Regional Concept Drift Detection and Density Synchronized Drift Adaptation

Anjin Liu, Yiliao Song, Guangquan Zhang, Jie Lu
Centre for Artificial Intelligence, School of Software
Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia
{Anjin.Liu, Yiliao.Song}@student.uts.edu.au, {Guangquan.Zhang, Jie.Lu}@uts.edu.au

Abstract

In data stream mining, the emergence of new patterns or a pattern ceasing to exist is called concept drift. Concept drift makes the learning process complicated because of the inconsistency between existing data and upcoming data. Since concept drift was first proposed, numerous articles have been published to address this issue in terms of distribution analysis. However, most distribution-based drift detection methods assume that a drift happens at an exact time point, and the data arrived before that time point is considered not important. Thus, if a drift only occurs in a small region of the entire feature space, the other non-drifted regions may also be suspended, thereby reducing the learning efficiency of models. To retrieve non-drifted information from suspended historical data, we propose a local drift degree (LDD) measurement that can continuously monitor regional density changes. Instead of suspending all historical data after a drift, we synchronize the regional density discrepancies according to LDD. Experimental evaluations on three benchmark data sets show that our concept drift adaptation algorithm improves accuracy compared to other methods.

1 Introduction

With the rapid development of our digital universe, the ever-growing amount of data poses a great challenge to statistical machine learning models: time-changing probability distributions in high-speed data streams, which is also called concept drift. In conventional models, the probability distribution of a target variable is assumed to be stationary. Under this condition, the statistical theory can minimums the discrepancy between predictions and actual values, namely the error rate. However, as pointed out by Zliobaite et al. [2014] , learning models based on such assumptions are becoming old-fashioned, whereas concept drift applications draw increased attention [Shao, *et al.*, 2014]. Typical concept drift applications include 1) weather prediction models that change from season to season, 2) customer preference recommender systems that vary with time, 3) suspicious stock exchange transaction detection and 4) the web, social media topic drifting [Li, *et al.*, 2016]. The terminology in other research fields may differ, such as covariate shift in machine learning [Sugiyama and Kawanabe, 2012], dataset shift in pattern recognition, and non-stationary learning in signal processing [Zliobaite, *et al.*, 2014].

The awareness of concept drift in machine learning has resulted in a series of in-depth studies of self-adaptive models [Gama, *et al.*, 2014]. Of these studies, concept drift detection is a fundamental technique, that is, the decisive factor in the performance of machine learning under a dynamic environment. The effectiveness of drift detection directly contributes to the performance of online adaptive learning.

At present, most concept drift detection and handling methods are focusing on time-related drift, namely when a concept drift occurs. They consider that a drift could occur suddenly at a time point, incremantaly, or gradualy in a time period [Harel, *et al.*, 2014]. As a result, their solutions are searching the best time to split the old and new concepts. The data received before the drift time point is considered as old concept, while the data received after is considered as new concept. Accordingly, the old concept data is discarded, while new concept data is used for updating or training new learners, which can be seen as a time-oriented "one-cut" process. However, in real-world scenarios, this assumption is not always true. A concept drift could only occur within some specific regions. Such a "one-cut" process does not consider the non-drifted regions in the old concept. Algouth some algorithms have introduced a buffer system to keep tracking drifting concepts and can find the best drift time point to identify concepts, they are not able to retrieve the location informtion related to the drifted regions [Liu, *et al.*, 2017].

To better address concept drift problems, we consider both time-related and spatial-related drfit informtion. In this paper, we propose a regional density inequality metric, called local drift degree (LDD), to measure the likelihood of regional drift in every suspicious region. By analyzing the density increasing or decreasing in a local region, learning systems are able to highlight dangerous regions and take relevant actions.

2 Literature Review

Concept drift is defined as a phenomenon whereby the statistical properties of a target domain change arbitrarily over time [Gama, *et al.*, 2014]. These changes are usually

driven by hidden variables or features that cannot be measured directly. In regard to online classification problems, concept drift is defined as follows: in a data stream, denoting data within a time period (from 0 to t) as $\mathbf{D}_t^0 = \{d_0, \cdots, d_t\}$, and each data instance d is a $(\boldsymbol{X}, y)$ pair, where $\boldsymbol{X}$ is the feature vector, $y \in Y$ is the classification label, we say a concept drift occurrs at time i if the distribution of $\mathbf{D}_i^0$ is statistically different from the distribution of $\mathbf{D}_t^{i+1}$, or simply denoted as $F(\mathbf{D}_i^0) \neq F(\mathbf{D}_t^{i+1})$. Concept drift is critical to online classification problems, because such inequality may lead to a inconsistency in decision boundaries, thereby increasing the error rate. In this paper, the term 'concept' refers to data distribution.

Changes in data distribution over time may manifest in different forms. Given F_1 and F_2 have different probability density functions, the transformation from F_1 to F_2 can be categorized into three types [Lu, *et al.*, 2016]: 1) a sudden/abrupt drift switching from F_1 to F_2 straightaway; 2) gradual drifts and 3) incremental drifts that consist of several intermediate concepts in the transformation process. The major difference between 2) gradual drifts and 3) incremental drifts is whether the intermediate concepts belong to any of the two distributions. For incremental drifts, the distribution of intermediate concepts are affected by both F_1 and F_2 and belongs to neither. By contrast, in gradual drift cases, the intermediate concepts can only be the repetitions of previous concepts or new concepts [Gama, *et al.*, 2014].

In real-world scenarios, one drift could be a mixture of all three types of drifts [Sarnelle, *et al.*, 2015]. How to handle the intermediate concepts in a drift is a challenging problem. So far, most the state-of-the-art drift detection algorithms only detect when the intermediate conepts drift. Very little research has discussed the intermediate concepts from a spatial perspective, such as research on where the drifted regions are. Considering both when and where a drift occurs is beneficial to reduce the risk of overestimating the drift regions, as shown in Figure 1.

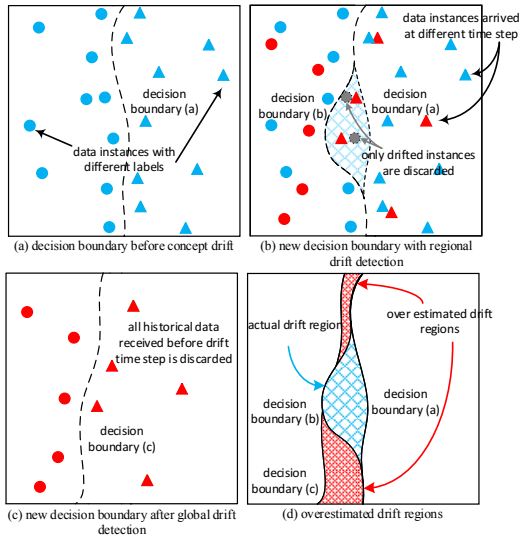

Figure 1. For any notable concept drift, if the distribution difference is only in a small region, as shown in (b), discarding the entire historical data and retraining the learner may result in a overestimation of the drift regions, such as the red shaded regions in (d), and thereby impairing the overall performance. By contrast, regional drift detection and adaptation only address targeted regions, and will not over estimate the drifts.

To the best of our knowledge, very little research has been conducted to explicitly address concept drift by detecting and adapting to regional drifts. In one related publication, [Gama and Castillo, 2006] , the authors applied a decision tree model to detect changes in the online error-rate in each internal tree node, thereby identifying drifted nodes and updating them, respectively. The experimental results showed a good performance in detecting drift and in adapting the decision model to the new concept. Similar algorithms are in [Ikonomovska and Gama, 2008, Ikonomovska, *et al.*, 2011, Ikonomovska, *et al.*, 2009]. However, these algorithms mainly focus on addressing concept drift on regression problems and they all are based on decision tree models, which have limited application areas.

A detailed reviewe of other drift detection algorithms can be found in [Lu, *et al.*, 2016]. The authors suggested to divid drift detection algorithms into three categories: 1) a statistical test monitoring raw data distribution [Lu, *et al.*, 2016, Lu, *et al.*, 2014]; 2) the outputs (error rates) of learners [Frias-Blanco, *et al.*, 2015], and 3) the changes of learner parameters [Su, *et al.*, 2008]. Generally, category 1) algorithms provide a statistical significance level to ensure that detected drifts are not caused by sampling errors, but they cannot easily explain the detected drift regions. Category 2) algorithms have much less computational complexity but are not sensitive to gradual and incremental drifts. To the best of our knowledge, only one attempt at category 3) has been proposed by [Su, *et al.*, 2008]. Their framework for modeling concept drift is creative and can be applied to many learning models. However, it is not suitable for knowledge-based learners and cannot interpret detected drifts well.

3 Local Drift Degree

In this section, we formally present the proposed test statistics, Local Drift Degree (LDD). The purpose of LDD is to quantify regional density discrepancies between two different sample sets, thereby, identifying density increased, decreased and stable regions.

The intuitive idea underlying LDD is that, given two d-dimensions populations $\mathbf{A}^d$ and $\mathbf{B}^d$, if two sample sets, A from $\mathbf{A}^d$ and B from $\mathbf{B}^d$, are independent and identically distributed, their local density discrepancies follow a certain normal distribtuion. Denote the feature space as V, for any subspace $W \subseteq V$, it has an equality that $|A_W|/n_A = |B_W|/n_B$ in an ideal situation where $|A_W|$,$|B_W|$ represents the number of data instances in W that belong to A, B, and n_A, n_B represents the total number of instances in A and B separately.

Definition 1. The local drift degree of a subspace W is defined as:

$$\delta_W = \frac{|B_W|/n_B}{|A_W|/n_A} - 1 \qquad (1)$$

Theorem 1. Given $\mathbf{A}^d$ and $\mathbf{B}^d$ have the same distribution, $\delta_W \sim N(0, \sigma^2)$, where σ^2 is the theoretical variance of δ_W.

Proof. Define A_i as Equation (2), and define B_i, $\mathbf{A}_i^d$ and $\mathbf{B}_i^d$ in the same way.

$$A_i = \begin{cases} 1 & ith\ point\ locates\ in\ W \\ 0 & otherwise \end{cases} \quad (2)$$

Then δ_W can be rewritten as Equation. (3)

$$\delta_W = \frac{\sum_{i=1}^{n} B_i/n_B}{\sum_{i=1}^{n} A_i/n_A} - 1 = \frac{\bar{B}}{\bar{A}} - 1 \quad (3)$$

Assuming $\bar{A}$ contains almost all $\{A_i = 1\}$ in $\overline{\mathbf{A}^d}$, $\bar{A}$ will be very closed to $\overline{\mathbf{A}^d}$. Therefore, $\delta_W \approx \bar{B}/\overline{\mathbf{A}^d} - 1$. In the Appendix, we have $E(\bar{B}) = \overline{\mathbf{B}^d}$, and the expectation of δ_W can be acquired by Equation (4)

$$E(\delta_W) = \frac{E(\bar{B})}{\overline{\mathbf{A}^d}} - 1 = \frac{\overline{\mathbf{B}^d}}{\overline{\mathbf{A}^d}} - 1 \quad (4)$$

If $\mathbf{A}^d$ and $\mathbf{B}^d$ have the same distribution, $\overline{\mathbf{B}^d} = \overline{\mathbf{A}^d}$ and $E(\delta_W) = 0$. According to the Central Limit Theorem, it obeys a normal distribution as it is constructed in terms of the sample average. The variance can be estimated by the Monte Carlo method. □

4 Drifted Instance Selection and Adaptation

4.1 Drifted Instance Selection

The LDD-based drifted instance selection algorithm (LDD-DIS) is shown in Algorithm 1. The core idea of LDD-DIS is to use LDD to identify density decreased ($\mathbf{D}^{dec}$), increased ($\mathbf{D}^{inc}$), and stable ($\mathbf{D}^{sta}$) instances within two batches of data. The inputs are the target data batches, $\mathbf{D}_1$ $\mathbf{D}_2$, the neighbourhood ratio ρ, and the drift significant level α.

The neighborhood ratio ρ controls the size of the neighbourhood. Instead of confining the neighbourhood within a certain range, selecting the k-nearest neighbours (KNN) with a certain proportion of a data set as the neighbourhood is more robust (the k value of KNN is equal to $|\mathbf{D}| \times \rho$). The reason is that KNN-based neighbourhood is independent of the shape of the feature space and is friendly to high-dimensional domains. With a proper data structure, like a binary tree, the complexity of the KNN-search can be reduced to $O(log(n))$, where n is the total number of data instances in a sample set.

The drift significance level α quantifies the statistical significance of concept drift. For example, in our case, if an observation is in the left tail, the system will be $(1-\alpha)\%$ confidence that it is a density-decreased region. Similarly, if an observation is in the right tail, the system will be $(1-\alpha)\%$ confidence that it is a density-increased region.

Without any specifications, LDD-DIS will be initialized by the default input values. LDD-DIS consists of two major steps. One is to estimate the distribution of LDD when no drift occurs, namely $\delta' \sim \mathcal{N}(\mu, \sigma^2)$, lines 1-13. The second is to compute LDD for each data instance according to the input data batches, and selects the drifted instances correspondingly, lines 14-31.

Lines 1 to 4 computes the k-nearest neighbor map of the entire data, where findKNN$(d_i, \mathbf{D}, |\mathbf{D}| * \rho)$ stands for finding the $(|\mathbf{D}| * \rho)$ nearest neighbor of d_i in $\mathbf{D}$. Then, at line 5, we shuffle the data and resample two new batches $\mathbf{D}_1'$, $\mathbf{D}_2'$ with the same size as $\mathbf{D}_1, \mathbf{D}_2$. The resampling guarantees that $\mathbf{D}_1'$ and $\mathbf{D}_2'$ follow an identical distribution. As per theorem 1, consequently, the δ' follows 0-mean normal distribution, and the density decreasing and increasing confidence interval can be calculated by the normal inverse cumulative distribution function, as shown in lines 12, 13, denoted as $\text{norminv}(\alpha, 0, \text{std}(\delta'))$, where α is the significance level, 0 is the mean, and $\text{std}(\delta')$ is the estimated standard deviation. Then, we compute the LDD for each data instance according to their original distributions. For each data instance, if its LDD is less than θ^{dec}, it will be identified as a density decreasing instance, while if its LDD is greater than θ^{inc}, it will be identified as density increasing instance, as shown in lines 14 to 31. Also, if the LDD is between $\theta^{dec} \le \delta_i \le \theta^{inc}$, that instance will be considered as a no drift instance, or a stable instance.

Algorithm 1: LDD Drifted Instance Selection (LDD-DIS)

Input: two batches of data instances, $\mathbf{D}_1, \mathbf{D}_2$
neighborhood ratio, ρ (default $\rho = 0.1$)
drift significance level, α (default $\alpha = 0.05$)

Output: drifted data sets, $\mathcal{D}_{drift} = \{\mathbf{D}_1^{dec}, \mathbf{D}_1^{sta}, \mathbf{D}_1^{inc}, \mathbf{D}_2^{dec}, \mathbf{D}_2^{sta}, \mathbf{D}_2^{inc}\}$

1. merge $\mathbf{D}_1$ and $\mathbf{D}_2$ as $\mathbf{D}$
2. **For** d_i : $\mathbf{D}$
3. retrieve d_i neighborhood, $\mathbf{D}_i^{knn}$ = findKNN$(d_i, \mathbf{D}, |\mathbf{D}| * \rho)$
4. **End**
5. shuffle $\mathbf{D}$ and resample $\mathbf{D}_1'$, $\mathbf{D}_2'$ without replacement
6. **For** d_i : $\mathbf{D}$
7. **If** $d_i \in \mathbf{D}_1'$
8. compute the LDD of d_i by $\delta_i' = \frac{|\mathbf{D}_i^{knn} \cap \mathbf{D}_2'|}{|\mathbf{D}_i^{knn} \cap \mathbf{D}_1'|} - 1$
9. **Else**
10. compute the LDD of d_i by $\delta_i' = \frac{|\mathbf{D}_i^{knn} \cap \mathbf{D}_1'|}{|\mathbf{D}_i^{knn} \cap \mathbf{D}_2'|} - 1$
11. **End**
12. density decrease threshold, $\theta^{dec} = \text{norminv}(\alpha, 0, \text{std}(\delta'))$
13. density increase threshold, $\theta^{inc} = \text{norminv}((1-\alpha), 0, \text{std}(\delta'))$
14. **For** d_i : $\mathbf{D}$
15. **If** $d_i \in \mathbf{D}_1$
16. compute the LDD of d_i by $\delta_i = \frac{|\mathbf{D}_i^{knn} \cap \mathbf{D}_2|}{|\mathbf{D}_i^{knn} \cap \mathbf{D}_1|} - 1$
17. **If** $\delta_i < \theta^{dec}$
18. $\mathbf{D}_1^{dec} = \mathbf{D}_1^{dec} \cup \{d_i\}$
19. **Else if** $\delta_i > \theta^{inc}$
20. $\mathbf{D}_1^{inc} = \mathbf{D}_1^{inc} \cup \{d_i\}$
21. **Else**
22. $\mathbf{D}_1^{sta} = \mathbf{D}_1^{sta} \cup \{d_i\}$
23. **Else**
24. compute the LDD of d_i by $\delta_i = \frac{|\mathbf{D}_i^{knn} \cap \mathbf{D}_1|}{|\mathbf{D}_i^{knn} \cap \mathbf{D}_2|} - 1$
25. **If** $\delta_i < \theta^{dec}$
26. $\mathbf{D}_2^{dec} = \mathbf{D}_2^{dec} \cup \{d_i\}$
27. **Else if** $\delta_i > \theta^{inc}$
28. $\mathbf{D}_2^{inc} = \mathbf{D}_2^{inc} \cup \{d_i\}$
29. **Else**
30. $\mathbf{D}_2^{sta} = \mathbf{D}_2^{sta} \cup \{d_i\}$
31. **End**
32. **return** $\mathcal{D}_{drift} = \{\mathbf{D}_1^{dec}, \mathbf{D}_1^{sta}, \mathbf{D}_1^{inc}, \mathbf{D}_2^{dec}, \mathbf{D}_2^{sta}, \mathbf{D}_2^{inc}\}$

4.2 Density Synchronized Drift Adaptation

In this section, a regional drift adaptation algorithm is developed to synchronize the density discrepancies based on the identified drifted instances. The set $\mathbf{D}_1^{dec}$, which returned by LDD-DIS, represents the data instances belonging to $\mathbf{D}_1$ and have decreased density compared to the set $\mathbf{D}_2$. Similarly, the set $\mathbf{D}_2^{inc}$ represents the data instances belonging to $\mathbf{D}_2$ and have increased density compared to data set $\mathbf{D}_1$. It is obvious that, if the size of $\mathbf{D}_1$ and $\mathbf{D}_2$ are the

Algorithm 2: Density Synchronized Drift Adaptation (LDD-DSDA)

Input: data instance arriving at each time step $d_0, \cdots, d_t$
minimum data batch size, w (default $w = 100$)
base learner, L (default $L =$ Naive Bayes Classifier)

Output: prediction results, $\hat{y}_0, \cdots, \hat{y}_t$

1. initial $\mathbf{D}_{train}$, $L = \text{buildLearner}(\mathbf{D}_{train})$
2. **while** stream not end, denote current time step as t
3. $\hat{y}_t = predict(L, d_t)$
4. **If** $|\mathbf{D}_{train}| < w$
5. $\mathbf{D}_{train} = \mathbf{D}_{train} \cup \{d_t\}$
6. **Else**
7. $\mathbf{D}_{buffer} = \mathbf{D}_{buffer} \cup \{d_t\}$
8. **If** $|\mathbf{D}_{train}| \geq w$ and $|\mathbf{D}_{buffer}| \geq w$
9. $\mathcal{D}_{drift} = \text{LDD-DIS}(\mathbf{D}_{train}, \mathbf{D}_{buffer})$, *where* $\mathcal{D}_{drift} = \{\mathbf{D}^{dec}_{train}, \mathbf{D}^{sta}_{train}, \mathbf{D}^{inc}_{train}, \mathbf{D}^{dec}_{buffer}, \mathbf{D}^{sta}_{buffer}, \mathbf{D}^{inc}_{buffer}\}$
10. merge the detected drift regions as per Equation (5)
11. based on the density of $\mathbf{D}_{buffer}$, resample the data as $\mathbf{D}_{train}$, as per Equation (6)
12. build new learner, $L = \text{buildLearner}(\mathbf{D}_{train})$
13. **Else**
14. $\text{updateLearner}(L, d_t)$
15. **End**
16. **return** $\hat{y}_0, \cdots, \hat{y}_t$

same, then the size of $\mathbf{D}_1^{dec}$ and $\mathbf{D}_2^{inc}$ will be the same, or is simply denoted as $|\mathbf{D}_1^{dec}|/|\mathbf{D}_1| = |\mathbf{D}_2^{inc}|/|\mathbf{D}_2|$. This property is the foundation of the proposed density synchronized drift adaptation (DSDA) in Algorithm 2.

The intuitive idea of LDD-DSDA is to merge existing data $\mathbf{D}_{train}$ with the recently buffered data $\mathbf{D}_{buffer}$ and resample a new batch of training data $\mathbf{D}_{train}$ so that the new $\mathbf{D}_{train}$ has the same distribution as $\mathbf{D}_{buffer}$. To achieve this goal, we present the following data merging rules: 1) if no density discrepancies are detected (which means $|\mathbf{D}_1^{dec}|/|\mathbf{D}_1| = |\mathbf{D}_2^{inc}|/|\mathbf{D}_2| = 0$) then return $\mathbf{D}_{train} \cup \mathbf{D}_{buffer}$ while if the unioned data set is considered redundant, return $\mathbf{D}_{buffer}$ instead; 2) if there are regional density drifts, merging the drifted data instances as per Equation 5.

$$\begin{cases} \mathbf{D}^{inc} = \mathbf{D}^{inc}_{train} \cup \mathbf{D}^{dec}_{buffer} \\ \mathbf{D}^{sta} = \mathbf{D}^{sta}_{train} \cup \mathbf{D}^{sta}_{buffer} \\ \mathbf{D}^{dec} = \mathbf{D}^{dec}_{train} \cup \mathbf{D}^{inc}_{buffer} \end{cases} \quad (5)$$

Then, to ensure the drifted regions have the same probability density as $\mathbf{D}_{buffer}$, a subset of instances in each drifted region will be resampled, denoted as $\mathbf{D}_{sample}$, shown as Equation 6.

$$|\mathbf{D}^{inc}_{sample}| : |\mathbf{D}^{sta}_{sample}| : |\mathbf{D}^{dec}_{sample}| = |\mathbf{D}^{inc}_{buffer}| : |\mathbf{D}^{sta}_{buffer}| : |\mathbf{D}^{dec}_{buffer}| \quad (6)$$

where $\mathbf{D}^{inc}_{sample}$ are sampled from $\mathbf{D}^{dec}$ in Equation 5. and so on. At last replace $\mathbf{D}_{train}$ by $\mathbf{D}_{sample}$, namely $\mathbf{D}_{train} = \mathbf{D}_{sample}$

In Algorithm 2, line 1 initializes the base learner L, and line 3 uses the base learner to predict data instances arriving at the current time, denoted as $predict(L, d_t)$. From lines 4 to 7, the system maintains the data for training and drift detection. When both training and buffered data sets reach the minium batch size, LDD-DIS is triggered, as shown in lines 8, 9. Then the detected drift regions are synchronized, based on Equations (5), (6). Otherwise, the system incrementally updates the base learner, denoted as $\text{updateLearner}(L, d_t)$, shown in line 14. Finally, in line 16, the predicted labels are returned for performance analysis.

5 Experiment and Evaluation

In this section, we describe three groups of experiments to evaluate the proposed LDD. The first group examines the normality of LDD under different circumstances. The second group evaluates the LDD-DIS algorithm. Lastly, we apply the LDD-DSDA algorithm to three real-world evolving data streams and compare the results with other methods. All experiments are conducted on a 2×3.1GHz 8 core CPU 128GB RAM cluster node with unique access.

5.1 Evaluation of the Validity of LDD

Experiment 1 The distribution of LDD

To verify the normality of LDD, two 1D data sets A, B of equal size (5,000,000) are generated. Given a neighborhood of randomly picked points in this domain, we randomly select 10,000 points from B, and observe the number of points located in 10,000 neighborhood of points in B. This process is repeated 10,000 times to approximately describe the distribution of δ. The histogram of 10,000 δ is shown in Figure 2.

According to Theorem 1, LDD is normally distributed with the parameters $(0, V(\delta))$. To compare the real and theoretical distribution of LDD, the line in Figure 2 represents the theoretical values of a normal distribution $N(0, V(\delta))$, where $V(\delta)$ stands for the variance of δ.

5.2 Evaluation of LDD-DIS

In this section, we evaluate the LDD-DIS algorithm in terms of two 2D synthetic drifted data sets. For each data set, 1,000 training cases are drawn from a specific distribution and 500 testing cases are drawn from a different distribution. We plot the estimated probability density curve of the selected feature in subfigure (a), the entire data set in subfigure (b) (blue dots represent the training set and red crosses represent the testing set), the detected drifted instances in subfigure (c) (blue dots represent decreasing density, red dots represent increasing density), and the value of LDD of the corresponding instances in subfigure (d) to demonstrate the relationship between LDD and regional density changes. The configurations of LDD are set as the default values described in Algorithm 1.

Experiment 2 Gaussian Data with Drifted Variance

This data set is generated from a bivariate Gaussian distribution. The training set has mean $\mu = [5, 5]$ and the

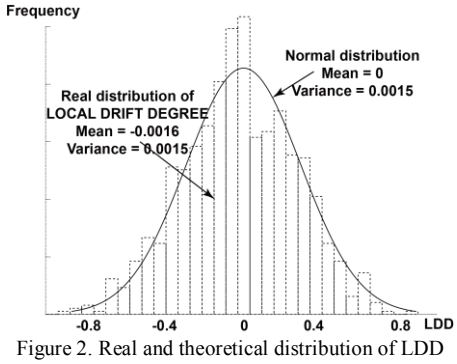

Figure 2. Real and theoretical distribution of LDD

covariance matrix is $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. The testing set has the same mean vector while its covariance matrix drifts to $\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$. This drift forms a ring-shaped drifted region as shown in Figure 3(c).

Since the covariance of x_1, x_2 is 0, the patterns of drifted regions are the same from only x_1 perspective. In Figure 3(a), we plot the kernel smoothing estimated density function of feature x_1 for both training set (red line) and testing set (blue line). If we consider the density drift from the training set perspective, when the blue line is higher than the red line, this means in that region, the training set density is decreasing, and vice versa. Accordingly, the instances located in these regions will be identified as $\mathbf{D}_{train}^{dec}$ or $\mathbf{D}_{train}^{inc}$, as shown in Figure 3(c), where the red dots belong to $\mathbf{D}_{train}^{inc}$, and the blue dots belong to $\mathbf{D}_{train}^{dec}$. The corresponding LDD value of each data instance is ploted in Figure 3(d). The higher the value of LDD, the higher the probability that this instance will drift.

Experiment 3 Mixture Distribution with Drifting Mean

To demonstrate how LDD works in a more general situation, we create a drifting Gaussian mixture data set and apply LDD to detect the drifted instances. The Gaussian mixture distribution consists of three bivariate Gaussian distributions. The three distributions for the testing set are generated based on the paramters given in Equation (7), while the testing set is generated based on the paramters given in Equation (8). The results of LDD-DIS are shown in Figure 4 with the same meaning as Figure 3.

$$\begin{cases} \mu_1 = [8,5], \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \mu_2 = [2,5], \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \\ \mu_3 = [5,5], \Sigma_3 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \end{cases} \quad (7) \qquad \begin{cases} \mu_1 = [10,5], \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \mu_2 = [4,5], \ \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \\ \mu_3 = [7,5], \ \Sigma_3 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \end{cases} \quad (8)$$

5.3 Evaluation of LDD-DSDA

In this section, we evaluate LDD-DSDA on three real-world evolving data streams. We compare its performance to several representatives of data stream classification paradigms. The selected comparison methods are: Hoeffding Adaptive Tree (HAT) [Bifet and Gavaldà, 2009], Accuracy Updated Ensemble (AUE) [Brzeziński and Stefanowski, 2011], SAMkNN [Losing, *et al.*, 2016], exponentially weighted moving average charts drift detection (ECDD) [Ross, *et al.*, 2012], and HDDM family algorithms [Frias-Blanco, *et al.*, 2015]. All these algorithms were implemented based on the MOA framework [Bifet, *et al.*, 2010], which is a commonly used software for evolving data stream analysis. To quantitatively evaluate LDD-DSDA, we consider the following performance metrics: accuracy (Acc.), precision (Pre.), recall (Rec.), f1-score (F1.) and computation time (Time), where Pre=TP/(TP+FP), Rec=TP/(TP+FN), and F1=2·Pre·Rec/(Pre+Rec). The term TP represents the number of true positive predictions, FN represents the number of false negative predictions. These metrics were evaluated in a prequential manner. To fairly compare these drift adaptation algorithms, the default parameters suggested by the authors are used, and the base classification model is set as Naïve Bayes classifier, except for SAMkNN, which is only designed for IBk classifier. The parameters for IBk classifier of SAMkNN are $k = 10$ and weighting method = uniformly weighted. Because LDD-DSDA involves a random sampling process, the results may vary at different runs, we run LDD-DSDA 50 times on each data set and state the mean and standard deviation of the results. To avoid confusion on the F1-score metric, we only take the mean of Pre. and Rec. to calculate the F1 for LDD-DSDA. The source code of LDD-DSDA is available at https://sites.google.com/view/anjin-concept-drift/home.

Experiment 4 Electricity Price Prediction Data Set (Elec)

Elec 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. We considered the classes as equally important, because both price up and down in the market is considered

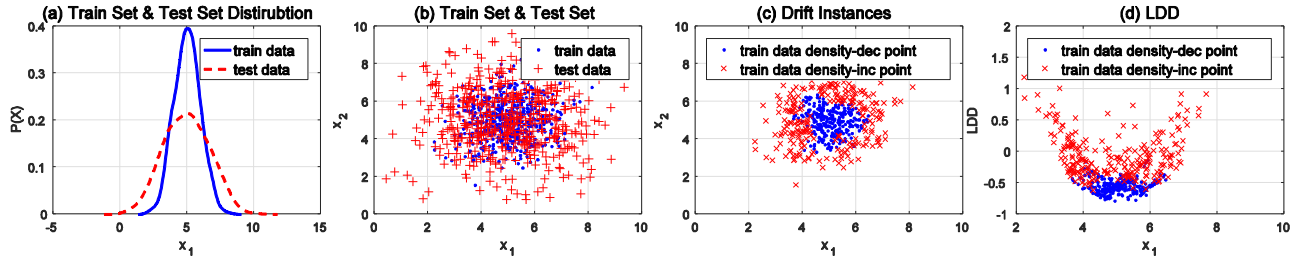

Figure 3 LDD-DIS on Gaussian Variance Drifted Distribution

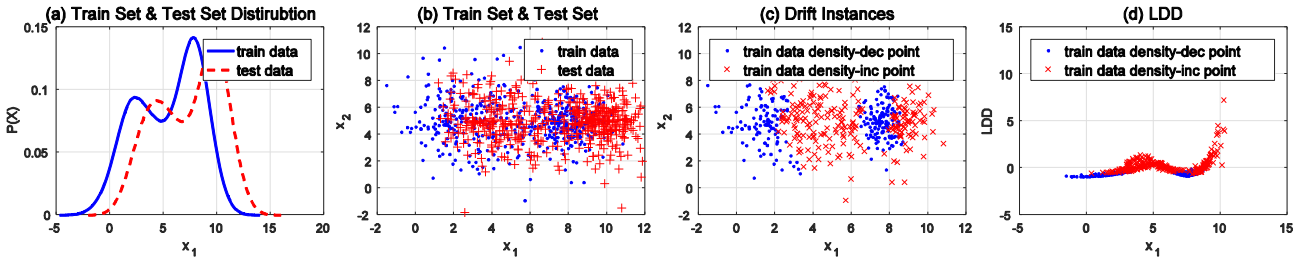

Figure 4 LDD-DIS on Mixture Mean Drifted Distribution

critical to users, and the ratio of classes is balanced, which is 58% down and 42% up. Therefore, we take the average precision (Pre.) and recall (Rec.) of both classes as the performance evaluation metrics, where average $\text{Pre} = (\text{Pre}_{class1} + \text{Pre}_{class2})/2$, average $\text{Rec} = (\text{Rec}_{class1} + \text{Rec}_{class2})/2$.

Experiment 5 Nebraska Weather Prediction Data Set (Weather)
This 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) class, and 69% negative (no-rain) class. This data set was summarized by [Elwell and Polikar, 2011]. In relation to the performance metrics Pre. Rec. and F1, only the positive class (rain) is considered. This is because a correct prediction of rain is considered more important than a correct prediction of no-rain.

Experiment 6 Spam Filtering Data Set (Spam Filtering)
This data set is a collection of 9,324 email messages derived from the Spam Assassin collection. 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 [Katakis, *et al.*, 2009]. According to [Katakis, *et al.*, 2009], 500 attributes were retrieved using the chi-square feature selection approach. The correct classification of legitimate emails is used to evaluate the algorithms [Katakis, *et al.*, 2009].

Experiment Summary
As shown in Table 1, LDD-DSDA achieve the best performance in most cases. Although the execution time of LDD is slightly longer than that of the other algorithms on high-dimensional data (experiment 6), the improvement is notable. LDD-DSDA addresses a critical problem in data stream mining from a novel perspective and achieved a competitive result compared to the state-of-the-art algorithms. LDD-DSDA has the best average rank (2.33) crossing the tested data sets. Compared to the second place, which is ECDD (2.67), LDD-DSDA improves the average F1 by 0.0114. According to the Friedman Test, the differences between LDD-DSDA and ECDD on Acc., Pre., Rec. and F1 are significant.

6 Conclusion and Further Study

In this paper, we analyzed existing concept drift adaptation algorithms and recognized the necessity of tracking regional drifts. Through investigating the distribution of data nearest-neighbors, we proposed a novel metric, called LDD, to detect regional concept drift. Accordingly, an LDD-based density synchronization algorithm was proposed to adapt the density discrepancies. The validity of LDD and the performance of LDD-based algorithms were thoroughly evaluated using six experiments.

Acknowledgments

This work is supported by the Australian Research Coun-cil (ARC) under discovery grant DP150101645. Also, the authors would like to thank the anonymous reviewers for their valuable feedback and all members of the CAI for discussion.

A Proof of Theorem 1

To demonstrate theorem1 we introduce a random variable I_i

$$I_i = \begin{cases} 1 & \mathbf{B}_i^d \epsilon B \\ 0 & otherwise \end{cases} \quad \text{(A.1)}$$

According to the sampling techniques, selecting n units from N, the probability that each unit will be selected in n draws is $C_{N-1}^{n-1}/C_N^n = n/N$ (C_N^n is the number of n-permutations of N) and the probability that two units will be selected in n draws is $n(n-1)/N(N-1)$. Under this condition, I_i satisfies the following equations:

$$E(I_i) = \frac{n}{N}, i = 1,2,\dots,N \quad \text{(A.2)}$$

By using I_i, $\bar{B}$ can be rewritten by Equation. (A.3) and its expectation equals $\overline{\mathbf{B}^d}$

$$\bar{B} = \frac{1}{n}\sum_{i=1}^{N} \mathbf{B}_i^d I_i \quad \text{(A.3)}$$

$$E(\bar{B}) = \frac{E(\sum_{i=1}^{N} \mathbf{B}_i^d I_i)}{n} = \frac{\sum_{i=1}^{N} \mathbf{B}_i^d E(I_i)}{n} = \frac{1}{n}\cdot\frac{n}{N}\sum_{i=1}^{N} \mathbf{B}_i^d = \frac{1}{N}\sum_{i=1}^{N} \mathbf{B}_i^d = \overline{\mathbf{B}^d} \quad \text{(A.4)}$$

Data Stream	#Insts	#Dim	#Class	Algorithms	Acc.	Pre.	Rec.	F1 (rank)	Time (ms)
Elec	45,312	8	2	LDD-DSDA	**0.8776±6.8E-5**	**0.8750±7.4E-5**	**0.8743±6.2E-5**	**0.8747** (1)	1776.4±243.5
				HAT	0.8131	0.8100	0.8057	0.8078 (5)	2071
				AUE	0.7544	0.7514	0.7408	0.7461 (7)	3005
				SAMkNN	0.7561	0.7506	0.74835	0.7495 (6)	305082
				ECDD	0.8676	0.8643	0.8650	0.8646 (2)	1005
				HDDM-A-Test	0.8492	0.8462	0.8446	0.8454 (3)	3063
				HDDM-W-Test	0.8409	0.8374	0.8367	0.8371 (4)	**1004**
Weather	18,159	8	2	LDD-DSDA	0.7256±4.7E-4	0.8070±4.2E-4	0.7876±7.2E-4	0.7972 (5)	**982.4±177.7**
				HAT	0.7248	0.8039	0.7922	0.7980 (4)	2069
				AUE	0.7243	0.7862	**0.8217**	0.8036 (2)	1006
				SAMkNN	**0.7486**	0.6778	0.3787	0.4859 (7)	121001
				ECDD	0.7279	0.7960	0.8183	**0.8070** (1)	1004
				HDDM-A-Test	0.7238	**0.8221**	0.7626	0.7912 (6)	1063
				HDDM-W-Test	0.7282	0.8018	0.8021	0.8019 (3)	1004
Spam Filtering	9,324	500	2	LDD-DSDA	**0.9412±4.3E-4**	**0.9503±5.1E-4**	0.9719±4.3E-4	**0.9610** (1)	9942.3±746.3
				HAT	0.8893	0.9430	0.9060	0.9241 (6)	5067
				AUE	0.8406	0.9246	0.8556	0.8888 (7)	7012
				SAMkNN	0.9247	0.9227	**0.9809**	0.9509 (2)	255923
				ECDD	0.8884	0.9012	0.9546	0.9271 (5)	2005
				HDDM-A-Test	0.9079	0.9341	0.9426	0.9383 (4)	**2003**
				HDDM-W-Test	0.9165	0.9293	0.9608	0.9448 (3)	**2003**

Table 1. Performance of Different Data Stream Classification Algorithms on Real-world Data Sets

References

[Bifet and Gavaldà, 2009] Albert Bifet and Ricard Gavaldà, "Adaptive learning from evolving data streams," in *Proceedings of the Eighth International Symposium on Intelligent Data Analysis*, pp. 249-260: Springer, 2009.

[Bifet, *et al.*, 2010] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer, "MOA: Massive online analysis," *The Journal of Machine Learning Research,* vol. 99, pp. 1601–1604, 2010.

[Brzeziński and Stefanowski, 2011] Dariusz Brzeziński and Jerzy Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *International Conference on Hybrid Artificial Intelligence Systems*, pp. 155-163: Springer, 2011.

[Elwell and Polikar, 2011] Ryan Elwell and Robi Polikar, "Incremental learning of concept drift in nonstationary environments," (in eng), *IEEE Transactions on Neural Networks,* Research Support, U.S. Gov't, Non-P.H.S. vol. 22, no. 10, pp. 1517-31, Oct 2011.

[Frias-Blanco, *et al.*, 2015] Isvani Frias-Blanco, Jose del Campo-Avila, Gonzalo Ramos-Jimenes, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yaile Caballero-Mota, "Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds," *IEEE Transactions on Knowledge and Data Engineering,* vol. 27, no. 3, pp. 810-823, 2015.

[Gama and Castillo, 2006] João Gama and Gladys Castillo, "Learning with local drift detection," in *International Conference on Advanced Data Mining and Applications*, pp. 42-55: Springer, 2006.

[Gama, *et al.*, 2014]João Gama, Indrė Zliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys,* vol. 46, no. 4, pp. 1-37, 2014.

[Harel, *et al.*, 2014] Maayan Harel, Shie Mannor, Ran El-Yaniv, and Koby Crammer, "Concept drift detection through resampling," in *Proceedings of the Thirty-first International Conference on Machine Learning*, pp. 1009-1017, 2014.

[Ikonomovska and Gama, 2008] Elena Ikonomovska and Joao Gama, "Learning Model Trees from Data Streams," in *Proceedings of the Eleventh International Conference on Discovery Science*, pp. 52-63, Berlin: Springer, 2008.

[Ikonomovska, *et al.*, 2011]Elena Ikonomovska, João Gama, and Sašo Džeroski, "Learning model trees from evolving data streams," *Data Mining and Knowledge Discovery,* vol. 23, no. 1, pp. 128-168, 2011.

[Ikonomovska, *et al.*, 2009]Elena Ikonomovska, João Gama, Raquel Sebastião, and Dejan Gjorgjevik, "Regression Trees from Data Streams with Drift Detection," in *Proceedings of the Twelfth International Conference on Discovery Science*, pp. 121-135, Berlin: Springer, 2009.

[Katakis, *et al.*, 2009] Ioannis Katakis, Grigorios Tsoumakas, Evangelos Banos, Nick Bassiliades, and Ioannis Vlahavas, "An adaptive personalized news dissemination system," *Journal of Intelligent Information Systems,* vol. 32, no. 2, pp. 191-212, 2009.

[Li, *et al.*, 2016] Peipei Li, Lu He, Xuegang Hu, Yuhong Zhang, Lei Li, and Xindong Wu, "Concept based short text stream classification with topic drifting detection," in *Proceedings of the Sixteenth International Conference on Data Mining*, pp. 1009-1014, 2016.

[Liu, *et al.*, 2017]Anjin Liu, Guangquan Zhang, and Jie Lu, "Fuzzy time windowing for gradual concept drift adaptation," in *Proceedings of the Twenty-sixth IEEE International Conference on Fuzzy Systems*, Naples Italy: IEEE, 2017.

[Losing, *et al.*, 2016] Viktor Losing, Barbara Hammer, and Heiko Wersing, "KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift," in *Proceedings of the Sixteenth IEEE International Conference on Data Mining*, 2016.

[Lu, *et al.*, 2016] Ning Lu, Jie Lu, Guangquan Zhang, and Ramon Lopez De Mantaras, "A concept drift-tolerant case-base editing technique," *Artificial Intelligence,* vol. 230, pp. 108-133, 2016.

[Lu, *et al.*, 2014] Ning Lu, Guangquan Zhang, and Jie Lu, "Concept drift detection via competence models," *Artificial Intelligence,* vol. 209, pp. 11-28, 2014.

[Ross, *et al.*, 2012] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognition Letters,* vol. 33, no. 2, pp. 191-198, 2012.

[Sarnelle, *et al.*, 2015] Joseph Sarnelle, Anthony Sanchez, RRobert Capo, Joshua Haas, and Robi Polikar, "Quantifying the limited and gradual concept drift assumption," in *The Proceedings of 2015 International Joint Conference on Neural Networks*, pp. 1-8, 2015.

[Shao, *et al.*, 2014] Junming Shao, Zahra Ahmadi, and Stefan Kramer, "Prototype-based learning on concept-drifting data streams," in *Proceedings of the Twentieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, USA, pp. 412-421, 2623609: ACM, 2014.

[Su, *et al.*, 2008] Bai Su, Yi-Dong Shen, and Wei Xu, "Modeling concept drift from the perspective of classifiers," in *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, Chengdu, China, pp. 1055-1060: IEEE, 2008.

[Sugiyama and Kawanabe, 2012] Masashi Sugiyama and Motoaki Kawanabe, *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT Press, 2012.

[Zliobaite, *et al.*, 2014] Indre Zliobaite, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes, "Active Learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 25, no. 1, pp. 27-39, 2014.