Concept Drift Detection Based on Equal Density Estimation

Feng Gu[1], Guangquan Zhang[1], Jie Lu[1], Chin-Teng Lin[2]
[1]Centre for Quantum Computation & Intelligent Systems
[1,2]Faculty of Engineering and Information Technology
[1,2]University of Technology Sydney
[1]Email: feng.gu@student.uts.edu.au, guangquan.zhang@uts.edu.au, jie.lu@uts.edu.au

***Abstract*—An important problem that remains in online data mining systems is how to accurately and efficiently detect changes in the underlying distribution of large data streams. The challenge for change detection methods is to maximise the accumulative effect of changing regions with unknown distribution, while at the same time providing sufficient information to describe the nature of the changes. In this paper, we propose a novel change detection method based on the estimation of equal density regions, with the aim of overcoming the issues of instability and inefficiency that underlie methods of predefined space partitioning schemes. Our method is general, nonparametric and requires no prior knowledge of the data distribution. A series of experiments demonstrate that our method effectively detects concept drift in single dimension as well as high dimension data, and is also able to explain the change by locating the data points that contribute most to the change. The detection result is guaranteed by statistical tests.**

I. Introduction

Data stream mining has received increasing attention as a result of the rapid growth of online applications such as sensor networks, telecommunication, mobile applications and cloud computing. Data generated by these systems, for example, network logs and electronic transaction flows, come in large volume and evolve over time, which means not only that data is generated at a very fast speed, but that the underlying distribution of the data is also constantly changing [1], [2]. This is known as the concept drift problem and it is considered to be one of the major challenges to online data mining and machine learning systems.

Formally, we denote stream data as an infinite sequence of (x, y, t), where x is the feature vector, y is the class label and t is time stamp. Concept drift is defined as the distribution (with possibility density function ϕ) of $p(x, y)$ having changed between the newly arrived data and the existing data at time T. Further, the change of $p(x, y)$ may have two sources: the change of $p(x)$ and/or the change of $p(y|x)$ [3]–[5].

$$x(t) \sim \begin{cases} \phi_0, & t < T \\ \phi_1, & t \geq T \end{cases}$$

Concept drift can be categorised into different types which present different challenges to concept drift handling algorithms. For example, based on the extent of the drift, concept drift can be defined as slow drift, moderate drift and slow or gradual drift [6]. Based on the scope of the drift, Tsymbal et al. [7] distinguished local drift from global drift and discussed the scenario in which data distribution only changes in a subregion of the data space.

To tackle the concept drift problem, existing methods typically consist of two steps: drift detection and model maintenance. In this article, our main effort is focused on developing a new detection method, although we will also discuss how our detection result has the potential to be used for model maintenance.

According to the literature, many algorithms for concept drift detection have been developed using a variety of strategies. One popular direction is to monitor the changes of certain statistics calculated from the data, such as cumulative sum [8] or classifier error rate [9], [10]. Another main branch of research aims to detect changes in the data distribution. Methods in this family, such as [11], [12] and [13], often involve two time windows, on which statistical tests are performed to check whether they have same distribution. One advantage of this windowing scheme is that it is especially suitable for detecting gradual drift, which many other methods may find challenging. The kdq-tree method [14] and competence model method [15] are two such state-of-the-art algorithms for detecting distribution changes. Both methods also share another advantageous philosophy, which is to examine the local density imbalance of two windows on subregions and then detect drift by computing a summarisation of all the local density inequalities. This approach by nature handles local drift and the detection result can be easily used to update training model in the maintenance step.

However, such methods rely heavily on space partitioning, each of them uses different partitioning schemes and often find it difficult to fit all possible distributions in the real world. For example, Dasu et el. [14] use kdq-tree to directly partition space, which may create inappropriate divisions in certain subregions, thus affecting the overall accuracy. Similarly, Lu et el. [15] use overlapping circles to construct a so-called competence modal, but the circles of a predefined radius may not always represent the shape of all regions, especially when the density of different regions varies extensively. Also, it is very costly to compute the competence model because of overlapping.

Motivated by these issues, instead of calculating local density inequality with a predefined partitioning scheme, we propose a novel distribution drift detection method based on estimating equal density regions on two data sets. Compared with other distribution detection methods, our method has the following advantages: 1) no parameter is needed for space partitioning, which means it is not biased to certain distributions; 2) it can be easily adopted for high-dimensional data without a drastic increase in the computational cost; 3) it is able to describe the drift by locating those data points that contribute most to the drift, which could be a very useful input for model maintenance.

Our main contributions are:

- A local density equality estimation technique without a predefined space partitioning scheme.
- A non-parametric generic detection method with statistical guarantee, which scales well in multi-dimensional data.
- An approach for identifying the data points that have the most significant impact on the drift.

This paper is organised as follows. Section 2 introduces the new local density equality estimation model and discusses its relationship with traditional space partitioning technique. Section 3 presents the concept drift detection method based on the local density equality estimation model, combined with the windowing scheme and statistical tests. Section 4 outlines the results of our experiments, with an analysis of the detection result, and illustrates how the result can be used to explain the nature of the drift. Lastly, Section 5 concludes our study with a discussion of the future work.

II. Equal density estimation

One popular way to check whether two data sets have the same distribution is to partition the data into subregions and then examine each subregion to see whether the density of two data sets in this region are equal, often by comparing the number of data points from each data set. Algorithms following this approach mainly distinguish themselves by the novelty of their partition strategies. Key to such methods is to find appropriate regions whose density is as uniform as possible and points inside these regions are evenly distributed. Different partition strategies aim to find such boundaries so that the domination effect of high density regions on low density regions is minimized. The accuracy of such partition methods inevitably reliant on - or limited by - the strategy, or may even be biased to certain distributions. For example, Dasu et el. [14] uses kdq-tree to split hyperspace into small hyper-squares. Such rigid splitting may cut up an region of higher density and group it with a lower density region. The competence model of Lu et el. [15] requires an circle radius parameter to be empirically chosen for different data distribution, and it is difficult to prove that it is the optimal value.

To overcome the problems that raise from fixed partitioning strategy, we instead try to find regions where the two data sets will have the same density. Clearly the smallest region in which two data sets have the same density is marked by the two points from each data set that are closest to each other. Figure 1 shows the differences between fixed regions and regions of equal density. If data points are generated from different distribution, represented by triangles and dots, they appear in different number in fixed regions, but remain the same in equal density regions.

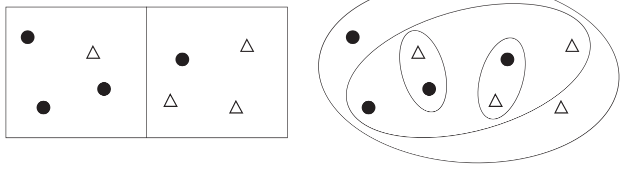


Fig. 1. Partitioning by fixed regions vs. equal density regions.

To describe the property of equal density regions, we give the following definitions.

Definition 1. *Let a space $S_n = \{a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n\}$, where $a \in A$ and $b \in B$, A, B are two data sets of size n, CrossScale denoted as $Cs(S_n) = \max(D(a, b))$, D is distance function.*

Given a certain distance function and a space consisted of two sets of equal size n, CrossScale is determined by the two points, each from one set and are farthest away from each other. Although, in practice, we need not actually compute CrossScale of all the regions. CrossScale will help us better understand following definitions.

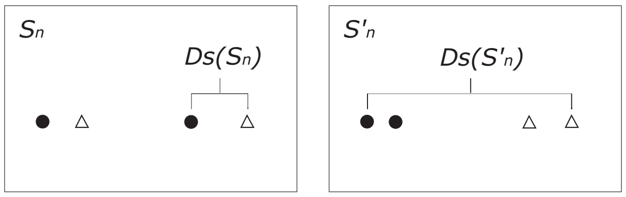
Definition 2. *Given a space S_n, for any subspace S_k, $k \leq n$, MinimumRegionScale denoted as $Mrs(S_n, k) = \min(Cs(S_k))$ and MinimumRegion denoted as $Mr(S_n, k)$ is the corresponding region.*

If $k < n$, space S_n will contain more than one subspaces ($\frac{n!}{(n-m)!\,m!}$ to be specific), $MinimumRegion(S_n, k)$ will be the subspace with minimum CrossScale.

Specially, when $k = n - 1$, any space S_n can be divided into two parts, $Mr(S_n, n-1)$ and a single pair of points that is left out. To examine $DensityScale$ on each level k, we give the following recursive definition.

Definition 3. *DensityScale denoted as $Ds(S_n) = \max(Ds(Mr(S_n, n - 1)), D(a, b))$, where $a, b \in S_n$ and $a, b \notin Mr(S_n, n-1)$. If $n = 1$, $Ds(Mr(S_n, 0) = 0$.*

We can see that if two data sets have different distributions, the data points from each set are imbalanced in subregions. Additional points will be included in $MinimumRegion$ of larger k and will have larger $CrossScale$, which results in larger $DensityScale$ of the space. Figure 2 is a simplified demonstration of how $DensityScale$ increases if the two data sets have different distribution.



(a) same distribution (b) different distribution

Fig. 2. $DensityScale$ increases when two data sets have different distribution.

III. Change detection method

From Section II, we see that an increase of $DensityScale$ can indicate local density difference between two data sets. To detect the overall concept drift, we need to measure this increase on each subregion of the data. Note that these subregions are allowed to overlap with others to reflect all possible density levels. This is the root reason why our method is non-parametric and is not biased to certain distribution.

To find all the $DensityScale$ of each level k of a subregion, we repeatedly search nearest pairs of points from two data sets, and record the distance D between the two points of each pair as $CrossScale$. After a pair is found, we remove it from the data so that each point can only be used once to guarantee that the density estimation of the $MinimumRegion$, which is marked by the pair of two points, remains the same between the two data sets. We call this removal process *Pairwise Reduction*.

To measure whether the $DensityScale$ of all the subregions of space S_n have a trend of increasing, we compute the integral of $DensityScale$, denoted as ω.

$$\omega = \int_0^n Ds(Mr(S_n, x))dx$$

In practice, since sample x is discrete, the integral of $DensityScale$ can be approximated by using the composite trapezoidal rule.

A. Statistical guarantee

The next step is to determine whether the observed increase of ω is significant enough to represent a concept drift with statistical test. In this work, we resort to the two-sample non-parametric permutation test method [16]. The permutation test makes no assumption of data distribution and is suitable when the theoretical distribution of the test statistic is complicated or unknown.

To test whether data have different distributions, the null hypothesis is that two distributions D_1 and D_2 are identical $H_0 : D_1 = D_2$.

Once the observation is made, in this case, the computed ω', the significance level or P-value is $P = Pr_{H_0}(\omega^* \geq \omega')$ where ω^* is measured ω under null hypothesis.

Given a predefined α, we can compute the critical region $(\hat{\omega}, \infty)$ through permutation test so that the possibility of ω^* lying in this range is equal to α. If our observed $\omega' > \hat{\omega}$, we reject H_0 and raise an alert of concept drift.

B. Window models

Our method is to detect the distribution difference of two windows. Formally, given a stream of objects $x_1, x_2, \ldots,$ we denote the sequence of points as $window\ W_t = (x_{t-n+1}, \ldots, x_t)$, where n is the window size. ω is computed between distributions of points from two windows W_t and $W_{t'}$.

Different window models can be used to detect different types of drift, such as sudden drift or gradual drift [6]. Here we consider two window models. The *adjacent sliding windows* model uses two windows W_t and W_{t-n}; the *fix-slide windows* model [17] uses a fixed window W_k and a sliding window W_t, and only updates W_k after detecting a drift, thus it is more suitable for detecting gradual drift.

Window size can also be adjusted to detect changes at different scales. A common practice is to choose window sizes that increase exponentially $(n, 2n, 4n, etc)$. Further, calculations on each window size can be processed in parallel for efficiency.

For the purpose of validating our method, we will use the *fix-slide windows* model with fixed window size n and implement it in single-threaded program.

IV. Empirical Evaluation

To evaluate the effectiveness of the $DensityScale$ and our proposed change detection method, we conduct a series of experiments with various types of data sets and/or different parameter settings. Our evaluation consists of three parts:

We first evaluate the $DensityScale$ and demonstrate how the change of the integral of $DensityScale$ ω can reflect the underlying difference of data distribution.

Then we evaluate our change detection method by applying it to various types of generated data sources, for which we can easily control the change of distribution and test the performance of our detection method in different scenarios. In addition, we compare our detection method to [14] and [15] on both accuracy and efficiency.

Lastly, we analyze the detection result and demonstrate that our detection method is not only able to detect a change but is also suitable for describing the changing area.

A. Evaluating ω

We rely on the integral of $DensityScale$ ω to represent the difference in underlying distribution of the two data sets. We first conduct several tests to see how ω varies as different changes in the data occur.

In these tests, we use data streams that are generated according to 2D normal distributions. Each data stream contains a total of 100,000 points divided into 10 groups. The data distribution within one group remains unchanged, yet different groups differ from each other on one of the three parameters, namely mean μ, standard deviation σ and correlation ρ. Window size is set to 5,000.

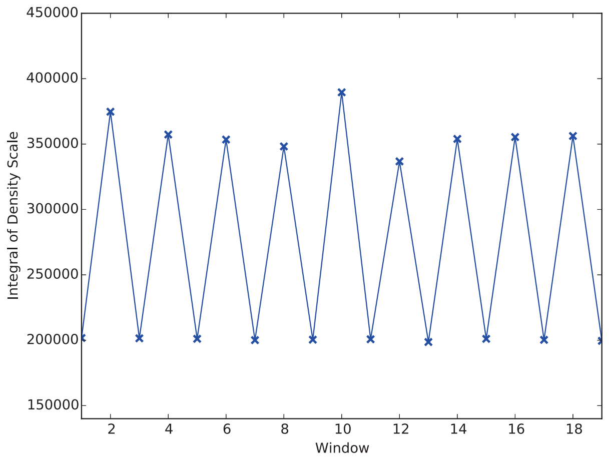


Fig. 3. ω varies as the mean μ of normally distributed data changes.

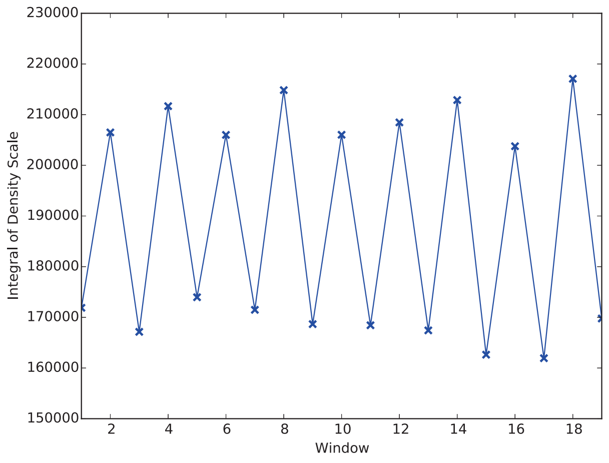


Fig. 5. ω varies as the correlation ρ of normally distributed data changes.

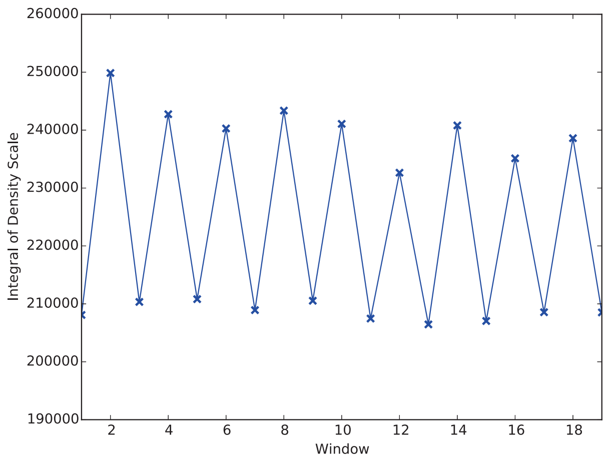


Fig. 4. ω varies as the standard deviation σ of normally distributed data changes.

1) Test 1: Varying μ: In the first test, we fix the standard deviation $\sigma = 0.2$ and the correlation $\rho = 0$; we vary the mean $\mu_i = 0.2 + 0.06(i - 1)$ for the ith group, $i = 1, ..., 10$, which means that the center of the normally distributed data moves gradually from $(0.2, 0.2)$ to $(0.8, 0.8)$. In Figure 3, we see that ω increases as the two windows slide across the drifting point.

2) Test 2: Varying σ: Next, we fix the mean $\mu = 0.5$ and the correlation $\rho = 0$ while varying the standard deviation $\sigma_i = 0.1 + 0.02(i - 1)$ for the ith group, $i = 1, ..., 10$. In Figure 4, again we see that ω increases as σ changes.

3) Test 3: Varying ρ: For the third data stream, we fix the mean $\mu = 0.5$ and the standard deviation $\sigma = 0.2$ while the correlation is set to $\rho_i = 0.08(i - 1)$ for the ith group, $i = 1, ..., 10$. This means that the two features of the data are independent of each other initially and then become more and more positively correlated. In Figure 5, the change is more subtle, but we can still clearly see the increase of ω.

B. Evaluating the change detection method

The experiments above demonstrate how ω fluctuates as the underlying data distribution changes. In the following experiments, we plugin the permutation test described in section III-A to determine whether a measured increase of ω is statistically significant enough to indicate a concept drift. When the observed significance level P drops below a given threshold α, we say that concept drift is occurring and an alert will be raised.

We compare our detection method PR with the two method KL [14] and CM [15]. They are appropriate comparisons for several reasons. Firstly, these methods do not require a specific classifier, which means that they have a boarder application scope than other detection methods based on error rates or decision trees. Secondly, we all resort to a similar approach which measures differences in subregions and then performs a hypothesis test on the calculated summation. Lastly, since [15] is proved to have higher accuracy, but [14] runs faster, we can evaluate our method by comparing with them on both accuracy and performance.

The same experimental environments are set up for the three methods to guarantee a fair comparison. For the data source, we generate artificial data sets according to the same procedure described in [14]. Detailed information of each data source is described in their experiment setup sections. For the window strategy, we use the *fix-sliding* window model mentioned in section III-B, that is, moving the sliding window while keeping the reference window fixed and only updating it when the next drift is detected. For the parameters, unless indicated otherwise, the same window size 10,000 is used for all three methods and the significant level α is set to 1%. Since both [14] and [15] require additional parameters to be used to define the subregions, their values are shown in Table I.

TABLE I
ADDITIONAL PARAMETERS OF COMPARISON METHODS USED IN THE EXPERIMENTS.

Method	Parameter	Symbol	Value
KL [14]	Minimum side length of a cell	δ	2^{-10}
	Maximum number of points in a cell	τ	100
CM [15]	Euclidean distance threshold	d_ε	0.05

We measure the accuracy of each method by the following rules. If the detection alert is raised when the sliding window is moving across the drifting point, we call it $Detected$; if the alert is late as the sliding window moves past the drifting point, but is raised before the next drift, it is $Late$; $Missed$ alert means that the alert is not raised at all and the next drift is reached; lastly, if the previous drift has already been detected but another alert is raised before the next drift comes, it is called a $False$ detection.

1) Test 4: Different types of concept drift: In this experiment, we use three artificial data sets described in [14]. The data streams are all generated according to 2D normal distribution with mean $\mu = 0.5$, standard deviation $\sigma = 0.2$ and correlation $\rho = 0.5$. Different parameters will be used to demonstrate different types of drift. The first group of streams M varies the mean μ in the range of $[0.2, 0.8]$ with step size $\Delta = 0.05, 0.02$. The second group of streams D varies the standard deviation σ in the range of $[0, 0.4]$ with step size $\Delta = 0.02, 0.01$. The third group of streams C varies the correlation between the two features ρ in the range of $[-1, 1]$ with step size $\Delta = 0.2, 0.1$. Each data stream consists of 4,000,000 points divided into 100 groups, which make up a total of 99 drifts.

The result is shown in Table II. We can see that our method out-performs both [15] and [14]. For data M(0.05) and D(0.02), the results of the three methods are similarly high since the scale of change is large and is easy to detect. However, the accuracy of our method is significantly higher than the other two methods for the other data stream. This shows that our method is more sensitive to smaller changes.

2) Test 5: Window size: In this experiment, we test the performance of our method with a smaller window size. All the environmental settings are the same as in the previous experiment except that the window size is reduced to 5,000. From the result shown in Table III, we can see that as expected, the detection rate of all three methods decreases, but our method is still able to maintain relatively higher accuracy. Also, it is notable that for data M(0.05) the accuracy of our method has remained almost the same, and for data M(0.02), D(0.01) and C(0.01), the accuracy of both comparison methods has dropped by 50% or more, whereas that of our method has dropped by about 40%. This shows that our method is more robust for small windows than the other two methods.

3) Test 6: High dimension: In the next experiment, we increase the dimensions of the data stream to test the scalability of our method. We choose the same setting as the previous normal data stream C(0.2) and add more dimensions to it. The distribution of all the added dimensions remains unchanged, so that we can test how our method performs in detecting changes in sub-spaces. For CM [15] especially, empirical setting of Euclidean distance threshold d_ε is required for each data stream to achieve a better result. We will use the same setting proposed in [15], ie, 0.15 for 4-dimensional, 0.3 for 6-dimensional and 0.5 for 10-dimensional.

The results are presented in Table IV, which shows that our method still out-performs the other two comparison methods and particularly shows its strength when the change becomes subtle as more stationary dimensions are added.

4) Test 7: Poisson distribution: In this experiment, we test our detection method on data of Poisson distribution, which is another important distribution commonly seen in the real world. The data streams are generated using Trivariate Reduction [18] according to $(X, Y) \sim Poisson(500(1-\rho), 500(1-\rho), 500\rho)$, where ρ starts at 0.5 and then performs a random walk between 0 and 1 with step size $\Delta = 0.2, 0.1$.

TABLE II
DRIFT DETECTION RESULT ON 2D NORMAL DISTRIBUTION DATA WITH WINDOW SIZE $10{,}000$.

Data	Method	Detected	Late	False	Missed
M(0.05)	PR	99	0	6	0
	CM	99	0	6	0
	KL	99	0	6	0
M(0.02)	PR	97	2	3	0
	CM	66	24	3	9
	KL	48	27	5	24
D(0.02)	PR	99	0	11	0
	CM	99	0	8	0
	KL	98	1	5	0
D(0.01)	PR	50	18	3	31
	CM	34	24	6	41
	KL	30	23	2	46
C(0.2)	PR	99	0	6	0
	CM	96	3	7	0
	KL	94	5	6	0
C(0.1)	PR	77	15	11	7
	CM	67	13	8	19
	KL	61	12	4	26

TABLE III
DRIFT DETECTION RESULT ON 2D NORMAL DISTRIBUTION DATA WITH WINDOW SIZE $5{,}000$.

Data	Method	Detected	Late	False	Missed
M(0.05)	PR	99	0	7	0
	CM	96	3	4	0
	KL	95	4	2	0
M(0.02)	PR	72	24	6	3
	CM	35	30	3	34
	KL	25	19	2	55
D(0.02)	PR	94	3	3	2
	CM	88	3	4	8
	KL	83	9	8	7
D(0.01)	PR	24	22	4	53
	CM	12	12	1	75
	KL	12	18	1	69
C(0.2)	PR	90	9	6	0
	CM	67	20	6	12
	KL	82	14	2	3
C(0.1)	PR	43	22	4	34
	CM	30	24	4	45
	KL	31	18	4	50

TABLE IV
DRIFT DETECTION RESULT ON HIGH-DIMENSIONAL NORMAL DISTRIBUTION DATA WITH WINDOW SIZE 10,000.

Data	Method	Detected	Late	False	Missed
D(4)	PR	98	1	5	0
	CM	93	0	4	6
	KL	89	1	7	9
D(6)	PR	95	2	4	2
	CM	91	5	4	3
	KL	84	10	8	5
D(10)	PR	88	4	4	7
	CM	75	5	5	19
	KL	65	12	6	22

TABLE V
DRIFT DETECTION RESULT ON 2D POISSON DISTRIBUTION WITH WINDOW SIZE 10000.

Data	Method	Detected	Late	False	Missed
P(0.2)	PR	99	0	7	0
	CM	99	0	7	0
	KL	98	1	2	0
P(0.1)	PR	91	6	11	2
	CM	87	4	7	8
	KL	76	6	9	17

Again, d_ε for the competence model is set to 10 according to [15].

As the results in Table V show, all three method perform well on the Poisson data set, but our method still has the best performance.

5) Test 8: Efficiency: The following tests aim to measure the speed of our method using data of different dimensions and different window size. Similar to the methods of CM [15] and KL [14], our method consists of three steps: initial construction of space partition, ie, competence model for CM [15] and kdq-tree for KL [14]; incremental update on the arrival of new data; calculation of test statistics to determine the critical region. The test statistics can be calculated directly using standard algorithms such as bootstrap or permutation test, thus we only focus on comparing our method in the first two steps, namely the construction and incremental update. The test programs are implemented in Python 2.7 and run on a PC with a 2.3 GHz Intel i5 processor and 8 GB memory.

Note that our method uses quad tree or k-d tree [19] to searching the nearest neighbor, which can be done in $O(\log n)$ time.

As shown in Table VI, the time cost of our method is slightly affected by the increase of dimensions, but will increase with larger window size, similar to CM [15]. The temporal complexity of our method is shown in Table VII.

6) Test 9: Finding the drifting region: This experiment demonstrates how those data points that contribute most to the change are located during the *pairwise reduction* process. The experiment setting is similar to that of section IV-A1, with two data streams both generated from normal distribution. The mean of the first data stream remains unchanged, whereas the mean of the second stream varies slightly. We compare the result of the *pairwise reduction* process on the two data streams of different k values. In Figure 6, we see that data points from each data set are always mixed together during *pairwise reduction* when there is no drift in data distribution. By comparison, we see in Figure 7, when there is a drift, data points that cause the drift are clearly separated from each other after several iterations of k, showing the trend and nature of the change. This output of the *pairwise reduction* process can be further used in the maintenance process for updating training models.

TABLE VI
RUNNING TIME WITH DIFFERENT DIMENSIONS AND WINDOW SIZES.

Dimension(d)	Window size(n)	Method	Construct(s)	Update(s)
2	5000	PR	0.254	0.019
		CM	7.582	0.027
		KL	1.680	0.029
4	5000	PR	0.295	0.021
		CM	7.637	0.028
		KL	2.125	0.039
10	5000	PR	0.318	0.031
		CM	8.599	0.044
		KL	3.387	0.062
10	10000	PR	0.691	0.094
		CM	25.85	0.108
		KL	3.598	0.178
10	50000	PR	17.27	0.410
		CM	231.1	0.583
		KL	4.210	0.366

TABLE VII
TEMPORAL COMPLEXITIES OF MODEL CONSTRUCTION AND INCREMENTAL UPDATING.

	PR	CM	KL
Construction	$O(n\log^2 n)$	$O(n^2)$	$O(nd\log\frac{1}{\delta})$
Updating	$O(\log^2 n)$	$O(n)$	$O(d\log\frac{1}{\delta})$

V. CONCLUSIONS AND FURTHER STUDIES

In this paper, we have proposed a nonparametric distributional change detection method based on the estimation of equal density regions, which aims to overcome the issues of instability and inefficiency that underlie predefined partitioning methods. Experiments have shown that this approach works accurately and efficiently in both one dimension and higher dimensions. The output of the method describing the change is also easily understood and could be used in model maintenance.

One of our future studies would aim to develop model maintenance methods designed specifically for equal density region estimation in order to improve its application scale and significance. Another interesting direction would be to conduct further investigations of mathematical distributions of $DensityScale$ and improve the performance of the statistical testing process. A range of different distance algorithms could also be adopted and evaluated for further improvement of detection accuracy.

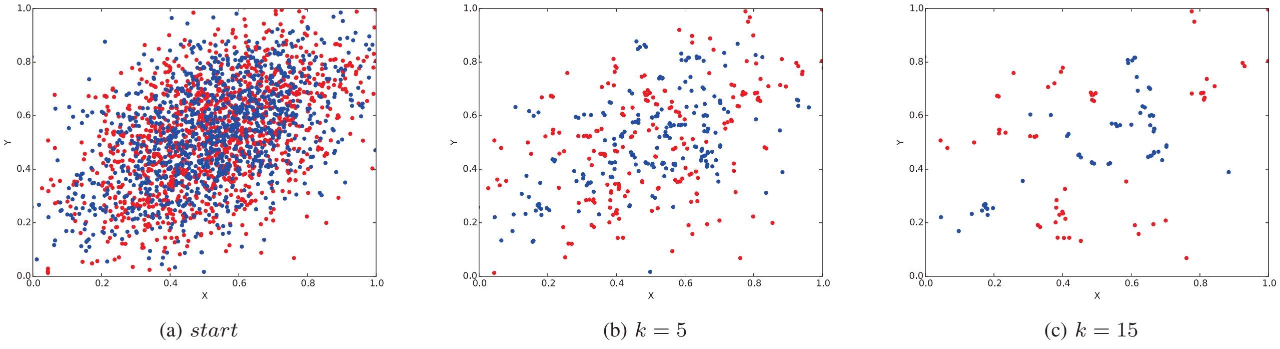

(a) *start* (b) $k = 5$ (c) $k = 15$

Fig. 6. Pairwise reduction on two normally distributed data sets of the same mean.

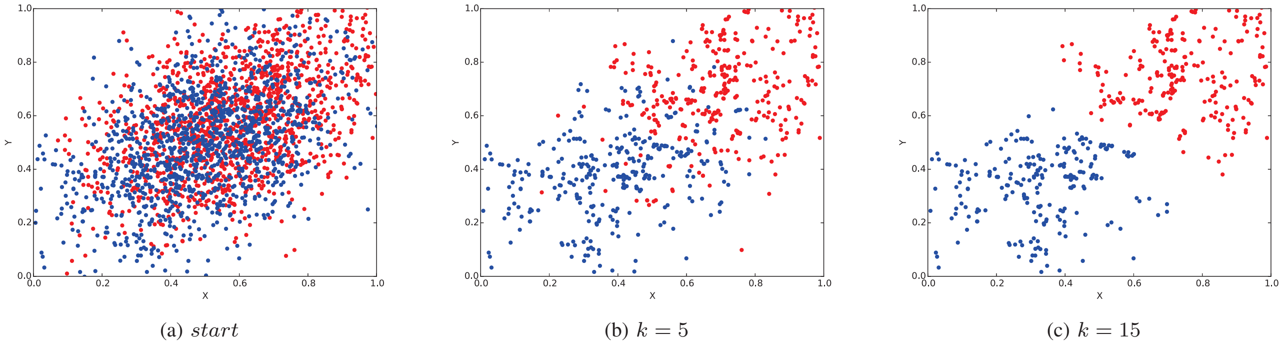

(a) *start* (b) $k = 5$ (c) $k = 15$

Fig. 7. Pairwise reduction on two normally distributed data sets of different mean.

Acknowledgment

The authors would like to thank Ning Lu, Anjin Liu and Fan Dong for their comments and suggestions on this study.

References

[1] J. C. Schlimmer and R. H. Granger Jr, "Incremental learning from noisy data," *Machine Learning*, vol. 1, no. 3, pp. 317–354, 1986.

[2] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.

[3] P. Zhang, X. Zhu, and Y. Shi, "Categorizing and mining concept drifting data streams," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 812–820.

[4] C. Alippi, G. Boracchi, and M. Roveri, "Hierarchical change-detection tests," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2016.

[5] G. J. Ross, D. K. Tasoulis, and N. M. Adams, "Nonparametric monitoring of data streams for changes in location and scale," *Technometrics*, vol. 53, no. 4, pp. 379–389, 2011.

[6] K. O. Stanley, "Learning concept drift with a committee of decision trees," *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*, 2003.

[7] A. Tsymbal, M. Pechenizkiy, P. Cunningham, and S. Puuronen, "Dynamic integration of classifiers for handling concept drift," *Information Fusion*, vol. 9, no. 1, pp. 56–68, 2008.

[8] E. Page, "Continuous inspection schemes," *Biometrika*, pp. 100–115, 1954.

[9] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence–SBIA 2004*. Springer, 2004, pp. 286–295.

[10] J. a. Gama and P. Kosina, "Learning about the learning process," in *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis X*, ser. IDA'11. Berlin, Heidelberg: Springer, 2011, pp. 162–172.

[11] A. Bifet and R. Gavaldà, "Kalman filters and adaptive windows for learning in data streams," in *Proceedings of the 9th International Conference on Discovery Science*, ser. DS'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 29–40.

[12] ——, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 443–448.

[13] J. Gama, R. Fernandes, and R. Rocha, "Decision trees for mining data streams," *Intelligent Data Analysis*, vol. 10, no. 1, pp. 23–45, 2006.

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

[15] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artificial Intelligence*, vol. 209, pp. 11–28, 2014.

[16] A. Kolmogoroff, "Confidence limits for an unknown distribution function," *The Annals of Mathematical Statistics*, vol. 12, no. 4, pp. 461–463, 1941.

[17] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases*. VLDB Endowment, 2004, pp. 180–191.

[18] K. V. Mardia, *Families of Bivariate Distributions*. London: Griffin, 1970.

[19] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.