in 2011. He became the Alexander von Humboldt Fellow in 2012. Currently, he is a professor of Computer Science at the University of Electronic Science and Technology of China. His research interests include data mining and machine learning, especially for clustering in high-dimensional data, graph mining and brain structural network analysis. He not only published papers on top-level data mining conferences like KDD, ICDM, SDM, IEEE/TKDE, but also published data mining-related interdisciplinary work in leading journals including Brain, Neurobiology of Aging, and Water Research.

Jay Kumar is currently pursuing his PhD and working in Data Mining Lab, School of Computer Science and Engineering, University of Electronics Science and Technology of China. He received his masters degree from Quaid-i-Azam University, Islamabad, in 2018. His main interest of research include data stream mining and natural language processing. His current research work has been published in top conference of ACL, IEEE Transactions on Cybernetics and Information Sciences.

Cobbinah Bernard Mawuli pursued his masters in Computer Science at the University of Electronic Science and Technology of China and currently pursuing his PhD in the same university. His research interests include federated learning, data stream mining, transfer learning and neuroimaging.

S. M. Hasan Mahmud received a PhD in Computer Science and Technology from the University of Electronic Science and Technology of China, China. He is serving as an Assistant Professor in the Department of Computer Science American International University–Bangladesh (AIUB), Dhaka, Bangladesh. He has published several international conference and journal papers. His research interests include clinical bioinformatics, drug discovery, machine learning, deep learning, and pattern recognition.

Wei Zhang received the masters degree from the Xi'an Jiaotong University in China. Currently, she is the professor of Science and Technology on Electronic Information Control Laboratory. Her major research interests are signal processing and object identification.

Qinli Yang received her PhD degree at the University of Edinburgh, UK. Currently, she is working at the University of Electronic Science and Technology of China. Her recent research interests focus on data mining-driven water resources research. She has published many papers in prestigious journals like Water Research, Environmental Modelling and Software, Journal of Environmental management, as well as several papers in the field of data mining.