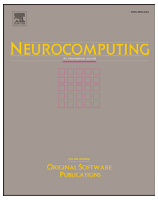
Neurocomputing 653 (2025) 131190

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Concept drift detection based on radial distance

Dan Shang, Guangquan Zhang, Jie Lu*

Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia

ARTICLE INFO

Communicated by Z. Wang

Keywords:
Concept drift
Classification
Stream data mining
Neural network

ABSTRACT

With the advancement of powerful computational hardware in recent years, neural networks, once popular for machine learning on pre-collected datasets, are becoming applicable for streaming data processing. Stream data have the characteristics of high velocity, variety and volume. In stream data mining a common challenge is concept drift, which refers to the phenomenon where the statistical properties of the target variable in predictive tasks change over time. For instance, in image prediction problems, the input facilities may be altered by the environment or device flaws. The materials generated from them could be distorted, such as blurring, discoloring or part-missing. Concept drift problem is considered a root cause of performance degradation in machine learning models on stream data. Traditional concept drift detection methods usually require large amount of historical data, which leads to substantial memory footprint and high computational cost, and tend to be overly sensitive to arbitrary distribution changes not related to prediction results. Such limitations are particularly evident in settings using neural network models, where input data are usually high dimensional images or videos. Aiming to improve the accuracy and efficiency of concept drift detection in neural network models, we propose a new concept drift detection method that specifically addresses these limitations and is applicable to general neural network models. Our method represents the original data with a distance-based statistic, extracted from the layer outputs of the neural network models, and is able to adjust its sensitivity to input distribution changes based on their relevance to neural network features. We evaluated our method with popular neural network architectures on both synthetic and real-world data sets. The results showed our method not only outperforms existing concept drift methods in accuracy, but is also significantly faster and consumes less resources.

1. Introduction

In the field of machine learning, stream data mining attracts lots of attention from researchers. Stream data is being generated from multiple applications in many different scenarios, for example, stock market quotes, outputs from various equipment sensors and social media feeds. Unlike traditional machine learning problems, stream data mining has its own characteristics. First, streaming data sources produce information continually and at a high rate [1]. The processing system may not have enough memory to store the endlessly arriving data elements [2]. Second, the underlying distributions of the data streams have potential changes in unforeseen ways. In literature, this characteristic of evolving is referred to as concept drift [3], which would likely degrade the accuracy of models over time. Algorithms designed for streaming data should be able to adapt to the concept drift in a timely manner while performing with lower memory and computation costs.

Concept drift is a well-studied research field. Existing approaches can be categorized into two branches: incremental learning and detection combined with retraining. Incremental learning algorithms [4] are widely used in streaming data settings. They use incoming data continuously to adapt to the data distribution change. Detection and then retraining is another popular strategy to handle concept drift problems. They use concept drift detection methods such as [5,6] to monitor the stability of the incoming data. If a drift alert is triggered, the model will be retrained with new data set. In this strategy, drift detection methods can be directly applied to the traditional models.

In the meanwhile, artificial neural networks have undergone tremendous growth in recent years. Many artificial neural network models have been favorably used in stream applications, such as image recognition, fraud detection [7], etc. For neural network models, the concept drift problem is also a significant concern. For example, in image prediction

* Corresponding author.
Email addresses: dan.shang@student.uts.edu.au (D. Shang), guangquan.zhang@uts.edu.au (G. Zhang), jie.lu@uts.edu.au (J. Lu).

https://doi.org/10.1016/j.neucom.2025.131190
Received 14 March 2023; Received in revised form 2 August 2025; Accepted 3 August 2025
Available online 7 August 2025

problems, the input facilities may be altered by the environment or device flaws. The materials generated from them could be distorted, such as blurring, discoloring or part-missing. Also, the proportion of input images containing targets of interest may vary seasonally. In order to deal with these evolving characteristics without changing the whole structure of the original models, concept drift detection methods are advantageous in monitoring the status of the data. When changes are detected, the weights of the networks should be adjusted to adapt to the novel information.

In unsupervised settings, traditional concept drift methods, such as [5,8], cannot achieve a desired effect. They first choose a window method to divide the data into smaller, manageable segments. Once the data is divided into batches, the next step is to apply a two-sample statistical test to compare the distributions between the batches. They need to store sufficient data to make the test statistically significant. In the context of neural networks, which usually process images, videos, and audios, retaining large amounts of data may not be feasible. Some methods approach this problem using output features of intermediate or the last layer of the model instead of the original input data, such as [9,10]. However, in deep neural network models, even these abstract feature representations are still high dimensional data. It will consume enormous memory to store these historical data. In addition, existing concept drift detection methods [11,12] treat all features in the whole feature space equally in terms of their test power. However, in neural network settings, data from different classes tend to have different subspaces of features. These features should be given more test power to enhance the detection accuracy, in contrast to other features. Thus, an efficient concept drift detection method for such problems becomes highly desirable, as we need to both avoid over-sensitiveness induced by irrelevant features and reduce the computation and storage cost.

Targeting to deal with these challenges in concept drift detection for the neural network settings, we proposed a method named Radial Distance Drift Detection (RDDD), aiming to avoid over-sensitive retraining, while eliminating the need to store high dimensional input data sets. We use the radial base distance to represent each data point to reduce the dimensionality of the data. The other advantage of using the radial distance is that we could adjust the sensitivity to specific features of each class by modifying the location of the radial base. Data from different classes tends to activate different groups of neurons in the same layer of a neural network. So for the features we want to focus on, we choose higher values for the radial base. This will make the radial distance more sensitive to changes in those important features. For features we hope to downplay, we choose the mean values for the radial base. This will effectively reduce the influence of those less relevant features in the data representation. This method can be applied to most classification models with neural network architectures. It only uses the activation of the neural network layers as input. Therefore the prediction model does not require extra computational costs. Finally, our method is suitable for both sudden and gradual drift detection by adjusting the test window size.

The main contribution of this work is: we propose a radial distance-based feature for drift detection in neural networks, that can adjust detection sensitivity based on the feature subspace. Changes within the target subspace are emphasized, while variations in irrelevant dimensions are suppressed, leading to more reliable detection. Experiments on two synthetic data sets and three real data sets with 5 neural network architectures demonstrate the competitive performance compared with existing concept drift methods. The remainder of the paper is organized as follows. Section 2 reviews the literature on traditional concept drift strategies for neural network settings. In Section 3, we present the preliminaries and articulate the problem addressed in this work. Section 4 introduces the radial distance concept and presents our proposed methodology. Section 5 experimentally evaluates our method on a variety of data sets. Finally, Section 6 provides conclusions and future research.

2. Related work

In this work, we aim to develop an efficient method to detect concept drifts in neural network scenarios. In this section, we review concept drift problems, concept drift detection methods, neural networks used on stream data and concept drift detection in neural network models. We also review distance-based feature extraction. This work builds upon distance-based feature extraction methods.

2.1. Concept drift

A key challenge in learning from dynamic data streams is concept drift, which occurs when the underlying data distribution changes over time. Formally, concept drift is defined as a change in the joint probability distribution $P(X, y)$ between input features X and target variable y over time, i.e., $P_t(X, y) \neq P_{t+1}(X, y)$ [3]. Concept drift may lead to performance degradation in machine learning models if the drift affects the target concept $P(Y \mid X)$ and the model fails to adapt. In order to be responsive to the ever changing stream data, machine learning models need to either identify when and where the drift occurs or adopt incremental learning to suit the development of a data stream [13]. Handling concept drift mainly includes concept drift detection and adaptation. Concept drift detection copes with the problem of identifying the point where change arises, whereas concept drift adaptation focuses on updating the models according to new data sets [14]. Machine learning models can adapt to changing data through several methods: periodic retraining, retraining combined with drift detection, or incremental learning. Our review mainly focuses on drift detection, which can be used as an external drift detection tool that operates alongside the model. This means the base learner can retain its own structure while performing the drift detection.

In supervised settings, in which class labels are available, detection work is generally performed by monitoring the accuracy degradation of classifiers. Gama's popular Drift Detection Method (DDM) [15] focuses on the number of prediction errors made by the model. DDM is based on the assumption that the increase in the error rates suggests that the distribution of the incoming data is unstationary. It relies on significant changes in error rates to detect drift. Thus, this method is not sensitive enough to slow gradual drifts. An extension method Early Drift Detection Method (EDDM) [16] was proposed to improve the drift detection performance when gradual drifts exist. Unlike methods that simply count errors, EDDM focuses on the distance between consecutive misclassifications. This makes it more sensitive to gradual changes in the data stream. Reactive Drift Detection Method (RDDM) [17] enhances the accuracy of DDM by periodically removing older instances of stable concepts. This helps DDM detect concept drift more effectively, especially in datasets where stable concepts have a large number of instances. Adaptive Windowing (ADWIN) by Bifet and Gavalda [13] is a variable window approach with rigorous performance guarantees. It supervises the mean value of the prediction results based on a sliding window. The length of the window is updated according to stability of the data stream. If there is no change, the length will increase. Otherwise, the window will shrink shorter to drop out the outdated data points. Drift Detection Methods based on the Hoeffding's bounds (HDDM) [6] proposed by Frias-Blanco et al. is also a window based method. This method utilizes Hoeffding's inequality to establish confidence bounds on the estimated mean of the data within the window. Instead of directly analyzing the data distribution as in HDDM, Fast Hoeffding Drift Detection Method (FHDDM) [18] monitors the accuracy of predictions.

In unsupervised settings, detection is usually performed by computing some predefined distance between two different parts of the stream data. Dasu [19] proposed a nonparametric detection method based on the relative entropy. Research [5] investigates the impact of concept drift on case-base competence and proposed detecting change via competence models. Maximum Mean Discrepancy (MMD) [20] is a powerful kernel-based method for measuring the dissimilarity between two probability distributions. MMD leverages the power of kernel methods to

embed probability distributions into a Reproducing Kernel Hilbert Space (RKHS). This space allows for measuring the distance between distributions based on their mean embeddings, which are representations of the distributions in the RKHS. Calculating MMD can be computationally expensive, especially for large datasets. Research [9] employs the outputs of the neural network as the input to MMD, and achieves better results. This approach leverages the feature extraction capabilities of neural networks to transform complex data into a more manageable representation, which is then fed into MMD for a more accurate comparison. This method is also nonparametric, but it comes with the trade-off of increased computational cost, because neural networks often have a large number of parameters. This method can lead to decreased accuracy when only the class distribution changes, but the underlying data distribution remains the same. Some works utilize the distances of learned representations in deep learning. Research [21] proposes an approach for detecting changes in graph streams by learning graph embeddings in deep autoencoders with manifold constraints. Though empirical studies suggest that activations do lie near nonlinear subspaces, they are not necessarily a specific well-defined manifold. Deep learning activations are often analyzed directly in Euclidean space without explicit manifold constraints. Research [22] uses KS test directly on each feature of the activation vector from the Variation Autoencoder to detect anomalous points. Research [23] uses PCA technique to reduce the dimension and employs f-distance to compare the distance of distributions under the multi-Gaussian distribution assumption. The computational cost of PCA and f-distance is high. With window method, the process still needs to store high-dimensional data. Finally, these distribution-based strategies all require performing permutation or bootstrap hypothesis tests to determine the statistical significance of the observed drift. This adds a significant computational burden to detection process. The LSDD method [24] is a kernel-based approach for directly estimating the difference between two probability density functions without going through density estimation. Compared to other kernel-based methods, LSDD maintains strong computational efficiency.

2.2. Concept drift handling in neural network settings

Neural networks are inspired by biological neural systems and attempt to model the human brain's functional style. Human brains can acquire new knowledge continuously. Artificial neural networks also should have the ability for continual learning. Draelos [25] proposes an incremental method by adding new neurons to neural networks. New neurons are added to identify new information in recently arriving data. In this method, neural network models adapt directly to changing data. ExStream [26] is a memory efficient replay-based method. Instead of storing the entire historical data, it maintains a small, carefully selected subset of past data points, called the "replay buffer." Then mixture of centroids and the new incoming data were used to update the parameters. A very fast versatile elliptic basis function neural network (VEBFNN) [27] is designed to learn the data set in one pass based on the hyperellipsoidal function. The function can be rotated and translated according to the distribution of the data set. VEBF neural network is also proposed, in which the number of neurons in the hidden layer can be automatically updated. Thus, the network continuously adapts to the distribution of the incoming data stream. This kind of network only has three layers, and the performance depends upon the initial values of the parameters. Research [27] proposed evolving granular neural network (eGNN) in which the fuzzy data streams serve as the input to the neural network. The size of the granulars can adapt incrementally to incoming data streams. There are also works dedicated to the deep neural networks for evolving stream data. Research [28] used hybrid architecture to provide better performance. The surface agent is used to adapt to the short term structure, and the deep learning agent searches for the long term structure of the data. Research [29] learned data streams with partial labels. This approach utilized the output of the Deep Belief Network or the Boltzmann machine as the inputs to the supervised classifiers. Then the model can be fine-tuned to the new data streams with the labeled examples. These adaptive learning techniques apply only to specific network structures. A significant constraint is that techniques developed for one type of network may not be directly applicable to others.

Deep learning provides a representation of each data point in a hierarchical manner through multiple transformation layers. More abstract representations are extracted from the less ones of last layer [30]. Thus, concept drift detection methods could also be applied directly to the outputs of the layers, which may help to identify the source of change. Research [31] utilized Restricted Boltzmann Machine to extract features, which are used as the input for the ADWIN detection method. This method can address the imbalanced class problem, but it cannot be applied directly to other types of neural network architectures. Research [32] performs multiple forward passes of a given data instance through the network and analyzes the resulting empirical distribution over the outputs or parameters. This requires significantly more computation than the prediction process. Monte Carlo Dropout [33,34] monitors concept drift in neural networks by tracking the error rate. However, the method cannot be used if the true labels are unavailable. When true labeled data is unavailable, detecting drifts by monitoring the performance of the model becomes infeasible. Drift detection is usually performed by distribution-based methods, which require storing as much historical data as possible. In neural network settings, the output of the neural networks is usually of high dimensions. Thus high storage capacity will be required. Feature extraction or reduction can be a powerful approach to resolve this type of dilemma.

2.3. Feature extraction based on distances

The foundation of this work is distance-based feature extraction, which is a fundamental technique in machine learning. It involves deriving new features from data by calculating various distance measures between data points or between data points and reference points. This approach can enhance the performance of modeling tasks, reducing costs of computation and data storage. Treating distance-based features as univariate variables also enables the application of certain statistical tests in machine learning, such as the Kolmogorov-Smirnov (K-S) test. In literature, some studies extract distance-based features to improve accuracy. Tsai and Lin [35] proposed a method for intrusion detection based on the new features extracted as triangle area formed by the data point and the class cluster centers. They evaluated the approach on the KDD-cup99 data set [36], and provided higher accuracy and the lower false alarm rate. Since data are represented by a number of triangle areas, if the number of the cluster centers is large, their approach will lead to the curse of dimensionality [37]. Research [38] proposed a feature extraction method using Delaunay triangulation to achieve better discrimination power for the task of online handwritten character recognition. The topological structure is reflected in the new feature space. They achieved good results in the HMM-based recognition system. Research [39] introduced a novel distance-based feature extraction method. They utilized the discrimination ability of the centroid for each cluster. The original feature space is concatenated with the newly extracted distance features. Classification experiments based on the Naive Bayes, kNN, and SVM algorithms show classification improvements for most of the data sets. However, this method cannot achieve the same good results in the image related datasets which usually come with high dimensions.

3. Preliminary

In this section, we first introduce the formal definitions of concept drift in data stream mining. Next, we justify our approach of applying detection on $P(X \mid y)$ (monitoring features per class). Finally, we analyze the properties of neural network hidden layer outputs and their relevance to our class-conditional distribution drift detection.

3.1. Data stream mining and concept drift definition

Data stream is defined as a sequence $(s_1, s_2, \dots, s_i, \dots)$, $i \in \mathbb{N}^+$ of instances that are generated continuously. s_i denotes the ith sample. The size of the sequence may be infinite. In unsupervised cases, each datum $s_i = [x_i^1, x_i^2, \dots, x_i^d]$ is a d-dimensional vector ($s_i \in \mathbb{R}^d$). x_i^j represents the jth feature of the ith sample s_i. In supervised classification tasks, the element is $s_i = [x_i^1, x_i^2, \dots, x_i^d, y_i]$. The last component y_i is the assigned class label and takes values from a finite set of classes $Y = \{c_1, c_2, \dots, c_m\}$. $m \in \mathbb{N}^+$ is a finite number. In a probabilistic setting, the observation s_i can be considered as a pair of random variables (X_i, y_i) drawn from the sample space $S \subset \mathbb{R}^d \times \mathbb{R}^1$. $X_i = [x_i^1, x_i^2, \dots, x_i^d]$ takes values in $X \subset \mathbb{R}^d$, $y_i \in Y \subset \mathbb{R}^1$. In supervised classification case, the classifier $f : X \to Y$, which is trained on the data set $\{(X_i, y_i)\}_{i=1}^n$, approximates the true data-generating process $P(X, y)$. The goal is to minimize the probability of misclassification: $P(f(X_j; \{(X_i, y_i)\}_{i=1}^n) \neq y_j)$ for new data points (X_j, y_j), which are drawn from the same distribution $P(X, y)$. $f(X_j; \{(X_i, y_i)\}_{i=1}^n)$ is the classifier's prediction for X_j, conditioned on the training data set $\{(X_i, y_i)\}_{i=1}^n$.

In real world settings, the underlying generating process $P(X, y)$ may change over time. Formally, we say that concept drift occurs if: $\exists$ time t_1, t_2, such that $P_{t_1}(X, y) \neq P_{t_2}(X, y)$. The evolution may occur in two ways: change originates from the probability of observing X or from the conditional probability of observing y given X, denoted as $P(y \mid X)$. If the change occurs only in P(X), this kind of drift is referred to as virtual drift. Change which only happens in $P(y \mid X)$ is referred to as real drift. If the posterior $P(y \mid X)$ changes after the model is deployed, the model's predictions may become less accurate over time, leading to higher error rates.When true labels are available, tracking error rate (misclassification rate) is an effective way to detect model performance degradation caused by data drift. When true labels are not available, practitioners typically monitor drifts in the input feature distribution $P(X)$. Though it is not a perfect proxy for actual performance degradation, this approach can be served as a practical early warning system. Two sample test methods are usually used for $P(X)$ monitoring to test whether two batches of data sets (for example, $\{X_i\}_{i=1}^n$, $\{X_i\}_{i=n+1}^{2\,n}$) are from the same distribution. If the test results are significant, it is considered that concept drift has occurred.

There are two critical limitations of monitoring $P(X)$. First, in high dimensional spaces two-sample tests lose statistical power. Second, when class ratio (prior probability $P(y)$) changes, two-sample tests on $P(X)$ will flag drift, but model performance may not degrade if the decision boundary remains valid. Instead of considering the overall $P(x)$, a more efficient way to detect concept drift that actually impacts model performance is focusing on the class-conditional distributions $P(X \mid c_k)$. We assume that if $P(X \mid c_k)$ changes, the decision boundary may become invalid, leading to real performance degradation. Then the model needs to be retrained to adapt to the new concept. For each class c_k, we compare samples $\{X_{train}^{c_k}\} \sim P_{train}(X \mid c_k)$ and $\{X_{test}^{c_k}\} \sim P_{test}(X \mid c_k)$. We supervise every class separately in the meantime. This method will also be more suitable to deal with class imbalanced classification problems, in which case the majority class will overwhelm the minority class. In practice, what our method used to test is $P(x \mid y' = c_k)$, where y' is the predicted label of the model, which is available without extra cost.

3.2. Properties of hidden layers' outputs

The jth node in the hidden layer Z denoted as z_j can be computed by $z_j = f(w \cdot x + b)$. w represents weights, b is the bias vector, and f is the activation function. The Tanh, Logistic and ReLU functions are common choices for activation functions. In many practical neural networks (especially efficiency-focused ones), a significant fraction of neuron outputs are zero. In well-trained classifiers, neurons fire selectively for specific classes. The network learns to separate dissimilar classes in activation space, so the activations of each class tend to cluster in lower dimensional affine subspaces within the high-dimensional activation space. We choose the vector Z to represent the original input data X. For each class c_k, comparing activation samples $\{Z_{train}^{c_k}\}$ and $\{Z_{test}^{c_k}\}$ will improve computational efficiency and accuracy compared to a global comparison across all classes. For a given class c_k, certain neurons (dimensions) in the network's representation Z have high activations, while others are near zero. These high-activation neurons are important for discriminating c_k. In our method, we aim to adjust the test power to the dimensions with higher values and ignore the other features that are less relevant to the corresponding class cluster. We accomplish this by introducing the concept of radial distance in the next section.

4. Methodology

In this section, we first present a distance feature called radial distance and its properties. Then, we propose our drift detection method for neural network models based on the radial distance feature.

4.1. Radial distance as a feature

In high dimensional situations, reducing the dimension of data sets has two benefits. First, rather than storing the whole set of features, we need only store part of the whole set, this will help save memory and computational resources. Second, removing the redundant features can also improve accuracy of models. In our method, we only extract one distance feature to use as the input of the detection algorithm. This will lead to a very high efficiency. The distance feature we have chosen should have more test power for the original features that are more related to the corresponding class cluster. Instead of choosing the centroid of the data sets, we choose some fixed point with high feature values in some features and mean values in other features. For the features we want to mainly focus on, we choose higher values for the radial base. For features we want to ignore, we choose the mean values for the radial base. The following will explain why we choose radial base in that way.

Definition 1 (Radial Distance and Radial Base). For a data set $\{s_i = (x_i^1, x_i^2, \dots, x_i^d)\}$, there is a fixed point $B = (x_*^1, x_*^2, \dots, x_*^d)$, the radial distance of s_i is $d(s_i) = \|s_i - B\|$. The point B is the radial base.

In this work, we use the Euclidean distance function. For some n dimensional data point $X = (x_1, x_2, \dots, x_n)$, the Euclidean distance from radial base B is:

$$d(X) = \|X - B\| = \sqrt{\sum_{i=k}^{n} (x_k - b_k)^2} \tag{1}$$

$B = (b_1, b_2, \dots, b_n)$ is the chosen radial base. $d(X)$ is a multivariable scalar function, which is differentiable. So the directional derivative exists along any unit vector v. $\nabla_v d(X) = \nabla d(X) \cdot v$. For the basis vector of Euclidean space $v_i = (0, \dots, 0, 1, 0, \dots, 0)$

$$\nabla_{v_i} d(X) = \nabla d(X) \cdot v_i = \frac{x_i - b_i}{\sqrt{\sum_{i=k}^{n} (x_k - b_k)^2}} = \frac{1}{\sqrt{\sum_{k=1}^{n} \left(\frac{x_k - b_k}{x_i - b_i}\right)^2}}. \tag{2}$$

From the above equation, we can infer that if the $x_i - b_i$ is larger relative to the other features $x_k - b_k$, the directional derivative along the direction v_i ($\nabla_{v_i} d(X)$) will be larger. That means the radial distance feature d will be more sensitive to the change of x_i. So we can adjust the location of the radial base to modify the sensitivity of d with respect to x_i. We illustrated this property in the example of (Experiment 1) 5.2.2.

4.2. Drift detection in artificial neural network models

In classification tasks, artificial neural networks learn activation vectors in a way that is driven by the classification objective. During the training process, neural network models help to filter out irrelevant features. Testing on the intermediate layers will give us the benefit of focusing on the task-relevant features, ignoring redundant or noisy ones. Due to their layered architecture, networks can automatically learn

hierarchical representations of data, from simple low-level features to complex high-level abstractions. In this paper, the input layer does not count in the number of layers in a neural network. The output layer is more constricted than hidden layers in structure. It usually represents the classification results. Many features will disappear in this layer. Our detection method can be applied to the intermediate layer activations generated during the reference stage.

In practice, we hope to avoid alerting concept drifts, which stem from the irrelevant features. A network has L layers, with $g_1, g_2, \ldots, g_L$ neurons in each layer respectively. The output of each layer l (where $l \in \{1, 2, \ldots, L\}$) is represented as a vector Z^l with dimensionality g_l. Our method will select one of the intermediate layer outputs $Z^1, Z^2, \ldots, Z^L$ as the input. Each layer's output Z^l captures a hierarchical representation of the input data. In contrast to the original input data, testing on Z^l filters out irrelevant noise and focuses on task-relevant features. By monitoring drift in Z^l instead of raw input data, the algorithm can ignore nuisance variations. This is efficient because the algorithm could avoid some retraining work triggered by irrelevant feature variations. Drift detection method used on the outputs of different layers will also help us to find out different sources of concept drift.

In our method, the data points will be stored in the form of a set of radial distances to save memory, since distances are only real numbers with one dimension. By adjusting the location of the radial base we can adjust the discrimination power of the radial distances for different subspaces of the data sets. For a given data set $\{s_i^{c_k}\}$ with predicted class label c_k, we choose the layer $Z^l(s_i^{c_k})$. Distances between $Z^l(s_i^{c_k})$ and the radial distance base B^{c_k} are computed, denoted as $dist_i^{c_k}$. We use $dist_i^{c_k}$ to represent the data point $s_i^{c_k}$. The new data set $\{dist_{s_i}^{c_k}\}$ is generated from the original data set $\{s_i^{c_k}\}$ during the referencing stage. Algorithm 2 provides the process of generating referential radial distances initialization. For the next batch of data set $\{t_j^{c_k}\}$ with the same predicted class label c_k, we perform the same process, computing corresponding distances $\{dist_{t_j}^{c_k}\}$. These distances also serve as compact, one-dimensional representations of the original data set $\{t_j^{c_k}\}$. For drift detection, we only need to choose methods suitable for one dimensional settings. We can choose K-S test. If the distributions of data set $\{dist_{s_i}^{c_k}\}$ and $\{dist_{t_j}^{c_k}\}$ are different, we consider the two original data sets $\{s_i^{c_k}\}$ and $\{t_j^{c_k}\}$ to come from two different distributions. A drift will be triggered. Algorithm 3 provides our detection method using the outputs of a given hidden layer.

To obtain the radial distance of a set of layer activations $\{Z_i^{l,c_k}\}$ (output of the lth layer with class label c_k), we should choose a suitable radial base vector $B = (b_1, b_2, \ldots, b_{g_l})$ for the data set $\{Z_i^{l,c_k}\}$. In practice, we could choose the radial base as follows:

$$B(\{Z_i^{l,c_k}\}, \gamma) = \left\langle \begin{cases} \lambda \max_i(z_{i,j}^{l,c_k}), & if \quad \max_i(z_{i,j}^{l,c_k}) > \underset{j}{percentile}(\max_i(z_{i,j}^{l,c_k}, \gamma)) \\ \overline{z_{i,j}^{l,c_k}}, & otherwise \end{cases} \right\rangle \quad (3)$$

$z_{i,j}^{l,c_k}$ is the activation of neuron j in layer l for the ith input of class c_k. For each neuron j, $\max_i(z_{i,j}^{l,c_k})$ is the maximum activation across all inputs i of class c_k. $t = \underset{j}{percentile}(\max_i(z_{i,j}^{l,c_k}, \gamma))$ is the $\gamma - percentile$ of all per-neuron maximum activations. γ is the threshold percentile of layer activation values (by default 0.5) to split activations into highly activated neurons and weakly activated neurons. λ is a positive scaling integer, by default 2. For each neuron j, if maximum activation $\max_i(z_{i,j}^{l,c_k})$ is lower than t, b_j is set to be $\overline{z_{i,j}^{l,c_k}}$(mean value of the $\{z_{i,j}^{l,c_k}\}$). If maximum activation is higher than t, $b_j = \lambda \max_i(z_{i,j}^{l,c_k})$. Algorithm 1 presents the radial base initialization process. Lines 1–6 get the activations of some predefined layer for each input in X (training data set of a given class). Then we find the maximum of each activation and set $b = \max(z_j^l(x))$. After that, lines 7–9 find threshold activation value t of all values in per-neuron maximum activations according to threshold percentile γ. Finally, we set the values which are smaller than the threshold value t as $b = \overline{z_j^l}$. Then return B as the radial base. This radial base will help to enlarge the test power compared to the centroid. In neural network settings, we choose different radial bases for different class clusters. Fig. 1 shows the relationship between the chosen radial base for the class cluster A and features of the other class B. The dot-dashed line represents the radial base of Class A. Its high value features are set to be twice the percentile (dashed line) of the activation value of Class A features (shaded area in color red). As comparison, percentile of Class B features is also shown as shaded area in color blue. The method is designed to avoid dependency on distribution density. Fewer sample points in minority class do not impact the calculation of radial base and K-S statistics. Thus, the proposed method is suitable for imbalanced datasets.

Algorithm 1: Radial distance base initialization.

input : X, training data set of a specific class;
Z^l, activation of layer l in neural network model;
γ, threshold percentile of layer activation values, by default 0.5.
output: radial distance base vector.

1 **forall** *input x in X* **do**
2 get layer activations $Z^l(x)$ for input x;
3 **foreach** *neuron j in layer L* **do**
4 find $\max(z_j^l(x))$ as the maximum activation value of neuron j given all x;
5 initialize vector B as radial distance base
6 B ={neuron j in l: $\lambda \max(z_j^l(x))$};
7 find threshold activation value t of all values in B according to threshold percentile γ;
8 **foreach** *value b in B* **do**
9 **if** $b < t$ **then**
10 set $b := \overline{z_j^l}$;
11 **return** *B as the radial distance base.*

Algorithm 2: Referential radial distances initialization.

input : X, referential data set of a specific class;
Z^l, activation of layer l in neural network model;
B, radial distance base;
$Dist$, distance function, by default Euclidean.
output: list of referential radial distances.

1 initialize $R :=$ empty list;
2 **forall** *input x in X* **do**
3 get activation $Z^l(x)$ for input x;
4 compute $Dist(Z^l(x), B)$, the distance from activation vector to base vector;
5 append $Dist(Z^l(x), B)$ to R;
6 **return** *R as the list of referential radial distances.*

4.3. Computational analysis and limitations of the proposed method

Radial distance is a feature we extracted to represent the original data to reduce the storage and computation cost. Since we only choose the radial distance feature to represent the data point, in applications we only need to compute the distance and store it. We do not need to store the whole high-dimensional data sets. In drift detection phase, we only compare two batches of one dimensional data sets. The computation cost is low. Compared with multidimensional two sample test methods which usually use permutation tests, we can choose K-S test.

Algorithm 3: Radial distance drift detection (RDDD).

input : x, new input data sample from test data set or stream, predicted as a specific class;
Z^l, activation of layer l in neural network model;
n, size of the list of testing radial distances;
B, radial distance base;
$Dist$, distance function, by default Euclidean;
R, the list of referential radial distances;
T, mutable list of testing radial distances, by default empty list;
θ, confidence threshold for drift detection, by default 0.05.

output: boolean detection result.

get activation $Z^l(x)$ for input x;
compute $Dist(Z^l(x), B)$, the distance from activation vector to base vector;
append $Dist(Z^l(x), B)$ to the end of T;
if $length(T) < n$ **then**
  return *False, as not enough testing samples*;
if $length(T) > n$ **then**
  remove one element from the beginning of T;
apply Kolmogorov-Smirnov test to R and T and get P-value $KS(R, T)$;
if $KS(R, T) < \theta$ **then**
  return *True (drift)*;
else
  return *False (no drift)*

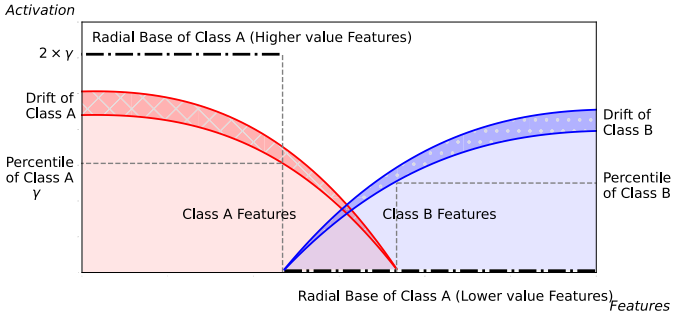


Fig. 1. Radial Base (dot-dashed line) of Class A features. Its high value features are set to be twice the percentile (dashed line) of the activation value of Class A features (shaded area in color red). As comparison, percentile of Class B features is also shown (shaded area in color blue).

Excluding permutation tests will save lots of computation cost. Our approach begins with a one-time preprocessing step that converts the input data into a suitable one-dimensional representation by computing the distance from a radial base. This step has a time complexity of $O(n)$, where n is the number of data points. In the subsequent Kolmogorov-Smirnov (K-S) test stage, the complexity is $O((n+m)log(n+m))$, where m is the window size.

Due to our method adding some kind of bias by choosing the location of the radial distance base, our method is mainly suitable for drift detection in neural network situations. In our method the data should have the property that it live in a lower dimensional affine space to achieve good results. Our method just focuses on the class clusters $P(X|y)$. If only the class prior changes, we will miss them. In the Section 3.2, we know that the data sets extracted from neural networks have two properties. First, the activated neurons by the same category samples are the same. Second, the activated neurons by different categories are different.

Table 1
Synthetic datasets with coordinate drift.

Data set	Initial mean	Drift description
D_{v_1}	(0.9, 0, 0.9)	X1-direction drift decrease by 0.1
D_{v_2}	(0.9, 0, 0.9)	X2-direction drift increase by 0.1
D_{v_3}	(0.9, 0, 0.9)	X3-direction drift decrease by 0.1
D_{v_4}	(0, 0, 0.9)	X1-direction drift increase by 0.1
D_{v_5}	(0, 0, 0.9)	X2-direction drift increase by 0.1
D_{v_6}	(0, 0.9, 0.9)	X1-direction drift increase by 0.1
D_{v_7}	(0, 0.9, 0.9)	X2-direction drift decrease by 0.1

These properties ensure the fitness. We could choose a radial base for each class cluster to perform concept drift detection. In our method, the related features will be given more test power. If the unrelated features change more, our method tends to ignore them.

5. Evaluation

In this section, we assess RDDD's capability to identify drift across different deep learning classifiers for synthetic and real world image and text data (described in Section 5.1) by analyzing its drift detection performance and runtime efficiency (in Section 5.2) against existing drift detection methods.

5.1. Experimental settings

5.1.1. Synthetic datasets

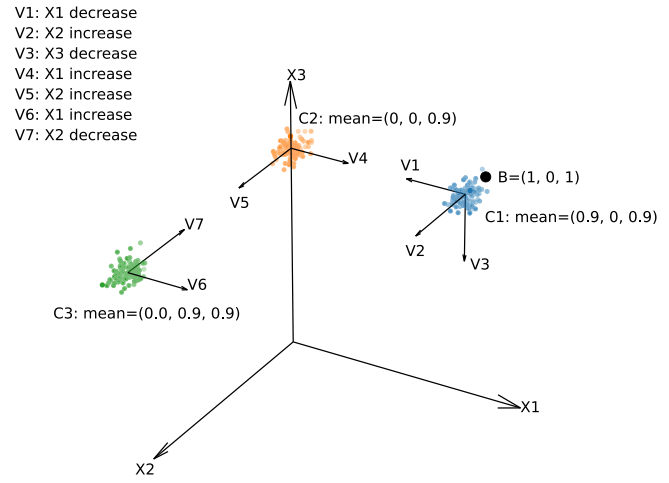
In experiment 1, we test radial distance locality sensitivity. We generate data from a 3D normal distribution with simulated drift by changing the mean along different axes. Three batches of data sets C_1, C_2, C_3 are used as baseline data with fixed means (0.9, 0, 0.9), (0, 0, 0.9), (0, 0.9, 0.9) and standard deviation 0.001. Seven datasets D_{v_i}, $i = 1, 2, \ldots, 7$ are generated where drift is introduced by shifting the mean of C_1, C_2, C_3 by 0.1 along different axes. They are summarized in Table 1 and also shown in Fig. 2.

In experiment 2, we first evaluate our method with a multivariate normal distribution comprising 15 features. We selected the first seven dimensions as the subspace of interest, while the remaining eight dimensions served as the subspace to be ignored. One baseline data set, was set to 0.9 in the first seven features, while the remaining eight dimensions were set to zero. The other baseline data set have zero mean in all features. Drift was introduced in only the first seven features at four levels: 0.004, 0.005, 0.006, and 0.007. The latter eight dimensions remained unchanged. The window size was 500, and a total of 100 drifts were simulated. The data generating random seed was set to 0. We also evaluated the method's accuracy when drift occurred in a different seven-dimensional subspace. In this scenario, the first seven dimensions remained unchanged, while the next seven dimensions experienced drifts of the same magnitude as in the previous tests.

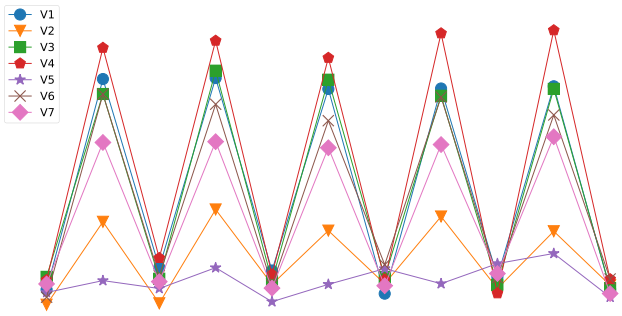
In experiment 3, we test the efficiency of our method in higher-dimensional situations. For datasets with higher dimensions (more than 40 features), the first 20 features are initialized with a mean value of 0.9. Concept drift is introduced in these 20 features by shifting their means by 0.005 over time. The remaining features (those beyond the first 20) have a constant mean of zero and do not experience any drift.

5.1.2. Realworld datasets

For real world data sets in experiment 4, we used MNIST [40], which consists of 70000 handwritten digit images. We choose the subset, which are labeled "3". Three image transformations (rotation, saturation loss, partial erase) are introduced to simulate concept drifts. In experiment 5 we used CIFAR10 [41], which contains 60000 32×32 color images in 10 classes. We choose one subset with the class label "car". We apply saturation, blur, erase, and rotate transformations to simulate drifts. In experiment 6, we used SST-2 (Stanford Sentiment Treebank) [42], which is a widely used benchmark dataset for binary sentiment classification.



(a) Blue data set are generated from the 3d normal distribution with mean (0.9,0,0.9) and standard deviation 0.001. Radial distance base $B = (1, 0, 1)$ locates in the subspace $S_{1,3}$ spanned by x_1 and x_3. Three data sets drift in the directions of arrows V_i (i = 1,2,3,4,5,6,7) with the same amount 0.1



(b) For the same amount of change in different features. The radial distances from radial distance base $B = (1, 0, 1)$ are much more sensitive to change in the direction x_1 or x_3. It ignores some amount of change in the direction of x_2.

Fig. 2. Sensitivity of radial distance to change in different direction.

It consists of movie reviews from the Rotten Tomatoes platform, where each sentence is annotated for sentiment polarity (positive or negative). Drift is simulated by introducing negative samples into the positive class by 5 %, 7 %, 10 % and 12 %. The window size on data sets is 500. We sampled 100 windows from the drifted data sets. The radial base is chosen from the baseline data set (reference distribution) and fixed.

5.1.3. The compared methods

We compared our method with existing methods MMD and CM in experiments 2, 3, and 5. For MMD method, we used the RBF kernel. Both MMD and CM have 100 permutations used for the permutation test. The permutation test p-value for both methods was set to 0.05. The default p-value threshold for distinguishing between drifted and non-drifted distributions is set at 0.05. The MMD and CM parameters are set to the default values by the original authors. In experiment 6, we compare our method with MMD, CM, LSDD, and K-S [43] in Alibi. For LSDD and K-S (in Alibi) methods, we keep the default parameter configuration.

5.2. Drift detection performance evaluation

We test the radial distance locality sensitivity in 3D situation. We also compare our method with MMD and CM using a synthetic dataset generated from a multivariate normal distribution, where only a subset of features undergoes a mean shift. In this part, we also compare our method with MMD and CM in terms of computational efficiency, measuring time and memory costs on higher-dimensional datasets. We evaluate our method on two benchmark image datasets: MNIST (using a simple network architecture) and CIFAR-10 (comparing performance across three popular DNNs). Finally, we conduct comparative analysis against MMD, CM, LSDD, and K-S Detector (in Alibi) on the SST-2 text dataset.

5.2.1. Evaluation metrics

We choose activation layers from different network models as the inputs. The comparison methods also use the same activation vectors as inputs. In experiment 1, the amplitude of the fluctuations reflects the magnitude of change in radial distance when drift occurs, indicating its sensitivity to a particular drift direction. In experiments 2–6, the number of detected drifts served as the metric for accuracy the more drifts detected, the higher the accuracy. P-value threshold is set to 0.05 by default for all methods. We repeated the full process 10 times and obtained the average accuracy result value.

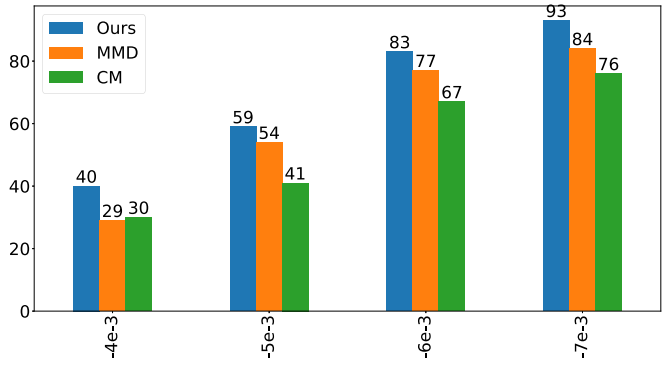
5.2.2. (Experiment 1) Radial distance locality sensitivity

We use radial distance to represent the data point. Therefore, we hope that certain drift in the data can be reflected through changes in radial distance in a synchronized manner. Thus, we want radial distance to be more sensitive to changes in the dimensions we focus on and less sensitive to changes we wish to ignore. In this experiment, we tested the sensitivity of radial distance to drifts in different directions. As shown in Fig. 2 (a), baseline data sets C_1, C_2, C_3 are used as reference distributions before introducing drift. A fixed radial base point B is set to be $(1, 0, 1)$, located in the subspace $S_{1,3}$ spanned by X_1 and X_3. The average radial distance from the radial base B is computed for each data batch. In Fig. 2 (b), the seven lines correspond to average radial distance (from B) changes of seven drift datasets D_{v_i}, which we generate by changing the mean separately in directions of X1, X2, X3 with the same drift amount 0.1. The bottom of each line (baseline) represents no drift occurring, while the top represents drift happening, pushing radial distances away from the baseline. We perform the test five times, so there are five peaks in the figure. From the Fig. 2, it can be observed that for drifts parallel to the $S_{1,3}$ direction (lines V1, V3), the fluctuation amplitude is larger, meaning that for the same amount of drift, radial distance is more sensitive to changes in these two directions X_1 and X_3. Conversely, in line V2 it is less sensitive to changes in the perpendicular direction X_2. Though the orange data set (C2) in Fig. 2(a) is positioned farther from the radial base B compared to the blue dataset (C1), a similar trend is observed. For the green data set (C3), changes in the direction $V7$ don't strongly affect the radial distance as $V6$. This shows our method will be sensitive if the change happens in the direction of X_1 or X_3. In the meantime, it will ignore some amount of change in the direction of X_2.

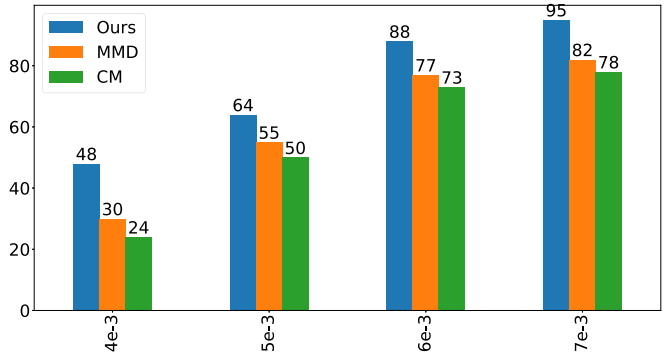
5.2.3. (Experiment 2) Synthetic data sets

To examine the impact of the radial base selection when drift occurs in a subspace, we evaluate the detection performance of our method using the same radial base when drifts occur in different subspaces. Two baseline datasets are generated. The mean of the first data set is set to be (0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0, 0, 0, 0, 0, 0, 0, 0), where the first seven features are non-zero. Drifts are intentionally induced by reducing the means of these seven features by 0.004, 0.005, 0.006, and 0.007, respectively, while the remaining eight features remain unchanged. For the second data set, the mean vector is initialized as (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0). Drifts are introduced by increasing the means of the first seven features by 0.004, 0.005, 0.006, and 0.007, respectively. For both datasets, the radial base is fixed at (1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0), emphasizing the first seven features while ignoring the others. Using the same radial base, we also tested the method when drift occurred in a different seven-dimensional subspace. The first seven dimensions remained unchanged, while the next seven dimensions underwent the same magnitude of drift.

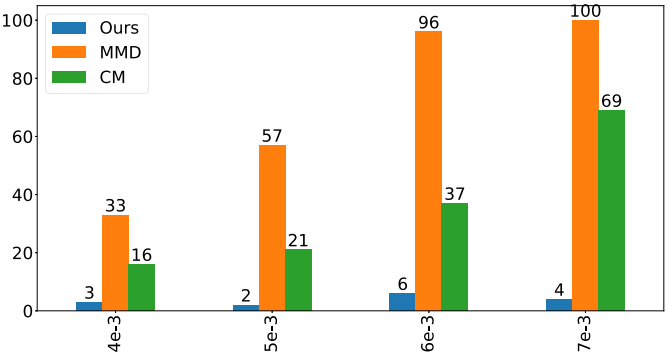
The results are depicted in Fig. 3(a) and (b). As the quantity of change increases, the accuracy of all three methods increases. The detection accuracy here is the number of detected drifts out of the 100 already known drifts. Our method outperforms the other two methods in drift detection accuracy. In contrast, when keeping the first seven features the



(a) Radial base in the subspace where changes happen (mean decrease from 0.9)



(b) Radial base in the subspace where changes happen (mean increase from 0)



(c) Radial base in the subspace where no changes happen

Fig. 3. Accuracy evaluation on synthetic data set.

same, the change occurs in other 7 dimensions. The radial base remains the same as above. That is the situation where the radial distance base is not in the subspace in which the drift occurs. In that situation, our method achieved lower detection rates than the other two. The results are shown in Fig. 3(c). Our method is a biased method. Thus, we can adjust the sensitivity to changes in different feature spaces by changing the choice of the radial distance base. We can achieve higher accuracy during tests in the drifts of the features we aim to focus on.

5.2.4. *(Experiment 3) Efficiency in higher-dimensional situations*

We also test the capability of our method in higher dimensional situations. We compare the accuracy, computation time and memory cost when the dimension is 20, 40, 60, and 80 separately. As shown in Fig. 4, our method outperforms the other two methods in accuracy in high dimensional situations. Besides, our method is very efficient in terms of time and memory cost compared with MMD and CM. The results are listed in Tables 2 and 3. The running times are recorded as the

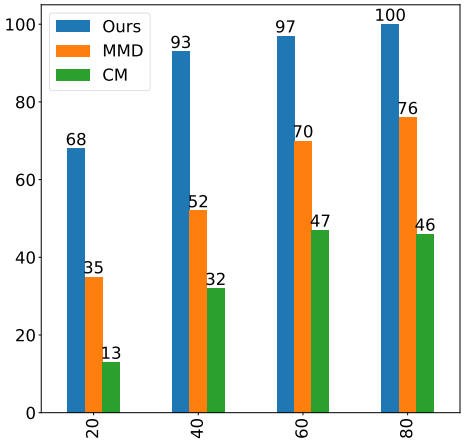


Fig. 4. Accuracy evaluation on high dimension synthetic data set.

Table 2
Running time cost.

Dim	CM(s)	RDDD(s)	MMD(s)
20	10.5	0.5	22.6
40	12.8	0.5	23
60	13.2	0.6	23.6
80	13.4	0.7	24.3

Table 3
Memory cost.

Dim	CM	RDDD	MMD
20	172789760	170369024	183304192
40	173645824	170774528	184774656
60	174854144	170844160	189812736
80	178028544	171114496	193384448

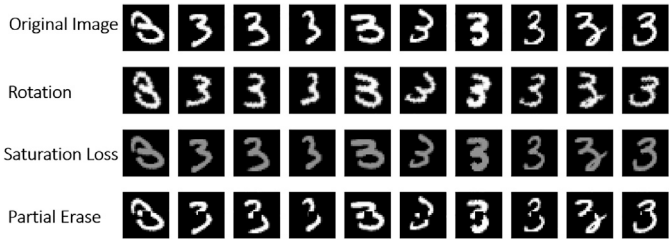
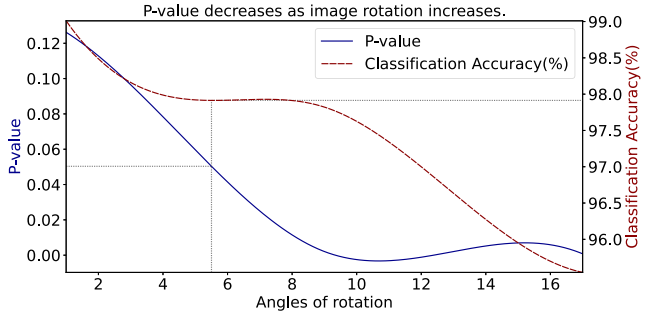


Fig. 5. Samples from the transformed MNIST data sets.

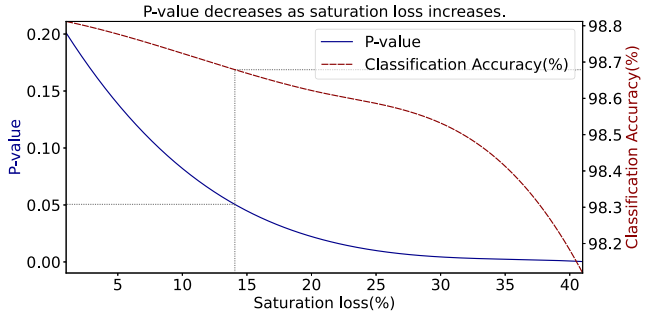
running times of 100 tests. Our method uses far less time resources than the other two methods. CM is faster than MMD. The running times are obtained in a server environment with Intel Xeon 2.40 GHz CPU, 256GB memory and 64bit Red Hat Linux Operating System. The programs are implemented in Python 2.7 with numpy and scipy library stack.

5.2.5. *(Experiment 4) Radial distance sensitivity on image data drifts*

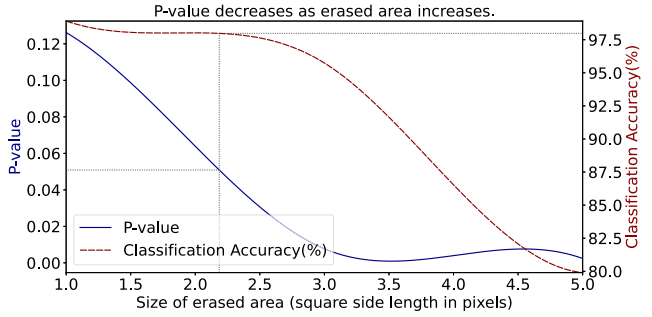
We apply our method on a simple convolutional neural network (CNN) to detect concept drift in the MNIST image data set. The network contains two convolutional layers, two max pooling layers and one fully-connected layer. The kernel size of each convolutional layer is 3. The detection is applied on the fully connected layer. Samples from the transformed MNIST data sets are shown in Fig. 5. Fig. 6 shows the relationships between the image changes and the accuracy degradation. The p-value changes of the KS test are also presented in the figure to show the accuracy change at drift alert points. In this experiment, P-value is the output of the KS 2-sample test in Python. When the magnitude of the changes increases, the P-value outputs decrease.



(a) When the rotation angle reach about 5.5, p-value decrease to 5% . The prediction accuracy drop to about 98%.



(b) P-value dropped below 5% when saturation decreases by 13.6 percent. The prediction accuracy drop to 98.7%



(c) P-value dropped below 5% when erase size increase to 2.2*2.2. The prediction accuracy drop to 97.5%

Fig. 6. Results on the MNIST data set.

When the P-value decreases to below 0.05, we consider that the two data set come from different distributions. We note that p-value of the KS-test decreases as image rotation (or saturation loss, partial erase) increases. Figure (a) shows that when the rotation angle reaches about 5.5, p-value decreases to 5 percent, at which point concept drifts will be triggered. In the meantime, the prediction accuracy decreases to about 98 %. For human eyes such small rotation is barely distinguishable. In Figure (b), P-value dropped below 5 percent when saturation decreases by 13.6 percent, and drift detection will trigger alarm. The prediction accuracy decreases to 98.7 %. In Figure (c), P-value dropped below 5 percent when erase size increases to 2.2*2.2. The prediction accuracy decreases to 97.5 %. This experiment shows that as the transformation degrees increase the p-value of the KS test drops more sharply than the accuracy. That means in prediction stage our method will help to detect drifts before the performance drops sharply.

5.2.6. *(Experiment 5) Drift detection on deep neural networks*

We evaluate our methods on some popular deep network architectures, including AlexNet [44], VGG [45] and ResNet50 [46] on CIFAR10. These architectures all contain massive parameter spaces and produce high-dimensional hidden vectors. These networks are pretrained on ImageNet and fine tuned on the CIFAR-10 data set. The AlexNet has eight layers, consisting of five convolutional layers and 3 fully connected layers followed by ReLU activation in each of these layers except for the output layer. We chose the feature activations from the ReLU activation layer after the second fully connected layer, which has about two thousand features. VGG is one of the most popular image recognition architectures. It consists of 13 convolutional layers and 5 maxpooling layers. We chose the feature activations from layer 44 AvgPool2d, which has 512 features. The ResNet has 48 convolutional layers, one MaxPool layer, and one average pool layer. We chose the outputs of the last Bottleneck, which has 2048 features. The results are presented in Table 4. As shown in the table, for the four kinds of drifts (Saturation, Blur, Erase, Rotate) our method (RDDD) outperforms the other two methods in most of settings. For the ResNet network, MMD has similar accuracy to our method. However, when the amount of change is small, our method is more sensitive to small drifts than MMD. We also found that change detection methods behave differently for different change types. In the Saturation and Erase situations, the MMD drift detection method performs better than CM. However, it achieves less accuracy in the Blur and Rotation change detection processes.

Table 4
Result of three neural networks on Cifar10 data set.

		AlexNet			VGG			ResNet		
		RDDD	MMD	CM	RDDD	MMD	CM	RDDD	MMD	CM
Saturation	0.4	96	91	84	67	54	64	91	100	23
	0.3	85	73	65	33	23	34	62	92	6
	0.2	58	51	50	9	4	10	40	23	0
Blur	0.5	98	58	89	78	26	75	97	80	6
	0.49	94	44	85	64	15	63	97	80	6
	0.48	91	35	73	49	7	48	72	18	3
Erase	4*4	100	99	96	100	100	100	99	100	93
	4*3	77	74	50	100	97	82	83	82	66
	3*3	44	40	9	89	83	66	60	44	49
Rotate	3.3	97	42	92	98	89	99	99	97	29
	3.2	94	28	78	85	62	83	76	77	10
	3.1	91	29	71	77	42	71	68	83	1

Table 5
Detection performance on text data sets.

Data set	CM	RDDD	MMD	LSDD	K-S(Alibi)
5 % negative samples	37	24	27	23	31
7 % negative samples	55	39	47	37	45
10 % negative samples	70	69	76	60	61
12 % negative samples	89	97	93	74	80
Running time	11 m	0.2 s	2m39s	7.9 s	1m36s

5.2.7. *(Experiment 6) Drift detection performance on text data sets*

We utilize the pretrained 'DistilBertForSequenceClassification' model from Hugging Face, extracting the outputs from its pre-classifier layer with 768 output features. In this experiment, we compared our method with four existing methods, MMD, CM, LSDD, and K-S Detector in Alibi [43]. The results are shown in Table 5. The parameters of LSDD and K-S Detector in Alibi are set as default in Alibi library. Results in Table 5 show that our technique achieves better drift detection accuracy than previous methods when the new class data exceeds 10 %, and it is at least 10 times faster.

6. Conclusion and further study

This work presents an efficient concept drift detection method dedicated to neural network classification settings. In neural network stream data mining tasks, input data are usually high dimensional images or videos. In literature, concept drift detection methods usually need to

store much historical data and are over sensitive, which leads to substantial memory footprint and high computational cost. The proposed method is developed to improve the efficiency of drift detection. We introduced the radial base distance as a univariate feature representing the original data point. Next, we monitor the univariate radial distance distribution. The proposed method demonstrates two advantages over alternative solutions: (1) it achieves higher time and memory efficiency by using the univariate radial distance feature; (2) it is able to efficiently detect concept drift happening in subspace of the entire space through choosing the location of the radial distance base. Our next attempt will aim to develop adaptive model maintenance algorithms based on the concept drift detection results.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

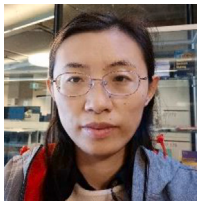
Acknowledgement

This study was supported by Australian Research Council (ARC) under Discovery Grant DP220102635 and Laureate project FL190100149.

References

[1] J. Lu, A. Liu, Y. Song, G. Zhang, Data-driven decision support under concept drift in streamed big data, Complex Intell. Syst. 6 (1) (2020) 157–163.
[2] P. Duda, L. Rutkowski, M. Jaworski, Stream Data Mining: Algorithms and Their Probability Properties, Springer, 2020.
[3] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: a review, IEEE Trans. Knowl. Data Eng. (2018).
[4] J. Read, A. Bifet, B. Pfahringer, G. Holmes, Batch-incremental versus instance-incremental learning in dynamic and evolving data, in: International Symposium on Intelligent Data Analysis, Springer, 2012, pp. 313–323.
[5] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, Artif. Intell. 209 (2014) 11–28..
[6] I. Frias-Blanco, J.D. Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2015) 810–823.
[7] J.I.-Z. Chen, K.-L. Lai, Deep convolution neural network model for credit-card fraud detection and alert, J. Artif. Intell. 3 (2) (2021) 101–112.
[8] A. Liu, Y. Song, G. Zhang, J. Lu, Regional concept drift detection and density synchronized drift adaptation, in: Proc. 26th Int. Joint Conf. Artificial Intelligence, Accept, 2017.
[9] F. Liu, W. Xu, J. Lu, G. Zhang, A. Gretton, D.J. Sutherland, Learning deep kernels for non-parametric two-sample tests 119, 01 2020.
[10] X. Wang, Z. Wang, W. Shao, C. Jia, X. Li, Explaining concept drift of deep learning models, in: International Symposium on Cyberspace Safety and Security, Springer, 2019, pp. 524–534.
[11] F. Gu, G. Zhang, J. Lu, C.-T. Lin, Concept drift detection based on equal density estimation, in: Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 24–30.
[12] T. Dasu, S. Krishnan, S. Venkatasubramanian, K. Yi, An information-theoretic approach to detecting changes in multi-dimensional data streams, in: In Proceedings of the Symposium on the Interface of Statistics, Computing Science, and Applications, May 2006.
[13] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, vol. 7, SIAM, 2007, pp. 2007.
[14] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 1–37.
[15] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of 17th Brazilian Symposium on Artificial Intelligence – SBIA 2004, Springer, 2004, pp. 286–295, Lecture Notes in Computer Science.
[16] M. Baena-García, J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, R. Morales-Bueno, Early drift detection method, in: Proceedings of the Fourth International Workshop on Knowledge Discovery from Data Streams, number 6, 2006, pp. 77–86.
[17] R.S.M. Barros, D.R.L. Cabral, P.M. Gonßalves, S.G.T.C. Santos, RDDM: reactive drift detection method, Expert Syst. Appl. 90 (2017) 344–355.
[18] A. Pesaranghader, H.L. Viktor, Fast hoeffding drift detection method for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16, Springer, 2016, pp. 96–111.
[19] T. Dasu, S. Krishnan, D. Lin, S. Venkatasubramanian, K. Yi, Change (detection) you can believe in: finding distributional shifts in data streams, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 21–34.
[20] A. Smola, A. Gretton, L. Song, B. Schölkopf, A hilbert space embedding for distributions, in: International Conference on Algorithmic Learning Theory, Springer, 2007, pp. 13–31.
[21] D. Grattarola, D. Zambon, L. Livi, C. Alippi, Change detection in graph streams by learning graph embeddings on constant-curvature manifolds, IEEE Trans. Neural Netw. Learn. Syst. 31 (6) (2019) 1856–1869.
[22] J. Li, K. Malialis, C.G. Panayiotou, M.M. Polycarpou, Unsupervised incremental learning with dual concept drift detection for identifying anomalous sequences, in: 2024 International Joint Conference on Neural Networks (IJCNN), IEEE, 2024, pp. 1–8.
[23] S. Greco, B. Vacchetti, D. Apiletti, T. Cerquitelli, Unsupervised concept drift detection from deep learning representations in real-time, arXiv preprint arXiv:2406.17813, 2024.
[24] M. Sugiyama, M. Kawanabe, Learning under nonstationarity: covariate shift and class-balance change, Wiley Interdiscip. Rev. Comput. Stat. 5 (6) (2013) 465–475.
[25] T.J. Draelos, N.E. Miner, C.C. Lamb, J.A. Cox, C.M. Vineyard, K.D. Carlson, W.M. Severa, C.D. James, J.B. Aimone, Neurogenesis deep learning: extending deep networks to accommodate new classes, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 526–533.
[26] T.L. Hayes, N.D. Cahill, C. Kanan, Memory efficient experience replay for streaming learning, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 9769–9776.
[27] S. Jaiyen, C. Lursinsap, S. Phimoltares, A very fast neural learning for classification using only new incoming datum, IEEE Trans. Neural Netw. 21 (3) (2010) 381–392.
[28] A.G. Ororbia Ii, C.L. Giles, D. Reitter, Online semi-supervised learning with deep hybrid Boltzmann machines and denoising autoencoders, arXiv preprint arXiv:1511.06964, 2015.
[29] J. Read, F. Perez-Cruz, A. Bifet, Deep learning in partially-labeled data streams, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, 2015, pp. 954–959.
[30] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[31] L. Korycki, B. Krawczyk, Concept drift detection from multi-class imbalanced data streams, CoRR abs/2104.10228 (2021).
[32] L. Baier, T. Schlör, J. Schöffer, N. Kühl, Detecting concept drift with neural network model uncertainty, arXiv preprint arXiv:2107.01873, 2021.
[33] B. Silva, N. Marques, G. Panosso, Applying neural networks for concept drift detection in financial markets, in: Workshop on Ubiquitous Data Mining, Citeseer, 2012, pp. 43.
[34] S. Disabato, M. Roveri, Learning convolutional neural networks in presence of concept drift, in: 2019 International Joint Conference on Neural Networks (IJCNN), 2019, pp. 1–8.
[35] C.-F. Tsai, C.-Y. Lin, A triangle area based nearest neighbors approach to intrusion detection, Pattern Recognit. 43 (1) (2010) 222–229.
[36] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.
[37] R. Bellman, Dynamic programming, Science 153 (3731) (1966) 34–37.
[38] W. Zeng, X. Meng, C. Yang, L. Huang, Feature extraction for online handwritten characters using Delaunay triangulation, Comput. Graph. 30 (5) (2006) 779–786.
[39] C.-F. Tsai, W.-Y. Lin, Z.-F. Hong, C.-Y. Hsieh, Distance-based features in pattern classification, EURASIP J. Adv. Signal Process. 2011 (1) (2011) 1–11.
[40] L. Deng, The mnist database of handwritten digit images for machine learning research, IEEE Signal Process. Mag. 29 (6) (2012) 141–142.
[41] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.
[42] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642.
[43] S. Rabanser, S. Günnemann, Z. Lipton, Failing loudly: an empirical study of methods for detecting dataset shift, Adv. Neural Inf. Process. Syst. 32 (2019).
[44] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, vol. 25, Curran Associates, Inc., 2012.
[45] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
[46] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

Author biography



Dan Shang is a Ph.D student in Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. Her research focuses on streaming data learning.

Guangquan Zhang is an Australian Research Council (ARC) QEII Fellow, Associate Professor and the Director of the Decision Systems and e-Service Intelligent (DeSI) Research Laboratory at the Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. He received his Ph.D. in applied mathematics from Curtin University, Australia, in 2001. From 1993 to 1997, he was a full Professor in the Department of Mathematics, Hebei University, China. His main research interests lie in fuzzy multi-objective, bilevel and group decision making, fuzzy measures, transfer learning and concept drift adaptation. He has published six authored monographs and over 500 papers including some 300 articles in leading international journals. He has supervised 40 Ph.D. students to completion and mentored 15 Postdoc fellows. Prof Zhang has won ten very competitive ARC Discovery grants and many other research projects. His research has been widely applied in industries.

Jie Lu (F'18) is an Australian Laureate Fellow, IFSA Fellow, ACS Fellow, Distinguished Professor, and the Director of Australian Artificial Intelligence Institute (AAII) at the University of Technology Sydney, Australia. She received a PhD degree from Curtin University in 2000. Her main research expertise is in transfer learning, concept drift, fuzzy systems, decision support systems and recommender systems. She has published over 500 papers in IEEE Transactions and other leading journals and conferences. She is the recipient of two IEEE Transactions on Fuzzy Systems Outstanding Paper Awards (2019 and 2022), NeurIPS2022 Outstanding Paper Award, Australia's Most Innovative Engineer Award (2019), Australasian Artificial Intelligence Distinguished Research Contribution Award (2022), Australian NSW Premier's Prize on Excellence in Engineering or Information & Communication Technology (2023), and the Officer of the Order of Australia (AO) 2023.