Accumulating regional density dissimilarity for concept drift detection in data stream

Anjin Liu[a], Jie Lu[a,*], Feng Liu[a], Guangquan Zhang[a]

[a]*Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia*

Abstract

In a non-stationary environment, newly received data may have different knowledge patterns to the data used to train learning models. As time passes, the performance of learning models becomes increasingly unreliable. This problem is known as concept drift and is a common issue in real-world domains. Concept drift detection has attracted increasing attention in recent years, however, hardly any existing methods pay attention to small regional drifts, and their drift detection accuracy may vary due to different statistical significance test. To address these problems, this paper presents a novel concept drift detection method that is based on regional density estimation, named nearest neighbor-based density variation identification (NN-DVI). It consists of three components. The first one is a k-nearest neighbor-based space partitioning schema (NNPS) which transforms unmeasurable discrete data instances into a set of shared subspaces for density estimation. The second one is a distance function that accumulates the density discrepancies in these subspaces and quantifies the overall discrepancies. The last component is a tailored statistical significant test by which the confidence interval of a concept drift can be accurately determined. The distance applied in NN-DVI is sensitive to regional drift, and has been proven to follow a normal distribution. As a result, both the accuracy and false alarm rate of NN-DVI are statistically guaranteed. In addition, several benchmarks have been used to evaluate the method, including both synthetic and real-world datasets. The overall results show that NN-DVI has better performance in terms of addressing concept-drift-detection-related problems.

Keywords: concept drift, dataset shift, changing environments, covariate shift, empirical density estimation

∗. Corresponding author

Email addresses: Anjin.Liu@student.uts.edu.au (Anjin Liu), Jie.Lu@uts.edu.au (Jie Lu), Feng.Liu-2@student.uts.edu.au (Feng Liu), Guangquan.Zhang@uts.edu.au (Guangquan Zhang)

1. Introduction

As technology advances, it has become increasingly easier to collect and organize data from different sources. Details of daily life that were previously unavailable can now be acquired and stored in mobile devices [15]. These seemingly insignificant details convey an enormous amount of valuable information and have certain patterns. When these details come with timestamp, it becomes streaming data. In the absence of outside interference, the patterns of a person's daily routine will not change. However, this assumption usually does not hold [36, 44]. We live in an interactive world. People's daily routines are easily changed due to special events, as well as the impact of companies' or countries' new policies. Learning models trained to discover knowledge patterns have to consider pattern drifts as time shifts [24, 39]. A wide range of machine learning problems need proper solutions to handle such a dynamic environment, for example, personal assistance applications that deal with information filtering, macroeconomic forecasts, bankruptcy prediction and individual credit scoring [15].

The term concept drift in machine learning field refers to a phenomenon in knowledge patterns where data distribution continues to change over time [41]. As concepts change, new data may no longer conform to the patterns induced from historical data [45], and such conflicts will exert a negative impact on subsequent analysis tasks. More importantly, in real-world scenarios, these types of changes are barely perceptible [46, 45]. For this reason, instead of making an assumption in a stationary environment, an effective learning model must always be alert to concept drift, and track and adapt to them quickly [18, 28, 47].

The root causes of concept drift is the variation in the distribution of the underlying data between different time periods. So far, the various concept drift adaptation algorithms that have been developed can be divided into two categories [15, 18, 44, 45] : 1) active drift detection and adaptive learning ; or 2) passive online learning with a forgetting mechanism. Category 1) methods actively detect concept drifts at every time step and react after confirming a drift. They can be further divide into three subcategories [46] : a) data distribution-based drift detection, b) learner output-based drift detection and c) learner parameter-based drift detection. These detection algorithms are also called drift trigger techniques. In contrast, category 2) methods attempt to learn drifts incrementally with each new piece of arriving data, unlike active detection that attempts to detect a drift at each time step, this strategy assists learning models to adapt to new environments gradually [18, 44].

In this paper, we focus on addressing the weaknesses in distribution-based drift detection algorithms. Since these algorithms directly address the root causes of concept drift, and are capable of representing corresponding confidence intervals, they have been reported as the most accurate drift detection methods [14, 34, 45, 53]. Although this type of drift detection algorithm has made remarkable achievements, they still face the following bottlenecks :

1) Regional drifts were not taken into consideration and drift sensitiveness was increased at a cost of increasing false alarms. Existing algorithms detect drifts in terms of the entire sample set, but do not consider any regional

changes in sub-sample sets. As a result, the test statistics of regional drifts may eventually be diluted by stable regions, which decreases sensitivity [24]. Even if the algorithms can successfully capture a distribution drift caused by regional density inequality, they are not able to distinguish whether this drift is caused by a serious regional drift or a moderate global drift ;

2) Existing distribution-based drift detection methods lack tailored statistical significance test. For example, Dasu, et al. [14] used bootstrapping [17] for statistical analysis, and Shao, et al. [53] used the Wilcoxon test. From their experiments, we can see that different significance tests result in different performance outcomes. Statistical analysis is critical to drift detection accuracy, and adequate explanation is indispensable to justify the relationship between significance tests and test statistics [34]. Therefore, to improve the sensitivity of drift detection and to propose a tailored significance test, this paper proposes a novel concept drift detection algorithm, called the nearest neighbor-based density variation identification (NN-DVI). NN-DVI requires no prior knowledge of the data distribution, and instead, estimates the dissimilarity between data sets in terms of instances' neighbors. Compared to other distribution-based drift detection algorithms, the proposed NN-DVI method demonstrates the following advantages :

— It is robust to one-dimensional data, as well as high-dimensional data.
— It is sensitive to concept drift caused by regional density changes and is robust to noise.
— The distribution of the proposed distance is proven theoretically which provides a statistical bound to guarantee the number of false alarms.
— It can describe detected changes by highlighting suspicious regions, and has been tested in real-world applications.

The rest of this paper is organized as follows. In section 2, the problem of concept drift is formally defined and some of the-state-of-art drift detection and adaptation algorithms are introduced. Section 3 introduces the preliminaries. Section 4 formally defines the proposed nearest neighbor-based density variation identification (NN-DVI) algorithm. Section 5 discusses how to select a proper neighborhood to construct the data model. Section 6 details the settings for the experiments and outlines the performance of NN-DVI on both synthetic and real-world datasets. Section 7 concludes this study with a discussion of future work.

2. Literature Review

2.1. Problem Description : Concept Drift and Related Research Topics

Concept drift was first proposed by [52]. It is formally defined as follows : at a time step t, a batch of observations is given, denoted as $S_t = d_1, \ldots, d_m$, where m is the batch size, $d_i = (X_i, y_i)$ is one observation (or a data instance), X_i is the feature vector, y_i is the label, and S_t follows a certain distribution $F_t(X, y)$. Concept drift occurs whenever there is a statistically significant difference between two consecutive observation sets S_t, S_{t+1} that have $F_t(X, y) \neq F_{t+1}(X, y)$, denoted as $\exists t : p_t(X, y) \neq p_{t+1}(X, y)$ [21, 43, 44, 56]. If we further decompose $p_t(X, y)$ into two parts as $p_t(X, y) = p_t(X) \times p_t(y|X)$, we could say there are two sources of concept drift : one is the drift of $p_t(X)$

along with the time variable t, which can also be written as $p(X|t)$; and the other is $p_t(y|X)$, which is the drift of the conditional probability of the feature vector X [46].

Concept drift has been given different names, such as dataset shift [48, 54] or concept shift [42, 50, 56]. Other related terminologies were introduced in [49]'s work. The authors proposing that concept drift or shift is only one subcategory of dataset shift and the dataset shift consists of covariate shift, prior probability shift and concept shift. Moreover, they formally defined that : 1) dataset shift appears when training and testing joint distributions are different which is $\exists t : p_t(X, y) \neq p_{t+1}(X, y)$; 2) covariate shift appears only in $X \rightarrow y$ problems, and the research focus is the drift in $p_t(X)$ while $p_t(y|X)$ remains unchanged ; 3) prior probability shift appears only in $y \rightarrow X$ problems, and the research focus is the drift in $p_t(y)$ while $p_t(X|y)$ remains unchanged ; and 4) concept shift (drift) focuses on the drift of $p_t(y|X)$ while $p_t(X)$ remains unchanged in $X \rightarrow y$ problems and the drift of $p_t(X|y)$ while $p_t(y)$ remains unchanged in $y \rightarrow X$ problems. These definitions clearly state the research scope of each research topic. However, since concept shift (drift) is usually associated with covariate shift and prior probability shift, an increasing number of publications [21, 44, 45, 40] refer to the term concept drift in relation to the problem that $\exists t : p_t(X, y) \neq p_{t+1}(X, y)$. Therefore, we use the term concept drift, instead dataset shift, in this paper.

2.2. Concept Drift Detection and Adaptation Algorithms

A wide range of algorithms has been developed to address concept drift problems. Some popular ensemble learning-based algorithms are the streaming ensemble algorithm (SEA) [55][dynamic weighted majority (DWM) [35], Learner++ NSE [18], leverage bagging [9], Diversity for Dealing with Drifts (DDD) [47], Dynamic adaptation to concept changes (DACC) [30], and Accuracy Updated Ensemble (AUE1, AUE2) [12, 13]. A detailed survey about these algorithms can be found in [22]. These algorithms take the advantages of ensemble learning to minimize the damage caused by concept drift. By changing the weights of committee members (base classifiers), they can smoothly adapt to new concepts. Another large group of concept drift detection algorithms is learner accuracy-based drift detection, which includes the drift detection method (DDM) [18], early drift detection method (EDDM) [4], adaptive windowing (ADWIN) [6], CI-CUSUM [2, 3], exponentially weighted moving average charts for detecting concept drift (ECDD) [51], concept drift detection through resampling [25], and Hoeffding's inequality-based drift detection method (HDDM) [19]. These algorithms track the fluctuations in a learner's error-rate. If the fluctuations exceed the predefined statistical bounds, a drift adaptation process is triggered. Other novel drift adaptation algorithms are : a) recurrence drift adaptation : SAMkNN [44], Just-in-time classifiers for recurrent concepts [1] ; b) active learning algorithms [38] ; c) time dependency algorithms : FISH1, FISH2, FISH3 [37] and some famous concept-drift friendly incremental learners, such as Concept-adapting Very Fast Decision Tree (CVFDT) [27].

These algorithms can effectively detect and adapt to the changes of newly arrived data, once the changes are considered significant. However, in some cases, if a drift happens in a local region, and the overall changes are not

considered significant, such changes will not be addressed. For example, in a spam filtering system, if one user only changes her interest in car sales-related emails and all other topics remain stable, considering car sales-related emails only comprise a very small percentage of all her emails, this small change may not be considered as significant to the overall user's interests.

To address local regional drifts, in some publications [7, 5], the authors applied a decision tree model to detect changes in the online error-rate in each internal tree node to identify drifted nodes, and then updating the nodes respectively. The experimental results showed a good performance in detecting and adapting the drifts. Similar algorithms for regression problems can be found in [20, 28, 29]. However, these algorithms are based on decision tree models, which may not be easy to highlight drifted data instances. In addition, for some decision tree models, constrains may be applied to construct tree nodes. For example, CVFDT normally requires to observe 200 data instances before attempting to split the nodes [27]. If a regional drift occurs within the 200 data instances, the entire nodes will be updated, as a result, regional drifts in this area will not be identified.

2.3. Drift Detection via Non-Parametric Distribution Analysis

In probability theory, the distribution of data is commonly described by a probability density function (PDF). Admittedly, if a data stream fits a certain distribution, the best way of tracking concept drift is to monitor the change in its PDF's parameters [33, 43]. However, this presumption may not always hold. In most cases, it is impossible to abstract a PDF from given data. Compared to tracking the parameter changes of PDFs, mapping data sets to non-parametric models and detecting the difference between these models is a more flexible approach [33, 43].

According to the literature, the first formal treatment of drift detection in data streams was proposed by [34]. In their study, they point out the most natural notion of distance between distributions is total variation as defined by :

$$TV(P_1, P_2) = 2sup_{E\in\varepsilon}|P_1(E) - P_2(E)| \tag{1}$$

or equivalently, when the distributions have the density functions f_1 and f_2

$$dist_{L^1} = \int |f_1(x) - f_2(x)|dx \tag{2}$$

This provides practical guidance on the design of a distance function for distribution discrepancy analysis. Accordingly, [34] proposed a family of distances, called relativized discrepancy, denoted as Ø_A and Ξ_A. In addition, [34] also presents the significance level of the distance according to the number of data instances. The bounds on the probabilities of missed detections and false alarms are theoretically proven, using Chernoff bounds and the Vapnik-Chervonenkis dimension. However, experiment results show that the Ø_A, Ξ_A statistics only outperform other statistics on uniform and normal distribution, while on exponential, binomial and Poisson distributions, Ø_A, Ξ_A performs similar to or even worse than the others. With regard to the data modelling stage, [34] does not propose any novel high-dimensional friendly data models. Instead, they stress that a suitable model choice is an open problem.

Later, [14, 46] published another two distribution-based algorithms for concept drift detection. In [14], a kdqTree-based space partitioning algorithm was exploited to estimate data distributions empirically. Intuitively, each cell (a leaf node on the kdqTree) can be seen as a bin of the empirical distribution. By dividing the number of instances located in a cell by the total number of instances, an estimated empirical-PDF (epdf) can be acquired. They then adopt the Kullback-Leibler divergence as the test statistics. Their statistical significance test is a form of bootstrapping. In [46], the authors proposed an enhanced competence model and applied an empirical distance as the test statistics. This statistical significance test is a non-parametric permutation test. According to their experiments, the competence model achieved much better results. Nevertheless, regional concept drift has still not been resolved.

3. Preliminary

An important element of this research is the multiset theory [11]. In contrast to classic set theory, multiset theory defines a multiset (or bag) as a collection of elements in which duplicates of the elements are allowed. The multiplicity of an element, denoted as $m(x)$, is the count of element x in a multiset. The cardinality is the total number of elements in a multiset, including repeated memberships. For example, in the multiset $A = \{a, a, b, c, c, c\}$, the multiplicities of the members a, b and c are respectively 2, 1 and 3, and the cardinality of A is 6. A multiset can be formally defined as :

Definition 1. [11]($Multiset$) A multiset is defined as a 2-tuple $(A,\ m)$ where A is a set of elements and $m : A \to \mathbb{N}^+$ is the multiplicity function from A to the set $\mathbb{N}^+ = \{1, 2, 3, \ldots\}$. Formally, a multiset is denoted as

$$\mathbf{M} = \{(a, m(a)) : a \in A, m : A \to \mathbb{N}_{\geq 1}\} \tag{3}$$

The multiplicity operations involved in this Chapter are listed below.

Definition 2. [11]($Multiset\ Indicator\ Function$) Indicator function $\boldsymbol{I}_A : X \to \mathbb{N}$, is defined by

$$\boldsymbol{I}_A(x) = \begin{cases} m(x), & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases} \tag{4}$$

Definition 3. [11]($Multiset\ Intersection$) Intersection indicator function of multiset A and B on element x is

$$\boldsymbol{I}_{A \cap B}(x) = \min\{\boldsymbol{I}_A(x), \boldsymbol{I}_B(x)\} \tag{5}$$

Definition 4. [11]($Multiset\ Union$) Union indicator function of multiset A and B on element x is

$$\boldsymbol{I}_{A \cup B}(x) = \max\{\boldsymbol{I}_A(x), \boldsymbol{I}_B(x)\} \tag{6}$$

Definition 5. [11]($Multiset\ Difference$) The set difference indicator function of multiset A to B on element x is

$$\boldsymbol{I}_{A \setminus B}(x) = \max\{0, \boldsymbol{I}_A(x) - \boldsymbol{I}_B(x)\} \tag{7}$$

Definition 6. [11]($Multiset\ Element\ Sum$) The sum indicator function of multiset A and B on element x is

$$\boldsymbol{I}_{A \uplus B}(x) = \boldsymbol{I}_A(x) + \boldsymbol{I}_B(x) \tag{8}$$

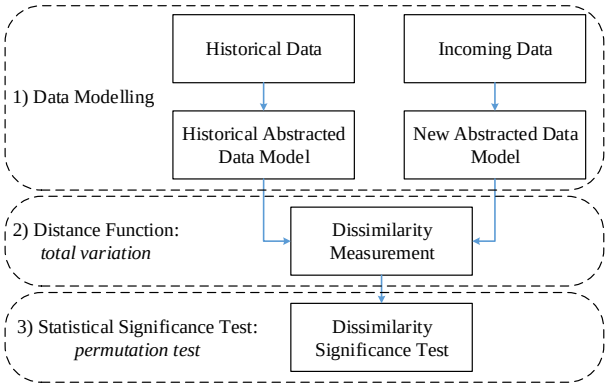

FIGURE 1: A general framework for distribution-oriented concept drift detection

Definition 7. [11]($Multiset\ Cardinality$) The cardinality of a multiset A is the sum of the indicator function values

$$|A| = \sum_{x \in X} \boldsymbol{I}_A(x) \tag{9}$$

where X is the set of unique element in A

These multiplicity functions provide the basic set of operators for our algorithm.

4. Nearest Neighbor-based Density Variation Identification

Data distribution-based concept drift detection has been reported as the most sensitive and convincing detection method [46, 45] because it directly addresses the root causes of concept drift and can represent the corresponding confidence interval. According to the literature, a typical distribution-based detection method consists of three components. The first component is a data representation model through which critical information is retrieved and irrelevant details are discarded. The second component is a specific dissimilarity function designed to measure the discrepancies between the data models. One of the most natural notions for the distance between distributions is the total variation or the L^1 norm [33]. The third component is a statistical significance test. Statistical significance, namely the p-value, is the probability of obtaining the least extreme result given that a null hypothesis is true. In drift detection, the null hypothesis is true when the detected discrepancies are not caused by concept drift. As a non-parametric test, a permutation test is a good option to empirically estimate statistical significance. An overall framework for distribution-oriented concept drift detection is summarized in Figure 1.

In this section, we formally present the proposed drift detection method (NN-DVI). NN-DVI consists of three parts. The first part is a data modeling approach which retrieves critical information and presents data sets as an abstracted model. This is explained in detail in Section 4.1. The second part is a distance function that accumulates regional density changes to quantify the overall discrepancy between data sets. Section 4.2 gives the definition and theorems related to this distance. The third part is a tailored statistical significant test for the distance. In Section 4.3,

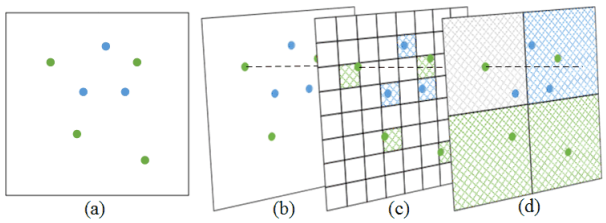

FIGURE 2: Conventional space partitioning schema maps data instances into larger bins, and uses these bins to measure the similarity between datasets. Different bin sizes will give different answers. For example, the bin size of Figure 2 (c) is too small so that each block only contains one instance. As a result, the intersection set of green dots and blue dots is empty. However, in Figure 2 (d), the intersection set has 2 elements.

we further discuss the property of the distance and theoretically prove its distribution. Finally, Section 4.4 provides the implementation details of NN-DVI.

4.1. Modelling Data as a Set of High-Resolution Partitions

Space partitioning is defined as the process of dividing a feature space into two or more disjoint subsets. Any data instance in the feature space can then be identified as residing in exactly one of the subsets. When dealing with discrete datasets, it is often infeasible to directly estimate the similarity between data instances [14]. One of the most popular solutions is to accumulate the differences of empirical probability density through the divided non-overlapping subsets [14]. For example, a histogram is one of the most popular ways to assess the probability distribution of a given variable by depicting the frequencies of observations occurring in certain ranges of values (bins), where the bins are the disjoint subsets. In concept drift problems, if two sample sets are drawn from an identical distribution, every partition should have a similar number of data points, namely the empirical probability density ($\frac{num\ of\ data\ points\ in\ one\ partition}{total\ num\ of\ data\ points}$) in each partition will be similar. Otherwise, a statistical significant difference between their empirical probability density will be found.

Currently, in related research, space partitioning methods directly contribute to the accuracy of drift detection [45]. As shown in Figure 2, if a partitioning schema cannot explicitly map similar items into the same partitions, like the 8×8 partitioning schema, the similarity between sample sets will always be zero. As a result, the detected density change will be invalid. By the same token, if a partitioning schema mistakenly maps non-similar items into the same partitions, the dissimilarity between testing sample sets will always be zero. A drift detection algorithm will lose its sensitivity to density drifts, especially when only a few data instances are available for evaluating the partitioning schema. Optimizing space partitioning methods for concept drift detection is still an open problem [34].

Motivated by this issue, we propose a novel nearest neighbor-based partitioning schema (NNPS). The fundamental idea behind the NNPS is to find the minimum shared particles between instances instead of the shared partitions to which instances belong. Instead of mapping a data instance into a lower granularity presented by partitions or bins, expanding the data points into a hypersphere can preserve much more of the instance's details. For example, as shown in Figure 3(b), in terms of a two-dimensional feature space, the expanded data points are indicated by blue and

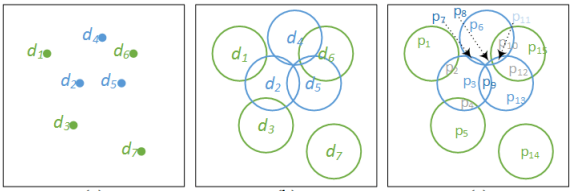

FIGURE 3: The proposed instance-oriented space partitioning schema aims to find the primary elements that consist of data instances, and then use these elements to measure the similarity between datasets. In a 2d domain, let us consider each data instance as a circle rather than a dot and the entire domain is a by n×n pixels square, as shown in Figure 3(b). Then the pixels are the primary elements which constitute the data instances, and data instances can be represented by a set of pixels located in their circles. Therefore, the similarity between instances can be simply estimated by counting the shared pixels located in the overlapping regions, such as the similarity between d_1 and d_2 can be estimated by $\frac{|p_2|}{|p_1|+|p_2|+|p_3|}$, where $|p_1|$ indicates the number of pixels in region p_1 as shown in Figure 3(c)

green circles. The partitions are the non-overlapping regions, as shown in Figure 3(c), $\{p_1, \ldots, p_{15}\}$, and the instance particles are the pixels of the figure. Then, the probability density discrepancies can be estimated by accumulating the number of overlapped pixels, namely the overlapping area.

This partitioning schema performs well with two-dimensional data. However, it becomes increasingly complex as the dimensionality increases. This is because the intersecting regions between hyperspheres are difficult to explicitly calculate in high dimensional space. Therefore, to further optimize our partitioning schema, we use a k-nearest neighbor model to replace the hyperspheres model.

An intuitive explanation for how a nearest neighbor model describes these instance particles is that close located data instances have hidden connections, and such connections can be used as the most basic element to constitute data instances. As shown in Figure 4(d), unlike conventional partitioning schemas, which group instances into different partitions, NNPS slices an instance into several particles, and uses these particles to constitute instances. An overview of the difference between the NNPS and conventional partitioning schemas is shown in Figure 4(e). To overcome the bias caused by high granule mapping, we propose our NNPS to extend instances into a more detailed granule. This process is also called instance discretization or instance quantization. Instead of measuring similarities in a conventional space partitioning schema, applying shared instance particles can pass more instance details to the sample sets, thereby making similarity measures more sensitive to small changes. The concept of instance particles is formally defined as follows.

Definition 8. $(Instance\ Particle)$ Given a $d_i \in \mathbf{D}$, if the set $\mathcal{K}_i$ contains all neighbors of d_i, then an instance particle based on d_i is defined as $\mathrm{p}_{d_i} = (d_i, \mathcal{K}_i)$.

Example 1. Denote the data sample set as $\mathbf{D} = \{d_1, d_2, d_3, d_4\}$ and the neighbor sets are $\mathcal{K}_1 = \{d_1, d_2\}$, $\mathcal{K}_2 = \{d_1, d_2, d_3\}$, $\mathcal{K}_3 = \{d_2, d_3, d_4\}$, $\mathcal{K}_4 = \{d_3, d_4\}$. Then the instance particles are :

$$\begin{cases} \mathrm{p}_{d_1} = \mathrm{p}(d_1, \mathcal{K}_1) = (d_1, \{d_1, d_2\}) \\ \mathrm{p}_{d_2} = \mathrm{p}(d_2, \mathcal{K}_2) = (d_2, \{d_1, d_2, d_3\}) \\ \mathrm{p}_{d_3} = \mathrm{p}(d_3, \mathcal{K}_3) = (d_3, \{d_2, d_3, d_4\}) \\ \mathrm{p}_{d_4} = \mathrm{p}(d_4, \mathcal{K}_4) = (d_4, \{d_3, d_4\}) \end{cases} \tag{10}$$

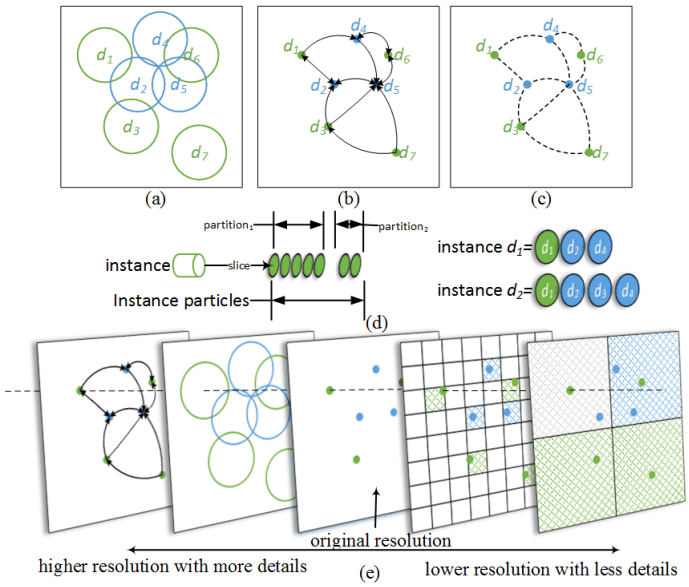

FIGURE 4: Using a k-nearest neighbor data presenting model to replace the hyperspheres model, the arrow in (b) indicates the relationship of k-nearest neighbors. For example, $d_1 \rightarrow d_4$ indicates that d_4 is a knn member of d_1 while d_1 is not a knn member of d_4. As long as there is a connection between two data instances, we consider that they are neighbors. Then the datasets can be presented as a set of connections, as shown in (c). If we consider each connection as one component of a data instance, namely a slice of instance, then the data instance d_1 is a composite of one slice of itself, one slice from d_2 and one slice from d_4, as shown in (d). Then, (e) illustrates the difference between conventional partitioning methods and the NNPS

The proposed NNSP breaks one data instance into a set of instance particles, thereby transforming each discrete data instance into a set of shared instance particles. As a result, the differences between data instances can be preserved for measuring the distance between sample sets.

Definition 9. ($Instance\ Particle\ Group$) Given an instance $d_k \in \mathbf{D}$, the particle group of d_k is defined as $\mathbf{P}(d_k) = \{\mathrm{p}_{d_i} : \text{if } d_k \in \mathcal{K}_i, i = 1, 2, \ldots, |\mathbf{D}|\}$, where $\mathrm{p}_{d_i} = (d_i, \mathcal{K}_i)$.

Example 2. Referring to example 1, the data instances belonging to $\mathbf{D}$ represented by the instance particles are :

$$\begin{cases} \mathbf{P}(d_1) = \{\mathrm{p}_{d_1}, \mathrm{p}_{d_2}\} \\ \mathbf{P}(d_2) = \{\mathrm{p}_{d_1}, \mathrm{p}_{d_2}, \mathrm{p}_{d_3}\} \\ \mathbf{P}(d_3) = \{\mathrm{p}_{d_2}, \mathrm{p}_{d_3}, \mathrm{p}_{d_4}\} \\ \mathbf{P}(d_4) = \{\mathrm{p}_{d_3}, \mathrm{p}_{d_4}\} \end{cases} \tag{11}$$

Definition 10. ($Instances\ Particle\ Group$) Given a sample set $S_k \subseteq \mathbf{D}$, the particle group of S_k is defined as $\mathbf{P}(S_k) = \{\bigcup_{d_i \in S_k} \mathbf{P}(d_i)\}$, where $\mathbf{P}(d_i)$ is the particle group of d_i.

Example 3. Referring to example 1, and given two sample sets $S_1 = \{d_1, d_2\}$, $S_2 = \{d_3, d_4\}$, then we have

$$\begin{cases} \mathbf{P}(S_1) = \mathbf{P}(d_1) \cup \mathbf{P}(d_2) = \{\mathrm{p}_{d_1}, \mathrm{p}_{d_2}, \mathrm{p}_{d_3}\} \\ \mathbf{P}(S_2) = \mathbf{P}(d_3) \cup \mathbf{P}(d_4) = \{\mathrm{p}_{d_2}, \mathrm{p}_{d_3}, \mathrm{p}_{d_4}\} \end{cases} \tag{12}$$

Since the data instances can be represented as a set of instance particles, this offers a higher resolution to calculate the distance between the sample sets. The next step is to even out the weight of the instance particles and the weight of instances represented by instance particles. As shown in Example 2, the number of particles in a data instance may

differ from instance to instance, $|\mathbf{P}(d_1)| = 2$, $|\mathbf{P}(d_2)| = 3$, $|\mathbf{P}(d_3)| = 3$, $|\mathbf{P}(d_4)| = 2$, resulting in inconsistencies between the instances' weight. To even out the weight of instance particles and instances simultaneously, we introduce the lowest common multiple (LCM), $LCM(\{|\mathbf{P}(d_j)| : \forall d_j \in \mathbf{D}\})$ as the multiplicity function. Then a uniform weighted data instance can be represented by a multiset of particles.

Definition 11. $(Instance\ Particle\ Set)$ Given an instance $d_k \in \mathbf{D}$, a multiset of instance particles of d_k in terms of $NNPS$ is defined as

$$\mathbf{M}_{d_k}^{nnps} = \{(\mathrm{p}_{d_i}, m(\mathrm{p}_{d_i})) : \mathrm{p}_{d_i} \in \mathbf{P}(d_k), m(\mathrm{p}_{d_i}) = \frac{Q}{|\mathbf{P}(d_k)|}\} \tag{13}$$

where Q is the lowest common multiple of $\{|\mathbf{P}(d_j)| : \forall d_j \in \mathbf{D}\}$.

Example 4. Referring to Example 1, we have multiset represented data instances :

$$\begin{cases} \mathbf{M}_{d_1}^{nnps} = \{(\mathrm{p}_{d_1}, 3), (\mathrm{p}_{d_2}, 3)\} \\ \mathbf{M}_{d_2}^{nnps} = \{(\mathrm{p}_{d_1}, 2), (\mathrm{p}_{d_2}, 2), (\mathrm{p}_{d_3}, 2)\} \\ \mathbf{M}_{d_3}^{nnps} = \{(\mathrm{p}_{d_2}, 2), (\mathrm{p}_{d_3}, 2), (\mathrm{p}_{d_4}, 2)\} \\ \mathbf{M}_{d_4}^{nnps} = \{(\mathrm{p}_{d_3}, 3), (\mathrm{p}_{d_4}, 3)\} \end{cases} \tag{14}$$

Remark. The multiplicity function can be generalized as $LCM(\{w \cdot |\mathbf{P}(d_j)| : \forall d_j \in \mathbf{D}\})$, where w is the weight of instance d_i and $w \in \mathbb{N}$. If a data set is uniformly weighted, the w will equal to one.

Theorem 1. $\mathbf{M}_{d_i}^{nnps} = \mathbf{M}_{d_j}^{nnps}$ *if and only if* $\mathbf{P}(d_i) = \mathbf{P}(d_j)$

Proof. *Obviously*

Accordingly, a data sample set S represented by instance particles is a cumulative multiset sum of $\mathbf{M}_{d_k}^{nnps}$ that $\forall d_k \in S$, namely $\mathbf{M}_S^{nnps} = \uplus_{d_k \in S} \mathbf{M}_{d_k}^{nnps}$

Definition 12. $(Instances\ Particle\ Set)$ Given a sample set $S \subseteq \mathbf{D}$, a multiset of instance particles of S in terms of $NNPS$ is defined as

$$\mathbf{M}_S^{nnps} = \{(\mathrm{p}_{d_i}, m(\mathrm{p}_{d_i})) : \mathrm{p}_{d_i} \in \cup_{d_k \in S} \mathbf{P}(d_k), m(\mathrm{p}_{d_i}) = \sum_{d_k \in S} I_{M_{d_k}^{nnps}}(\mathrm{p}_{d_i})\} \tag{15}$$

Example 5. Referring to Example 1 and 3, we have multiset represented sample sets :

$$\begin{cases} \mathbf{M}_{S_1}^{nnps} = \mathbf{M}_{d_1}^{nnps} \biguplus \mathbf{M}_{d_2}^{nnps} = \{(\mathrm{p}_{d_1}, 5), (\mathrm{p}_{d_2}, 5), (\mathrm{p}_{d_3}, 2)\} \\ \mathbf{M}_{S_2}^{nnps} = \mathbf{M}_{d_3}^{nnps} \biguplus \mathbf{M}_{d_4}^{nnps} = \{(\mathrm{p}_{d_2}, 2), (\mathrm{p}_{d_3}, 5), (\mathrm{p}_{d_4}, 5)\} \end{cases} \tag{16}$$

Theorem 2. *Given a data instance* d_i *and a data sample set* S*, if* $d_i \in S$*, then* $\mathbf{M}_{d_i}^{nnps} \subseteq \mathbf{M}_S^{nnps}$

Proof. *Obviously*

Corollary 1. *Given a sample set* $S_i \subseteq \mathbf{D}$*, then* $\mathbf{M}_{S_i}^{nnps} \subseteq \mathbf{M}_{\mathbf{D}}^{nnps}$

Proof. *Obviously*

In summary, NNPS aims to extend instances into a lower granularity to provide a more detailed shared subspace for measuring similarity and dissimilarity. In NNPS, each data instance is a partition, and the hidden relationships between

instances are the primary elements that lie in the regions. Compared to conventional "bin"-based partitioning methods, which only consider the similarity between data instances as 1 (located in the same bin) or 0 (located in different bins), NNPS can preserve the similarity between data instances, therefore becoming more sensitive to small discrepancies. In addition, the datasets presented by NNPS are the accumulation of instance particles from every partition. As a result, the density discrepancies in partitions will also be accumulated and reflected in the similarity or dissimilarity measurement.

4.2. The Distance Measurement of Sample Sets in terms of NNPS

Since data instances and datasets are now presented by multisets, the distance between them can be quantified by set-related metrics in terms of Definition 1, Definition 2, Definition 3, Definition 4, Definition 6, Definition 5, Definition 7. The distance function applied in this paper quantifies the number of different instance particles between two given multisets, denoted as d^{nnps}.

Definition 13. ($NNPS$ *dissimilarity measurement*) Given two sample sets A,$B \subseteq \mathbf{D}$, the $NNPS$-based distance between them is calculated by accumulating the differences in the number of instance particles, and it can be normalized by dividing the number of unique instance particles, denoted as

$$d^{nnps}(\mathbf{M}_A^{nnps}, \mathbf{M}_B^{nnps}) = \frac{1}{|\mathbf{P}(A) \cup \mathbf{P}(B)|} \sum_{\mathrm{p}_{d_i} \in \mathbf{P}(A) \cup \mathbf{P}(B)} \frac{|\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) - \boldsymbol{I}_{\mathbf{M}_B^{nnps}}(\mathrm{p}_{d_i})|}{\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) + \boldsymbol{I}_{\mathbf{M}_B^{nnps}}(\mathrm{p}_{d_i})} \tag{17}$$

Example 6. Referring to Example 5, we have the

$$\begin{aligned} d^{nnps}(\mathbf{M}_{S_1}^{nnps}, \mathbf{M}_{S_2}^{nnps}) &= \frac{1}{|\mathbf{P}(S_1) \cup \mathbf{P}(S_2)|} \cdot \left(\frac{|\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_1}) - \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_1})|}{\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_1}) + \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_1})} \right. \\ &+ \frac{|\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_2}) - \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_2})|}{\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_2}) + \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_2})} + \frac{|\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_3}) - \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_3})|}{\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_3}) + \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_3})} + \left. \frac{|\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_4}) - \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_4})|}{\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_4}) + \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_4})} \right) \\ &= \frac{1}{4}\left(\frac{|5-0|}{5} + \frac{|5-2|}{7} + \frac{|2-5|}{7} + \frac{|0-5|}{5}\right) = \frac{5}{7} \end{aligned} \tag{18}$$

4.3. A Tailored Statistical Significance Test for d^{nnps}

Measuring the difference between two sample sets is only one aspect of detecting concept drift. Another aspect is to provide adequate evidence to determine how likely it is that there will be such a difference if the given sample sets are drawn from an identical distribution. Null hypothesis testing, or the so-called p-$value$ approach, is widely used to address similar problems. In our case, the null hypothesis H_0 is : "It is very likely to observe such a d^{nnps}, if two given sample sets are independently drawn from the same distribution". The smaller the probability (the p-$value$), the stronger the evidence against H_0.

The most intuitive way to achieve this goal is using the Monte Carlo permutation test [16]. The achieved significance level (ASL), also known as the p-$value$, can be attained by observing all possible values of the test statistics. A

permutation test holds no assumptions for the test statistics, however, as the sample size increases, it become unfeasible to calculate the exact ASL, and it can only be roughly estimated which may increase the error rate.

Alternatively, if a given test statistics fit a known distribution, then the decision regions (to accept/reject the null hypothesis) can be obtained explicitly from its distribution function. In this research, according to Definition 13, we prove that the d^{nnps} of two i.i.d. sample sets fits a normal distribution. As a result, we can use max likelihood to estimate the mean and variance of its distribution and determine the critical region with less Monte Carlo error.

Lemma 1. *$\forall A$ and $B \subseteq \mathbf{D}$, if $\mathrm{p}_{d_i} \in \mathbf{P}(A) \cup \mathbf{P}(B)$, then the random variable $\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) \sim N(\mu_i, \sigma_i^2)$ if the following conditions can be satisfied :*

1. *$|\mathbf{D}| \to \infty$;*
2. *there is no elements $d_{k_0} \in \mathbf{D}$, such that $\boldsymbol{I}_{\mathbf{M}_{d_{k_0}}^{nnps}}(\mathrm{p}_{d_i}) \gg \boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i}), k = 1, \ldots, k_0 - 1, k_0 + 1, \ldots, |\mathbf{D}|$, where $A \cup B = \mathbf{D}$ and $A \cap B = \emptyset$.*

Proof. *Because $\boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i})$ is unchangeable no matter what A is selected ($k = 1 \ldots, |\mathbf{D}|$), $\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) = \sum_{d_k \in A} \boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i})$ is only related to elements in A. Considering a random variable I_k, expressed as follows,*

$$I_k = \begin{cases} \boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i}), & \text{if } d_k \in A \\ 0, & \text{otherwise.} \end{cases}, k = 1, \ldots, |\mathbf{D}|$$

so we have $\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) = \sum_{k=1}^{|\mathbf{D}|} I_k$ and all of $I_k (k = 1, \ldots, |D|)$ are independent. For each I_k we can express its distribution as follows :

I_k	$\boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i})$	*0*
p	$p_{d_i \in A}$	$p_{d_i \in B}$

where $p_{d_i \in A}$ is the probability that data instance d_i belongs to sample set A. Because $A \cup B = \mathbf{D}$ and $A \cap B = \emptyset$, we have $p_{d_i \in A} + p_{d_i \in B} = 1$. Thus, $\mathbf{E}(I_k) = \boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i}) \cdot p_{d_i \in A}$ and $\mathbf{Var}(i_k) = \boldsymbol{I}^2_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i}) \cdot [p_{d_i \in A} \cdot (1 - p_{d_i \in A})^2 + p_{d_i \in B} \cdot p^2_{d_i \in A}]$. Because Condition 2 can be satisfied and L^2 norm is grater than L^3 norm when $|\mathbf{D}| \to \infty$, meaning that

$$\lim_{|\mathbf{D}| \to \infty} \frac{\sum_{k=1}^{|\mathbf{D}|} \mathbf{E}\Big(\big(I_k - \mathbf{E}(I_k)\big)^3\Big)}{s^3_{|\mathbf{D}|}} = 0, \; s^2_{|\mathbf{D}|} = \sum_{k=1}^{|\mathbf{D}|} \mathbf{Var}(I_k)$$

based on Lyapunov central limit theorems, when $|\mathbf{D}| \to \infty$, we have

$$\frac{1}{s_{|\mathbf{D}|}} \sum_{k=1}^{|\mathbf{D}|} (I_k - \mathbf{E}(I_k)) \to N(0, 1)$$

so, $\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) = \sum_{k=1}^{|\mathbf{D}|} I_k \sim N(\mu_i, \sigma_i^2)$.

Remark. In this paper, we assume that instances in time windows are independent. So the constructed random variable I_k is independent.

Theorem 3. *$\forall A$ and $B \subseteq \mathbf{D}$, if $\mathrm{p}_{d_i} \in \mathbf{P}(A) \cup \mathbf{P}(B)$, then the random variable $(\boldsymbol{I}_{\mathbf{M}_A^{nnps}(\mathrm{p}_{d_i})} - \boldsymbol{I}_{\mathbf{M}_B^{nnps}(\mathrm{p}_{d_i})}) \sim N(\mu_i, \sigma_i^2)$ if following conditions can be satisfied :*

1. *$|\mathbf{D}| \to \infty$;*
2. *there is no elements $d_{k_0} \in \mathbf{D}$, such that $\boldsymbol{I}_{\mathbf{M}_{d_{k_0}}^{nnps}}(\mathrm{p}_{d_i}) \gg \boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i}), k = 1, \ldots, k_0 - 1, k_0 + 1, \ldots, |\mathbf{D}|$, where $A \cup B = \mathbf{D}$ and $A \cap B = \emptyset$.*

Proof. *Based on Lemma 1,* $\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i})$ *and* $\boldsymbol{I}_{\mathbf{M}_B^{nnps}}(\mathrm{p}_{d_i})$ *satisfy normal distributions, therefore* $(\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) - \boldsymbol{I}_{\mathbf{M}_B^{nnps}}(\mathrm{p}_{d_i})) \sim N(\mu_i, \sigma_i^2)$.

Theorem 4. $\forall A$ *and* $B \subseteq \mathbf{D}$, *if* $\mathrm{p}_{d_i} \in \mathbf{P}(A) \cup \mathbf{P}(B)$, *then* $d^{nnps}(\mathbf{M}_A^{nnps}, \mathbf{M}_B^{nnps}) \sim N(\mu, \sigma^2)$ *if following conditions can be satisfied*

1. $|\mathbf{D}| \to \infty$;
2. *there is no elements* $d_{k_0} \in \mathbf{D}$, *such that* $\boldsymbol{I}_{\mathbf{M}_{d_{k_0}}^{nnps}}(\mathrm{p}_{d_i}) \gg \boldsymbol{I}_{\mathbf{M}_{d_k}^{nnps}}(\mathrm{p}_{d_i}), k = 1, \ldots, k_0 - 1, k_0 + 1, \ldots, |\mathbf{D}|$, *where* $A \cup B = \mathbf{D}$ *and* $A \cap B = \emptyset$.

Proof. *Based on the definition of* $d^{nnps}(\mathbf{M}_A^{nnps}, \mathbf{M}_B^{nnps})$, *Definition 13, and Theorem 3, we know* $d^{nnps}(\mathbf{M}_A^{nnps}, \mathbf{M}_B^{nnps})$ *is the sum of* $|\mathbf{D}|$ *independent half-normal distributions. Thus, we need to verfy whether the sum of half-normal distributions can satisfy the Lyapunov condition. We denote* σ_i *as the parameter of the* i^{th} *half-normal distribution and* $r_i = |\boldsymbol{I}_{\mathbf{M}_A^{nnps}}(\mathrm{p}_{d_i}) - \boldsymbol{I}_{\mathbf{M}_B^{nnps}}(\mathrm{p}_{d_i}) - \mu_i| \sim HalfN(0, \sigma_i^2)$ *as the* i^{th} *random variable. We have*

$$\mathbf{E}(r_i^3) = \int_0^{+\infty} \frac{\sqrt{2}}{\sigma_i\sqrt{\pi}} \cdot e^{-\frac{r_i^2}{2\sigma_i^2}} \cdot r_i^3 \, dr_i = \frac{\sqrt{2}}{\sigma_i\sqrt{\pi}} \int_0^{+\infty} e^{-\frac{r_i^2}{2\sigma_i^2}} \cdot r_i^3 \, dr_i$$

using integration by parts, we arrive at the following equations

$$\mathbf{E}(r_i^3) = \frac{\sigma_i\sqrt{2}}{\sqrt{\pi}} \int_0^{+\infty} e^{-\frac{r_i^2}{2\sigma_i^2}} \cdot \Big(-\frac{r_i^1}{\sigma_i^2} \Big) \cdot (-r_i^2) \, dr_i = \frac{\sigma_i\sqrt{2}}{\sqrt{\pi}} \left(e^{-\frac{r_i^2}{2\sigma_i^2}} \Big|_0^{+\infty} + 2 \int_0^{+\infty} e^{-\frac{r_i^2}{2\sigma_i^2}} \cdot r_i \, dr_i \right)$$

thus,

$$\mathbf{E}(r_i^3) = \frac{\sigma_i 2\sqrt{2}}{\sqrt{\pi}} (-\sigma_i^2) \cdot e^{-\frac{r_i^2}{2\sigma_i^2}} \Big|_0^{+\infty} = \frac{2\sqrt{2}}{\sqrt{\pi}} \sigma_i^3$$

Because L^2 *norm is greater than* L^3 *norma when* $|\mathbf{D}| \to \infty$,

$$\lim_{|\mathbf{D}| \to \infty} \frac{\sum_{i=1}^{|\mathbf{D}|} \mathbf{E}(r_i^3)}{s_{|\mathbf{D}|}^3} = 0, \; s_{|\mathbf{D}|}^2 = \sum_{i=1}^{|\mathbf{D}|} \sigma_i^2$$

Hence $\sum_{i=1}^{|\mathbf{D}|} r_i$ *satisfies the Lyapunov condition, meaning* $d^{nnps}(\mathbf{M}_A^{nnps}, \mathbf{M}_B^{nnps}) \sim N(\mu, \sigma^2)$.

4.4. Stream Learning with Nearest Neighbor-based Density Variation Identification

In this section, we explain the implementation of NN-DVI from the computer logic perspective. The overall stream learning algorithm is given in Algorithm 1 and illustrates how NN-DVI is integrated with online learning models. Then, the implementation of the NN-DVI and the calculation of d^{nnps} are given in Algorithm 2 and Algorithm 3, respectively.

The proposed algorithm considers data stream learning as two scenarios. One is concept drift in which a drift adaptation process has to be initialized to correct the entire system. The other is static online learning which requires no intervention from an extra process. In the first scenario, the proposed algorithm applies a sliding windowing strategy to detect concept drift. The sliding window is initialized with training data or the first w_{min} elements in the data stream, where w_{min} is an input parameter used to decide the minimum drift detection window size. This process is implemented in Algorithm 1, lines 5-13, where the win_{fix} is the fixed time window while win_{slide} is the sliding time

Algorithm 1: Stream Learning with NN-DVI

input : training data, $\mathbf{D}_{train}$
data stream, $\mathbf{D} = \{d_0, \ldots, d_n\}$
min drift detection window size, w_{min} (default $w_{min} = 100$)
base learner, L (default L is set as Naive Bayes Classifier)
distance function, (default $dist(d_i, d_j)$ is set as Euclidean distance)
k value for NNPS, (default $k = 30$, an optimization of k is discussed in Section 5)
sampling times, (default $s = 500$)
drift significance level, α (default $\alpha = 0.01$)
output: classification results $\hat{Y} = \{\hat{y}_0, \ldots, \hat{y}_n\}$

1 **Initial** $win_{fix} = \mathbf{D}_{train}, win_{slide} = null, buff_{train} = \mathbf{D}_{train}$;
2 **Initial** buildLearner(L, $buff_{train}$);
3 **while** *not end of stream,* $d_t \in \mathbf{D}$ **do**
4 $\hat{Y} = \hat{Y} \cup \{$classify($L$, d_t)$\}$;
5 **if** $size(win_{fix} < win_{min})$ **then**
6 insert d_t at the end of win_{fix};
7 add d_t into $buff_{train}$;
8 **else**
9 **if** $win_{slide} = null$ **then**
10 $win_{slide} = win_{fix}$;
11 insert d_t at the end of win_{slide};
12 remove the first element of win_{slide};
13 **if** $nndviDriftDetection(win_{fix}, win_{slide}, dist(d_i, d_j), k, s, \alpha)$ *is true* **then**
14 $win_{fix} = win_{slide}$;
15 $buff_{train} = win_{fix}$;
16 $win_{slide} = null$;
17 **else**
18 add d_t into $buff_{train}$;
19 **if** $size(buff_{train} > w_{max})$ **then**
20 remove the first $size(buff_{train}) - w_{max}$ elements form $buff_{train}$;
21 updateLearner(L, $buff_{train}$);
22 **return** $\hat{Y}$

window. If a concept drift is detected, the current time window, namely win_{slide} will become the representation of current concept, and the training buffer ($buff_{train}$) will be the rest, as shown in lines 14-17. In contrast, lines 18-23 are the implementation of the second scenario. If no concept drift is detected, the newly arrived data will be kept in the training buffer. Once the training buffer has reached the maximum allowed buffer size, which is an input parameter denoted as w_{max}, the oldest training instance will be removed.

According to Theorem 4, the core idea of $nndviDriftDetection$ is to estimate the mean (μ) and variance (σ^2) of d^{nnps} of the normal distribution, then use the cumulative distribution function to compute the drift threshold θ_{drift} with a certain confidence level, namely the significant level α. Algorithm 2 lines 1-4 are the construction of NNPS, where $I_{|\mathbf{D}|\times|\mathbf{D}|}$ is a $|\mathbf{D}|$ by $|\mathbf{D}|$ Identity Matrix. Line 5 computes the actual distance between sample sets S_1 and S_2. From line 6 to line 9, we shuffle S_1 and S_2 to create two new sample sets. The shuffling process will ensure the new sample sets are i.i.d. Finally, in lines 10 to 12, we compare the actual NNPS distance with the drift threshold and determine if a significant drift is observed.

Algorithm 3 is the pseudo-code of Definition 13. Lines 2 and 3 transform the NNPS matrix to sample sets presented by normalized multiset. Then the discrepancies are accumulated as shown in Line 4-6.

Algorithm 2: Nearest Neighbor based Density Variation Identification ($nndviDriftDetection$)

input : two data sample sets S_1 and S_2
distance function, (default $dist(d_i, d_j)$ is set as Euclidean distance)
k value for NNPS, (default $k = 30$, an optimization of k is discussed in Section 5)
sampling times, (default $s = 500$)
drift significant level, (default $\alpha = 0.01$)
output: drift detection results $\{True \mid False\}$

merge sample sets, $\mathbf{D} = S_1 \cup S_2$, and save the indices of S_1, S_2 as V_{S_1}, V_{S_2};
constructing adjacent matrix, $M_{\mathbf{D}}^{adj} = nearestNeighbors(dist(d_i, d_j), k, \mathbf{D})$;
constructing particle matrix, $P_{\mathbf{D}}^{nnps} = M_{\mathbf{D}}^{adj} + I_{|\mathbf{D}|\times|\mathbf{D}|}$;
constructing NNPS matrix $M_{\mathbf{D}}^{nnps}$ by normalizing $P_{\mathbf{D}}^{nnps}$ according to instances weights;
compute actual NNPS distance $d^{act} = distance^{nnps}(M_{\mathbf{D}}^{nnps}, V_{S_1}, V_{S_2})$;
for $i = 1 : s$ **do**
    $V'_{S_1}, V'_{S_2} = shuffle(V_{S_1}, V_{S_2})$;
    compute shuffled NNPS distance $d_i^{shuffle} = distance^{nnps}(M_D^{nnps}, V'_{S_1}, V'_{S_2})$;
estimate the distribution of $d^{shuffle} \sim N(\mu, \sigma^2)$;
compute the drift threshold θ_{drift} according to $N(\mu, \sigma^2)$ with significant level α;
if $d^{act} > \theta_{drift}$ **then**
    return $True$;
return $False$;

Algorithm 3: NNPS distance measurement ($distance^{nnps}$)

input : NNPS matrix of samples $\mathbf{D} = S_1 \cup S_2$, denoted as $M_{\mathbf{D}}^{nnps}$
indices vectors of sample sets S_1 S_2, denoted as V_{S_1}, V_{S_2}
output: d^{nnps}

Initial initial $d^{nnps} = 0$;
$\mathbf{M}_{S_1}^{nnps} = V_{S_1} \cdot M_{\mathbf{D}}^{nnps}$;
$\mathbf{M}_{S_2}^{nnps} = V_{S_2} \cdot M_{\mathbf{D}}^{nnps}$;
for $d_i \in |\mathbf{D}|$ **do**
    $d^{nnps} = d^{nnps} + |\boldsymbol{I}_{\mathbf{M}_{S_1}^{nnps}}(\mathrm{p}_{d_i}) - \boldsymbol{I}_{\mathbf{M}_{S_2}^{nnps}}(\mathrm{p}_{d_i})|$;
return $d^{nnps} = d^{nnps}/|\mathbf{P}(S_1) \cup \mathbf{P}(S_2)|$

5. Information Granularity Indicator for NNPS

The selection of the number of nearest neighbors (the k value of knn) is critical to controlling the resolution and is the only parameter of the $NNPS$. To select the best resolution to construct $NNPS$, we propose a novel discretization controlling method, called the instance particle independence coefficient, or simply $independence$, to quantify the granule information contained at different granule levels.

In the context of the $NNPS$, we define the granule information indicator as how likely it is that a group of instance particles will appear together, namely the joint probability of a group of instance particles occurring. Intuitively, in the proposed data model, the primary element of a data set is the instance particles. The $NNPS$ is constructed based on these elements. If a group of such elements is always shown together, with a joint probability equal to 1, it would be impossible to detect drifts inside this group, as shown in Figure 5.

In other words, the granule information indicator of the proposed data model describes the average region size assumed to have no concept drift. From a practical point of view, investigating the joint probability of every combination is neither computationally-friendly nor storage efficient. Alternatively, such indicators can be acquired through mea-

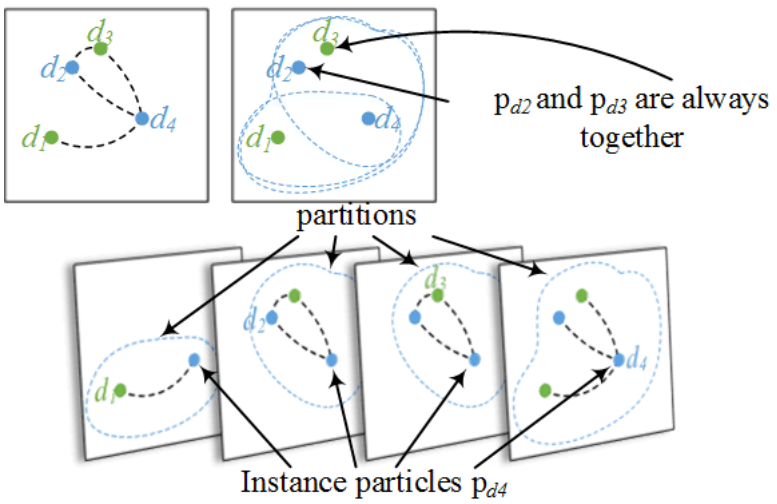

FIGURE 5: An example of grouped instance particles. Since $\mathrm{p}_{(}d_2)$ and $\mathrm{p}_{(}d_3)$ are always grouped together, the difference between d_2 and d_3 is equal to zero. If such a group is large enough, it will be impossible to identify whether there is a drift within them. Consequently, the sensitiveness of $NNPS$ on concept drift will decrease.

suring the independence of every instance particle. The higher the probability that an instance particle will appear in a certain instance particle group, the less independence that instance particle has. To estimate the overall independence of a given instance particle $\mathrm{p}_{(}d_i)$, we define the indicator of its independence as the average value of the conditional probability of its connected instance particles.

Definition 14. $(Independence\ of\ Instance\ Particle)$ The independence indicator of a given instance particle is defined as

$$independence(\mathrm{p}_{d_i}) = 1 - \frac{1}{|\mathbf{P}(d_i)|} \sum_{\mathrm{p}_{d_k} \in \mathbf{P}(d_i)} p(\mathrm{p}_{d_k}, \mathrm{p}_{d_i}) \tag{19}$$

where $p(\mathrm{p}_{d_k}, \mathrm{p}_{d_i})$ is the probability that $\mathrm{p}_{d_k} \in \mathbf{P}(d_i)$ and $\mathrm{p}_{d_i} \in \mathbf{P}(d_i)$

Definition 15. $(Independence\ of\ NNPS)$ The independence of the $NNPS$ is defined as the average value of instance particles' independence, denoted as :

$$independence = \frac{1}{|\mathbf{D}|} \sum_{\mathrm{p}_{d_i} \in \bigcup_{d_k \in \mathbf{D}} \mathbf{P}(d_k)} independence(\mathrm{p}_{d_i}) \tag{20}$$

With information granularity indicator, we can recognize and exploit the meaningful pieces of knowledge present in data so that different features and regularities can be revealed. This provides theoretical guidance for selecting the $k\text{-}value$, which is $\max_k\{independence\}$.

6. Experiments and Evaluation

The evaluation of NN-DVI for concept drift detection consists of four sections, through twelve experiments.

Section 6.1 Evaluating the effectiveness of $\mathrm{d}^{\mathbf{nnps}}$. To achieve this goal, we generated three synthetic drifting

data streams and compared the performance of d^{nnps} with the test statistics of two-sample Kolmogorov-Smirnov (KS) test.

1. Experiment 6.1 : Sudden drift streams by varying the data mean
2. Experiment 6.2 : Gradual drift streams by varying data variance
3. Experiment 6.3 : Regional drift streams by varying regional data mean

Section 6.2 Comparing NN-DVI with other distribution-oriented drift detection algorithms. To evaluate the performance of NN-DVI, we compared it with Dasu et al's [14] and Lu et al.'s work [46]. Since it is important to know the drift severity in advance, we generated 10 synthetic data sets with different predefined drift margins.

1. Experiment 6.4 : Drift streams with normal distributions
2. Experiment 6.5 : Drift streams with Poisson distributions
3. Experiment 6.6 : Drift streams with higher dimensional distributions

Section 6.3 Evaluation of NN-DVI in terms of real-world datasets. We applied NN-DVI on four real-world applications that consist of five datasets and compared the results to six state-of-the-art concept drift adaptation algorithms. The selected algorithms include two distribution-based, four learner output-based drift detection algorithms, one ensemble-based drift adaptation algorithm, one decision tree-based drift adaptation algorithm, and one drift adaptation algorithm considered all past concepts.

1. Experiment 6.7 : Electricity price prediction [19, 44]
2. Experiment 6.8 : Weather prediction [18, 44, 45]
3. Experiment 6.9 : Spam filtering with concept drift [19, 31]
4. Experiment 6.10 : UCI twenty newsgroups datasets [19, 32]

Section 6.4 Evaluating the performance of NN-DVI with different parameters In this section, we selected different parameters for NN-DVI to evaluate the parameters' sensitiveness.

1. Experiment 6.11 : Evaluating the performance of NN-DVI with different window size
2. Experiment 6.12 : Evaluating the performance of NN-DVI with different distance functions

6.1. Evaluating the Effectiveness of d^{nnps}

In this sub-section, we test and verify whether the proposed d^{nnps} can reflect the dissimilarity between sample sets. The experiments aim to : 1) establish how d^{nnps} varies according to changes between two distributions ; 2) investigate the impact of the selection of k on d^{nnps} ; and 3) evaluate the sensitiveness of d^{nnps} on regional drifts. We ran three experiments with generated artificial data sets following 1D normal distributions and compared the results with the test statistics of the two-sample K-S test. All the results were calculated as the mean of 100 independent tests.

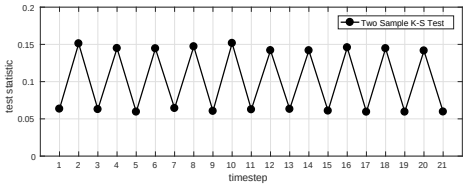

FIGURE 6: The test statistics of Two-sample K-S test between normal distributed data with varying μ. For each odd timestep, we compared two i.i.d. sample sets, while for each even timestep, we compared two sample sets with drifted mean $(\mu + 0.06)$. All the statistics were computed based on the average of 100 runs.

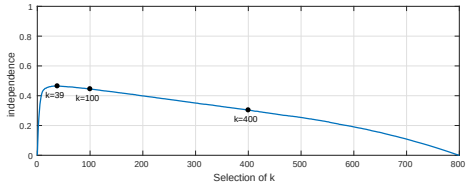

FIGURE 7: The relationship between $independence$ and the selection of k

Experiment 6.1. (Varying the mean μ) In this experiment, we compared the test statistics acquired by d^{nnps} with different k values. The experimental setting was as follows : a series of instances was generated from a synthetic data stream with 22 timestamps from 11 consecutive 1-D normal distributed data, with a batch size of 400 at each timestamp, and with $22 \times 400 = 8800$ data instances in total. Each normal distribution had a fixed standard deviation of $\sigma = 0.2$ and a moving mean of $\mu = 0.2 + 0.06 \times (i - 1)$, where i denotes the i^{th} distribution. As a result, when time step $t = 2 \times i - 1$, we compared two samples that were both drawn from the i^{th} distribution. When time step $t = 2 \times i$, we compared two samples drawn from the i^{th} distribution and $(i + 1)^{th}$ distribution. We ran the two-sample K-S test on 100 synthetic data sets with the same setup, then plotted the average test statistics in Fig.6. The results demonstrate how the two-sample KS test statistics change as the distributions vary. Because the difference only depends on the mean values, we found a similar height on all peaks and valleys for each series. Thus, for any valid distance measurement we should be able to construct a similar graph.

As outlined in Section 5, we chose the following k values for our distance measurement. The selections of k were : 39 ($independence = 0.46$), 100 ($independence = 0.45$) and 400 ($independence = 0.3$), as shown in Fig.7. independence reached its max at $k = 39$, while $k = 100$ and $k = 400$ belongs to the $independence$ decreasing stage, which means that the sensitivity of the distance is decreasing (Fig.7). The corresponding test statistics are outlined in Fig.8.

The d^{nnps} has the same pattern as the two-sample K-S test. For larger k values, the test statistic values are smaller (shown in blue). This is because larger k values have a larger number of instance particles, therefore, the ratio between the intersection set and union set will decrease. Even though the test statistic means are different, this will not affect the discrepancy measurement, as long as the peak-valley difference between the data structures is correctly reflected.

By the same token, we reduced the batch size to 50 at each time step and selected the k with the highest indepen-

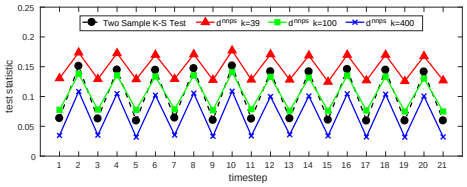

FIGURE 8: The d^{nnps} between normal distributed data (400 instances each time step) with varying μ. For each odd timestep, we compared two i.i.d. sample sets, while for each even timestep we compared two sample sets with drifted mean $(\mu + 0.06)$. All the statistics were computed based on the average of 100 runs.

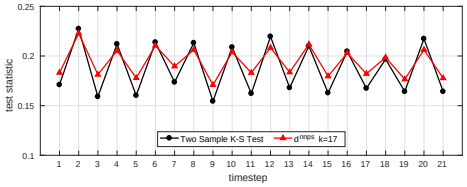

FIGURE 9: The d^{nnps} between normal distributed data (50 instances each time step) with varying μ.

dence value. The corresponding test statistics are plotted in Fig.9. The red lines with red triangles are the d^{nnps} with $k = 17$, the black lines with black dots is the two-sample K-S test. This experiment illustrates that d^{nnps} can perform well with an extremely small sample size.

Experiment 6.2. (Varying the standard deviation σ). In this experiment, we fixed the mean $\mu = 0.5$, but varied the standard deviation $\sigma = 0.1+0.02\times(i-1)$ for the i^{th} distribution, $i = 1, 2, \ldots, 11$. Again, when time step $t = 2\times i-1$, we compared two samples drawn from the i^{th} distribution; when time step $t = 2 \times i$, we compared two samples drawn from the i^{th} and $(i + 1)^{th}$ distribution. The batch size was set as 1000.

We show the two-sample K-S test statistics in Fig.10. In this figure, the valley values are similar, while the peak values gradually decrease. Since the valley values are determined by two identical distributions, they are steady and smooth. The peak values, by contrast, are calculated by two distributions with an increasing σ. As a result, the difference between these distributions will shrink as σ increases. Intuitively, this is because the distribution becomes less concentrated, thus the relative test statistic is smaller. The test statistics of d^{nnps} are shown in Fig.11. It can be seen that d^{nnps} has the same pattern as the two-sample K-S test, indicating that d^{nnps} is capable of precisely representing differences in distribution.

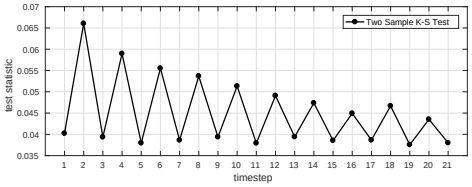

FIGURE 10: The test statistics of two-sample K-S test between normal distributed data with varying σ. For each odd timestep, we compared two i.i.d. sample sets, while for each even timestep we compared two sample sets with drifted standard deviation $(\sigma + 0.02)$. All the statistics were computed based on the average of 100 runs.

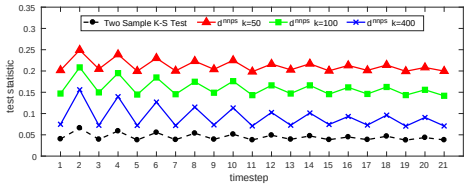

FIGURE 11: d^{nnps} between normal distributed data that varies σ.

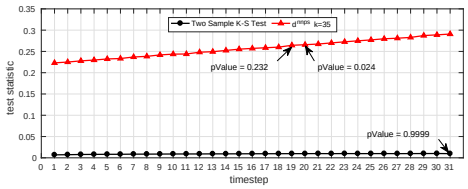

FIGURE 12: 1-D normal distribution with regional drift detection. At each timestep, we changed the mean $(\mu + 0.06)$ of a group of data instances and plot the d^{nnps} between the current timestep and last timestep. The proposed d^{nnps} reported a significant difference at after 19 groups drifted, whereas the K-S test cannot detect such a difference.

Experiment 6.3. (Varying the regional mean μ_j). In this experiment, we compared the distances between two data samples that comprise 30 independent 1-D normal distributed data groups. In the first data group, the normal distribution was set as $\mu = 0.2$, $\sigma = 0.2$. For the remaining 29 data groups, the normal distribution was set as $\mu = 0.2 + j \times (20 \times \sigma)$, $\sigma = 0.2$ for $j = 2, 3, \ldots, 30$. We intentionally left a $(20 \times \sigma)$ gap between each group to ensure that a change occurring in one group would not affect the others. Each group contained 100 data instances, equating to 3000 instances for each time step. In the first-time step, we compared two data sample sets without any drift in any data group. Later, we manually changed the mean of each data group by $\mu_j + 0.06$ successively for each time step to create 30 minor regional drifts.

Fig.12 shows the test statistics for the two-sample K-S test and the d^{nnps} at different time steps. At $t = 1$, we compared 30 data groups with another 30 groups that were generated by the same distribution. At $t = 2$, we compared 30 data groups with another $29 + 1$ groups where the mean μ_1 of group 1 was changed to $\mu_1 + 0.06$, and so on till $t = 31$. It can be clearly seen that d^{nnps} increased after each local drift occurred. In fact, the K-S test statistics also increased slightly. However, as the value of its test statistics is too small, the trend cannot be observed in this figure. In addition, to compare the test statistics, we also applied the corresponding significance test to examine whether the drifts were statistically significant. The p-value of the two-sample K-S test was above 95% all the time while the p-value of d^{nnps} dropped below 5% after timestep 19, which means that d^{nnps} can detect simulated drifts after 19/30 local drifts. This experiment demonstrates that the proposed d^{nnps} is sensitive to regional drifts.

6.2. Evaluating the NN-DVI Drift Detection Accuracy

In the above experiments, we have shown how the d^{nnps} varies as the underlying distribution changes. To determine whether a given measurement is statistically significant as a drift, we compute the ASL as described in Section

4.3. Given a desired significance level α, we say there is a concept drift when the $ASL < \alpha$. In the following experiments, we compare NN-DVI with Dasu [14] and Lu's [46] work in terms of drift detection accuracy on 10 synthetic drifting data streams. The reason we choose these two algorithms for comparison is that all three methods (including NN-DVI) detect concept drift based on the estimated data distribution, and do not hold assumptions on the distribution of data. In addition, we also compared one of the most popular two-sample tests for multivariate data, maximum mean discrepancy (MMD) [23], as the base line. NN-DVI(permutation) represents that using permutation test to estimate the ASL of NN-DVI, and NN-DVI(normal) represents that using the tailored significance test, which has been proven following normal distribution, to calculate the ALS of NN-DVI.

For a fair comparison, we used the same experimental configurations and evaluation criteria as introduced in Dasu and Lu's papers [14, 46]. The evaluation criteria are : $detected$, $late$, $false$, and $missed$. Given a concept that exists for a period of time $[t+1, t+z]$, where z is the length of the concept. In this experiment, the distribution parameters are updated at $t+1$ and $t+z+1$, that is, a new concept starts from $t+1$ and ends at $t+z$. For each timestep, it involves three processes : updating distribution parameters if new concept starts, generating data and detecting drift. The $detected$ indicates the number of times that a drift detection algorithm reports a drift at $t+1$. The $late$ indicates the number of times a drift reported at $t+2$ or $t+3$. The $false$ accumulates the number of false alarms that multiple drifts are reported in $[t+1, t+z]$. A $missed$ detection is counted if no drift reported within $[t+1, t+3]$. A detected drift within $[t+4, t+z]$ is marked as $missed$ because such a detection is considered as too late to be useful [46]. We denote Dasu et al.'s algorithm as KL (Kullback-Leibler distance) and Lu et al.'s work as CM (competence model). With regard to the data source, we implemented the same synthetic data sets introduced by Dasu et al. [14] and adopted by Lu et al. [46]. For each data stream, we generated 5,000,000 data instances, and for every 50,000 instances, the parameters of the corresponding distribution were varied randomly within a certain interval. These variations produced 99 controllable drifts for each data stream. For ease of comparison, we kept all the parameters the same for both the bootstrapping test and the permutation test, including the desired significance level of $\alpha = 1\%$, and the size of bootstrapping and permutation $N = 500$. KL and CM's parameters were set as per their papers. The distance function used in this section is Euclidean distance, as the same as the distance used by [14, 46]. Because the permutation test does not hold any assumptions and can be applied on KL without modification, we also ran KL with a permutation test for comparison. To avoid any bias in NNPS-DVI caused by parameter selection, we first conducted an experiment with a k-value that was chosen based on the average number of instances within the same partitioning size as KL and CM. Then we ran our algorithm with a dynamic k-value that we selected based on $\max_k\{independence\}$, and applied the normal distribution estimation described in Section 4.3 as the significance test.

Experiment 6.4. (Normal distributions). In this experiment, we compared KL, CM and NN-DVI using five bivariate normal distributed data streams, denoted as $M(\triangle)$ and $C(\triangle)$ streams. Drift severity was controlled by a predefined

TABLE 1: Drift detection results on $M(\triangle)$ stream

Data Stream	Drift Detection Method	Parameters	Window Size	Detected	Late	False	Missed
M(0.02)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	76	13	2	10
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	68	14	3	17
	CM (permutation)	$d_\epsilon : 0.05$	10000	87	5	12	7
	NN-DVI (permutation)	$k : 185$	10000	90	8	9	1
	NN-DVI normal)	$k : \max\{independence\}$	10000	91	6	6	2
	MMD	RBF kernel, σ median	10000	99	0	3	0
M(0.05)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	99	0	9	0
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	99	0	3	0
	CM (permutation)	$d_\epsilon : 0.05$	10000	99	0	2	0
	NN-DVI (permutation)	$k : 185$	10000	99	0	5	0
	NN-DVI (normal)	$k : \max\{independence\}$	10000	99	0	5	0
	MMD	RBF kernel, σ median	10000	99	0	3	0

TABLE 2: Drift detection results on $C(\triangle)$ stream

Data Stream	Drift Detection Method	Parameters	Window Size	Detected	Late	FALSE	Missed
C (0.1)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	33	18	4	48
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	30	16	1	53
	CM (permutation)	$d_\epsilon : 0.05$	10000	33	19	2	47
	NN-DVI (permutation)	$k : 160$	10000	41	19	0	39
	NN-DVI (normal)	$k : \max\{independence\}$	10000	54	12	3	33
	MMD	RBF kernel, σ median	10000	34	19	1	46
C (0.15	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	85	8	9	6
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	77	13	4	9
	CM (permutation)	$d_\epsilon : 0.05$	10000	81	10	7	8
	NN-DVI (permutation)	$k : 160$	10000	87	10	7	2
	NN-DVI (normal)	$k : \max\{independence\}$	10000	89	9	4	1
	MMD	RBF kernel, σ median	10000	81	13	3	5
C (0.15)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	5000	79	7	15	13
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	5000	63	14	6	22
	CM (permutation)	$d_\epsilon : 0.05$	5000	55	21	14	23
	NN-DVI (permutation)	$k : 80$	5000	67	19	13	13
	NN-DVI (normal)	$k : \max\{independence\}$	5000	70	19	5	10
	MMD	RBF kernel, σ median	5000	31	6	1	62
C (0.2)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	5000	96	2	11	1
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	5000	89	6	10	4
	CM (permutation)	$d_\epsilon : 0.05$	5000	82	9	8	8
	NN-DVI (permutation)	$k : 80$	5000	93	3	13	3
	NN-DVI (normal)	$k : \max\{independence\}$	5000	93	3	10	3
	MMD	RBF kernel, σ median	5000	44	5	2	50

drift margin $\triangle$. In the $M(\triangle)$ streams, the features x_1 and x_2 followed an independent normal distribution with the means μ_1 and μ_2 moving within $[0.2, 0.8]$. For every 50,000 instances, μ_1 and μ_2 were randomly updated with a step size $[-\triangle, -\triangle/2 \cup [\triangle/2, \triangle]$. In the $C(\triangle)$ streams, the two features have a moving correlation ρ, which started at 0 and then was randomly walked within $[-1, 1]$, with step size chosen randomly from $[-\triangle, -\triangle/2] \cup [\triangle/2, \triangle]$. To investigate the impact of the window size on drift detection accuracy, we conducted the experiment on the $C(0.15)$ stream twice with two window sizes. All the results are summarized in Tables 1 and 2.

From the results, we can see that our method outperformed the others in most cases. In terms of the false and missed rates, only bootstrapped KL on the $C(0.2)$ stream is slightly better than ours. With regard to the other streams, our results are much better. If we only consider drift detection based on permutation tests, no other methods can surpass ours. According to the results, the excellent performance of KL with bootstrap is more likely attributed to the bootstrap test rather than to their drift detection model. Nevertheless, Dasu et al. [14] did not give any explanation

TABLE 3: Drift detection results on $P(\triangle)$ stream

Data Stream	Drift Detection Method	Parameters	Window Size	Detected	Late	False	Missed
P(0.1)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	64	7	1	28
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	66	8	4	25
	CM (permutation)	$d_\epsilon : 10$	10000	63	8	1	28
	NN-DVI (permutation)	$k : 545$	10000	81	7	7	11
	NN-DVI (normal)	$k : \max\{independence\}$	10000	81	9	5	9
	MMD	RBF kernel, σ median	10000	61	8	2	30
P(0.2)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	95	1	11	3
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	98	0	1	1
	CM (permutation)	$d_\epsilon : 10$	10000	99	0	5	0
	NN-DVI (permutation)	$k : 545$	10000	99	0	6	0
	NN-DVI (normal)	$k : \max\{independence\}$	10000	99	0	6	0
	MMD	RBF kernel, σ median	10000	98	1	10	0

concerning the selection of the significance test, instead, they only suggest researchers investigate the pros and cons in other publications. It is also noteworthy that all the detection algorithms achieved high detection rates on $M(0.05)$ and $C(0.2)$. According to the data generation settings, $M(0.05)$ and $C(0.2)$ have relatively large drift margins. This would make the drifts easier to detect, and the drift detection results may be very similar as long as the drift margins exceed the sensitivity thresholds of the tested algorithms. For example, in this experiment, the drift margins for $M(0.05)$ exceed the sensitivity threshold of all the tested algorithms, therefore, all the results are very close (the missed rate are all zero and the false rate are all low). By the same token, we infer that the $C(0.2)$ reached the sensitivity threshold of both KL with bootstrap and NN-DVI. Unlike the $M(0.05)$ and $C(0.2)$ streams, the rest of the streams have smaller drift margins. The high detection rates on those streams prove that NN-DVI is more sensitive to small concept drifts.

Experiment 6.5. (Poisson distributions). In $P(\triangle)$ streams, the two features follow a Poisson distribution $Poisson(500(1-\rho), 500(1-\rho), 500\rho)$, where ρ starts at 0.5 and then performs a random walk between 0 and 1 with step size $\triangle = 0.2, 0.1$. The evaluation results are shown in Table 3.

From Table 3, we can see that KL using a bootstrap test suffers a manifest failure on Poisson distributions. On one hand, KL using a permutation test detected most drifts correctly on $P(0.2)$. It is evident that the $P(0.2)$ drift margin is larger than the KL sensitivity threshold. On the other hand, KL suffers high false rate on $P(0.2)$ when combined with the bootstrap test. This paradox indicates that a null hypothesis test could affect the sensitivity of a drift detection algorithm, and the bootstrap test may not be suitable for the KL algorithm in all circumstances. In general, according to the results in Table 3, NN-DVI clearly outperforms the others.

Experiment 6.6. (Higher dimensional distributions). To test the scalability and performance of NN-DVI in high dimension space, we extended the $C(0.2)$ stream to three d-dimensional multivariate normal distributed streams as per [14, 46]. Only the first two dimensions had a correlation value equal to ρ, and the remaining (d-2) dimensions were configured with correlation values equal to zero. Additionally, the standard deviation of all the added dimensions was configured to be the same as the $C(\triangle)$ streams. These settings are managed to retain the same marginal distribution of all dimensions, so that the overall distance between the instances contributed by each dimension is equally important

TABLE 4: Drift detection results on HD $C(\triangle)$ streams

Data Stream	Drift Detection Method	Parameters	Window Size	Detected	Late	False	Missed
4D C(0.2)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	91	5	5	3
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	87	8	5	4
	CM (permutation)	$d_\epsilon : 0.15$	10000	89	3	8	7
	NN-DVI (permutation)	$k : 95$	10000	94	5	5	0
	NN-DVI (normal)	$k : \max\{independence\}$	10000	95	4	6	0
	MMD	RBF kernel, σ median	10000	83	10	2	6
6D C(0.2)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	84	5	6	10
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	65	11	3	23
	CM (permutation)	$d_\epsilon : 0.3$	10000	91	4	7	4
	NN-DVI (permutation)	$k : 200$	10000	97	2	6	0
	NN-DVI (normal)	$k : \max\{independence\}$	10000	95	4	6	0
	MMD	RBF kernel, σ median	10000	69	12	7	18
10D C(0.2)	KL (bootstrap)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	81	7	4	11
	KL (permutation)	$\delta : 2^{-10}, \tau : 100, \gamma : 5\%$	10000	68	13	5	18
	CM (permutation)	$d_\epsilon : 0.5$	10000	78	10	4	11
	NN-DVI (permutation)	$k : 220$	10000	89	5	3	5
	NN-DVI (normal)	$k : \max\{independence\}$	10000	88	6	5	5
	MMD	RBF kernel, σ median	10000	26	18	0	55

TABLE 5: Average Drift Detection Results

Drift Detection Method	Detected	Late	FALSE	Missed
KL (bootstrap)	80.27	6.64	7	12.09
KL (permutation)	73.64	9.36	4.09	16
CM (permutation)	77.91	8.09	6.36	13
NN-DVI (permutation)	85.18	7.09	6.73	6.73
NN-DVI (normal)	**86.73**	**6.55**	5.82	**5.73**
MMD	65.91	8.36	**3.09**	24.73

[14, 46]. As a result, it will be easier to calculate the distance between instances, and the drift detection results will not be affected by the selection of the distance functions used to compare data instances.

The experimental results for different dimensions are listed in Table 4. As expected, as more stationary dimensions are added, the missed rates of all detection algorithms increase, implying that such a change makes drift detection harder. However, NN-DVI preserves much of its power as the number of dimension increases. It is very likely that this can be attributed to the k-nearest neighbor-based data representation model defined in Section 4.1.

To summarize, we calculated the average detected, late, false and missed rates of all synthetic streams, as shown in Table 5. It shows that the proposed NN-DVI makes a notable improvement on drift detection in various situations. Although the false rate of NN-DVI is slightly higher than the existing methods under the permutation test, the missed rate is almost halved. The performance of NN-DVI with normal distribution as significance test is even better. It shows higher accuracy with less false rate which indicates the tailored significance test introduced in Section 4.3 is more reliable. In addition, the complexity of data model construction can be reduced to $O(knlog(n))$ with a kd-Tree data structure while CM is $O(n^2log(n))$ [46], KL is $O(dnlog(1/\delta))$ where the d is the number of data dimensions and δ is the given parameter of KL [14] and MMD is $O(n^2)$ [23].

6.3. Evaluating the NN-DVI on Real-world Datasets

To demonstrate how our drift detection algorithm improves the performance of learning models in real-world scenarios, we compared our detection method with the other two closely related algorithms, 1) KL [14] and 2) CM [46]), on five benchmark real-world concept drift data sets. These datasets are the top referenced benchmark and include both low and high dimensionality data. In addition, we also compared NN-DVI with four state-of-the-art drift adaptation algorithms that address concept drift using different strategies. These algorithms are 3) SAMkNN [44] which keeps past concepts in memory and considers all past concept to make final prediction; 4) AUE2 [13] which uses ensemble methods to handle concept drift; and 5) HDDM family tests (HDDM-A, HDDM-W) [19] which detects drift by monitoring learner outputs; 6) ADWIN-Volatility (ADW-Vol) [26] which first introduced volatility shift into concept drift detection. At last, we use HAT [7] and ADWIN [6] as the base line. The algorithms were implemented with the authors' recommended settings. The selection of the base learner for the HDDM family tests, KL, CM and NN-DVI is independent of the drift detection algorithm, therefore, we implement these algorithms with Naïve Bayes, Hoeffding Tree and IBk classifiers. Because the SAMkNN is only compatible with IBk and AUE2 is only compatible with Hoeffding Tree, these two algorithms were only compared with their own base learners. All algorithms were implemented based on MOA platform [8] which is written in Java. The parameters of IBk were set as $k = 5$ and instances weighting method is $uniformly\ weighted$. All the distance function used in this section is Euclidean distance. The parameters of Heoffding Tree were set as the same as those recommended by AUE2. Table 6 summarizes the classification accuracy and gives the overall rank of different algorithms. The average processing time in second and average memory usage (RAM-Hour in Gigabyte) are also listed, since the computational resources management is also an important issue for data stream learning [10].

Experiment 6.7. (Electricity Price Prediction) The electricity data set contains 45,312 instances, collected every thirty minutes from the Australian New South Wales Electricity Market between 7 May 1996 and 5 Dec 1998. In this market, prices are not fixed and are affected by demand and supply. This data set contains eight features and two classes (up, down) and has been widely used for concept drift adaptation evaluation [19, 44]. The data set is available from MOA's website [8]

Experiment 6.8. (Spam Filtering)This data set is a collection of 9,324 email messages derived from the Spam Assassin collection, which is available at http ://spamassassin.apache.org/. The original data set contains 39,916 features, and 9,324 emails (around 20% spam emails and 80% legitimate emails). It is commonly considered to be a typical gradual drift data set [31]. According to Katakis's work [31], 500 attributes were retrieved using the chi-square feature selection approach.

Experiment 6.9. (Weather Prediction)The Nebraska weather prediction data set was compiled by the U.S. National Oceanic and Atmospheric Administration. It contains 8 features and 18,159 instances with 31% positive (rain) classes,

TABLE 6: Classification accuracy of real-world datasets. The rank of each algorithm is shown in the brackets. The final score is the average rank of that algorithm combined with different base learners. The Time (s) and RAM-Hour (GB) are calculated on the average of the five datasets.

Algorithms	Learner	Elec	Spam	Weather	Usenet1	Usenet2	AVG. Rank	Score	Time	RAM-Hour
	NB	84.10 (6)	92.03 (9)	72.50 (12)	68.93 (12)	77.13 (2)	8.2 (2)		190.6	2.85E-05
NN-DVI	IBk	87.39 (2)	93.47 (2)	74.63 (7)	66.80 (17)	72.47 (7)	7.0 (1)	2.0 (1)	208.5	2.85E-05
	HTree	82.85 (10)	92.06 (8)	71.68 (17)	70.27 (9)	76.73 (4)	9.6 (3)		195.2	2.85E-05
SAMkNN	IBk	82.54 (14)	95.67 (1)	78.26 (1)	66.13 (19)	70.33 (17)	10.4 (5)	5.0 (2)	266.5	3.75E-02
	NB	84.09 (7)	91.65 (14.5)	72.82 (11)	75.07 (2)	70.93 (16)	10.1 (4)		1.4	4.91E-08
HDDM-W	IBk	82.47 (15)	92.93 (3)	75.04 (6)	69.67 (10)	67.93 (22)	11.2 (10)	6.7 (3)	249.9	2.07E-03
	HTree	85.06 (4)	91.65 (14.5)	72.41 (13)	74.73 (3)	70.00 (18)	10.5 (6)		1.4	7.52E-08
	NB	84.92 (5)	90.79 (20)	72.38 (14)	75.20 (1)	71.00 (14.5)	10.9 (8)		1.4	4.97E-08
HDDM-A	IBk	82.10 (16)	92.00 (10)	76.13 (4)	69.33 (11)	68.00 (21)	12.4 (11.5)	9.5 (4)	132.2	1.05E-04
	HTree	85.71 (3)	91.78 (13)	71.71 (16)	74.60 (4)	68.87 (19.5)	11.1 (9)		1.6	7.55E-08
	NB	82.66 (12)	91.10 (19)	71.89 (15)	71.07 (7)	77.20 (1)	10.8 (7)		137.2	4.54E-04
CM	IBk	78.98 (23)	89.34 (23)	73.71 (8)	67.47 (14)	71.33 (13)	16.2 (23)	14.7 (5)	133.3	4.54E-04
	HTree	81.33 (20)	91.11 (18)	71.23 (20)	71.73 (5)	76.27 (5)	13.6 (14)		144.8	4.54E-04
	NB	82.62 (13)	91.19 (16)	71.00 (22)	70.73 (8)	77.06 (3)	12.4 (11.5)		191.7	7.90E-05
KL	IBk	78.95 (24)	89.37 (22)	73.68 (9)	67.27 (15)	71.00 (14.5)	16.9 (24)	16.8 (6)	186.7	7.91E-05
	HTree	81.34 (19)	91.16 (17)	71.18 (21)	71.33 (6)	76.20 (6)	13.8 (15)		171.7	7.94E-05
	NB	80.64 (22)	91.83 (12)	71.40 (18)	68.13 (13)	72.27 (8.5)	14.7 (21)		1.3	5.73E-08
ADW-Vol	IBk	81.72 (18)	92.81 (4)	76.20 (2.5)	59.73 (23.5)	67.27 (23.5)	14.3 (17.5)	17.2 (7)	201.5	5.93E-05
	HTree	82.84 (11)	92.59 (5)	71.28 (19)	66.67 (18)	72.00 (10.5)	12.7 (13)		1.5	9.28E-08
AUE2	HTree	87.74 (1)	84.29 (24)	75.24 (5)	63.47 (22)	68.87 (19.5)	14.3 (17.5)	17.5 (8)	4.4	5.37E-06
HAT	HTree	83.39 (8)	90.69 (21)	73.51 (10)	63.93 (21)	71.87 (12)	14.4 (19)	19.0 (9)	2.0	7.92E-08
	NB	81.03 (21)	91.90 (11)	70.31 (24)	67.00 (16)	72.27 (8.5)	16.1 (22)		1.4	5.04E-08
ADWIN	IBk	81.76 (17)	92.46 (6)	76.20 (2.5)	59.73 (23.5)	67.27 (23.5)	14.5 (20)	19.3 (10)	200.7	5.63E-05
	HTree	83.23 (9)	92.29 (7)	70.88 (23)	64.47 (20)	72.00 (10.5)	13.9 (16)		1.4	7.90E-08

and 69% negative (no-rain) classes. This data set is summarized in [18] and is available at http ://users.rowan.edu/ polikar/research/NSE

Experiment 6.10. (Usenet1 and Usenet2) The Usenet datasets were first introduced by [32], and were used to evaluate HDDM family algorithms [19]. They are two subsets of the 20 newsgroups collection which is owned by the UCI Machine Learning Repository. Each data sets consists of 1500 instances and 99 features. In these data sets, each instance is a message from different newsgroups that is sequentially presented to a user, who then labels it as interesting or not interesting. These messages involve three topics : medicine, space and baseball. The user was simulated with a drifted interest for every 300 messages. A more detailed explanation of these datasets can be found in [32].

To better demonstrate the improvements of NN-DVI on real-world problems, we used a ranking method to summarize the overall performance, which is similar to [44]. The average rank shows that NN-DVI achieves the best performance with the IBk classifier and has the best final score, which is the average of all base classifiers' ranks.

The analysis results of Table 6 can be summarized as follows :

— The performance of base learners is variable on different data sets. For example, under the same drift adaptation algorithm, the IBk classifier always outperforms the other classifiers on the Weather data set while it performs the worst on Usenet1 and Usenet2 data sets. In some situations, the selection of base learners may be more important than the selection of drift detection methods. This outcome inspired us to reconsider the learning strategy of using the same base classifier settings after a drift. Instead, changing a base learner or updating the base learner's parameters may further improve the overall prediction accuracy.

— The proposed NN-DVI method with IBk, NB and HTree have the best performance in terms of accuracy. Considering the overall performance, it is evident that the highlighted results of NN-DVI do not depend on the base classifiers and NN-DVI contributes to the learning process in dynamic environment.

— Regarding to computational complexity, it can be seen that distribution-based drift detection algorithms (NN-DVI, CM, KL) have very similar computational costs, namely the Time and RAM-Hour, for all the tested three base learners. The reason is that these algorithms have to keep the representative data instances in memory to detect the distribution changes. It looks like a common issue of distribution-based drift detection methods. Nevertheless, they have shown no manifest disadvantages comparing to other drift detection methods using IBk learner. Although error-rate based and ensemble-based drift adaptation methods can handle concept drift in an online manner, using IBk as base learner may slow down their learning speed.

6.4. Evaluating the Stream Learning with NN-DVI with Different Parameters

In general, the algorithm for stream learning with NN-DVI has six input parameters : 1) minimum drift detection window size ; 2) base learner ; 3) distance function for constructing NNPS ; 4) the number of nearest neighbours for NNPS ; 5) the sampling times for estimate the σ of d^{nnps} ; and 6) drift significance level. Amount these parameters, the parameter 2) is evaluated in Section6.3, parameter 4) is discussed in Section 5, parameter 5) is related to Monte Carlo Error which has been discussed in details in the section 5.2 of [46] parameter 6) is a drift sensitivity setting chosen by users which is highly depend on the system requirements. Therefore, in this section, we focus on evaluating the impacts of the parameters : 1) minimum drift detection window size and 3) distance functions for constructing NNPS. The rest of the parameters are fixed as the default values.

Experiment 6.11. (Changing window size) As a critical parameter of time window-based drift detection algorithms, the window size or chuck size [53] controls the length of the most recent concept, namely the win_{slide} in our algorithm. On the one hand, if the window size is too small, the most recent concept may be incomplete, as a result, false alarms will be raised. On the other hand, if the window size is over large, multiple concepts may be included in one window, and such mixed data will lead to bad performance. To evaluate how window size affects NN-DVI, we set win_{min} =50, 75, 100, 150, 200, and plotted the classification accuracy of the five real-world datasets in Figure 13.

Experiment 6.12. (Changing distance functions) To evaluate how distance function affects NN-DVI on real-world datasets, we choose another four most commonly used distance functions to replace Euclidean distance, which are Manhattan distance, Chebyshev distance, Jaccard distance, and Radial basis function kernel distance. The overall classification accuracy of these distances with different window size is plotted in Figure 13.

It can be seen that along with the increasing of window size, the average accuracy is decreasing, especially for dataset Usenet1. It could be caused by there are mixed concepts in time windows. In contrast, the accuracy of Spam

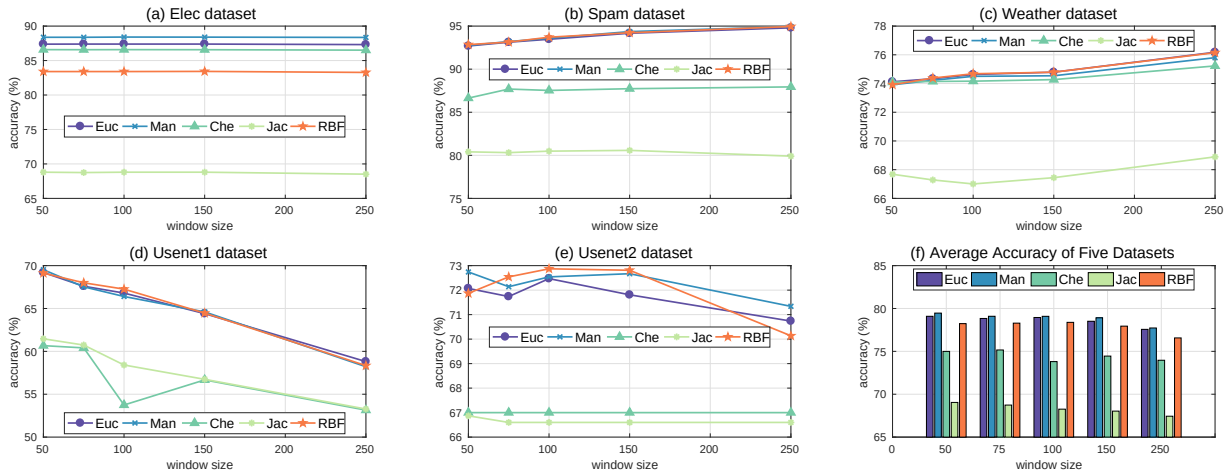

FIGURE 13: Classification accuracy of real-world datasets with different window size and different distance functions. Each line represents a distance function. A marker on a line represents the classification accuracy of NN-DVI with a given distance function and on the window size. The figure (f) shows the average accuracy of the tested five datasets for each distance function with different window size.

and Weather dataset is slightly increasing along with the window size. We consider that NN-DVI might overly trigger false alarms with small window size in these datasets. Increasing the window size will reduce the overall false alarms so that the accuracy can be increased. Regarding to the distance functions, Euclidean and Manhattan has very similar performance and are the top two best distance for the evaluated datasets, while Jaccard distance performs the worst all the time. Therefore, we recommend to use Euclidean or Manhattan distance for most situation.

7. Conclusions and Further Work

In this paper, we analyzed distribution-based drift detection algorithms, and summarized the fundamental components of this type of drift detection methods. Under the guidance of the summarized framework (Fig. 1), we proposed a novel space partitioning schema, called NNPS, to improve the sensitiveness of regional drift detection. Accordingly, a novel distance d^{nnps} and a nearest neighbor-based density variation identification (NN-DVI) algorithm is proposed. This research also defines a k-nearest neighbor-based data discretization controlling method to represent the granular information in data samples. Regarding the drift significance test, this study theoretically proved the proposed measurement, namely the d^{nnps}, fits a normal distribution. According to this finding, an efficient and reliable critical interval identification method has been integrated. The experiment results show that NN-DVI can detect concept drift accurately on the synthetic datasets and is beneficial to solving real-world concept drift problems.

The innovation and contributions of this paper lie in its aims to discover the relationships between distribution drifts and the changes in data instances' neighbors. A regional density-oriented similarity measurement is given. This measurement can effectively detect concept drift caused by either regional or global distribution changes.

Future research endeavors will aim to improve the current sliding windowing method by proposing an online version of d^{nnps}, so that the performance of NN-DVI will not be limited by window size. Another improvement may

be achieved by parallelizing the calculation of d^{nnps}, through which overall complexity can be further reduced. Finally, this paper is part of our work on adaptability learning for data stream mining. Successive instance selection methods and adaptive learning methods that can take advantage of drift detection are desired.

Acknowledgment

The work presented in this paper was supported by the Australian Research Council (ARC) under Discovery Project DP150101645.

References

[1] C. Alippi, G. Boracchi, M. Roveri, Just-In-Time classifiers for recurrent concepts, IEEE Transactions on Neural Networks and Learning Systems 24 (4) (2013) 620–634, ISSN 2162-237X.

[2] C. Alippi, M. Roveri, Just-in-Time Adaptive Classifiers Part I : Detecting Nonstationary Changes, IEEE Transactions on Neural Networks 19 (7) (2008) 1145–1153, ISSN 1045-9227 1941-0093.

[3] C. Alippi, M. Roveri, Just-in-Time adaptive classifiers part II : designing the classifier, IEEE Transactions on Neural Networks 19 (12) (2008) 2053–2064, ISSN 1045-9227.

[4] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in : Fourth International Workshop on Knowledge Discovery from Data Streams, Vol. 6, pp. 77–86, 2006.

[5] J. P. Barddal, H. Murilo Gomes, F. Enembreck, B. Pfahringer, A. Bifet, On dynamic feature weighting for feature drifting data streams, in : Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, ISBN 978-3-319-46227-1, 129–144, 2016.

[6] A. Bifet, R. Gavaldà, Learning from Time-Changing Data with Adaptive Windowing, in : Proceedings of the Seventh SIAM International Conference on Data Mining, vol. 7, SIAM, 2007, 2007.

[7] A. Bifet, R. Gavaldà, Adaptive Learning from Evolving Data Streams, in : The Proceedings of the Eighth International Symposium on Intelligent Data Analysis, Springer Berlin Heidelberg, ISBN 978-3-642-03915-7, 249–260, 2009.

[8] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA : Massive online analysis, Journal of Machine Learning Research 99 (2010) 1601–1604.

[9] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in : Proceedings of the 2010 Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 135–150, 2010.

[10] A. Bifet, G. Holmes, B. Pfahringer, E. Frank, Fast Perceptron Decision Tree Learning from Evolving Data Streams, in : Proceedings of the Fourteenth Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, ISBN 978-3-642-13672-6, 299–310, 2010.

[11] W. D. Blizard, Multiset theory, Notre Dame Journal of Formal Logic 30 (1) (1988) 36–66, ISSN 0029-4527.

[12] D. Brzeziński, J. Stefanowski, Accuracy updated ensemble for data streams with concept drift, in : Proceedings of the 2011 International Conference on Hybrid Artificial Intelligence Systems, Springer, 155–163, 2011.

[13] D. Brzeziński, J. Stefanowski, Reacting to different types of concept drift : the accuracy updated ensemble algorithm, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 81–94, ISSN 2162-237X.

[14] T. Dasu, S. Krishnan, S. Venkatasubramanian, K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in : Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications, Citeseer, 1–24, May 24-27, 2006.

[15] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments : a survey, IEEE Computational Intelligence Magazine 10 (4) (2015) 12–25, ISSN 1556-603X.

[16] M. Dwass, Modified randomization tests for nonparametric hypotheses, The Annals of Mathematical Statistics (1957) 181–187ISSN 0003-4851.

[17] B. Efron, R. J. Tibshirani, An introduction to the bootstrap, CRC press, ISBN 0412042312, 1994.

[18] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Transactions on Neural Networks 22 (10) (2011) 1517–31, ISSN 1941-0093 (Electronic) 1045-9227 (Linking).

[19] I. Frias-Blanco, J. d. Campo-Avila, G. Ramos-Jimenes, R. Morales-Bueno, A. Ortiz-Diaz, Y. Caballero-Mota, Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds, IEEE Transactions on Knowledge and Data Engineering 27 (3) (2015) 810–823, ISSN 1041-4347.

[20] J. Gama, G. Castillo, Learning with local drift detection, in : International Conference on Advanced Data Mining and Applications, Springer, 42–55, 2006.

[21] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Computing Surveys 46 (4) (2014) 1–37, ISSN 03600300.

[22] H. M. Gomes, J. P. Barddal, F. Enembreck, A. Bifet, A survey on ensemble learning for data stream classification, ACM Computing Surveys 50 (2) (2017) 1–36, ISSN 0360-0300.

[23] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, Journal of Machine Learning Research 13 (Mar) (2012) 723–773.

[24] A. Haque, L. Khan, M. Baron, B. Thuraisingham, C. Aggarwal, Efficient handling of concept drift and concept evolution over stream data, in : Proceedings of the Thirty-second IEEE International Conference on Data Engineering, 481–492, 2016.

[25] M. Harel, S. Mannor, R. El-Yaniv, K. Crammer, Concept drift detection through resampling, in : Proceedings of the Thirty-first International Conference on Machine Learning, 1009–1017, 2014.

[26] D. T. J. Huang, Y. S. Koh, G. Dobbie, A. Bifet, Drift Detection Using Stream Volatility, in : Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, ISBN 978-3-319-23528-8, 417–432, 2015.

[27] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in : Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining, ACM, ISBN 158113391X, 97–106, 2001.

[28] E. Ikonomovska, J. Gama, S. Džeroski, Learning model trees from evolving data streams, Data Mining and Knowledge Discovery 23 (1) (2011) 128–168, ISSN 1384-5810.

[29] E. Ikonomovska, J. Gama, R. Sebastião, D. Gjorgjevik, Regression Trees from Data Streams with Drift Detection, in : Proceedings of the Twelfth International Conference on Discovery Science, Springer, Berlin, ISBN 978-3-642-04747-3, 121–135, 2009.

[30] G. Jaber, A. Cornuéjols, P. Tarroux, Online learning : searching for the best forgetting strategy under concept drift, in : Proceedings of the Twentieth International Conference on Neural Information Processing, Springer, Berlin, Heidelberg, ISBN 978-3-642-42042-9, 400–408, 2013.

[31] I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades, I. Vlahavas, An adaptive personalized news dissemination system, Journal of Intelligent Information Systems 32 (2) (2009) 191–212, ISSN 0925-9902.

[32] I. Katakis, G. Tsoumakas, I. P. Vlahavas, An ensemble of classifiers for coping with recurring contexts in data streams, in : ECAI, 763–764, 2008.

[33] Y. Kawahara, M. Sugiyama, Sequential change-point detection based on direct density-ratio estimation, Statistical Analysis and Data Mining 5 (2) (2012) 114–127, ISSN 1932-1872.

[34] D. Kifer, S. Ben-David, J. Gehrke, Detecting change in data streams, in : Proceedings of the Thirtieth International Conference on Very Large Databases, vol. 30, VLDB Endowment, ISBN 0120884690, 180–191, 2004.

[35] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority : An ensemble method for drifting concepts, Journal of Machine Learning Research 8 (2007) 2755–2790, ISSN 1532-4435.

[36] P. Li, L. He, X. Hu, Y. Zhang, L. Li, X. Wu, Concept based short text stream classification with topic drifting detection, in : Proceedings of the Sixteenth International Conference on Data Mining, 1009–1014, 2016.

[37] I. Žliobaitė, Combining similarity in time and space for training set formation under concept drift, Intelligent Data Analysis 15 (4) (2011) 589–611, ISSN 1088-467X.

[38] I. Žliobaitė, A. Bifet, B. Pfahringer, G. Holmes, Active Learning with drifting streaming data, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 27–39, ISSN 2162-237X.

[39] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, G. Holmes, Evaluation methods and decision theory for classification of streaming data with temporal dependence, Machine Learning 98 (3) (2015) 455–482, ISSN 1573-0565.

[40] I. Žliobaitė, J. Hollmén, Optimizing regression models for data streams with missing values, Machine Learning 99 (1) (2015) 47–73, ISSN 1573-0565.

[41] A. Liu, Y. Song, G. Zhang, J. Lu, Regional concept drift detection and density synchronized drift adaptation, in : Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence, Accept, 2280–2286, 2017.

[42] B. Liu, Y. Xiao, S. Y. Philip, L. Cao, Y. Zhang, Z. Hao, Uncertain one-class learning and concept summarization learning on uncertain data streams, IEEE Transactions on Knowledge and Data Engineering 26 (2) (2014) 468–484, ISSN 1041-4347.

[43] S. Liu, M. Yamada, N. Collier, M. Sugiyama, Change-point detection in time-series data by relative density-ratio estimation, Neural Networks 43 (2013) 72–83, ISSN 0893-6080.

[44] V. Losing, B. Hammer, H. Wersing, KNN classifier with self adjusting memory for heterogeneous concept drift, in : Proceedings of the Sixteenth IEEE International Conference on Data Mining, 291–300, 2016.

[45] N. Lu, J. Lu, G. Zhang, R. L. De Mantaras, A concept drift-tolerant case-base editing technique, Artificial Intelligence 230 (2016) 108–133, ISSN 0004-3702.

[46] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, Artificial Intelligence 209 (2014) 11–28, ISSN 0004-3702.

[47] L. L. Minku, Y. Xin, DDD a new ensemble approach for dealing with concept drift, IEEE Transactions on Knowledge and Data Engineering 24 (4) (2012) 619–633, ISSN 1041-4347.

[48] B. Mirza, Z. Lin, Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification, Neural Networks 80 (2016) 79–94, ISSN 0893-6080.

[49] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recognition 45 (1) (2012) 521–530, ISSN 0031-3203.

[50] H. Raza, G. Prasad, Y. Li, EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments, Pattern Recognition 48 (3) (2015) 659–669, ISSN 0031-3203.

[51] G. J. Ross, N. M. Adams, D. K. Tasoulis, D. J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognition Letters 33 (2) (2012) 191–198, ISSN 0167-8655.

[52] J. C. Schlimmer, R. H. Granger Jr, Incremental learning from noisy data, Machine Learning 1 (3) (1986) 317–354, ISSN 0885-6125.

[53] J. Shao, Z. Ahmadi, S. Kramer, Prototype-based learning on concept-drifting data streams, in : Proceedings of the Twentieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2623609, 412–421, 2014.

[54] A. Storkey, When training and test sets are different : characterizing learning transfer, Dataset Shift in Machine Learning (2009) 3–28.

[55] W. N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in : Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining, ACM, 502568, 377–382, 2001.

[56] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101, ISSN 0885-6125.