Fast concept drift detection using unlabeled data

Dan Shang, Guangquan Zhang and Jie Lu

Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia

E-mail: dan.shang@student.uts.edu.au; {guangquan.zhang, jie.lu}@uts.edu.au

Streaming data mining is in use today in many industrial applications, but performance of the models is deteriorated by concept drift, especially when true labels are unavailable. This paper addresses the need of detecting concept drifts under unsupervised situation and proposes the Unsupervised Concept Drift Detection (UCDD) method. A cluster technique is first applied to determine artificial labels of the data set, then a fast drift detection algorithm is used to detect the boundary change between the labeled clusters. Through the empirical evaluation, the method demonstrates effectiveness on detecting various types of concept drifts.

Keywords: Concept Drift; Unsupervised Learning; Stream Data Mining.

1. Introduction

Data streams mining has attracted much attention of researchers in recent years. One of the main characteristics of stream data mining is that the distribution underlying may change over time. In traditional data mining models, one assumption is that training data share the same distribution with the test data. It means that data is static over the process of training and predicting. However, it is not the case in many real world applications. For example, text posts on social media website can be seen as a data stream. The target is to predict if one post is the given users' interested type. However, the concerns of the user may vary for some special period of time. This phenomenon is known as concept drift [1].

Detecting concept drift is challenging because real world stream data is continuously generated in high speed and volume. A common constraint of stream data mining is that data points can only be read once, which leads to difficulties in acquiring true labels timely. In most applications, true label data may not be available at all.

The most direct and efficient drift detection method is monitoring the

performance of models. For example, DDM[2,3] detects concept drift by supervising the change of error rate. The accuracy performance deteriorating implies that the underlying concept has changed, and adaptation to the change needed be adopted. However this strategy of monitoring error rate needs labels to compute the error rate, which are rarely available immediately in stream data mining situations. Thus this method is not suitable in the unsupervised applications because of the lacking of true labels.

Another commonly used strategy for handling concept drift is to update the models by training a new model to replace the older one periodically with latest data[4,5]. This kind of methods does not need drift detection process. The main drawback is that the performance overhead of retraining the models can be significant when the stream data is stationary most time. A recent work aims to estimate which data should be collected from the new concept[6]. However, it still heavily relies on the true label data to set the dynamic threshold.

This work thus focuses on detecting concept in the situation of lacking true labels. This is more common in real world applications and is also more challenging. In literature, many unsupervised methods have been proposed to handle this problem[7,8]. These methods are designed to test the density change of data points to detect virtual drift of data set. Such methods assume that if the distribution of data points changes, there is high possibility that the current model no longer fits the data[9]. The inherited limitation of distribution-test-based methods is that data distribution change does not necessarily leads to concept drift. In fact, it is possible that most concept drift detected by such methods are false alarms that do not affect the performance of the prediction model.

In this paper, we propose a method targeting to detect concept drift in unsupervised scenarios without labels. Our method is based on the assumption that data points of the same cluster are more likely to have the same class labels. Thus, if the cluster boundaries evolve, the decision boundary of the prediction models may also change. In other words, the margin changes of the clusters reflect the classification boundary change. The proposed method addresses the issue of lacking true labels for concept drift detection by clustering techniques.

The rest of the paper is organized as follows: Section 2 provides a formal problem definition of concept drift. Section 3 explains the proposed method. Section 4 includes a set of experiments for evaluating the proposed method. Finally, Section 5 summarizes our work and future studies.

2. Problem Definition and Preliminary

In classification problems, the target variable y is predicted by calculating the posterior probability of the classes, which is presented as $P(y|X) = \frac{P(y)P(X|y)}{P(X)}$, X the input of the model. Accordingly, concept drift can be defined as change of the relation between between the input X and the target y, which is referred as the the joint distribution of X and y, denoted as $P_t(X, y)$. t presents the time point. Concept drift may be caused by the change of $P(x)$ or $P(y|X)$. Concept drift can be categorized into two groups according to the source of change. If only the distribution of X changes, it is called virtual drift. If the relationship between inputs and their target variable vary, it is called real concept drift. Take the text post as an example, the change of post contents are virtual drift, whereas the change of users' preference is real drift. According to the speed of drifts, concept drifts can also be categorized into gradual drifts and sharp drifts [1].

The main challenge we are dealing with is that detecting real concept drifts with high probability in unsupervised situations. When labels are unavailable, current solution of concept drift detection is through virtual drift detection. However, most of virtual drifts will not necessarily lead to classification accuracy degradation. Thus, there will be lots of false alarm to hinder the overall application performance.

3. Proposed Method

The proposed method includes two steps: 1) generate artificial labels for unlabeled data using clustering technique; 2) detect boundary changes of the clusters. The assumption of our algorithm is that the observations that cluster together have the same labels. Under the assumption, we first employ some clustering learner on the data sets to distribute labels to the data points. Then we test the margin change between the clusters using a real drift detection method. If density changes near the boundary region, it is referred that there is a real concept drift.

The choice of clustering method and initial parameter setting just depend on characteristics of individual data set and requirements of the training task. K-means algorithm is the most used clustering approach in research and real world applications, because of its low computing cost and high time efficient. It is an clustering method based on partition. It partition the data sets into k clusters, in which each instance can be assigned the label the same as cluster with the nearest mean. The basic idea of K-means is to updated the center of cluster through iterative computation.

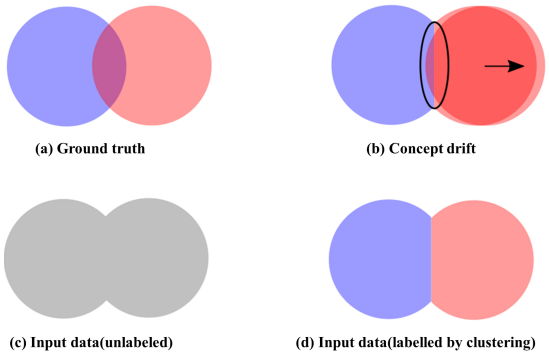

Fig. 1. Concept drift reflected in artificially labeled data by clustering.

In this work, we use K-means clustering method by default.

In Figure 1, data points of two classes are illustrated as blue and red spheres. The ground truth (a) shows that the two classes overlap each other, while the actual input data is unlabeled (c). However, artificial labels can be generated (d) using clustering method. When concept drift occurs (b), the new data points of the red class tend to locate towards the right direction. Such drift can be detected from the data distribution change near the classification boundary. We can count the number of points on the boundary. Under the hypothesis that the two data batches obey the same distribution, we can calculate probability P of the event that there are n_0 points from data window X_0 and n_1 points from data window X_1 fall in G. The distribution has the PDF of Beta distribution $\mathrm{CDF}_{\mathrm{Beta}(n_0,n_1)}(0.5)$. If p is very small, we reject the null hypothesis and consider X_0 and X_1 following different distribution, thus a concept drift is detected. The pseudo-code is listed in Algorithm 1.

4. Experimental Evaluation

In this section, we conduct experiments with synthetic data sets of various types of concept drift to evaluate effectiveness and efficiency of the proposed concept drift detection method.

We compare our method with two state of the art concept drift detection methods, namely detection - competence model-based drift detection(CM)[10] and information-theoretic drift detection(KDQ)[8]. Notice that these methods support both detection with and without true label data. For a fair comparison, in our experiments these methods are implemented

Algorithm 1: Unsupervised Concept Drift Detection (UCDD)

input : X_0, data points of first window.
X_1, data points of second window.
C, clustering model.
c, number of clusters, default 2.
k, number of neighbours for searching, default 1.
θ, confidence threshold for drift detection, default 0.05.
output: boolean concept drift detection result,
true (drift)
false (no drift)

let $X :=$ union(X_0, X_1);
let $X^+, X^- :=$ predict(C, c, X);
let $X_0^+, X_1^+ :=$ separate(X^+);
let $X_0^-, X_1^- :=$ separate(X^-);
let $beta^- =$ ComputeBeta(X_0^+, X_0^-, X_1^-);
let $beta^+ =$ ComputeBeta(X_0^-, X_0^+, X_1^+);
if $beta^+ < \theta$ *or* $beta^- < \theta$ **then**
  return *true (drift)*
else
  return *false (no drift)*

Procedure `ComputeBeta` *(U, V_0, V_1)*
  let $W_0 := \emptyset$;
  let $W_1 := \emptyset$;
  for x *in* U **do**
    $W_0 := W_0 \bigcup$(neigbour of x in V_0);
    $W_1 := W_1 \bigcup$(neigbour of x in V_1);
  return $Beta(|W_0|, |W_1|, 0.5)$

without true labels.

The default window sizes of sample data for all methods are set to 500, unless indicated otherwise. The significant threshold is 5%. Table 1 lists the parameters that are specific to each method. These parameters are chosen according to their authors' recommendations.

4.1. *Detecting concept drift of various types*

In this experiment, we evaluate the detection accuracy of the proposed method by comparing it with CM and KDQ. The test data sets are generated with multivariate Gaussian distribution. The data points of two classes are generated with different mean value. The concept drift is in-

Table 1. Parameters of comparison methods used in the experiments.

Method	Parameter	Symbol	Value
UCDD	Number of neighbours	k	1
CM	Euclidean distance threshold	d_ε	0.05
KDQ	Minimum side length of a cell	δ	2^{-10}
	Maximum number of points in a cell	τ	100

Table 2. Drift detection accuracy (number detection out of 100 drift) with data sets of different concept drift types.

Drift type	Drift magnitude	UCDD	CM	KDQ
One class mean increases	0.05	41	23	16
	0.07	59	34	21
	0.09	78	57	43
One class mean decreases	0.05	32	26	17
	0.07	36	31	19
	0.09	49	65	42
Both class mean changes	0.05	68	45	29
	0.07	84	73	46
	0.09	90	92	83

troduced by changing the mean value of one or both classes to simulate various concept drift types.

The result is shown in Table 2. We can see that the proposed UCDD outperforms the comparing methods in most cases, especially when the drift magnitude is small. This means that UCDD is very sensitive to small boundary changes. KDQ on the other hand, has the relatively worst performance for all drift types.

4.2. *Detection efficiency*

In this experiment, we evaluate the efficiency of the proposed method by measuring its computational time and compare it with the reference methods. The experiment environment is Intel 32GHz CPU, Linux operating system. All the methods are implemented with Python 2.7 numpy library.

The experiment settings are similar as previous. The two classes of the data points are generated with multivariate Gaussian distribution. The first three batches of the data varies in dimensions from 2 to 8, with fixed window size 500. Then for dimension 8, two more batches are generated with larger window sizes 1000 and 1500. This strategy will help us to identify the impact of both dimension increase and window size increase on the methods' performance.

The result is shown in Table 3. We can see that the proposed UCDD

Table 3. Drift detection time with data sets of different dimensions and window sizes.

Dimension	Size	Method	Modeling(s)	Detection(s)
2	500	UCDD	0.034	0.004
		CM	10.981	0.032
		KDQ	2.730	0.005
4	500	UCDD	0.036	0.004
		CM	11.852	0.035
		KDQ	3.349	0.005
8	500	UCDD	0.043	0.004
		CM	16.803	0.041
		KDQ	3.204	0.006
8	1000	UCDD	0.082	0.008
		CM	39.295	0.083
		KDQ	5.011	0.008
8	1500	UCDD	0.141	0.017
		CM	86.337	0.174
		KDQ	7.416	0.016

is more efficient than CM and KDQ by nearly two magnitude. The main reason is that CM and KDQ both include permutation process to estimate the significance level, whereas UCDD can directly compute the P-value for given significance threshold. Another reason is that CM and KDQ compute distribution difference on all the data points, but UCDD only detect the differences on the clusters' boundaries. From the result we can also see that all the methods can efficiently process high-dimensional data without much performance decrease. On the other hand, their performance is significantly impacted by the window size increase.

5. Conclusion

We presented a fast concept drift detection method for detecting real drift in unsupervised situation. The proposed method Unsupervised Concept Drift Detection (UCDD) first assigned labels using cluster strategy, then performed a fast real drift detection method to supervise the boundary variation. Our experiments showed that the proposed method outperforms other two state of the art concept drift detection methods. How to adapt the models under unsupervised situation is an interesting field for our future work.

Acknowledgments

The work presented in this paper was supported by the Australian Research Council (ARC) under Discovery Project DP190101733.

References

1. J. Lu, A. Liu, F. Dong, F. Gu, J. Gama and G. Zhang, Learning under concept drift: A review, *IEEE Transactions on Knowledge and Data Engineering* (2018).
2. J. Gama, P. Medas, G. Castillo and P. Rodrigues, Learning with drift detection, in *Proceedings of 17th Brazilian Symposium on Artificial Intelligence - SBIA 2004*, (Springer, 2004).
3. I. Frias-Blanco, J. d. Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz and Y. Caballero-Mota, Online and non-parametric drift detection methods based on hoeffding's bounds, *IEEE Transactions on Knowledge and Data Engineering* **27**, 810 (2015).
4. J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy and A. Bouchachia, A survey on concept drift adaptation, *ACM Computing Surveys* **46**, 1 (2014).
5. C. Alippi, G. Boracchi and M. Roveri, Just-in-time classifiers for recurrent concepts, *IEEE Transactions on Neural Networks and Learning Systems* **24**, 620 (2013).
6. Z. Yang, S. Al-Dahidi, P. Baraldi, E. Zio and L. Montelatici, A novel concept drift detection method for incremental learning in nonstationary environments, *IEEE transactions on neural networks and learning systems* (2019).
7. A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf and A. Smola, A kernel two-sample test, *The Journal of Machine Learning Research* **13**, 723 (Mar 2012).
8. 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).
9. Y. Yasumura, N. Kitani and K. Uehara, Quick adaptation to changing concepts by sensitive detection, in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems - IEA/AIE 2007*, eds. H. G. Okuno and M. Ali (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007).
10. N. Lu, G. Zhang and J. Lu, Concept drift detection via competence models, *Artificial Intelligence* **209**, 11 (2014).