Concept Evolution Detecting over Feature Streams

PENG ZHOU, YUFENG GUO, HAORAN YU, YUANTING YAN, and YANPING ZHANG, Key Laboratory of Intelligent Computing and Signal Processing (Anhui University), Ministry of Education, School of Computer Science and Technology, Anhui University, China
XINDONG WU, Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei University of Technology, China

The explosion of data volume has gradually transformed big data processing from the static batch mode to the online streaming model. Streaming data can be divided into instance streams (feature space remains fixed while instances increase over time), feature streams (instance space is fixed while features arrive over time), or both. Generally, online streaming data learning has two main challenges: infinite length and concept changing. Recently, feature stream learning has received much attention. However, existing feature stream learning methods focus on feature selection or classification but ignore the concept changing over time. To the best of our knowledge, this is the first work that studies concept evolution detection over feature streams. Specifically, we first give the formal definition of concept evolution over feature streams, which include three different types: concept emerging, concept drift, and concept forgetting. Then, we design a novel framework to detect the concept evolution over feature streams that consists of a sliding window, an improved density peak-based clustering algorithm, and a weighted bipartite graph-based concept detecting method. Extensive experiments have been conducted on several synthetic and high-dimensional datasets to indicate our new method's ability to cluster and detect concept evolution over feature streams.

CCS Concepts: • **Computing methodologies** → **Online learning settings**.

Additional Key Words and Phrases: Online Learning, Feature Streams, Stream Learning, Concept Evolution Detecting

ACM Reference Format:
Peng Zhou, Yufeng Guo, Haoran Yu, Yuanting Yan, Yanping Zhang, and Xindong Wu. 2020. Concept Evolution Detecting over Feature Streams. 1, 1 (July 2020), 32 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Big data involves large, complex, and growing datasets with multiple independent sources [3, 46]. In many applications, the knowledge extraction process must be very efficient and near real-time since storing all the observed data is almost infeasible [53]. The unprecedented amount of data requires practical data analysis and prediction platforms for fast response and real-time classification of big data. Besides, technological developments have led to the emergence of data streams and have changed how people store, communicate, and process data [21] .

Corresponding Author: Yuanting Yan.
Authors' addresses: Peng Zhou, doodzhou@ahu.edu.cn; Yufeng Guo, zghnay333@gmail.com; Haoran Yu, E22301154@stu.ahu.edu.cn; Yuanting Yan, ytyan@ahu.edu.cn; Yanping Zhang, zhangyp2@gmail.com, Key Laboratory of Intelligent Computing and Signal Processing (Anhui University), Ministry of Education, School of Computer Science and Technology, Anhui University, Heifei, Anhui, China, 230601; Xindong Wu, Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei University of Technology, China, xwu@hfut.edu.cn.

XXXX-XXXX/2020/7-ART $15.00
https://doi.org/10.1145/1122445.1122456

Generally speaking, static data do not change over a long period, while stream data are constantly updated and changed over time [42]. Furthermore, stream data can be further divided into instance streams (known in many pieces of literature as “data stream"), feature streams, or both [24]. For instance streams, the feature space of the target dataset is fixed, while the number of instances in the time domain keeps increasing due to the constant arrival of new data [2]. On the contrary, for feature streams, the instance space is fixed while features are generated and arrived one by one over time [18]. For example, Facebook, Twitter, and LinkedIn have millions of active users, and such networks generate significant streams of online data, such as text, multimedia, links, and interactions [35]. Take Twitter as an example, it produces more than 500 million tweets daily, and many slang words (features) are continuously being generated [40]. Specifically, when presenting a new hot topic, a set of new keywords appears, leading to an increase in the dimensionality of the data over time [24].

In general, streaming data mining has two main challenges: infinite length and concept changing [21]. Due to the large or potentially infinite amount of data arriving in the stream, only a portion of the entire data can be processed. Therefore, for instance streams, we use the time window as a common technique for the infinite length challenge [35]. Meanwhile, the rapidly changing environment of new instances inevitably results in the appearance of the concept drift problem, which means that the statistical properties of the target variable change over time in unforeseen ways [12]. Concept drift describes unforeseeable changes in the underlying distribution of streaming data over time, such as new products, new markets, new customers, and so on [28, 32, 50]. Besides, another issue of concept changing, which refers to the emergence of new classes during the evolution of instance streams, has been studied [31]. Traditional new class detection techniques work by assuming or constructing normal data models and identifying data points as new concepts that deviate from “normal" points [39]. Some efficient methods have recently been proposed to handle both concept drift and concept emerging over instance stream data [14, 34].

Feature steam is defined as features that are generated and arrived one by one over time while the number of instances remains fixed [52]. Most existing feature stream learning methods focus on feature selection [18], or online classification [16, 17, 57]. However, besides potentially infinite volume, feature stream mining also has the inherent property of dynamic concept change. For example, in the areas of new disease detection, such as COVID-19, we first diagnosed it as a common cold. As patients continue to add new medical tests (feature streams) to their repertoire, our understanding of the disease has changed (concept evolution). Fig.1 indicates an example of feature streams in a practical production line. Specifically, from the beginning of entering the assembly line to the end of completion, a specific product P_i generally needs to go through multiple processing equipments (from device 1 to device m). Different devices generate different streaming features for the same product P_i over time. Suppose there are three levels of product quality (A: Excellent, B: Qualified, and C: Failed). During the processing of the product P_i, with different devices (generate different streaming features), the “concept" of the quality of a specific product P_i may change over time. In other words, the quality of a specific product P_i may change from “B” to “C” with the arrival of some new streaming features. Therefore, the concepts can change over feature streams. With the detection of concept evolution, we can identify the processing equipment (the timestamp of streaming features) that caused this change, then rectify it to improve product quality.

The significance of this research lies in the proposition of a new framework for detecting concept evolution in feature streams, offering a new perspective for streaming data in dynamic environments. Specifically, by monitoring changes in concepts in feature streams, patterns evolving within the data can be identified in a timely or near-real-time manner, enabling adjustments to systems to adapt to new circumstances or enhance performance. This is crucial for fields such as production line optimization and disease monitoring. For instance, monitoring feature streams of specific products in production lines can promptly detect quality variations during the production process and swiftly adjust equipment to improve product quality [10]. In the medical field, concept evolution detection in feature streams can assist doctors in diagnosing diseases more accurately, adjusting treatment plans promptly, and enhancing patient treatment outcomes and survival rates [4]. Therefore, this study expands the theoretical realm of feature stream data mining and holds broad practical application prospects.

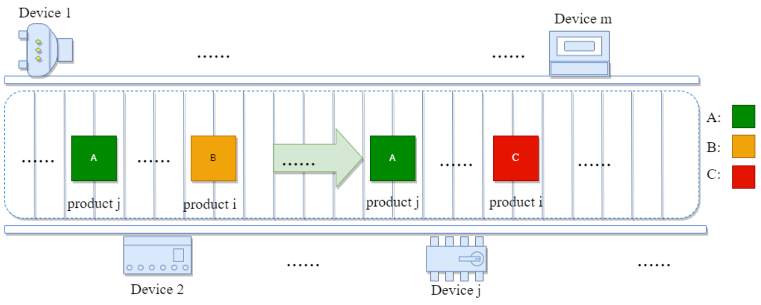


Fig. 1. Different devices (from 1 to m) generate different stream features for the same product i over time in a production line. Suppose there are three levels of product quality (A:Excellent, B:Qualified, and C:Failed). During the processing of the product, with different streaming features, the “concept" of the quality of a specific product may change over time (e.g., from “Qualified” to “Failed”).

In general, we divide the concept evolution over feature streams into three categories: concept emerging, concept drift, and concept forgetting, as shown in Fig. 2. With the arrival of new streaming features, the concepts between different time windows (feature spaces) may change over time. Concept emerging refers to the appearance of novel classes while feature streams evolve. As shown in Fig. 2(a), a new concept emerged between different timestamps over the feature streams. Concept drift indicates the gradual change in the distribution of concepts, as shown in Fig. 2(b). Besides, Fig. 2(c) explains concept forgetting in feature streams, which means the existing concepts that have disappeared in recent streams. Data analysis has revealed that data mining and machine learning in a concept-changing environment will result in poor learning results if the change is not addressed [28]. Detecting concept drift and concept forgetting helps us to analyze the causes of model performance degradation or failure. The task of detecting emerging concepts will positively consider “new concepts" as learning resources for future use. For example, a new virus detector can detect new concepts used by the medical community for research. In sum, it is critical to detect concept evolution in feature stream mining. However, to the best of our knowledge, this is the first work focusing on concept evolution detecting over feature streams.

For real-world applications, obtaining labels for streaming data in a timely manner is almost impossible, and most of the streaming data is unlabeled. Therefore, we apply clustering algorithms to discover the possible clusters. We assume different clusters indicate different concepts. Meanwhile, for streaming data, we cannot know the number of clusters before learning, and the clusters’ shape may be arbitrary. Thus, we need a clustering algorithm that detects non-spherical clusters without specifying the number of clusters. The density peak clustering (DPC) algorithm clusters the data set by building a decision graph and finding the cluster centers in the decision graph [45]. Without specifying the number of clusters, DPC can detect non-spherical clusters with no iterative process. Besides, the authors highlight the advantage of DPC in requiring fewer parameters compared to DBSCAN [9, 45]. They demonstrate that DPC can achieve effective clustering with just one

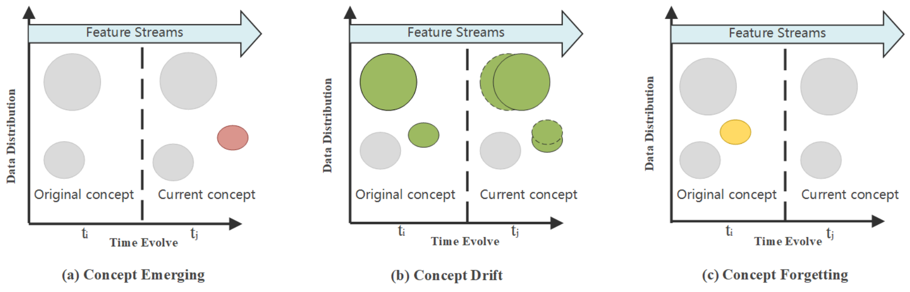


Fig. 2. There are three types of concept evolution in the feature streams: concept emerging, concept drift, and concept forgetting. Specifically, (a) represents an emerging new concept (with red color) with the arrival of new feature streams, whereas the concepts with gray color indicate no concept changing. (b) indicates the concept drift (the gradual change in the distribution of concepts) for two concepts under different feature spaces with green colors. (c) explains concept forgetting in feature streams, i.e., concept (with yellow color) that has disappeared in current feature streams.

parameter: the cutoff distance d_c. In contrast, DBSCAN typically requires at least two parameters: the neighborhood radius ε and the minimum number of points required to form a dense region $MinPts$. This reduced parameter dependency makes DPC simpler and less sensitive to parameter tuning, offering practical advantages in real-world applications. However, DPC performs poorly and may generate the wrong number of clusters on high-dimensional datasets. Therefore, we introduce kernel principal component analysis into the DPC algorithm to solve the curse of dimensionality. In addition, the local density in the DPC algorithm is defined by the parameter cutoff distance, and the improper selection of the cutoff distance will lead to the wrong selection of cluster centers. To avoid the possible detrimental effects of cutoff distances and to distinguish the density peaks of all data points, we consider the influence of one point on the other points and calculate the local density of each point using reverse k-nearest neighbors. Based on these, we propose a new, improved DPC-based clustering algorithm for concept evolution detection that does not need to specify the number of clusters and can distinguish the density peaks of all data points.

In this paper, we propose a new Concept Evolution Detecting framework over Feature Streams, named CED-FS, that consist of a sliding window, an improved DPC-based clustering algorithm, and a weighted bipartite graph-based concept evolution detection mechanism. Specifically, we use a sliding window mechanism to cache the latest streaming features as data retrieval. Then, we apply an improved DPC-based clustering algorithm on the two adjacent feature windows. After that, we detect concept evolution by measuring the similarity between each cluster of the clustering results. We model the two adjacent cluster sets together as a weighted bipartite graph. Then we calculate the weights between each pair of clusters in the adjacent clustering sets to construct a weight matrix. We summarize the characteristics of concept emerging, concept drift, and concept forgetting in the weight matrix. Finally, we let the maximum value of each row or column in the weight matrix be compared with a given threshold to determine the types of concept evolution. The main contributions are as follows:

- To the best of our knowledge, this is the first work that studies the issue of concept evolution detecting in feature stream learning. We formally define three different types of concept evolution over feature streams: concept emerging, concept drift, and concept forgetting. We present a case study to illustrate concept evolution's existence over feature streams.
- We design a novel framework to detect the concept evolution over feature streams that consist of a sliding window, a clustering algorithm, and a concept-detecting mechanism. We

propose a new, improved DPC-based clustering algorithm, KDPC-RkNN, combining kernel principal component analysis and reverse k-nearest neighbor. KDPC-RkNN does not need to specify the number of clusters and can distinguish the density peaks of all data points. Meanwhile, we measure the similarity between each cluster of two adjacent feature windows and construct a weighted bipartite graph. Then, we can detect different types of concept evolution with the weight matrix.
- In experiments, we first verify the clustering performance of KDPC-RkNN on synthetic and real-world datasets with some state-of-the-art clustering algorithms. Then, we conduct extensive experiments on several high-dimensional real-world datasets to validate our framework's ability to detect concept evolution. We experimentally indicate the ability of CED-FS to detect concept emerging, concept drift, and concept forgetting over feature streams containing multiple concepts.

The rest of the paper is organized as follows. Section II presents a brief introduction of the related works. Section III gives a formal definition of the concept evolution and a case study to illustrate the existence of concept evolution in feature streams. Section IV presents the proposed framework. Section V gives the experimental results. Finally, the paper concludes in Section VI.

2 RELATED WORK

2.1 Clustering for Concept Detection

Clustering algorithms aim to group objects so that the objects within the same group exhibit greater similarity than objects in other groups. In this paper, we assume different clusters indicate different concepts and we apply clustering to detect different concepts in the datasets.

Traditional clustering methods, such as k-means and its variants, are limited to clustering data with spherical clusters [1, 37]. For example, K-means++ is an improvement to the K-means clustering algorithm, primarily enhancing the selection of initial cluster centers [1]. By intelligently choosing initial cluster centers, K-means++ reduces the number of algorithm iterations and improves convergence speed. K-medoids is a K-means variant that defines each cluster's center as the point with the smallest average distance to other points in the cluster [37]. Unlike K-means, K-medoids is insensitive to outliers because it selects actual data points instead of the mean of the data. However, real-life data clusters often do not conform to spherical shapes. Consequently, new methods have emerged to cluster data with arbitrary-shaped clusters, including density-based clustering [9, 56], graph-based methods [49], exemplar-based clustering [29], and so on. For instance, DBSCAN is a density-based clustering algorithm that discovers clusters by defining density in the data space [9]. DBSCAN classifies data points into core, boundary, and noise points. Core points have a sufficient number of neighbors within a specified radius, boundary points have an insufficient number of neighbors but fall within the core point's neighborhood, while noise points are neither core nor boundary points. CDC is a clustering algorithm that combines diversity and connectivity [38]. CDC clusters data points by considering the diversity (the extent to which a point differs from other points within the cluster) and connectivity (similarity between points within the cluster). The algorithm uses an objective function that includes terms for maximizing diversity and maximizing connectivity to find diverse and connected clusters.

Density peak clustering (DPC) [45] identifies clusters by leveraging the insight that cluster centers typically reside within dense regions and are encircled by points of lower density. Initially, the algorithm computes the densities of all points and subsequently determines the distances to their nearest points with higher density (δ). Cluster centers are designated to possess high values of both δ and density. Subsequently, the remaining points are assigned to clusters by amalgamating with the nearest higher-density point. Two types of strategies are proposed in the literature to extend the DPC algorithm. Firstly, the performance of DPC is significantly influenced by the threshold parameter, i.e., local density d_c, which requires fine-tuning concerning different datasets. To address the parameter setting issue, Ding et al.[7] developed an automatic DPC algorithm based on the generalized extreme value distribution. DPC-KNN is a density-based clustering algorithm that combines the concepts of density peaks and K-nearest neighbors [56]. The algorithm first computes the local density and the nearest neighbor distance for each point, then determines the clustering result based on these two properties. Density peaks represent the core of a cluster, while the relative nearest neighbor distance is used to distinguish distances between different clusters. Secondly, other data points may be misallocated when the cluster centers are inaccurately chosen. To better discern the cluster centers, Xie et al. [54] employed fuzzy-weighted K-nearest neighbors to enhance the robustness of data point allocation. Du et al.[8] improved the clustering outcomes, especially for high-dimensional datasets, using K-nearest neighbors and principal component analysis. DPC-DLP incorporates a novel dynamic graph-based label propagation strategy by considering the correlation between instances and the local structure of the data [49]. The algorithm starts by detecting density peaks by calculating each data point's local density and relative nearest neighbor distance. Density peaks typically represent the core of clusters. The algorithm iteratively optimizes the clustering result by repeatedly performing density peak detection, dynamic graph modeling, and label propagation, updating the structure and labels of the dynamic graph until a certain stopping condition is met.

2.2 Concept Detection over Instance Stream

The instance stream assumes the feature space of the target dataset is fixed, while the number of instances in the time domain keeps increasing due to the constant arrival of new data [2]. Instance stream is known in much literature as “data stream". Data stream (instance stream) learning has been studied for many years and has produced a large number of excellent works [21, 35, 42]. There are two main types of concept detection in instance streams, namely concept drift and concept evolution [31].

Concept drift was first identified as the change in the data stream distribution [47]. Considerable work has been done to detect concept drift in data streams effectively. [28] reviewed more than 130 high-quality publications in research areas related to conceptual drift, analyzed recent developments in methods and techniques, and developed a framework for learning under conceptual drift. [14] present a semi-supervised framework for classifying evolving data streams that detect concept drift and determine chunk boundary dynamically by finding any significant change in classifier confidence. [22] described and evaluated the online classification system, which dynamically adjusts the size of the training window and the number of new examples between model reconstructions according to the current concept drift rate. [32] automatically detected the emergence of new classes in the presence of concept drift by quantifying cohesion between unlabeled test instances and separating test instances from training instances. [34] proposed a flow-based active learning strategy, which handles the sampling bias problem and queries samples that cause changes, i.e., drift samples or samples from new classes. [13] proposed the general framework for combining a diverse range of meta-features into a single representation. The proposed approach enabled state-of-the-art feature selection methods, such as mutual information, to be applied to concept representation meta-features for the first time. [27] proposed a novel threshold selection algorithm to align the drift thresholds of a set of algorithms so that they are all at the same sensitivity level. Higher detection sensitivity can improve the accuracy of data streams with frequently changing distributions. The evaluation results show that the drift threshold should not be fixed during flow learning.

Concept evolution (also named concept emerging) in data streams was defined as the emergence of new classes [50]. Specifically, [12] used adaptive outlier detection, discrete Gini coefficients, and multiple invisible class detection to detect invisible classes in the data stream. [50] based on the k-means clustering algorithm, which considers the problem of detecting concepts from a one-class classification perspective and aims to deal with novelty detection and concept drift through a single strategy. [31] proposed an adaptive threshold for outlier detection and a probabilistic class detection method based on the discrete Gini coefficient, which solves the problem of simultaneous emergence of multiple new classes problem. [39] have presented a review of the current state-of-the-art in novelty detection. It was found that a precise definition of novelty detection is difficult to achieve, nor is it possible to suggest what an “optimal” method of novelty detection would be. [15] based on clustering algorithms, discrete cosine transforms are used to build compact generative models, which are then used to detect new classes and concept drift efficiently. In [36], a semi-supervised approach was proposed to detect concept drift and concept evolution in the data stream by using a limited number of labeled data and a dynamic sliding window. [30] proposed a data stream classification technique that integrates a novel class detection mechanism into traditional classifiers, enabling automatic detection of novel classes before the true labels of the novel class instances arrive.

2.3 Online Feature Stream Learning

Feature stream is defined as features that are generated and arrived one by one over time while the number of instances remains fixed [52]. Most of existing feature stream learning work focus on feature selection [18] and online classification [57].

Recently, online feature selection with dynamic features has become an active research area. Specifically, [52] proposed a new online stream feature selection (OSFS) method to select strongly correlated and non-redundant features dynamically, and an efficient Fast-OSFS algorithm is proposed to improve feature selection performance. [60] proposed a new online streaming feature selection method based on adaptive density neighborhood relation, named OFS-Density. Besides, there is group structure in the feature stream in some real-world applications, such as in image analysis, where features are generated in groups to represent the color, texture, and other visual information. Group feature selection discovers meaningful subsets of features by structural information between features. [23] used stream features for group feature selection while performing feature selection at the group and individual feature levels, taking full advantage of the group structure to reduce the cost of evaluating flow features. [51] proposed approach consists of two stages: online within-group selection and online between-group selection. In within-group selection, a spectral analysis-based criterion was devised to select discriminative features. A linear regression model was used to select the optimal subset in between-group selection. [61] proposed a new online scalable streaming feature selection framework from a dynamic decisionmaking perspective, which dynamically classifies input features as selected, discarded, or delayed to minimize decision risk.

Meanwhile, some works focus on the classification of streaming features. For instance, [57] proposed a semi-streaming approach based on the well-known emerging pattern classification method that can be divided into two steps: online and offline. [16] explored a new online learning problem where the input sequence lives in an over-time varying feature space, and the ground-truth label of any input point is given only occasionally, making online learners less restrictive and more applicable. [17] proposed a novel learning paradigm: Feature Evolvable Streaming Learning, where old features would vanish and new features would occur that attempt to recover the vanished features and exploit them to improve performance. Besides, [58] present a boosting framework covering gradient ascent and online gradient ascent. They revisited Stochastic Continuous Submodular Maximization in both offline and online settings, which can benefit wide applications in machine learning and operations research areas. [59] present two communication-efficient decentralized online algorithms for the monotone continuous DR-submodular maximization problem, both of which reduce the number of per-function gradient evaluations and per-round communication complexity from $T3/2$ to 1. Maximizing a monotone submodular function is a fundamental task in data mining, machine learning, economics, and statistics.

In sum, existing online feature stream learning methods focus on the task of feature selection and classification. Motivated by this, this paper first studies the issue of concept evolution detecting over feature streams.

3 PROBLEM DEFINITION

In this section, we first define instance streams, feature streams, and concept evolution over feature streams. Then, we present a case study to illustrate the existence of concept evolution over feature streams. Table 1 summarises the notation used in this paper.

Table 1. Summary of Mathematical Notations

Notations	Definition
$\mathbb{D}$	the dataset
t	the timestamp
n	total number of instances
m	total number of features
x_i	the i^{th} instance in $\mathbb{D}$
f_j	the j^{th} feature in $\mathbb{D}$
$F_{[i,j]}$	$F_{[i,j]} = \{f_i, f_{i+1}, \cdots, f_j\}$ contains all the streaming features from timestamp i to j
$W_{[i,j]}$	$W_{[i,j]} = \{c_1, c_2, \cdots, c_k\}$ denotes the k different concepts over feature stream window $F_{[i,j]}$
$dist(x_i, x_j)$	the distance between x_i and x_j
$G = (V, E)$	bipartite graph G with vertex set V and edge set E

3.1 Formal Definition of Concept Evolution

In a natural streaming feature environment, as the number of features in new data collected from the temporal domain increases over time, the concept of these objects may evolve due to changes in the feature space. Therefore, we present a formal definition of concept evolution over feature streams that contains three types: concept emerging, concept drift, and concept forgetting.

Definition 3.1. **[Instance Streams]** Suppose dataset $\mathbb{D} = [x_1; x_2; ...; x_n] \in R^{n\times m}$, where the feature space is fixed while instances arrive one by one over time. At each timestamp t, we can require a new streaming instance $x_t \in R^{1\times m}$, and we cannot know the exact number of n in advance.

Definition 3.2. **[Feature Streams]** Suppose dataset $\mathbb{D} = [f_1, f_2, ..., f_m] \in R^{n\times m}$, where the instance space is fixed while features arrive one by one over time. At each timestamp t, we can require a new streaming feature $f_t \in R^{n\times 1}$, and we cannot know the exact number of m in advance.

Definition 3.3. **[Concept in Feature Streams]** In a feature stream environment with a fixed number of instances, objects belonging to the same cluster maintain roughly the same behavioral performance. We call these objects in the same cluster as a concept in feature streams.

Definition 3.4. **[Concept Evolution over Feature Streams]** Suppose that in a feature stream environment, at each timestamp t, a new streaming feature f_t is generated and arrives. The feature stream window $F_{[i,j]} = \{f_i, f_{i+1}, \cdots, f_j\}$ contains all the streaming features from timestamp i to j, and $W_{[i,j]} = \{c_1, c_2, \cdots, c_k\}$ denotes the k different concepts over $F_{[i,j]}$. For two adjacent feature windows F_T and F_{T+1}, **concept evolution** occurs if $W_T \neq W_{T+1}$, where $T = [i, j]$ and $T + 1 = [j + 1, k]$.

In general, there are three types of concept evolution, as shown in Fig.3. We use the binary relations E, D, and F to represent concept emerging, concept drift, and concept forgetting respectively as follows:

Definition 3.5. **[Three Types of Concept Evolution]**

- **Concept Emerging** $E_{(T,T+1)} = \{\langle \emptyset, c_j \rangle | c_j \notin W_T \wedge c_j \in W_{T+1}\}$: concept c_j exists in W_{T+1} and does not exist in W_T;
- **Concept Drift** $D_{(T,T+1)} = \{\langle c_i, c_j \rangle | c_i \in W_T \wedge c_j \in W_{T+1}\}$: concept c_i in W_T and concept c_j in W_{T+1} are similar, but they are not exactly the same;
- **Concept Forgetting** $F_{(T,T+1)} = \{\langle c_i, \emptyset \rangle | c_i \in W_T \wedge c_i \notin W_{T+1}\}$: concept c_i exists in W_T and does not exist in W_{T+1}.

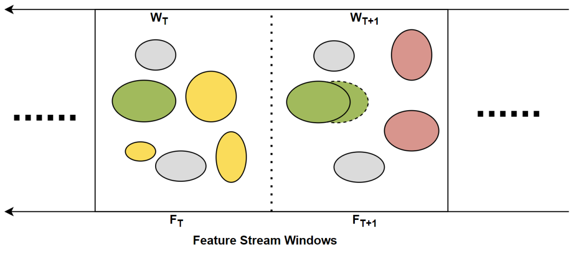


Fig. 3. Concept evolution between two feature windows F_T and F_{T+1}, where 'grey' indicates concepts that have not evolved, 'red' indicates emerging concepts, 'green' indicates concepts that have drifted, and 'yellow' is the forgotten concepts.

3.2 Case Study of Concept Evolution

In a general instance stream, the feature space is fixed, and many existing methods use model error rates to detect whether concepts have changed. However, the feature space continues to arrive in a feature stream, and we cannot require the label information in real-time. For the same object, with the increase in new streaming features, our perception of it may change. Therefore, we clustered all the samples into different clusters and used the variations of the clusters to indicate the concept changes. In this section, we use three metrics to evaluate the clustering results among different feature windows to determine whether concept evolution exists over different feature streams. We conduct the following experiments on three high-dimensional datasets (Lung2, Prostate, Dlbcl) as shown in Table 5.

Specifically, we divide the whole feature space of these three datasets into several feature windows, where the size of each feature window is fixed at 1,000. For example, since the number of features in Lung2 is 3312, there are three feature windows: $F1 = F_{[1,1000]}$, $F2 = F_{[1001,2000]}$, $F3 = F_{[2001,3000]}$. Similarly, since the numbers of features of datasets Prostate and Dlbcl are 5966 and 6285, they are divided into six feature windows, $F1$ to $F6$. Then, we apply k-means to each feature window, and the values of parameter k are selected from 2 to 6. We use Dunn Index (DI) [33], Davies-Bouldin Index (DBI) [33] and Rand Index (R) [43] to evaluate the clustering performance of different k among different feature windows to potentially judge whether evolutionary behavior will occur. DI and DBI are internally valid indicators that evaluate the clustering classification in terms of tightness, separability, connectivity, and overlap based mainly on the set structure information of the dataset. The R is a measure of agreement between two clustering solutions, which rewards true positives and true negatives and penalizes false positives and false negatives.

The DI metric calculates the shortest distance between any two cluster elements (between classes) divided by the longest distance between any two clusters (within classes). DI is defined as follows:

$$diam(C) = \max_{\forall x_i, x_j \in C} \left\| x_i - x_j \right\|_2, \tag{1}$$

$$d_{min}(C_i, C_j) = \min_{\forall x_i \in C_i, \forall x_j \in C_j} \left\| x_i - x_j \right\|_2, \tag{2}$$

$$DI = \frac{\min\limits_{1 \le i < j \le k} d_{min}(C_i, C_j)}{\max\limits_{1 \le l \le k} diam(C_l)}, \tag{3}$$

where k is the total number of clusters, $diam(C)$ denotes the farthest distance between samples within cluster C, $d_{min}(C_i, C_j)$ denotes the distance between the nearest samples of cluster C_i and cluster C_j, and $\left\| x_i - x_j \right\|_2$ denotes the distance between cluster elements in two clusters. A larger value of DI means a larger inter-class distance while a smaller intra-class distance.

DBI uses the distance of sample points within a class to its cluster centers to estimate intra-class tightness and the clustering between cluster centers to indicate inter-class separability. DBI is defined as follows:

$$d_{cen}(C_i, C_j) = dist(\mu_i, \mu_j), \tag{4}$$

$$avg(C) = \frac{2}{|C| \, (|C| - 1)} \sum_{\forall x_i, x_j \in C} dist(x_i, x_j), \tag{5}$$

$$DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \ne i} \left(\frac{avg(C_i) + avg(C_j)}{d_{cen}(C_i, C_j)} \right), \tag{6}$$

where $dist(x_i, x_j)$ is used to calculate the distance between two samples, μ represents the centroid of cluster C, $\mu = \frac{1}{|C|} \sum_{1 \le i \le |C|} x_i$, $avg(C)$ represents the average distance between samples within cluster C, and $d_{cen}(C_i, C_j)$ represents the distance between the centroids of C_i and C_j. A smaller value of DBI means smaller intra-class clusters and larger inter-class distances.

In simple terms, R measures the accuracy of the algorithm as follows:

$$R = \frac{TP + TN}{TP + FP + TN + FN}, \tag{7}$$

where TP, TN, FP and FN denote the number of true positive, true negative, false positive and false negative decisions, respectively. For the Rand index, the ideal clustering solution will have a value close to 1, while the poorer solution will have a value close to 0. The larger the R value, the better the clustering performance.

Table 2 indicates the experiment results of DI, DBI, and R obtained from k-means when the number of clusters in each feature window varies from 2 to 6 on three high-dimensional datasets. Specifically, on dataset Lung2, the maximum value of DI in difference feature window are $k = 4$, $k = 2$, and $k = 3$. Meanwhile, on datasets Prostate and Dlbcl, the optimal number of clusters also changes during different feature windows. We use different clusters to indicate different concepts in the experiments. In other words, for dataset Lung2, the optimal number of concepts from feature window F_1 to F_3 are 4, 2, and 3. Therefore, there exists concept changing over feature streams. This paper aims to detect this concept evolution over feature streams.

Table 2. Concept changing was verified on three high-dimensional datasets using the k-means algorithm (from $k = 2$ to $k = 6$) on different feature windows (F_i). The bold fonts indicate the best clustering performance among different feature windows with different values of k.

(a) Clustering results on dataset Lung2

	DI			DBI			R		
	F1	F2	F3	F1	F2	F3	F1	F2	F3
k=2	0.399	**0.474**	0.422	**1.721**	1.497	**1.591**	0.678	0.696	0.696
k=3	0.418	0.442	**0.453**	2.898	**1.212**	2.714	**0.781**	0.631	0.691
k=4	**0.486**	0.434	0.423	2.815	1.967	3.098	0.678	0.715	**0.782**
k=5	0.471	0.398	0.429	2.527	2.969	2.947	0.766	0.665	0.754
k=6	0.467	0.422	0.423	2.916	3.128	2.139	0.702	**0.771**	0.622

(b) Clustering results on dataset Prostate

	DI						DBI						R					
	F1	F2	F3	F4	F5	F6	F1	F2	F3	F4	F5	F6	F1	F2	F3	F4	F5	F6
k=2	0.311	**0.693**	0.303	0.308	0.338	0.318	0.997	**1.070**	1.251	**1.326**	**1.208**	1.138	**0.511**	0.511	0.514	0.505	0.511	0.511
k=3	**0.382**	0.291	0.293	0.295	**0.352**	0.311	**0.824**	1.382	**1.010**	1.536	1.535	1.422	0.506	0.512	0.529	0.526	0.512	0.509
k=4	0.313	0.273	**0.336**	0.333	0.314	0.291	1.507	1.738	2.270	1.922	1.960	1.893	0.508	0.505	0.526	0.522	**0.546**	0.513
k=5	0.325	0.300	0.253	**0.349**	0.317	0.312	1.837	2.189	2.057	1.452	1.635	**1.108**	0.506	**0.518**	0.515	**0.532**	0.506	0.521
k=6	0.303	0.277	0.287	0.322	0.313	**0.325**	1.993	1.749	1.935	1.874	2.256	1.147	0.506	0.500	**0.538**	0.518	0.524	**0.540**

(c) Clustering results on dataset Dlbcl

	DI						DBI						R					
	F1	F2	F3	F4	F5	F6	F1	F2	F3	F4	F5	F6	F1	F2	F3	F4	F5	F6
k=2	0.434	0.455	0.441	0.464	**0.773**	0.482	2.359	**2.406**	2.460	2.459	**0.511**	2.661	**0.513**	**0.598**	0.513	0.518	**0.547**	**0.576**
k=3	0.434	0.460	0.470	0.455	0.476	0.474	2.220	2.618	2.711	2.625	2.231	3.020	0.479	0.491	**0.577**	**0.577**	0.517	0.487
k=4	**0.454**	0.457	0.452	**0.480**	0.448	0.482	2.266	2.668	2.602	2.472	2.418	2.351	0.457	0.513	0.518	0.471	0.472	0.440
k=5	0.432	**0.481**	**0.518**	0.474	0.448	0.470	2.532	2.862	**2.231**	**2.283**	2.386	2.639	0.465	0.452	0.442	0.451	0.506	0.437
k=6	0.439	0.475	0.474	0.458	0.443	**0.490**	**2.211**	2.916	2.369	2.434	2.653	**2.233**	0.459	0.439	0.441	0.469	0.429	0.461

4 THE PROPOSED FRAMEWORK

Unlike traditional data mining, where static datasets can be iterated over repeatedly, feature streams arrive with very large or potentially infinite amounts of data. Therefore, feature stream mining requires a window technique, and we can only compute and store a small fraction of the whole feature streams. Besides, we use clustering to approximate the concepts in each feature window and detect the concept changing by analyzing the relationships among different clusters. Therefore, our new proposed framework for concept evolution detection over feature streams, named CED-FS, consists of three components: a sliding window, a clustering algorithm, and a concept-changing detection mechanism. The details of CED-FS are shown in Fig. 4.

4.1 Data Retrieval by a Sliding Window

In some practical applications, we are faced with datasets that are large in sample size and high in dimensionality while having streaming characteristics. Therefore, storing and using all the feature streams for data mining is impractical. In other words, we can only deal with a portion of the whole stream. The sliding window mechanism is commonly used to solve this memory constraint problem [35].

In this paper, we aim to detect concept evolution over feature stream. However, we cannot know precisely when concept evolution occurs before learning. Therefore, we divide the feature streams into equal-sized blocks using sliding windows. The size of a data block is equal to the size of a sliding window [55].

Concept evolution involves multiple influencing factors, such as environmental changes, user behavior, or system dynamics. These factors may interact in a complex manner. However, when aggregated, their combined effect could approximate a normal distribution due to the contributing elements' diversity and randomness. While the assumption of a normal distribution for concept

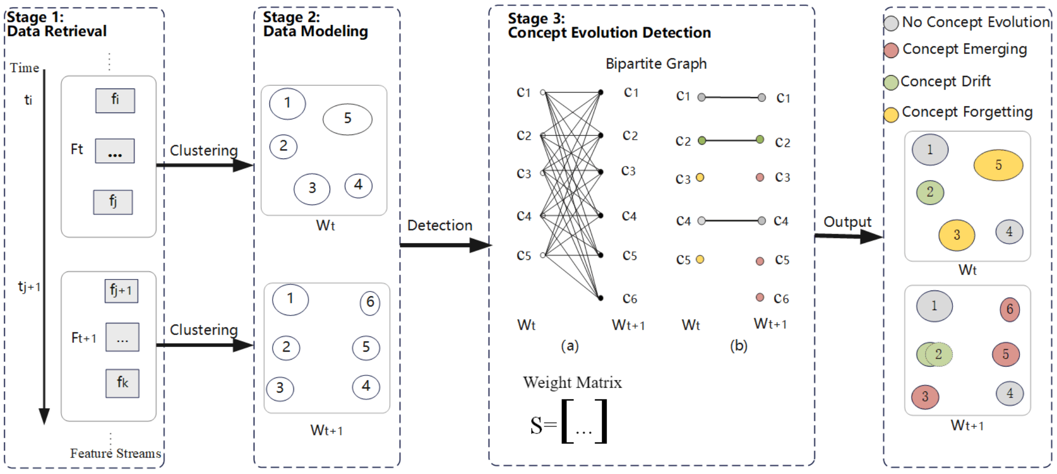


Fig. 4. The CED-FS framework consists of three main phases: data retrieval, data modeling, and concept evolution detection. Specifically, data retrieval uses a sliding window mechanism to cache the latest streaming features. Data modeling constructs an effective clustering model based on the current feature stream window. Concept evolution detection uses weighted bipartite graphs to detect concept evolution based on the clustering results between two adjacent windows. In Stage 3, a weight matrix S of the bipartite graph (a) is obtained according to the weighting formula. Then the maximum value of each row and column of S is compared with the given threshold to obtain the graph (b). In (b), the clusters that did not undergo concept evolution are colored gray and connected with black lines, while the drifting concepts are colored green and connected with black lines. Besides, in the concept set W_t, the forgotten concepts are colored yellow and connected without lines. Meanwhile, the emerging concepts are colored red and also connected without lines in the concept set W_{t+1}.

evolution in sliding windows may not be universally applicable, it can be substantiated by theoretical considerations, empirical evidence, and the convenience it offers for mathematical modeling and analysis. Hence, we assume that the probability of concept evolution within the sliding window conforms to a normal distribution.

Theorem 1. Let w be the size of a sliding window. Assume that the probability of concept evolution in the sliding window obeys a normal distribution:

$$f(x) = \frac{\epsilon}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, (1 \le x \le w) \tag{8}$$

therefore, the size of a sliding window w should be $w \le \sqrt{-2ln(\frac{\epsilon^2 e^{-\frac{1}{2}}-2}{\epsilon^2})}$ and $\sqrt{\frac{2}{e^{-\frac{1}{2}}}} \le \epsilon$, where ϵ is a predefined parameter.

Including the value of ϵ in the function $f(x)$ allows for scaling or adjusting the amplitude of the Gaussian distribution. This scaling factor ϵ enables the adjustment of the height or magnitude of the probability density function, making it suitable for various applications where different scales or amplitudes are required. Here this scaling factor ϵ still maintains $f(x)$ as a probability distribution. It is important to note that for any probability distribution function $f(x)$ the integral over its entire domain must be equal to 1. By appropriately including ϵ, $f(x)$ on the interval $1 \le x \le w$ can still be normalized to 1, ensuring that $f(x)$ remains a valid probability density function in that interval.

Proof. For a sliding window, let $S = \int_1^w \frac{\epsilon}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx = \int_1^w \frac{\epsilon}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy$

$\Rightarrow S^2 = \int_1^w \int_1^w \frac{\epsilon}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \frac{\epsilon}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dxdy = \int_1^w \int_1^w \frac{\epsilon^2}{2\pi} e^{-\frac{x^2+y^2}{2}} dxdy$

Let $x = \rho cos\theta, y = \rho sin\theta \Rightarrow dxdy = \rho d\rho d\theta, 1 \leq \rho \leq w \; and \; 0 \leq \theta \leq \pi$;

$S^2 = \frac{\epsilon^2}{2\pi} \int_0^\pi \int_1^w e^{-\frac{\rho^2}{2}} \rho d\theta d\rho = \frac{\epsilon^2}{2\pi} \cdot \pi \int_1^w e^{-\frac{\rho^2}{2}} \rho d\rho = \frac{\epsilon^2}{2} \int_1^w e^{-\frac{\rho^2}{2}} \rho d\rho = \frac{\epsilon^2}{2} (e^{-\frac{1}{2}} - e^{-\frac{w^2}{2}})$

$\because S^2 \leq 1 \therefore \epsilon^2 e^{-\frac{1}{2}} - \epsilon^2 e^{-\frac{w^2}{2}} \leq 2 \Rightarrow e^{-\frac{w^2}{2}} \geq \frac{\epsilon^2 e^{-\frac{1}{2}} - 2}{\epsilon^2} \Rightarrow -\frac{w^2}{2} \geq ln(\frac{\epsilon^2 e^{-\frac{1}{2}} - 2}{\epsilon^2}) \Rightarrow w \leq \sqrt{-2ln(\frac{\epsilon^2 e^{-\frac{1}{2}} - 2}{\epsilon^2})}$

$\therefore \frac{\epsilon^2 e^{-\frac{1}{2}} - 2}{\epsilon^2} \leq 1 \; and \; \epsilon^2 e^{-\frac{1}{2}} - 2 \geq 0 \Rightarrow \sqrt{\frac{2}{e^{-\frac{1}{2}}}} \leq \epsilon.$

In our experiments, we set $\epsilon = \sqrt{\frac{2}{e^{-\frac{1}{2}}}}$ and $w = 1,000$. It is easy to verify that $w \leq \sqrt{-2ln(\frac{\epsilon^2 e^{-\frac{1}{2}} - 2}{\epsilon^2})}$.

4.2 Data Modeling by an Improved Clustering Algorithm

We use clustering to approximate the concepts in each streaming feature window before concept evolution detection. Therefore, we cannot know the number of concepts in advance. In other words, the clustering algorithm should not depend on the number of clusters before learning. Besides, the clustering algorithm should efficiently meet the online learning demands.

4.2.1 Density Peak Clustering. The density peak clustering (DPC) [45] argues that a cluster center is characterized by a higher density than its neighbors and a relatively large distance from any locally dense point. DPC has two quantities that need to be calculated: first, the local density ρ_i of a point; second, the distance δ_i from the point with higher density. If ρ_i and δ_i are high at a point, the point is likely to be a cluster center. These two quantities correspond to the two hypotheses of cluster centers.

Specifically, suppose dataset $\mathbb{D} = [x_1; x_2; \ldots; x_n]$ contains n objects, and each data object contains m dimensional attributes, denoted as $x_i = [x_i^1, x_i^2, \ldots, x_i^m]$. The distance between any two objects x_i and x_j in dataset X is calculated by Euclidean distance and expressed as:

$$dist(x_i, x_j) = ||x_i - x_j||_2. \tag{9}$$

Then, the local density ρ_i of the data object x_i is defined as:

$$\rho_i = \sum_j \chi(dist(x_i, x_j) - d_c), \tag{10}$$

$$\chi(x) = \begin{cases} 1 \,, x \leq 0 \\ 0 \,, x > 0 \end{cases} \tag{11}$$

where d_c represents the cut-off distance. Equation (10) finds the number of data points whose distance from the i_{th} data point is less than the cut-off distance d_c, and use it as the density of the i_{th} data point.

Another way to express ρ_i, which is defined as a Gaussian kernel function [45], as follows:

$$\rho_i = \sum_j exp(-\frac{dist^2(x_i, x_j)}{d_c^2}), \tag{12}$$

where d_c is an adjustable parameter controlling the weight degradation rate, and the point j is eligible when $dist(x_i, x_j)$ is less than d_c.

δ_i is measured by computing the minimum distance between point x_i and any other point of higher density:

$$\delta_i = \min_{j:\rho_j > \rho_i} (dist(x_i, x_j)). \tag{13}$$

For the point with the highest density, we usually take $\delta_i = max_j(dist(x_i, x_j))$. Note that δ_i is much larger than the typical nearest neighbor distance only for points that are local or global maxima in the density.

After calculating these two quantities, the clustering will be executed in two steps: finding the cluster centers and assigning labels to the other points. ρ_i and δ_i depend on the distance between data points $dist(x_i, x_j)$, and the cluster centers are thought to have higher values of ρ_i and δ_i. The points with high ρ_i and δ_i values are also called peaks with higher densities than other points.

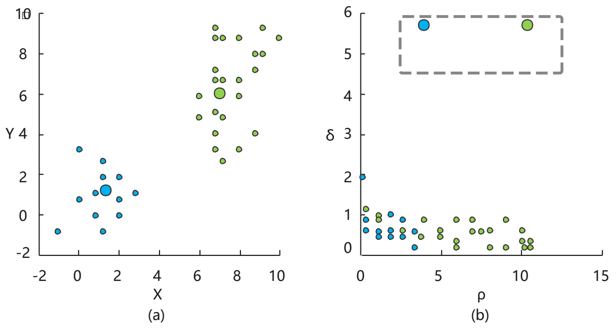


Fig. 5. An illustrated example of DPC clustering. (a) The distribution of data. (b) An example of a decision graph, where the X-coordinate is local density ρ and the Y-coordinate is δ. The manual selection is in the upper right corner, while those points in the box are labeled cluster centers.

An example of DPC clustering is shown in Fig. 5, where the target data points are plotted in Fig. 5(a) and Fig. 5(b) represents a plot of δ_i as a function of each point's ρ_i. Those points are sampled in two Gaussian distributions. The two points of larger size serve as the centers of the two clusters with the largest cluster density. The user can select a region containing individual points on the decision graph, and the points within this region are selected as centroids. Fig. 5(b) gives an example of the selection in the decision graph. A rectangle is drawn in the upper right, and the points within this rectangle are marked as centers. Since the structural information of the data is visualized in the decision graph, it can be understood and utilized by the user. After the cluster centers have been found, each remaining data point is assigned to the cluster of its nearest neighbors with a greater density than its data points.

4.2.2 Improved DPC Algorithm. DPC has many advantages, such as not needing to specify the number of clusters, requiring fewer parameters, and does not require an iterative process to detect non-spherical clusters. However, DPC also has some drawbacks [25, 54, 56]. Specifically, in some practical applications, when the dimensionality of the dataset is relatively high, the number of clusters generated by DPC may be wrong, and the model performance decreases. In feature stream learning, the dimensionality is consistently high or even infinite. Therefore, we introduced kernel principal component analysis in the DPC to handle this drawback.

Principal component analysis (PCA) is a well-known feature extraction method [6]. By computing the eigenvectors of the original input covariance matrix, PCA linearly transforms high-dimensional input vectors into low-dimensional input vectors whose components are uncorrelated. Kernel principal component analysis (KPCA) is a nonlinear PCA developed by extending the kernel method to PCA [48]. Specifically, KPCA uses a kernel function to map the original input to high-dimensional feature space and then computes principal component analysis in the high-dimensional feature space. The linear principal component analysis in the high-dimensional feature space corresponds to the nonlinear principal component analysis in the original input space. However, in practical applications, there are many nonlinear data, and it is difficult for ordinary PCA techniques to make corresponding optimization choices in dealing with nonlinear problems. The KPCA algorithm is a reliable choice for processing nonlinear data, both in terms of theoretical explanation and practical results. The common kernel functions are Gaussian kernel $k = exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2})$, exponential kernel $k = exp(-\frac{||x_i - x_j||_2}{2\sigma^2})$, Laplace kernel $k = exp(-\frac{||x_i - x_j||_2}{\sigma})$, etc, where σ is kernel function parameter.

In order to address the impact of the dimensional disaster on the performance of the DPC algorithm, we introduce KPCA to make the samples linearly separable in low-dimensional space. The details of KPCA are shown in Algorithm 1.

Algorithm 1 KPCA.

```
Input:
    Original data $X \in^{N \times D}$;
    Kernel function parameter $\sigma$;
    Target dimension $d$;
Output:
    Reduced dimensional data $Y \in^{N \times d}$;
1: Select the kernel function $k$ to calculate the kernel matrix $K_{ij} = k(x_i, x_j)$, where $1 \leq i, j \leq N$, $K \in N \times N$;
2: Center the kernel matrix according to $KI = K - I * K/N - K * I/N + I * K * I(N * N)$ , I is the identity matrix of $N \times N$ ;
3: Calculate the eigenvalues and eigenvectors of $KI$;
4: Obtain the largest $d$ eigenvalues and their corresponding eigenvectors, then unit-orthonormalized eigenvectors;
5: Collect $d$ unit orthogonal vectors, generate a dimension reduction matrix $U(U \in [N, d])$;
6: Obtain the new subspace $Y = KI * U$;
7: return $Y$;
```

Besides, the choice of parameter cutoff distance d_c significantly impacts the performance of DPC. If the selected d_c is very low, although the distance between cluster centers can be distinguished on the decision graph, it will lead to the wrong selection of the initial cluster centers. To avoid the possible detrimental effects of the cutoff distance, we distinguish the density peaks of all data points by introducing the idea of reverse k-nearest neighbor (RkNN) into DPC.

k Nearest Neighbors (kNN) is one of the most representative classification methods and has also been applied to clustering [19, 41]. The basic idea of kNN is to find the k nearest neighbors of the target instance among N samples. Usually, after calculating the Euclidean distances between the target instance and its neighbors, we sort these distances in ascending order and find the top k nearest neighbors. kNN and reverse kNN are symmetric neighborhood relations [20], and the reverse can also be obtained in the process of kNN. Suppose $NN_k(x_i)$ be the k_{th} nearest point to x_i, kNN (x_i) can be defined as:

$$kNN(x_i) = \left\{x_j | dist(x_i, x_j) \leq dist(x_i, NN_k(x_i))\right\}, \tag{14}$$

where if x_j satisfies (14), we call x_j as one of the kNN of x_i.

Suppose the point x_i is regarded as the kNN of x_j, the definition of RkNN (x_i) is:

$$RkNN(x_i) = \left\{x_j | x_i \in kNN(x_j)\right\}, \tag{15}$$

where x_j satisfies (15), we call x_j as one of the RkNN of x_i. Fig.6 visualizes the difference between kNN and RkNN.

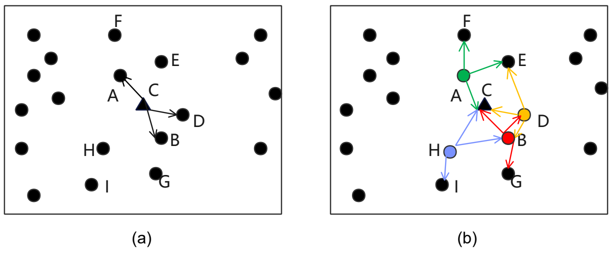


Fig. 6. Intuitionistic presentation for (a) kNN and (b) RkNN when $k = 3$. The target instance C has three nearest neighbors, A, B, and D in kNN. While in RkNN, the instance C has four reverse k nearest neighbors A, B, D, and H, since it falls into the 3-nearest neighborhoods of these four instances.

Generally, the kNN of point x_i will return at least k results, while the RkNN may have zero or more results. We use the RkNN of a point to calculate the local density instead of the kNN, which allows the difference between the local density of clustered centroids and non-clustered centroids to be magnified. The new definition of local density in terms of RkNN is as follows:

$$\rho_i = \sum_{x_j \in RkNN(x_i)} exp(-\frac{dist^2(x_i, x_j)}{r_k^2}), \tag{16}$$

where $RkNN(x_i)$ is the reverse kNN of point x_i, and r_k is the number of reverse k nearest neighbors of point x_i. The new definition indicates that the local density ρ_i of point x_i is influenced by the distribution information of its reverse kNN. The value of r_k is easier to determine than d_c. Specifically, the value of k can be specified as a percentage (p) of the number of points N, that is, $k = p \times N$.

To sum up, we propose a new, improved DPC-based clustering algorithm (KDPC-RkNN) with kernel principal component analysis and reverse k-nearest neighbor. Algorithm 2 is a summary of KDPC-RkNN.

Algorithm 2 KDPC-RkNN.

```
Input:
    A data matrix $B_i (i = 1, 2, 3, \cdots)$;
    Kernel function parameter $\sigma$;
    Target dimension $d$;
    The number of nearest neighbors $k$;
Output:
    The set of clusters $C = \{c_1, c_2, \cdots, c_m\}$;
1: Calculate the reduced dimensional dataset $Z$ by KPCA as shown in Algorithm 1;
2: Calculate the distances between each pair of point $x_i$ and point $x_j$ on $Z$ by formula (9);
3: Calculate $r_k$ and $rdist$ for each point $x_i$ in terms of RkNN, where $r_k$ is the number of reverse $k$
   nearest neighbors of point $x_i$, and $rdist$ is the reverse nearest neighbor distance;
4: Calculate $\rho_i$ and $\delta_i$ for each point $x_i$ respectively by using formula (16) and (13);
5: Draw the decision graph according to $\rho$ and $\delta$, and manually select those points which stand
   out in the decision graph as cluster centers;
6: Assign each remaining points to the same cluster as its nearest neighbors with higher density;
7: return $C$;
```

4.2.3 Complexity Analysis of KDPC-RkNN. The space complexity of the DPC is $O(n^2)$, mainly due to the storage of the distance matrix, where n is the number of instances for the target dataset. For KDPC-RkNN, besides the storage of the neighbors of kNN and RkNN $O(kn + n)$, KPCA needs to store and compute the eigenvectors of a $n \times n$ kernel matrix $O(n^2)$. Thus, the space complexity of KDPC-RkNN is $O(n^2)$.

The time complexity of KDPC-RkNN depends on the following four aspects: (a) For step 1, the time complexity of KPCA is $O(n^3)$; (b) Step 2 calculates the distance between each pair of two points with a time complexity of $O(n^2)$; (c) Step 4 calculates the local density ρ_i using RkNN with a time complexity of $O(r_k * n)$, where r_k is the number of points in the reverse kNN of point x_i and r_k is not greater than n. Meanwhile, the time complexity of calculating the distance δ_i of point x_i is $O(n^2)$; (d) Step 6 assigns points to the most appropriate cluster with a time complexity of $O(n^2)$. In sum, the time complexity of KDPC-RkNN is $O(n^3)$.

4.3 Concept Evolution Detection by a Weighted Bipartite Graph

Using the KDPC-RkNN clustering algorithm, we can get the clusters for each feature window. To detect the concept evolution over feature streams, we should measure the similarity of each pair of clusters between two adjacent feature windows. In this paper, we construct a bipartite graph using the set of clusters from two adjacent feature windows by introducing a parsimonious method (bipartite graph) [11]. The resulting graph models the clusters as vertices in the graph, allowing similarities between clusters to be considered in forming the final cluster. Therefore, we propose a weighted bipartite graph-based concept evolution detection method that converts the problem to a bipartite graph partitioning problem.

Specifically, in the mathematical field of graph theory, a bipartite graph is a type of graph where the set of vertices can be separated into two distinct, non-overlapping sets U and V. Each edge in the graph links a vertex in set U to a vertex in set V, and there are no edges connecting vertices within the same set. These sets U and V are often referred to as the two parts of the graph[44]. Given a bipartite graph $G = (V, E)$, with vertex partition $V = V_1 \cup V_2$, and edge set $E = \{(u, v) \mid u \in V_1, v \in V_2\}$. G is called a weighted bipartite graph if every edge (u, v) is associated with a weight $w(u, v)$. Bipartite graphs are widely used in areas such as image segmentation, cluster ensemble and graph edit distance computation [26].

Let $W_T = \{c_i\}_{i=1}^{|W_t|}$ and $W_{T+1} = \left\{c_j\right\}_{j=1}^{|W_{t+1}|}$ denote two sets of clustering results in two adjacent feature windows, where c_i and c_j represent the concepts in each set. We model W_T and W_{T+1} together as a weighted bipartite graph $G = (V, E)$, where $V = W_T \cup W_{T+1}$, $E = \left\{(c_i, c_j) \mid c_i \in W_T, c_j \in W_{T+1}\right\}$, and $w(c_i, c_j)$ is determined by the similarity between a pair of concepts c_i and c_j. In our method, all weights are calculated in the following way:

$$w(c_i, c_j) = \frac{2\left|c_i \cap c_j\right|}{|c_i| + \left|c_j\right|}, \tag{17}$$

where $\left|c_i \cap c_j\right|$ is the number of common instances. Then we can obtain a weight matrix $S_{(t,t+1)} \in R^{|W_T| \times |W_{T+1}|}$ in terms of W_T and W_{T+1}.

With the weight matrix S, we compute the maximum value S_{max} for each row (i) of S and its column index value (j). There are four different cases:

- If $0.5 \le S_{max} < 1$, which means the difference in distribution between c_i and c_j is not particularly large but not exactly similar, then we consider c_j undergoes a concept drift with relative to c_i.
- If $S_{max} = 1$, which means c_i is exactly similar to c_j. In other words, there is no concept evolution between these two concepts.

- If $S_{max} < 0.5$, which indicates the distribution between c_i and each cluster c_j in W_{T+1} is very different, then we consider c_i undergoes a concept forgetting.
- The judgment of concept emerging is relatively symmetric to concept forgetting. We first get the transpose matrix T of S ($T = S^{\top}$). If the maximum value of $T_{max} < 0.5$ for each row (j) in T, which means a great difference between each cluster c_i in W_T and c_j, then we consider c_j undergoes a concept emerging.

To verify the validity of the proposed concept evolution detection method, we apply an experiment test on dataset Lung2. Specifically, we divide the whole feature space of dataset Lung2 into three feature windows where the number of features in each window is 1,000. After the clustering by our proposed algorithm KDPC-RkNN, the number of clusters for each $W_T (T = 1, 2, 3)$ are 4, 6, and 5, respectively. The details of weight matrix S for each two adjacent feature windows are shown in Tables 3.

Table 3. Weighting Matrix S of clustering results in two adjacent feature windows on Lung2 dataset. The bold fonts indicate the S_{max} per row.

(a) The weight matrix $S_{(1,2)}$ for W_1 and W_2

W_1 \ W_2	c_1	c_2	c_3	c_4	c_5	c_6
c_1	0.027	**0.944**	0.066	0	0.059	0
c_2	0	0.039	0	**0.903**	0	0
c_3	0	0.013	0	0	**0.857**	0
c_4	0	0	0.074	0	0	**0.976**

(b) The weight matrix $S_{(2,3)}$ for W_2 and W_3

W_2 \ W_3	c_1	c_2	c_3	c_4	c_5
c_1	0	0	0	**0.160**	0
c_2	0	**0.986**	0.013	0	0
c_3	**0.909**	0.014	0	0	0
c_4	0	0	**0.966**	0	0
c_5	0	0.245	0	**0.913**	0
c_6	0	0	0	0	**1.000**

In Tables 3 (a), $S_{max} = 0.944 > 0.5$ for the first row which means there is a concept drift for c_1 in W_1 relative to the concept c_2 in W_2. The second, third, and fourth rows are similar. Therefore, we can get concept drift relation $D_{(1,2)} = \{\langle c_1, c_2\rangle, \langle c_2, c_4\rangle, \langle c_3, c_5\rangle, \langle c_4, c_6\rangle\}$. For the transpose matrix T of S, $T_{max} < 0.5$ for the first and third rows. Thus, we can get concept emerging relation $E_{(1,2)} = \{\langle \emptyset, c_1\rangle, \langle \emptyset, c_3\rangle\}$. In Tables 3 (b), $S_{max} < 0.5$ for the first row. Therefore, there is a concept forgetting $F_{(2,3)} = \{\langle c_1, \emptyset\rangle\}$. Besides, $S_{max} = 1$ for the last row, which means there is no concept evolution for these two concepts. Besides, we can get concept drift relation $D_{(2,3)} = \{\langle c_2, c_2\rangle, \langle c_3, c_1\rangle, \langle c_4, c_3\rangle, \langle c_5, c_4\rangle\}$.

The general process of CED-FS is shown in Algorithm 3 below. The codes are publicly available on Github [1].

[1] https://github.com/doodzhou/Steam-Learning

Algorithm 3 CED-FS.

```
Input:
        Streaming feature $f_i$;
        The length of sliding window $w$;
        The target dimension size $d$;
        The Kernel function parameter $\sigma$;
        The ratio of the nearest neighbors $p$;
Output:
        Three types of concept evolution: $E, D, F$;
1:  Set $T = 1, C = \{\}, E = \{\}, D = \{\}, F = \{\}$;
2:  Repeat
3:  Receive streaming features to form data matrix $B_T = \{f_i, ..., f_{i+w-1}\}$;
4:  $k = p \times N$, where $N$ is the number of instances;
5:  $C_T = KDPC - RkNN(B_T, \sigma, d, k)$;
6:  $C = C \cup \{C_T\}$;
7:  If $T == 1$
8:      Continue;
9:  End If;
10: Calculate the weight matrix $S_{(T-1,T)}$ betwee $C_{T-1}$ and $C_T$;
11: Use $S_{(T-1,T)}$ to determine the concept evolution behaviors: $E_{(T-1,T)}, D_{(T-1,T)}, F_{(T-1,T)}$;
12: $E = E \cup \{E_{(T-1,T)}\}, D = D \cup \{D_{(T-1,T)}\}, F = F \cup \{F_{(T-1,T)}\}$;
13: $T = T + 1$;
14: Until no more streaming features
15: return $E, D, F$;
```

5 EXPERIMENTS

This section first presents the experiment setup, including datasets, evaluation metrics, competing algorithms, and statistical tests. Since our new method is the first algorithm designed for concept evolution detection over feature streams, no competing algorithms exist. Meanwhile, existing concept evaluation detection methods for data streams cannot be applied to feature streams. For our new concept evolution detection framework, the effectiveness of clustering is fundamental. Therefore, we first validate the effectiveness of our new clustering algorithm KDPC-RkNN on five synthetic datasets. Meanwhile, we compared KDPC-RkNN with state-of-the-art clustering methods on several real-world datasets. Then, we apply our new framework, CED-FS, on nine high-dimensional datasets to illustrate the effectiveness of concept evaluation detection. Finally, we present the parameter analysis of CED-FS.

5.1 Experiment Setup

5.1.1 Datasets. We first verify the clustering performance of KDPC-RkNN on five synthetic datasets [1] as shown in Fig. 7. The number of data points for these five datasets varies from small to large with different numbers of clusters, and the distribution of points is a challenge in detecting clusters.

Specifically, Dataset (a) is generated from 15 similar two-dimensional Gaussian distributions located in a ring and contains 600 points. A total of ten spherical clusters are contained in it, with seven nearby clusters. Dataset (b) consists of seven groups of perceptually different points where non-Gaussian clustering exists and contains 788 points. Compared with other datasets, it has two

[1] http://cs.uef.fi/sipu/datasets.

weakly connected cluster pairs. Dataset (c) has different sizes and shapes and contains 240 points. It contains a spherical cluster and a close crescent-shaped cluster. Dataset (d) consists of three spherical and three irregular data, containing 399 points. Clusters in this are more challenging in distributions, which contain two weakly connected spherical clusters, a non-spherical dense cluster, and a sparse cluster. Meanwhile, a ring cluster surrounds a spherical cluster, appearing as an island distribution. Dataset (e) has 300 points and consists of a circular cluster with an opening near the bottom and two Gaussian distribution clusters. It contains two dense clusters in the shape of spheres surrounded by a ring-shaped cluster with a significant density difference.

Meanwhile, we compare KDPC-RkNN with six state-of-the-art clustering algorithms on nine real-world datasets, as shown in Table 4. The details of these datasets can be seen in the UC Irvine Machine Learning Repository[2].

Table 4. Nine real-world datasets for clustering

Datasets	**Instances**	**Features**	**Classes**
Iris	150	5	3
BreastCancer	699	10	2
Seeds	210	8	3
Zoo	101	17	7
MovementLibras	360	90	15
Glass	214	9	6
Pima	768	9	2
Leaves1	96	64	6
Leaves2	96	64	6

Besides, to validate the effectiveness of our new framework CED-FS, we conduct experiments on nine high-dimensional real-world datasets for concept evolution detection over feature streams, as shown in Table 5 [3]. Specifically, The LUNG2 dataset comprises 203 samples categorized into five groups: adenocarcinoma, squamous cell lung cancer, lung carcinoid, small cell lung cancer, and normal lung. Specifically, there are 139 samples of adenocarcinoma, 21 of squamous cell lung cancer, 20 of lung carcinoid, 6 of small cell lung cancer, and 17 of normal lung. Each sample in this dataset consists of 3312 features. The GLIOMA dataset comprises four categories: cancerous glioblastoma (CG), non-cancerous glioblastoma (NG), cancerous oligodendroglioma (CO), and non-cancerous oligodendroglioma (NO). It consists of 50 samples: 14 CG, 14 NG, 7 CO, and 15 NO samples. Each sample is represented as a one-dimensional vector of length 4434. The Gisette dataset comprises 1000 instances, each characterized by a vast array of 5000 features. These features represent various attributes or measurements associated with each instance. Such features could include pixel intensities in image data, gene expression levels in biological data, or numerical values representing different aspects of a phenomenon in scientific datasets. Each instance in the dataset is a data point that contains values for all 5000 features, forming a high-dimensional representation of the underlying data. The Mll dataset comprises 57 instances with 5848 features each. This dataset is characterized by its comprehensive array of features, covering a diverse range of variables and attributes. Each instance within the dataset is associated with many descriptive characteristics, providing rich and detailed information for analysis and modeling purposes. The prostate dataset comprises 102 instances and 5966 features, each representing a patient diagnosed with prostate cancer, described by diverse demographic attributes (e.g., age, race), medical history (treatment

[2] https://archive.ics.uci.edu/datasets.
[3] http://www.cs.binghamton.edu/ lyu/KDD08/data/.

history, familial cancer background), and diagnostic outcomes (e.g., PSA levels, tumor size). The DLBCL dataset comprises 77 samples categorized into two classes: diffuse large B-cell lymphomas (DLBCL) and follicular lymphoma (FL), with 58 and 19 instances, respectively. Each sample is characterized by 6285 features. The CAR data set comprises 174 samples distributed across eleven distinct cancer types: prostate, bladder/ureter, breast, colorectal, gastroesophagus, kidney, liver, ovary, pancreas, lung adenocarcinomas, and lung squamous cell carcinoma. The sample counts for each class are as follows: 26 for prostate, 8 for bladder/ureter, 26 for breast, 23 for colorectal, 12 for gastroesophagus, 11 for kidney, 7 for liver, 27 for ovary, 6 for pancreas, 14 for lung adenocarcinomas, and 14 for lung squamous cell carcinoma. ARCENE is obtained by merging three mass spectrometry datasets to obtain sufficient training and testing data for benchmarking. Raw features indicate the abundance of proteins with a given mass value in human serum. Based on these characteristics, cancer patients (56 samples) had to be separated from healthy patients (44 samples).

Table 5. Nine high-dimensional real-world datasets for concept evolution detection

Datasets	**Instances**	**Features**	**Clusters**
Lung2	203	3312	5
Glioma	50	4434	4
Gisette	1000	5000	2
Mll	57	5848	3
Prostate	102	5966	2
Dlbcl	77	6285	2
Car	174	9182	11
Arcene	100	10000	2
Real-sim	72309	20958	2

5.1.2 Evaluation Metrics. ARI (Adjusted Rand Index), F-score, and ACC (Accuracy) are three common metrics to evaluate the performance of competing clustering algorithms.

ARI (Adjusted Rand Index): ARI is a metric used to measure the effectiveness of clustering algorithms, taking into account the consistency between the actual class labels and the clustered results. It has a range of [-1, 1]. The formula for ARI is as follows:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{N}{2}}, \tag{18}$$

where (n_{ij}) represents the intersection size of samples between actual class (i) and cluster (j), (a_i) and (b_j) represent the number of samples in actual class (i) and cluster (j) respectively, and (N) is the total number of samples.

Accuracy is the simplest evaluation metric in classification, indicating the proportion of correctly classified samples. The formula is:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \tag{19}$$

where TP, TN, FP and FN denote the number of true positive, true negative, false positive and false negative decisions, respectively.

F-score is the harmonic mean of precision and recall, used to comprehensively evaluate the accuracy of a classification algorithm. We apply F_1 to evaluate the performance of competing algorithms as follows:

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}, \tag{20}$$

where $Precision = TP/(TP + FP)$ and $Recall = TP/(TP + FN)$.

Before starting the experiment, we normalize the data using a min-max normalization to make all data linearly map into $[0, 1]$.

5.1.3 Competing Clustering Algorithms. We compare our new clustering algorithm KDPC-RkNN with six clustering methods, including DBSCAN [9], K-means++ [1], K-medoids [37], DPC-KNN [56], DPC-DLP [49], and CDC [38]. A brief introduction to these competing algorithms can be seen in Section 2.1.

The parameter k for algorithms K-means++ and K-medoids is set to the actual number of classes for each dataset in Table 4. For DBSCAN, the parameter values of eps and $minPts$ with the best performance are $(0.13, 9), (0.48, 15), (0.34, 30), (0.5, 1), (0.5, 40), (0.4, 20), (0.2, 1), (0.5, 1), (0.85, 7)$ for each data set in Table 4 respectively. For DPC-KNN, the optimal values of the parameter p that control the nearest neighbors are 0.11, 0.2, 0.05, 0.7, 0.05, 0.02, 0.02, 0.05, and 0.05 for each data set. Similar to DPC-KNN, the parameter p with the best performance in DPC-DLP for each dataset are 0.11, 0.4, 0.01, 0.03, 0.04, 0.01, 0.08, 0.36, and 0.36, respectively. CDC has two parameters, k and $ratio$. The optimal values of these two parameters for each data set are $(6, 0.90), (8, 0.95), (7, 0.85), (10, 0.85), (15, 0.90), (8, 0.80), (9, 0.70), (8, 0.90), (8, 0.90)$, respectively. Besides, for our new proposed algorithm KDPC-RkNN, the parameter p exhibits the best performance for each data set are 0.05, 0.10, 0.02, 0.05, 0.05, 0.05, 0.02, 0.05, and 0.05.

For each dataset in Table 4, we run all these competing algorithms ten times and report the mean value as the final results.

5.1.4 Statistical Tests. To verify whether the clustering performance of KDPC-RkNN and its competitors on different metrics is significantly different, we performed the Friedman test at 95% significance level under the null hypothesis [5]. If the null hypothesis is rejected, there is a significant difference in the performance of KDPC-RkNN and its competitors. When the null hypothesis of the Friedman test was rejected, we proceeded to the Nemenyi test as a post-hoc test [5].

All experiments were run on a computer with Windows 10, AMD Ryzen 7 3700X 8-Core Processor 3.6 GHz, and 16 GB of RAM running Matlab R2021b.

5.2 Performance of KDPC-RkNN

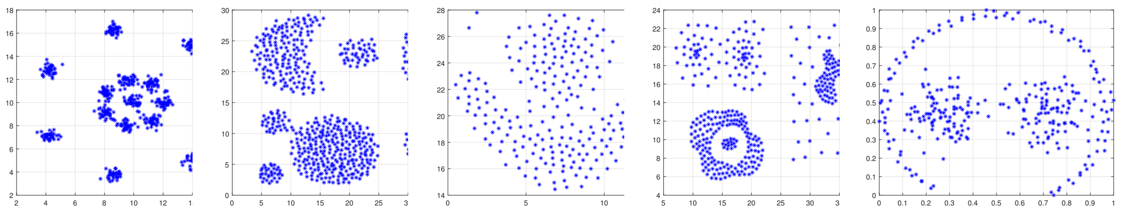

(a) (b) (c) (d) (e)

Fig. 7. Five original synthetic datasets.

5.2.1 The clustering results on synthetic datasets. Since the dimensions of these five synthetic datasets are relatively low, the value of the target dimension d in KPCA is set to the same dimension as the original dataset. Besides, there are two parameters for KDPC-RkNN: kernel function parameter δ, the ratio of the nearest neighbors p ($k = p \times N$).

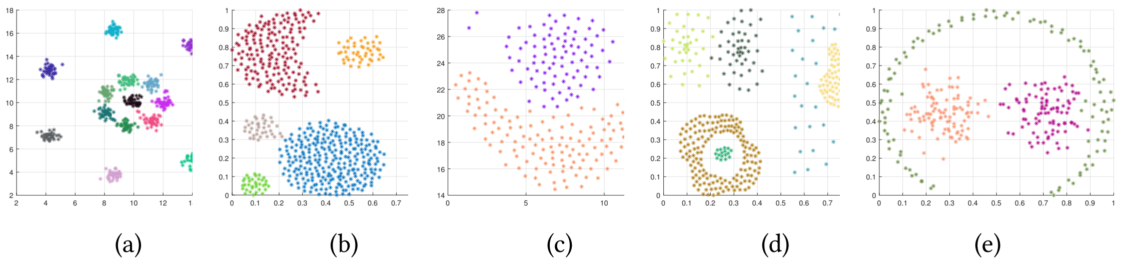


Fig. 8. The clustering results of KDPC-RkNN on five synthetic datasets.

For dataset (a), with parameter p chosen from interval $[0.1\%, 5\%]$, and parameter σ is taken from interval $[2, 20]$ for Gaussian kernel, exponential kernel or Laplace kernel, all yielding the results in Fig. 8(a). Therefore, our improved algorithm is robust for σ and p on the dataset (a).

For datasets (b) and (c), we obtain the results as shown in Fig. 8(b) and Fig. 8(c), respectively, which indicate the effectiveness of our new algorithm for such clusters with different sizes and shapes. We used the Laplace kernel on dataset (b), and the best experimental result is achieved when the parameter p is taken as 0.5% and σ taken from interval $[2, 20]$. On dataset (c), by using the Gaussian kernel, good results are obtained when the parameter p is taken around 20% and σ is taken as interval $[12, 20]$. In addition, the experimental effect is also valid by using the Laplace kernel with the parameter p is taken around 10% or 20% and σ is taken as interval $[6, 18]$.

Through extensive testing on datasets (d) and (e), we can get perfect results as shown in Fig. 8(d) and Fig. 8(e). KDPC-RkNN can take any values for the parameter p from the interval $(0.1\%, 1)$ which indicates the insensitive parameter p on these two datasets. In other words, our new method of computing local densities with reverse k nearest neighbors is very effective for dealing with distributions like datasets (d) and (e).

In sum, these experiments on synthetic datasets indicate that our new algorithm can be very effective for clusters with different distributions, shapes, and densities.

5.2.2 The clustering results on real-world datasets. To validate the effectiveness of our new clustering algorithm in real-world applications, we compare KDPC-RkNN with six state-of-the-art clustering methods on nine real-world datasets. Tables 6 to 8 present the clustering performance of ARI, FScore, and ACC, respectively. The p-values of the Friedman test for ARI, FSCore, and ACC are 4.4327e-05, 3.4520e-06, and 7.8811e-04, respectively. Therefore, there is a significant difference among these competing algorithms on the clustering performance. According to the Nemenyi test, the value of CD is 3.003. Fig.9 indicates the statistical test of these competing algorithms with ARI, FScore, and ACC.

From Tables 6 to 8 and Fig. 9, we can indicate that:

- KDPC-RkNN vs. DBSCAN: According to the statistical test graph, KDPC-RkNN performs significantly better than DBSCAN in cases of FScore and ACC. In the ARI index, KDPC-RkNN also gets higher average values and lower average ranks than DBSCAN. DBSCAN may struggle with datasets exhibiting uneven density, as it might not adapt well to regions with varying density levels. Sensitivity to noise and border points poses another challenge, potentially leading to misclassification at the borders of different-density regions. The algorithm's reliance on density reachability limits its effectiveness in identifying clusters with non-convex shapes.

Table 6. Clustering performance with ARI

Datasets	**DBSCAN**	**K-means++**	**K-medoids**	**DPC-KNN**	**DPC-DLP**	**CDC**	**KDPC-RkNN**
Iris	0.6812	0.6788	0.7565	0.8015	0.886	**0.9038**	0.9037
BreastCancer	0.8662	0.8391	0.8284	0.7395	0.8557	0.8718	**0.872**
Seeds	0.5843	0.6934	0.7064	0.7289	0.7431	0.7440	**0.7754**
Zoo	0.8229	0.748	0.666	0.8058	0.7466	**0.9501**	0.9249
MovementLibras	0.331	0.3350	0.3363	0.3450	0.3289	**0.4251**	0.4171
Glass	0.8662	0.8391	0.8284	0.7395	0.8557	0.8698	**0.8716**
Pima	**0.1533**	0.0976	0.0957	0.1043	0.1429	0.1514	0.1469
Leaves1	**0.9748**	0.7580	0.8053	0.7625	0.8135	0.8059	0.9745
Leaves2	0.7788	0.8646	0.8646	0.9286	0.7307	0.8163	**1.0**
AVG.	0.6621	0.6504	0.6542	0.6617	0.6781	0.7265	**0.7584**
AVG. RANKS	4.111	5.500	5.389	4.667	4.333	2.222	**1.778**

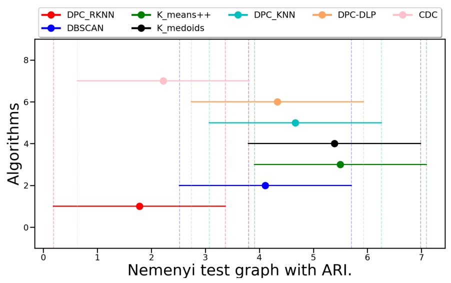
Table 7. Clustering performance with FScore

Datasets	**DBSCAN**	**K-means++**	**K-medoids**	**DPC-KNN**	**DPC-DLP**	**CDC**	**KDPC-RkNN**
Iris	0.7728	0.7893	0.837	0.8668	0.9236	**0.9355**	0.9354
BreastCancer	0.8662	0.8391	0.8284	0.7395	0.8557	0.8718	**0.877**
Seeds	0.7263	0.795	0.8038	0.8094	**0.8283**	0.8093	0.818
Zoo	0.8594	0.8017	0.728	0.8484	0.8127	**0.9618**	0.943
MovementLibras	0.2698	0.4329	0.4333	0.4559	0.4466	**0.522**	0.5089
Glass	0.8229	0.7480	0.6660	0.8058	0.7466	0.9148	**0.9249**
Pima	0.5679	0.6013	0.6025	**0.6884**	0.6090	0.5563	0.6734
Leaves1	0.3440	0.4654	0.4605	0.5899	0.3471	0.5751	**0.5940**
Leaves2	0.2181	0.4329	0.4333	0.4559	0.4466	0.4768	**0.5089**
AVG.	0.6053	0.6562	0.6436	0.6956	0.6685	0.7359	**0.7510**
AVG. RANKS	5.556	5.444	5.444	3.444	3.889	2.556	**1.667**

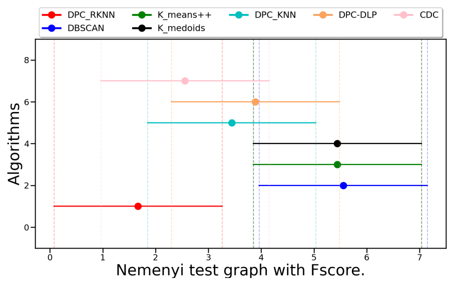
Table 8. Clustering performance with ACC

Datasets	**DBSCAN**	**K-means++**	**K-medoids**	**DPC-KNN**	**DPC-DLP**	**CDC**	**KDPC-RkNN**
Iris	0.8000	0.8485	0.9067	0.9187	0.9600	**0.9667**	**0.9667**
BreastCancer	0.8662	0.8391	0.8284	0.7395	0.8557	**0.8718**	0.8716
Seeds	0.8381	0.8857	0.8905	0.8905	0.9048	0.8905	**0.9190**
Zoo	0.8416	0.7228	0.7228	0.8020	0.7921	**0.9208**	0.8614
MovementLibras	0.4514	0.5298	0.5298	0.4881	0.5238	0.4940	**0.5417**
Glass	0.8662	0.8391	0.8284	0.7395	0.8557	0.8718	**0.8770**
Pima	0.5130	0.6665	0.6654	0.6497	**0.6940**	0.6016	0.6875
Leaves1	0.9792	0.8125	0.8958	0.7917	0.8587	0.8854	**0.9896**
Leaves2	0.8021	0.9375	0.9375	0.9688	0.8333	0.8333	**1.0000**
AVG.	0.7731	0.7868	0.8006	0.7765	0.8087	0.8151	**0.8572**
AVG. RANKS	5.111	4.833	4.500	5.111	3.722	3.333	**1.389**

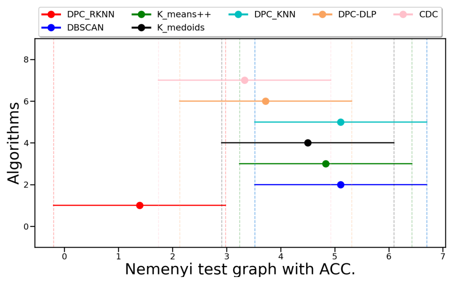
- KDPC-RkNN vs. K-means++: Based on the Nmenyi test, KDPC-RkNN performs significantly better than K-means++ in cases of all these three metrics. K-means++ assumes spherical and equally sized clusters, akin to the standard K-means, making it less suitable for datasets with clusters of non-spherical shapes or varying sizes. Another consideration is the dependence on the predefined number of clusters, requiring users to specify this parameter beforehand. This can be challenging when the actual number of clusters is unknown or varies in the dataset.



(ARI)



(FScore)



(ACC)

Fig. 9. The statistical test graph of KDPC-RkNN vs. competing clustering algorithms.

- KDPC-RkNN vs. K-medoids: According to the statistical test, KDPC-RkNN performs significantly better than K-medoids in cases of ARI and Fscore. In the case of ACC, KDPC-RkNN gets higher accuracy than K-medoids on all nine datasets. The K-medoids algorithm's performance is contingent on the initial selection of medoids, making it dependent on the choice of starting points, and finding an optimal set of medoids can be computationally intensive. Besides, as the number of data points increases, the computational complexity of K-medoids rises, making it less scalable for large datasets.
- KDPC-RkNN vs. DPC-KNN: Based on the Nmenyi test, KDPC-RkNN performs significantly better than DPC-KNN in ACC. In the ARI metric, KDPC-RkNN outperforms DPC-KNN on all nine datasets. Meanwhile, in the Fscore metric, KDPC-RkNN is better than DPC-KNN on eight of nine datasets. Like our new method, DPC-KNN is also based on the DPC algorithm. However, the DPC-KNN algorithm's performance is sensitive to parameter choices, such as thresholds for determining density peaks and k-nearest neighbors, requiring careful consideration or domain knowledge for optimal selection. Additionally, the effectiveness of DPC-KNN is contingent on the accurate identification of initial density peaks, and different initial selections may lead to varying clustering outcomes.
- KDPC-RkNN vs. DPC-DLP: According to the statistical test, there is no significant difference between KDPC-RkNN and DPC-DLP in these three metrics. On most datasets, KDPC-RkNN has a higher performance than DPC-DLP. DPC-DLP is also based on the DPC algorithm. The effectiveness of DPC-DLP heavily relies on accurately identifying initial density peaks, and different initial selections can yield diverse clustering outcomes, emphasizing the critical nature of this step. DPC-DLP might face difficulties when handling clusters with non-convex shapes, potentially lacking flexibility for clusters with intricate shapes.
- KDPC-RkNN vs. CDC: There is no statistically significant difference in clustering performance between KDPC-RkNN and CDC. On the average values and ranks, KDPC-RkNN performs

slightly better than CDC. CDC uses an objective function that includes terms for maximizing diversity and maximizing connectivity to find diverse and connected clusters. Compared with KDPC-RkNN, CDC may not be suitable for high-dimensional data sets because of the exponential increase in time complexity.

In sum, KDPC-RkNN performs significantly better than classical clustering algorithms, such as DBSCAN, K-means++, and K-medoids. Meanwhile, KDPC-RkNN performs better than some DPC-based methods on most datasets, such as DPC-KNN and DPC-DLP. The effectiveness of our new method in clusters with different distributions, shapes, and densities was thoroughly validated by these experimental comparisons on real-world datasets.

5.3 Performance of CED-FS

Since the feature stream has a potentially infinite volume, only a fraction of the entire stream can be processed. Therefore, we use sliding windows to solve the memory constraint problem and detect concept evolution between two adjacent feature windows. The effectiveness of the clustering algorithm KDPC-RkNN can be seen in the above experiments. After obtaining the clusters of two adjacent feature windows, we will illustrate the concept evolution on the nine high-dimensional datasets.

We apply CED-FS on these nine datasets and obtain the results as shown in Table 9, where $E_{(m,n)}$, $D_{(m,n)}$, and $F_{(m,n)}$ indicate the three different types of concept evolution between feature window m and n. In general, the change in the number of clusters indicates the concept evolution between two adjacent feature windows. For datasets Gisette and Arcene, the number of clusters does not change among different feature windows, and there is no concept evolution too. However, the unchanged number of clusters does not necessarily mean that there is no concept evolution. For example, on dataset Mll, the number of clusters remains 3, but there is concept drift between feature windows $(1, 2)$, $(2, 3)$, $(5, 6)$.

Table 9. Concept evolution detection results on nine datasets, where "#Windows" denotes the number of windows and "#Clusters" denotes the number of clusters for each feature window.

Data sets	#Windows	#Clusters	Concept emerging	Concept drift	Concept forgetting
Lung2	3	$\langle 4, 6, 5 \rangle$	$E_{(1,2)}$	$D_{(1,2)}$,$D_{(2,3)}$	$F_{(2,3)}$
Glioma	4	$\langle 5, 5, 4, 4 \rangle$	$E_{(1,2)}$,$E_{(3,4)}$	$D_{(1,2)}$,$D_{(2,3)}$,$D_{(3,4)}$	$F_{(1,2)}$,$F_{(3,4)}$
Gisette	5	$\langle 3, 3, 3, 3, 3 \rangle$	-	-	-
Mll	6	$\langle 3, 3, 3, 3, 3, 3 \rangle$	-	$D_{(1,2)}$,$D_{(2,3)}$,$D_{(5,6)}$	-
Prostate	6	$\langle 6, 3, 3, 4, 4, 5 \rangle$	$E_{(3,4)}$,$E_{(5,6)}$	$D_{(1,2)}$,$D_{(2,3)}$ $D_{(3,4)}$,$D_{(4,5)}$,$D_{(5,6)}$	$F_{(1,2)}$
Dlbcl	6	$\langle 3, 3, 2, 2, 2, 4 \rangle$	$E_{(3,4)}$,$E_{(5,6)}$	$D_{(1,2)}$,$D_{(2,3)}$ $D_{(3,4)}$,$D_{(4,5)}$,$D_{(5,6)}$	$F_{(2,3)}$
Car	9	$\langle$11,12,12, 10,8,12, 8,9,14 $\rangle$	$E_{(1,2)}$,$E_{(2,3)}$ $E_{(3,4)}$,$E_{(4,5)}$$E_{(5,6)}$ $E_{(6,7)}$,$E_{(7,8)}$,$E_{(8,9)}$	$D_{(1,2)}$,$D_{(2,3)}$ $D_{(3,4)}$,$D_{(4,5)}$,$D_{(5,6)}$ $D_{(6,7)}$,$D_{(7,8)}$,$D_{(8,9)}$	$F_{(1,2)}$,$F_{(2,3)}$ $F_{(3,4)}$,$F_{(4,5)}$,$F_{(5,6)}$ $F_{(6,7)}$,$F_{(7,8)}$,$F_{(8,9)}$
Arcene	10	$\langle$2,2,2,2,2, 2,2,2,2,2$\rangle$	-	-	-
Real-sim	21	$\langle$2,3,2,2, 4,2,2,4, 3,2,2,5, 3,2,2,3, 5,2,2,3,3$\rangle$	$E_{(1,2)}$,$E_{(2,3)}$ $E_{(4,5)}$ $E_{(5,6)}$,$E_{(6,7)}$ $E_{(7,8)}$,$E_{(11,12)}$,$E_{(12,13)}$ $E_{(14,15)}$,$E_{(15,16)}$,$E_{(16,17)}$ $E_{(17,18)}$,$E_{(19,20)}$,$E_{(20,21)}$	$D_{(1,2)}$,$D_{(4,5)}$,$D_{(5,6)}$ $D_{(6,7)}$,$D_{(7,8)}$,$D_{(12,13)}$ $D_{(13,14)}$,$D_{(14,15)}$,$D_{(15,16)}$ $D_{(16,17)}$,$D_{(17,18)}$,$D_{(18,19)}$	$F_{(1,2)}$,$F_{(2,3)}$,$F_{(4,5)}$ $F_{(5,6)}$$F_{(6,7)}$,$F_{(7,8)}$ $F_{(8,9)}$,$F_{(9,10)}$$F_{(12,13)}$ $F_{(13,14)}$,$F_{(14,15)}$,$F_{(15,16)}$ $F_{(16,17)}$,$F_{(17,18)}$,$F_{(20,21)}$

Fig. 10 presents the bipartite graph of concept evolution detection on dataset Lung2. It can be easily found that there exists concept emerging ($E_{(1,2)} = \{\langle \emptyset, c_1 \rangle, \langle \emptyset, c_3 \rangle\}$) and concept drift ($D_{(1,2)} = \{\langle c_1, c_2 \rangle, \langle c_2, c_4 \rangle, \langle c_3, c_5 \rangle, \langle c_4, c_6 \rangle\}$) between feature windows W_1 and W_2. Meanwhile, for the two adjacent feature windows W_2 and W_3, the concept evolution includes: concept drift

$D_{(2,3)} = \{\langle c_2, c_2\rangle, \langle c_3, c_1\rangle, \langle c_4, c_3\rangle, \langle c_5, c_4\rangle\}$ and concept forgetting $F_{(2,3)} = \{\langle c_1, \emptyset\rangle\}$. Besides, there is no concept evolution between the concept c_6 in W_2 and concept c_5 in W_3.

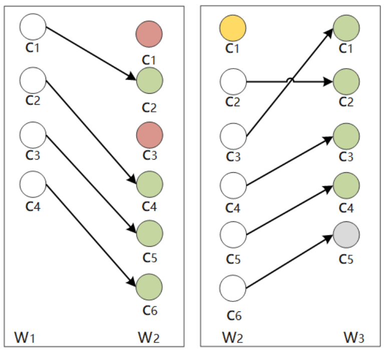


Fig. 10. The bipartite graph of the concept evolution detection on dataset Lung2.

Similarly, we can easily obtain the details of concept evolution on dataset Glioma from Fig. 11. Specifically, we can get concept emerging relations $E_{(1,2)} = \{\langle\emptyset, c_2\rangle\}$, $E_{(3,4)} = \{\langle\emptyset, c_1\rangle\}$, concept drift relations $D_{(1,2)} = \{\langle c_1, c_1\rangle, \langle c_3, c_3\rangle, \langle c_4, c_4\rangle, \langle c_5, c_5\rangle\}$, $D_{(2,3)} = \{\langle c_1, c_1\rangle, \langle c_2, c_1\rangle, \langle c_3, c_2\rangle, \langle c_4, c_3\rangle, \langle c_5, c_4\rangle\}$, $D_{(3,4)} = \{\langle c_1, c_2\rangle, \langle c_2, c_4\rangle\}$, and concept forgetting relation $F_{(3,4)} = \{\langle c_3, \emptyset\rangle\}$.

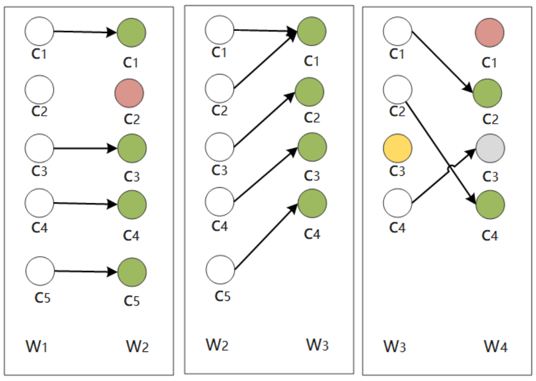


Fig. 11. The bipartite graph of the concept evolution detection on dataset Glioma.

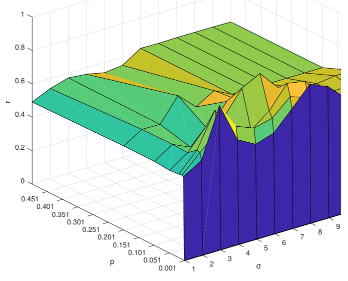
In terms of our proposed CED-FS framework, it can intuitively and effectively analyze and mine the concept evolution over feature streams. As the first work in this field, it will provide meaningful guidance for conceptual analysis and evolutionary studies on streaming data.

5.4 Parameter Analysis of CED-FS

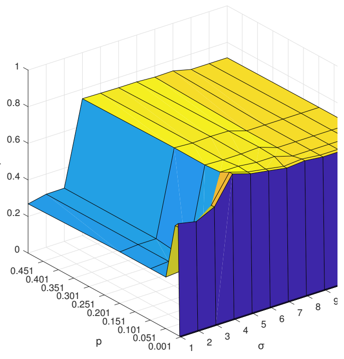
In general, there are four parameters for CED-FS:

- The length of sliding window w: in our experiments, we set $w = 1,000$, which we have already analyzed in Section 4.1;
- The target dimension size d in KPCA: due to the kernel matrix K is $N \times N$, the upper limit of the principal component dimension (d) is N, we set $d = N/3$ as an experience value, where N is the number of instances for each dataset;

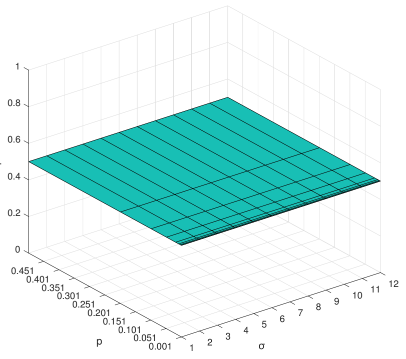
- Kernel function parameter σ in KDPC-RkNN: in terms of the experiment results of Section 5.2, we choose the Gaussian kernel in this and next experimental subsection. Meanwhile, to analyze the optimal values for the nine high-dimensional datasets in Table 5, We choose the value of σ from 1 to 12 with 1 interval;
- The ratio of the nearest neighbors p in KDPC-RkNN: we set the number of nearest neighbors $k = p \times N$. To test the effect of different values of p, we choose p as $\{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ in our experiments.



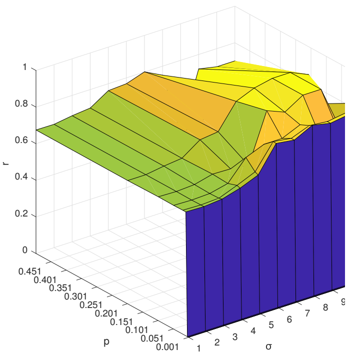
(d) Lung2



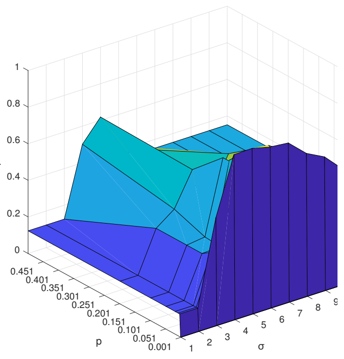
(a) Glioma



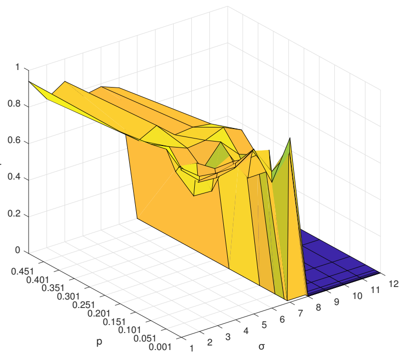
(b) Gisette



(c) Mll



(e) Car



(f) Real-sim

Fig. 12. Illustration of the mean value of R for all feature windows varying with σ and p.

Due to space constraints, Fig. 12 illustrates the clustering performance of KDPC-RkNN on six datasets (Lung2, Glioma, Gisette, Mll, Car, and Real-sim) varying with different values of σ and p, where r indicates the mean value of R for all feature windows ($w = 1,000$ for each feature window). From Fig. 12, we can observe that different datasets have different sensitivities to the parameters. For example, on datasets Glioma, Mll and Lung2, there are more flat areas in the figure than in datasets Car and Real-sim. In general, for parameter p, a smaller value indicates higher performance. This is because a large value of p, which means considering more nearest neighbors, may decrease discrimination. Besides, for parameter σ, different datasets prefer different values. Among these six datasets, it should be noted that on dataset Gisette, the changing of parameters p and σ do

Table 10. Optimal values of parameter σ and p on these nine datasets, where '-' means that the change of these values does not affect the results.

Datasets	σ	p
Lung2	6	0.05
Glioma	4	$\{0.01, 0.02\}$
Gisette	–	–
Mll	$\{9, 10\}$	$\{0.1, 0.2\}$
Prostate	7	0.1
Dlbcl	8	0.1
Car	5	0.005
Arcene	–	0.2
Real-sim	1	0.05

not affect the performance. In other words, for different feature windows, the data distribution is almost not changed. From Table 9, we can also find that there is no concept evolution on dataset Gisette during different feature windows. Table 10 presents the optimal values of parameters σ and p on these nine datasets, which we will use in the subsequent experiments.

6 CONCLUSION

This paper studies the issue of concept evolution detection over feature streams for the first time and proposes a new framework, CED-FS, to handle it. CED-FS consists of a sliding window, an improved DPC-based clustering algorithm, and a weight bipartite graph-based concept evolution detection method. We first give the formal definition of concept evolution detection over feature streams and present a case study on real-world datasets to illustrate the existence of concept evolution. By analyzing the drawbacks of the DPC algorithm, we propose the new clustering algorithm (KDPC-RkNN) based on kernel principal component analysis and reverse k-nearest neighbors. Then, we construct a weight bipartite graph with the concept sets of two adjacent feature windows and detect different types of concept evolution based on the corresponding characteristics. Extensive experiments on synthetic and real-world datasets demonstrate the algorithm's effectiveness. Besides, we applied CED-FS on several high-dimensional datasets and experimentally indicated its ability to detect concept emerging, concept drift, and concept forgetting over feature streams.

Since this is the first work on concept evolution detection over feature streams, there are many issues for further study and improvement. First, some specialized benchmark high-dimension datasets should be constructed for the concept evolution detection task in the future. Secondly, new metrics should be designed to measure the performance of concept evolution detection over feature streams. Thirdly, in this paper, we use a fixed-size sliding window in our framework. Variable-size adaptive sliding windows will be more adaptable to the needs of real-world applications. Fourthly, more efficient clustering algorithms that do not require the number of clusters in advance should be investigated. Finally, we firmly believe that the near future will involve new concept evolution detection methods to gain higher recognition and apply this detection framework to more practical applications.

7 ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China under grants (62376001, 62376002, 62120106008), the Science Foundation of Anhui Province of China under grant 2308085MF215.

REFERENCES

[1] David Arthur and Sergei Vassilvitskii. 2007. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 1027–1035.

[2] Maroua Bahri, Albert Bifet, João Gama, Heitor Murilo Gomes, and Silviu Maniu. 2021. Data stream analysis: Foundations, major tasks and tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 11, 3 (2021), e1405.

[3] Ewan Birney. 2012. Lessons for big-data projects. *Nature* 489, 7414 (2012), 49–51.

[4] Jianguo Chen, Kenli Li, Huigui Rong, Kashif Bilal, Nan Yang, and Keqin Li. 2018. A disease diagnosis and treatment recommendation system based on big data mining and cloud computing. *Information Sciences* 435 (2018), 124–149.

[5] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.

[6] Konstantinos I Diamantaras and Sun Yuan Kung. 1996. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc.

[7] Jiajun Ding, Xiongxiong He, Junqing Yuan, and Bo Jiang. 2018. Automatic clustering based on density peak detection using generalized extreme value distribution. *Soft Computing* 22 (2018), 2777–2796.

[8] Mingjing Du, Shifei Ding, and Hongjie Jia. 2016. Study on density peaks clustering based on k-nearest neighbors and principal component analysis. *Knowledge-Based Systems* 99 (2016), 135–145.

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, Vol. 96. 226–231.

[10] Qihang Fang, Gang Xiong, MengChu Zhou, Tariku Sinshaw Tamir, Chao-Bo Yan, Huaiyu Wu, Zhen Shen, and Fei-Yue Wang. 2022. Process monitoring, diagnosis and control of additive manufacturing. *IEEE Transactions on Automation Science and Engineering* 21, 1 (2022), 1041–1067.

[11] Xiaoli Zhang Fern and Carla E Brodley. 2004. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the twenty-first international conference on Machine learning*. 36.

[12] Gajendra Singh Gurjar and Sharda Chhabria. 2015. A review on concept evolution technique on data stream. In *2015 International Conference on Pervasive Computing (ICPC)*. IEEE, 1–3.

[13] Ben Halstead, Yun Sing Koh, Patricia Riddle, Mykola Pechenizkiy, and Albert Bifet. 2023. Combining diverse meta-features to accurately identify recurring concept drift in data streams. *ACM Transactions on Knowledge Discovery from Data* 17, 8 (2023), 1–36.

[14] Ahsanul Haque, Latifur Khan, Michael Baron, Bhavani Thuraisingham, and Charu Aggarwal. 2016. Efficient handling of concept drift and concept evolution over stream data. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 481–492.

[15] Morteza Zi Hayat and Mahmoud Reza Hashemi. 2010. A DCT based approach for detecting novelty and concept drift in data streams. In *2010 International Conference of Soft Computing and Pattern Recognition*. IEEE, 373–378.

[16] Yi He, Xu Yuan, Sheng Chen, and Xindong Wu. 2021. Online Learning in Variable Feature Spaces under Incomplete Supervision. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. 4106–4114.

[17] Bo Jian Hou, Lijun Zhang, and Zhi Hua Zhou. 2021. Learning with Feature Evolvable Streams. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2021), 2602–2615.

[18] XueGang Hu, Peng Zhou, PeiPei Li, Jing Wang, and XinDong Wu. 2018. A Survey on Online Feature Selection with Streaming Features. *Frontiers of Computer Science* 12, 3 (2018), 479–493.

[19] Jinlong Huang, Qingsheng Zhu, Lijun Yang, Dongdong Cheng, and Quanwang Wu. 2017. QCC: a novel clustering algorithm based on Quasi-Cluster Centers. *Machine Learning* 106, 3 (2017), 337–357.

[20] Wen Jin, Anthony KH Tung, Jiawei Han, and Wei Wang. 2006. Ranking outliers using symmetric neighborhood relationship. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 577–593.

[21] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, and Jerzy Stefanowski. 2014. Open Challenges for Data Stream Mining Research. *SIGKDD Explor. Newsl.* 16, 1 (sep 2014), 1–10.

[22] Mark Last. 2002. Online classification of nonstationary data streams. *Intelligent data analysis* 6, 2 (2002), 129–147.

[23] Haiguang Li, Xindong Wu, Zhao Li, and Wei Ding. 2013. Group feature selection with streaming features. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 1109–1114.

[24] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. 2017. Feature Selection: A Data Perspective. *Acm Computing Surveys* 50, 6 (2017), 1–45.

[25] Zejian Li and Yongchuan Tang. 2018. Comparative density peaks clustering. *Expert Systems with Applications* 95 (2018), 236–247.

[26] Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. 2012. Segmentation using superpixels: A bipartite graph partitioning approach. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 789–796.

[27] Anjin Liu, Jie Lu, Yiliao Song, Junyu Xuan, and Guangquan Zhang. 2022. Concept drift detection delay index. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2022), 4585–4597.

[28] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2019. Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering* 31, 12 (2019), 2346–2363.

[29] Ioannis A Maraziotis, Stavros Perantonis, Andrei Dragomir, and Dimitris Thanos. 2019. K-Nets: Clustering through nearest neighbors networks. *Pattern Recognition* 88 (2019), 470–481.

[30] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M Thuraisingham. 2010. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering* 23, 6 (2010), 859–874.

[31] Mohammad M. Masud, Qing Chen, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. 2010. Addressing concept-evolution in concept-drifting data streams. In *Proceedings - IEEE International Conference on Data Mining, ICDM*. 929–934.

[32] Mohammad M Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani Thuraisingham. 2009. Integrating novel class detection with classification for concept-drifting data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 79–94.

[33] Ujjwal Maulik and Sanghamitra Bandyopadhyay. 2002. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on pattern analysis and machine intelligence* 24, 12 (2002), 1650–1654.

[34] Saad Mohamad, Moamar Sayed-Mouchaweh, and Abdelhamid Bouchachia. 2017. Active learning for data streams under concept drift and concept evolution. *CEUR Workshop Proceedings* 2069 (2017), 1–18.

[35] Hai-Long Nguyen, Yew-Kwong Woon, and Wee-Keong Ng. 2015. A survey on data stream clustering and classification. *Knowledge and information systems* 45, 3 (2015), 535–569.

[36] Le T Nguyen, Ming Zeng, Patrick Tague, and Joy Zhang. 2015. Recognizing new activities with limited training data. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. 67–74.

[37] Hae-Sang Park and Chi-Hyuck Jun. 2009. A simple and fast algorithm for K-medoids clustering. *Expert systems with applications* 36, 2 (2009), 3336–3341.

[38] Dehua Peng, Zhipeng Gui, Dehe Wang, Yuncheng Ma, Zichen Huang, Yu Zhou, and Huayi Wu. 2022. Clustering by measuring local direction centrality for data with heterogeneous density and weak connectivity. *Nature communications* 13, 1 (2022), 5455.

[39] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. 2014. A review of novelty detection. *Signal processing* 99 (2014), 215–249.

[40] Joseph Prusa, Taghi M Khoshgoftaar, and Naeem Seliya. 2015. The effect of dataset size on training tweet sentiment classifiers. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 96–102.

[41] Yikun Qin, Zhu Liang Yu, Chang-Dong Wang, Zhenghui Gu, and Yuanqing Li. 2018. A novel clustering method based on hybrid k-nearest-neighbor graph. *Pattern recognition* 74 (2018), 1–14.

[42] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. 2017. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* 239 (2017), 39–57.

[43] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[44] Kaspar Riesen and Horst Bunke. 2009. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing* 27, 7 (2009), 950–959.

[45] Alex Rodriguez and Alessandro Laio. 2014. Clustering by fast search and find of density peaks. *science* 344, 6191 (2014), 1492–1496.

[46] Amanpreet Kaur Sandhu. 2021. Big data with cloud computing: Discussions and challenges. *Big Data Mining and Analytics* 5, 1 (2021), 32–40.

[47] Jeffrey C Schlimmer and Richard H Granger. 1986. Incremental learning from noisy data. *Machine learning* 1, 3 (1986), 317–354.

[48] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation* 10, 5 (1998), 1299–1319.

[49] Seyed Amjad Seyedi, Abdulrahman Lotfi, Parham Moradi, and Nooruldeen Nasih Qader. 2019. Dynamic graph-based label propagation for density peaks clustering. *Expert Systems with Applications* 115 (2019), 314–328.

[50] Eduardo J Spinosa, André Ponce de Leon F. de Carvalho, and Joao Gama. 2007. Olindda: A cluster-based approach for detecting novelty and concept drift in data streams. In *Proceedings of the 2007 ACM symposium on Applied computing*. 448–452.

[51] Jing Wang, Meng Wang, Peipei Li, Luoqi Liu, Zhongqiu Zhao, Xuegang Hu, and Xindong Wu. 2015. Online feature selection with group structure analysis. *IEEE Transactions on Knowledge and Data Engineering* 27, 11 (2015), 3029–3041.

[52] Xindong Wu, Kui Yu, Wei Ding, Hao Wang, and Xingquan Zhu. 2012. Online feature selection with streaming features. *IEEE transactions on pattern analysis and machine intelligence* 35, 5 (2012), 1178–1192.

[53] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. 2013. Data mining with big data. *IEEE transactions on knowledge and data engineering* 26, 1 (2013), 97–107.

[54] Juanying Xie, Hongchao Gao, Weixin Xie, Xiaohui Liu, and Philip W Grant. 2016. Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors. *Information Sciences* 354 (2016), 19–40.

[55] Shuliang Xu, Lin Feng, Shenglan Liu, and Hong Qiao. 2020. Self-adaption neighborhood density clustering method for mixed data stream with concept drift. *Engineering Applications of Artificial Intelligence* 89 (2020), 103451.

[56] Liu Yaohui, Ma Zhengming, and Yu Fang. 2017. Adaptive density peak clustering based on K-nearest neighbors with aggregating strategy. *Knowledge-Based Systems* 133 (2017), 208–220.

[57] Kui Yu, Wei Ding, Dan A Simovici, Hao Wang, Jian Pei, and Xindong Wu. 2015. Classification with streaming features: An emerging-pattern mining approach. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9, 4 (2015), 1–31.

[58] Qixin Zhang, Zengde Deng, Zaiyi Chen, Haoyuan Hu, and Yu Yang. 2022. Stochastic Continuous Submodular Maximization: Boosting via Non-oblivious Function. In *Proceedings of the 39th International Conference on Machine Learning*, Vol. 162. PMLR, 26116–26134.

[59] Qixin Zhang, Zengde Deng, Xiangru Jian, Zaiyi Chen, Haoyuan Hu, and Yu Yang. 2023. Communication-Efficient Decentralized Online Continuous DR-Submodular Maximization. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*. New York, NY, USA, 3330–3339.

[60] Peng Zhou, Xuegang Hu, Peipei Li, and Xindong Wu. 2019. OFS-density: A novel online streaming feature selection method. *Pattern Recognition* 86 (2019), 48–61.

[61] Peng Zhou, Shu Zhao, Yuanting Yan, and Xindong Wu. 2022. Online scalable streaming feature selection via dynamic decision. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 5 (2022), 1–20.