INFORMATICA, 2008, Vol. 19, No. 1, 135–156

An Efficient and Sensitive Decision Tree Approach to Mining Concept-Drifting Data Streams

Cheng-Jung TSAI
Department of Computer Science, National Chiao Tung University
1001, Ta Hsueh Rd., Hsinchu 300, Taiwan, Republic of China
e-mail: tsaicj@cis.nctu.edu.tw

Chien-I LEE
Department of Information and Learning Technology, National University of Tainan
33, Sec. 2, Shu-Lin St. Tainan 700, Taiwan, Republic of China
e-mail: leeci@mail.nutn.edu.tw

Wei-Pang YANG
Department of Information Management, National Dong Hwa University
No.1, Sec. 2, Da Hsueh Rd., Shoufeng, Hualien 97401, Taiwan, Republic of China
e-mail: wpyang@mail.ndhu.edu.tw

Received: November 2006

Abstract. Data stream mining has become a novel research topic of growing interest in knowledge discovery. Most proposed algorithms for data stream mining assume that each data block is basically a random sample from a stationary distribution, but many databases available violate this assumption. That is, the class of an instance may change over time, known as concept drift. In this paper, we propose a Sensitive Concept Drift Probing Decision Tree algorithm (SCRIPT), which is based on the statistical X^2 test, to handle the concept drift problem on data streams. Compared with the proposed methods, the advantages of SCRIPT include: a) it can avoid unnecessary system cost for stable data streams; b) it can immediately and efficiently corrects original classifier while data streams are instable; c) it is more suitable to the applications in which a sensitive detection of concept drift is required.

Key words: data mining, data streams, incremental learning, decision tree, concept drift.

1. Introduction

Data mining, an important technique used in searching for knowledge in databases, has attracted many researchers' attention in recent years. Among several functionalities of data mining, classification is crucially important and has been applied successfully to several areas (Jegelevičius *et al.*, 2002; Remeikis *et al.*, 2004). The popular techniques developed for classification includes Bayesian classification, Neural Networks, Genetic Algorithms, and Decision Trees (Gehrke *et al.*, 2000; Han and Kamber, 2001; Mehta *et al.*, 1996; Misevičius, 2006; Quinlan, 1993; Rastogi and Shim, 1998; Shafer *et al.*,

1996). However, in addition to displaying comparable classification accuracy to other techniques, decision tree is more efficient and easily interpreted by human (Rastogi and Shim, 1998). In the research domain of decision tree, several important issues, including scalability (Gehrke *et al.*, 2000), imbalanced dataset (Japkowicz and Stephen, 2002), ensemble classifier (Chawla *et al.*, 2002), incremental learning (Schlimmer and Fisher, 1986; Utgoff, 1989) etc., also have been widely studied. Since the real-world data nowadays might come in the form of consecutive data blocks (Domingos and Hulten, 2000), researchers have put more and more attention on data streams mining. The related applications include e-mail sorting (Cohen, 1996), calendar scheduling (Blum, 1997), and computer intrusion detection (Maloof and Michalski, 2000) etc. Nevertheless, most proposed approaches of data stream mining assumed data blocks come under *stationary distribution*. Such an assumption is unreasonable since the *concept* (also called *target class*) of an instance might change as time goes by. That is, an instance with concept "yes" in current data block may be with concept "no" in the next one. Such a change of concept is known as *concept drift* (Harries *et al.*, 1998; Kifer *et al.*, 2004; Schlimmer and Fisher, 1986; Wang *et al.*, 2003), *changing concepts* (Klinkenberg and Renz, 1998), or *time-varying concepts* (Kuh *et al.*, 1991).

Window-based approaches (Hulten *et al.*, 2001; Klinkenberg and Renz, 1998; Maloof, 2003; Widmer and Kubat, 1996) are the common solutions for the concept drift problem on data stream. They use a fixed or sliding window (Jin and Agrawa, 2003) to select appropriate training data for different time points. Weighting-based (Koychev, 2000; Kolter and Maloof, 2003) and ensemble classifier (Fan, 2004; Street and Kim, 2001) were also introduced to handle the concept drift problem. However, while the concept is stationary, the methods mentioned above spend a lot unnecessary system cost, including computational cost to rebuild the decision tree or storage cost to record similar data blocks. Moreover, they generally are not sensitive enough to the concept drift problem. For some real-time applications such as fraudulent credit card transactions or computer virus detection, a sensitive approach to detect drifting concepts would be very important since serious damage can be therefore reduced. Finally, it is interesting to note that drifting instances should gather in some specific areas in the dimensional space of attributes, otherwise they can be regarded as noise instances. These foregoing observations motivate us to propose a more efficient and sensitive approach to mine drifting concepts on data streams.

In this paper, we propose a *Sensitive Concept Drift Probing Decision Tree algorithm* (*SCRIPT*). The main contributions of SCRIPT are: a) it can avoid unnecessary system cost for stable data streams; b) it can efficiently rebuild classifier while data streams are instable; c) it is more suitable for the applications in which a sensitive detection of concept drift is required. We evaluated SCRIPT on UCI Database (Agrawal *et al.*, 1992) to demonstrate its accuracy, efficiency, and sensitivity on the detection of concept drift. The remainder of this paper is organized as follows. Section 2 introduces the concept drift problem and some traditional data stream mining algorithms. In Section 3, we propose our Sensitive Concept Drift Probing Decision Tree algorithm. The experimental evaluation is presented in Section 4. Finally, we concludes this paper.

2. The Problem of Concept Drift on Data Stream Mining

VFDT (Very Fast Decision Tree Learner) (Domingos and Hulten, 2000) have been proposed to solve the scalable problem when learning from very large data stream. It starts with a single leaf and starts collecting training examples from a data stream. When VFDT gets enough data to know, with high confidence that it knows which attribute is the best to partition the data with, it turns the leaf into an internal node and goes on splitting it. However, as most incremental learning methods, it assumes that the data is a random sample drawn from a stationary distribution and is inappropriate for the concept drift mining such as credit card approval and fraud detection. CVFDT (Hulten *et al.*, 2001) (Concept-adapting Very Fast Decision Tree Learner), which is formerly VFDT, is a representative window-based approach for mining concept drift on data stream. It solves the concept drift problem by maintaining only fixed amount of data within the window. CVFDT keeps its learned tree up-to-date with this window by monitoring the quality of its old decisions as data moves into and out of the window. In particular, whenever a new instance is read it is added to the statistics at all the nodes in the tree that it passes through, the last example in the window is forgotten from every node where it had previously had an effect, and the validity of all statistical tests are checked. If CVFDT detects a change, it starts growing an alternate tree in parallel which is rooted at the newly-invalidated node. When the alternate is more accurate on new data than the original, the original will be replaced by the alternate tree.

WAH (Window-Adjustment-Heuristic) (Widmer and Kubat, 1996) and DNW (Klinkenberg and Renz, 1998) are also window-based algorithms, however, they use sliding window. WAH take the actual condition of decision tree into account to dynamically adjust the window size. After new data stream join, the doubt for concept drift will reduce the size of windows by 20%. Contrarily, when data are stable, a unit of window is deleted to avoid maintaining too many unused data. When the concept seems to be stable, the original window size is maintained. If none of the conditions mentioned above are valid, it means that more information will be needed to build classifiers. As a result, old data will not be left out of the window and new data will also be added in it. Although WAH can solve the problem of concept drift according to actual conditions, but it is suitable only for small databases. DNW deals with the learning of training data by way of data block, which is suitable for data stream environment. DNW has a similar way of learning to WAH; however, they are different in condition and way of assessment. DNW builds a classifier for each block, and compares the three parameters: accuracy, recall, and precision for classifiers on the current blocks with the ones for the previous classifiers. Weighting-based (Koychev, 2000; Kolter and Maloof, 2003) and ensemble classifier (Fan, 2004; Street and Kim, 2001) were also introduced to handle the concept drift problem on data stream. Weighting-based approach provides each example with a weight according to their age and utility for the classification task. Ensemble classifier built separate sub-classifiers and then combines the prediction of each sub-classifier to classify the unseen data. The main disadvantage of an ensemble classifier is the huge system cost caused by the building and maintenance of all sub-classifiers.

While the concept is stable, the methods mentioned above would spend unnecessary system cost, including computational cost to build or rebuild a decision tree or storage cost to record similar data streams. Moreover, when concept is drifting, they generally are not sensitive enough to the concept drift problem. That is, if the proportion of drifting instances to all instances in a data block is small, the proposed solutions can detect the changes until the number of drifting instances reaches a threshold to cause obvious difference in accuracy or information gain. For some applications such as fraudulent credit card transactions, the sensitivity to detect drifting concepts would be very important. In an ensemble classifier, the fraudulent transactions might be ignored due to the predictions of old sub-classifiers. For weighting-based approaches, even giving a high weight to the transactions in the new data block, they might also make a wrong prediction since the influence of old transactions. Fixed window-based approaches have a similar problem to that in weighting-based approaches, and sliding window-based approaches would also disregard such changes since these drifting transactions would not cause obvious variance of accuracy or information gain.

3. Sensitive Concept Drift Probing Decision Tree Algorithm

In this section, we first give some discussions of the concept problem in Section 3.1. We then define the CDAV which is used to probe the drifting concept in Section 3.2. In Section 3.3, we present the correct mechanism to efficiently handle concept drift problem. Section 3.4 is our Sensitive Concept Drift Probing Decision Tree algorithm (SCRIPT) Algorithm. Finally, we compare the system cost among SCRIPT, DNW, and CVFDT in Section 3.5.

3.1. *Concept Stable, Concept Drift, and Concept Shift*

To make readers easily understand the problem we will address later, in this paper we divide the concept drift into concept stable, concept drift, and concept shift (Hsieh, 2004). We refer to the examples in (Wang *et al.*, 2003) and modify the figures to illustrate the problem in Fig. 1. Fig. 1 represents a two-dimensional data stream and is divided into six continuous data blocks according to the arriving time of data. Instances arriving between t_i and t_{i+1} form block B_i, and the separating line in each block stands for the optimum classification boundary in this block. During time t_0 to t_1, data blocks B_0 and B_1 have similar data distribution. That is, data stream during this period is stable. Thereafter in B_2, some instances shows concept drift and the optimum boundary changes. This is defined as "concept drift". Finally, data blocks B_4 and B_5 have opposite sample distribution and this is defined as "concept shift". Obviously, since the sample distributions of the first two blocks B_0 and B_1 are quite close, we can use decision tree DT_0 built by B_0 as the classifier for B_1 to save the computational and recording cost. Meanwhile, B_2 shows slight differences when compared with the sample distribution of B_1 and an efficient approach should make correction according to the original decision tree in stead of rebuilding it.

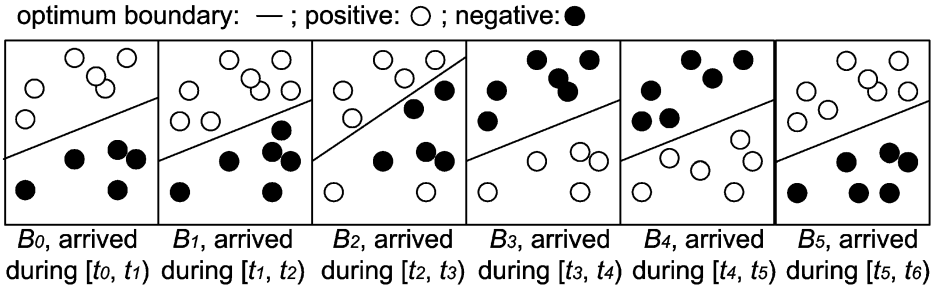

Fig. 1. A data stream with the occurrence of concept drift.

Besides, in Section 2, we have showed that the proposed solutions are not sensible enough to the drifting concepts. That is, the proposed solutions can detect the changes until the number of drifting instances reaches a threshold to cause obvious difference in accuracy or information gain. Here we describe another concept drift problem which would enforce some proposed solutions such as CVFDT and DNW make a wrong prediction. In order to introduce this problem, we subdivide concept drift into *one-way drift* and *two-way drift* (Hsieh, 2004). Take Fig. 1 as the example again, we can find that some negative data in B_2 drift to be positive data in B_3, known as one-way drift. However, the positive data in B_4 drift to be negative in B_5, and vice versa, known as two-way drift. We can regard two-way drift as a kind of "local" concept shift if it occurs in the internal or leaf node of a decision tree. If the variation of information gain or gini index is used as the criterion to judge the occurrence of concept drift, e.g., the difference of information gain adopted in CVFDT, we can detect only one-way drift since the information gain obtained from B_4 would the same as B_5. It is worth to note that for the real data, two-way drift might happen. For example, a hacker in turn uses two computers with IP address x and y to send attack packages. When an internal node, which is learned from the first data block, splits the packages form x as safe and that from y as attack, there might be a contrary result learned from another data block. A similar condition might be found in trash mail protection, image comparison and so on.

3.2. *Class Distribution on Attribute Values*

Since the proposed solutions to mine concept-drifting data stream check the occurrence of concept drift on the level of instance or attribute, they generally are not sensitive enough. Besides, they are also unable to detect the two-way drift illustrated in Fig. 1. To solve these problems, SCRIPT probe the changes at a more detailed level, which is called *CDAV* (*Class Distribution on Attribute Values*) and defined as follows.

DEFINITION 1. Assuming that a data block contains m target classes c_k ($k = 1, \ldots, m$), n attributes a_i ($i = 1, \ldots, n$), and each attribute a_i having v attribute values a_{ij} ($j = 1, \ldots, v$), then the distribution of target class c_k on the attribute value a_{ij} is defined as a $CDAV_{ij}$ (*Class Distribution on Attribute Values*).

With Definition 1, we can use the X^2 test to check if there are concept drift between two data blocks. X^2 is a statistical measure used to test the hypothesis that two discrete attributes are statistically independent. Applied to the concept problem, it tests the hypothesis that the class distribution on an attribute value of two data blocks is identical. The formula to computing the X^2 value is

$$X^2 = \frac{(f'_{ijk} - f_{ijk})^2}{f_{ijk}}, \tag{1}$$

where f_{ijk} represents the number of instances having attribute value a_{ij} and class c_k in D and f'_{ijk} is that in D'. With Formula (1), we can then define the variance of a $CDAV_{ij}$ in the two data blocks as follows.

DEFINITION 2. For a given significant level α, the variance $CDAV_{D\to D'}(i,j)$ of the a $CDAV_{ij}$ between two data blocks D and D' in a data stream is defined as

$$CDAV_{D\to D'}(i,j) = \sum_{k=1}^{m} \frac{(f'_{ijk} - f_{ijk})^2}{f_{ijk}}. \tag{2}$$

PROPOSITION. For the two data blocks D and D', if all $CDAV_{D\to D'}(i,j) < \varepsilon$, then the concept distribution on all attribute value a_{ij} in the two data blocks show no significant difference, and neither do the accuracy of decision tree built according to D and D', respectively.

Proof. Since $CDAV_{D\to D'}(i,j) < \varepsilon$, we can obtain $f_{ijk} \cong f'_{ijk}$ for target classes c_k ($k = 1, \ldots, m$), attributes a_i ($i = 1, \ldots, n$) and attribute value a_{ij}.

For attribute a_i, the Entropy before the splitting is

$$I(a_i) = -\sum_{k=1}^{m} P_{ik} \log P_{ik},$$

where $P_{ik} = f_{i+k}/N$, f_{i+k} denotes the total number of instances belonging to class c_k as shown in Table 1, and N denotes the total number of instances in the data block.

Since $f_{ijk} \cong f'_{ijk}$ and $N = N'$ we can obtain

$$P_{ik} = f_{i+k}/N = \sum_{j=1}^{v} f_{ijk}/N \cong -\sum_{j=1}^{v} f'_{ijk}/N' = f'_{i+k}/N' = P'_{ik}.$$

Continually, we can obtain that

$$-\sum_{k=1}^{m} P_{ik} \log P_{ik} \cong -\sum_{k=1}^{m} P'_{ik} \log P'_{ik} \quad \text{and} \quad I(a_i) \cong I(a'_i). \tag{3}$$

That is, the Entropy of attribute a_i before splitting in data blocks D and D' is similar.

Suppose we splitting all instances N into v subset by attribute a_i, the Entropy of attribute a_i after splitting is

$$E(a_i) = \sum_{j=1}^{v} \frac{f_{ij+}}{N} \times \left(-\sum_{k=1}^{m} P_{ijk} \log P_{ijk} \right), \tag{4}$$

where $P_{ijk} = f_{ijk}/f_{ij+}$, f_{ij+} denotes the total number of instances having attribute value a_{ij} as shown in Table 1.

Since $f_{ijk} \cong f'_{ijk}$ we can infer that for the attribute value a_{ij}

$$f_{ij+} \cong f'_{ij+}. \tag{5}$$

As a result, we can also obtain that

$$P_{ijk} \cong P'_{ijk} \quad \text{and} \quad -\sum_{k=1}^{m} P_{ijk} \log P_{ijk} =\cong -\sum_{k=1}^{m} P'_{ijk} \log P'_{ijk}. \tag{6}$$

From (4), (5) and (6), we can get that

$$E(a_i) \cong E(a'_i). \tag{7}$$

From (3) and (7), we can get that

$$Gain(a_i) = I(a_i) - E(a_i) \cong I(a'_i) - E(a'_i) = Gain(a'_i).$$

That is, the *Information gain* of attribute a_i in data blocks D and D' is similar.

As a result, the two decision trees which are respectively built by using blocks D and D' will be similar.

By this Proposition and Formula 2, we can detect any kind of concept drift between two data blocks and then build an accurate decision tree. The significance level can be set to be smaller or larger according to the needs of applications. With a given significance level, we can obtain the ε by checking the X^2 table in a statistical book. The degree

Table 1

The class distribution on an attribute a_i

Class \ value	a_{i1}	a_{i2}	$\ldots$	a_{iv}	Summation
c_1	f_{i11}	f_{i21}		f_{iv1}	f_{i+1}
c_2	f_{i12}	f_{i22}		f_{iv2}	f_{i+2}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
c_m	f_{i1m}	f_{i21}		f_{ivm}	f_{i+m}
Summation	f_{i1+}	f_{i2+}		f_{iv+}	N

of freedom will be 1 less than the number of classes. Suppose that we set the level of significance $\alpha = 5\%$ and there are three classes, if all $CDAV_{D\to D'}(i,j)$are less than $\varepsilon = 5.991$, that means the class distribution on all attributes shows no significant difference between D and D' with 95% confidence. As a result, the information gain obtained from any attribute will show no significant difference and the decision tree need not to be rebuilt. Note that the purpose of our Proposition is to claim that a rebuild tree will have very similar accuracy to that of original one, rather than to guarantee the rebuild tree will be a copy of the original one.

For clearly understand our idea, a case with two datasets D and D' is presented in Table 2. Each of the two sets has two attributes A_1 and A_2, and each attribute has three attribute values $(a_{11}, a_{12}, a_{13}; a_{21}, a_{22}, a_{23})$. There are total 500 instances and two classes are c_1 and c_2 in each dataset. Assuming that the level of significance $\alpha = 5\%$ (*degree of freedom* $= 1$ and $\varepsilon = 3.841$), we can infer the following by Formula 2:

$$CDAV_{D\to D'}(1,1) = 0.6723 < \varepsilon; \quad CDAV_{D\to D'}(1,2) = 0.5948 < \varepsilon;$$
$$CDAV_{D\to D'}(1,3) = 0.5326 < \varepsilon; \quad CDAV_{D\to D'}(2,1) = 2.7763 < \varepsilon;$$
$$CDAV_{D\to D'}(2,2) = 1.7223 < \varepsilon; \quad CDAV_{D\to D'}(2,3) = 1.5948 < \varepsilon.$$

Since all $CDAV$s have no significant difference, by our Proposition mentioned above, the decision trees built respectively with D and D' would be very similar. To verify this, we build the two decision trees and show the corresponding rules. The rules obtained from data set D are:

(1) $A_1 =$ "a_{12}" $\to c_2$; (2) $A_1 =$ "a_{13}" $\to c_2$;
(3) $A_1 =$ "a_{11}" $\cap A_2 =$ "a_{21}" $\to c_2$; (4) $A_1 =$ "a_{11}" $\cap A_2 =$ "a_{22}" $\to c_1$;
(5) $A_1 =$ "a_{11}" $\cap A_2 =$ "a_{23}" $\to c_2$.

And the rules obtained from data set D' are:

(1) $A_1 =$ "a_{12}" $\to c_2$; (2) $A_1 =$ "a_{13}" $\to c_2$;
(3) $A_1 =$ "a_{11}" $\cap A_2 =$ "a_{21}" $\to c_2$; (4) $A_1 =$ "a_{11}" $\cap A_2 =$ "a_{22}" $\to c_1$;
(5) $A_1 =$ "a_{11}" $\cap A_2 =$ "a_{23}" $\to c_2$.

We can find that the two decision tree have identical rules. This result corresponds to our Proposition.

Table 2
Two data sets D and D' without occurrence of concept drift

Dataset		D						D'					
attribute		A_1			A_2			A_1			A_2		
Attribute value		a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}
class	c_1	192	41	13	18	216	12	198	42	12	25	211	15
	c_2	33	142	79	74	122	58	37	133	73	76	108	65

COROLLARY. By above Proposition, we can infer that if the variance of $CDAV$ for the two data blocks D and D' is greater than or equivalent to a threshold ε, (i.e., $CDAV_{D\to D'}(i,j) \geqslant \varepsilon$), then concept drift may occur between D and D'. As a result, the original decision tree needs to be corrected.

Here, we use the two datasets in Table 3 modified from Table 2 to illustrate this Corollary. Again assuming that the level of significant $\alpha = 5\%$ (*degree of freedom*$= 1$ and $\varepsilon = 3.841$), we can infer the following by Formula 2:

$$CDAV_{D\to D'}(1,1) = 3.0848 < \varepsilon; \qquad CDAV_{D\to D'}(1,2) = 1.5402 < \varepsilon;$$
$$\boldsymbol{CDAV_{D\to D'}(1,3) = 5.9085 > \varepsilon}; \qquad CDAV_{D\to D'}(2,1) = 1.8754 < \varepsilon;$$
$$CDAV_{D\to D'}(2,2) = 0.4274 < \varepsilon; \qquad CDAV_{D\to D'}(2,3) = 2.7299 < \varepsilon.$$

Since $CDAV_{13}$ achieves significant difference, by above Corollary, we can claim that concept drift occurs and the decision trees built respectively with D and D' would be different. To verify this, we again show the corresponding rules for two trees as follows. The rules obtained from data set D are:

(1) $A_1 =$ "a_{12}" $\to c_2$; (2) $A_1 =$ "a_{11}" $\cap\, A_2 =$ "a_{21}" $\to c_2$;
(3) $A_1 =$ "a_{11}" $\cap\, A_2 =$ "a_{22}" $\to c_1$; (4) $A_1 =$ "a_{11}" $\cap\, A_2 =$ "a_{23}" $\to c_2$;
(5) $\boldsymbol{A_1 =}$ **"$\boldsymbol{a_{13}}$"** $\boldsymbol{\to c_2}$.

And the rules obtained from data set D' are:

(1) $A_1 =$ "a_{12}" $\to c_2$; (2) $A_1 =$ "a_{11}" $\cap\, A_2 =$ "a_{21}" $\to c_2$;
(3) $A_1 =$ "a_{11}" $\cap\, A_2 =$ "a_{22}" $\to c_1$; (4) $A_1 =$ "a_{11}" $\cap\, A_2 =$ "a_{23}" $\to c_2$;
(5) $\boldsymbol{A_1 =}$ **"$\boldsymbol{a_{13}}$"** $\boldsymbol{\cap\, A_2 =}$ **"$\boldsymbol{a_{21}}$"** $\boldsymbol{\to c_2}$; (6) $\boldsymbol{A_1 =}$ **"$\boldsymbol{a_{13}}$"** $\boldsymbol{\cap\, A_2 =}$ **"$\boldsymbol{a_{22}}$"** $\boldsymbol{\to c_1}$;
(7) $\boldsymbol{A_1 =}$ **"$\boldsymbol{a_{13}}$"** $\boldsymbol{\cap\, A_2 =}$ **"$\boldsymbol{a_{23}}$"** $\boldsymbol{\to c_2}$.

By comparison, we can find that the rule $A_1 = a_{13} \to c_2$ in dataset D have some changes in data set D'; the results correspond to our Corollary.

3.3. *Correcting Mechanism in SCRIPT*

As described in Introduction, drifting instances should gather in some specific areas in the dimensional space of attributes, otherwise they can be regarded as noise instances.

Table 3
Two data sets D and D' with occurrence of concept drift

Dataset		D						D'					
attribute		A_1			A_2			A_1			A_2		
Attribute value		a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}
class	c_1	192	41	13	18	216	12	203	34	20	23	208	16
	c_2	33	142	79	74	122	58	42	135	66	68	118	67

Accordingly, another advantage of $CDAV$ is that it can reveal which attribute values cause concept drift before building the decision tree by aggregating the drifting $CDAV$s. This will enable SCRIPT to efficiently and immediately amend the original decision tree. In the example in Table 3, we can recognize the concept drift is caused by attribute value a_{13}. Therefore, we can only correct the subtree rooted at a_{13} to efficiently correct the decision model.

We use Fig. 2 to further illustrate the idea of correcting mechanism in SCRIPT. Fig. 2(a) is a decision tree trained from old customer's data to predict if a customer will apply for credit cards. For better understanding, only the subtree rooted at attribute "salary" is shown. A similar decision tree, except that it is trained from new customer's data stream, is shown in Fig. 2(b). By comparison with the CDAVs in Fig. 2(a) and Fig. 2(b), we can find that some concept in new data block is significantly different from that in old one. More importantly, we can find that these changes gather up in the branch of "age < 20 and 20 $\leqslant$ age < 40". Accordingly, the aggregated drifting CDAVs

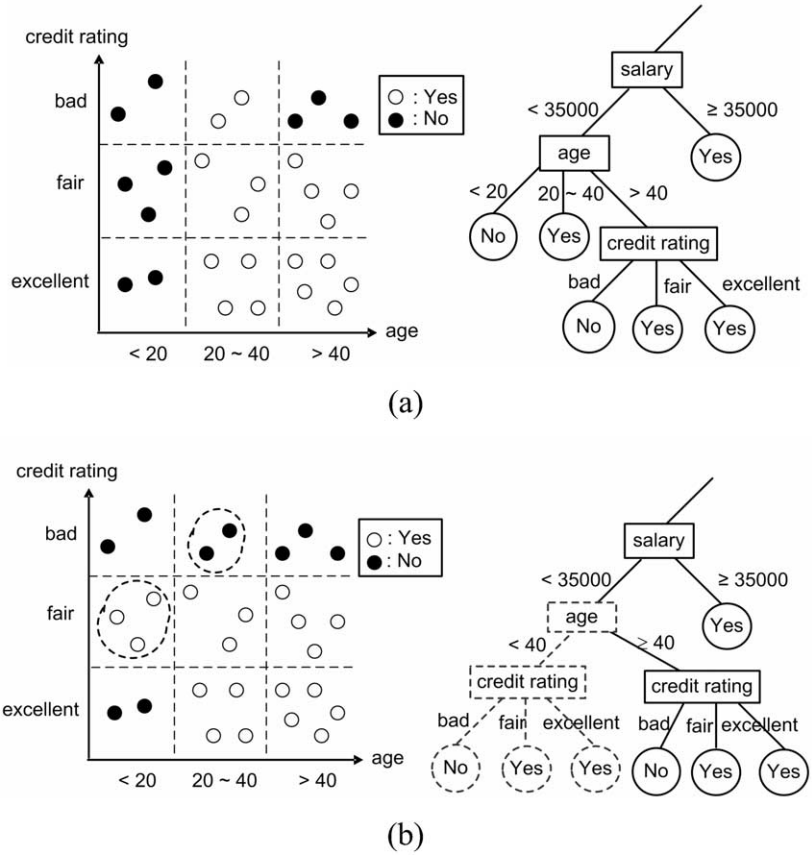

Fig. 2. Two data blocks with the occurrence of concept drift: (a) original data block and the corresponding sub-tree; (b) new data block and the corresponding sub-tree.

is $0 \leqslant$ age $<$ 40 and it means that a people younger than 40 have changed his concepts in this example. To efficiently provide a decision tree suitable for new customers' data block, we can only correct the subtree rooted at $0 \leqslant$ age $<$ 40 as in Fig. 2(b).

Now, we detail the correcting mechanism in SCRIPT. In the processing of data stream, when the difference of CDAV between new data block B_t and original data block B_{t-i} ($t \geqslant i \geqslant 1$) is greater than the given threshold (the level of significance is set 0.05 as the default), the correct methods in SCRIPT can be divided into the following cases. The corresponding illustration of each case is shown in Fig. 3. In each case of Fig. 3, the dotted node in the left tree (original tree) denoted the occurring of concept drift and the dotted subtree in the right tree (new tree) is an alternate tree built by SCRIPT.

For each aggregated drifting CDAV in attribute a_i with value(s) a_{ij}

a. If attribute a_i was not a split attribute of a node in the original decision tree; SCRIPT will use this attribute to split all leaf nodes by using data block B_t. Such a variation of CDAV indicates that an attribute with originally little information changes into an optimal split attribute due to concept drift. We illustrate this condition in Fig. 3(a).

b. If attribute a_i was a split attribute of a node in the original decision tree and all

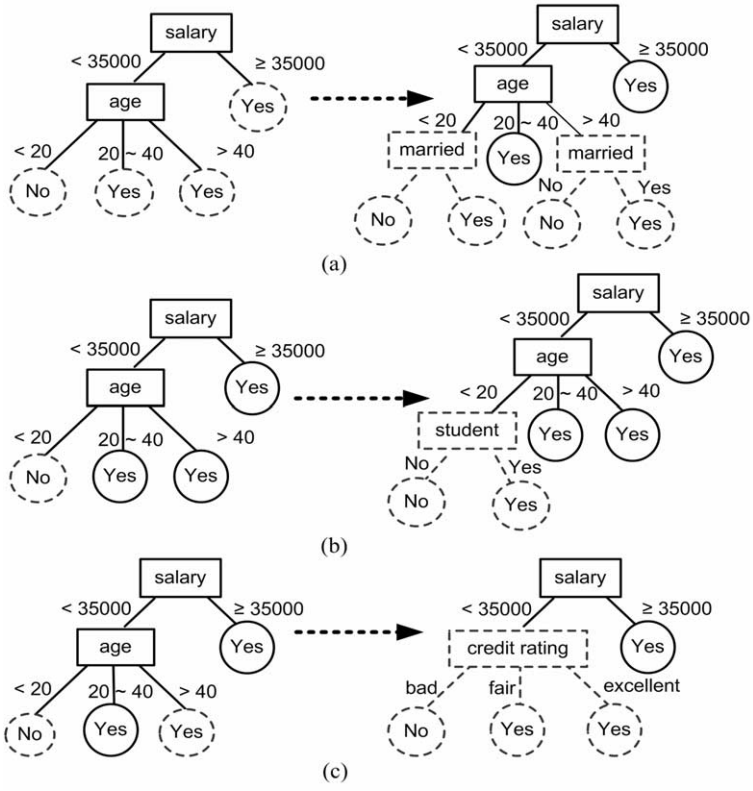

Fig. 3. The illustrations of correcting mechanism in SCRIPT when concept drift occurs.

CDAVs group in an interval a_{ij}, SCRIPT will remove the subtree rooted at the attribute value a_{ij} from the original tree and use data block B_t to build the alternative tree. Such a variation means concept drift is caused by a fixed range a_{ij} of the attribute a_i. Take Fig. 3(b) for example, for the attribute age, those under 20 were originally inclined not to apply for credit cards; however, with the growing consuming ability of students, more and more are applying.

c. If attribute a_i was a split attribute of a node in the original decision tree but all CDAVs are scattered in several interval, SCRIPT will removes the subtree rooted at this attribute a_i from the original tree and use data block B_t to build the alternative tree. Such a variation represents concept drift is caused by the attribute a_i but within multiple ranges of the attribute. For instance, for the attribute of age, people younger than 20 and older than 40 were originally both inclined not to apply for credit cards; however, with the change of payment types, more and more are applying. In this case, "age", no longer the optimal split attribute, is replaced by "credit rating", according to a test result. This case is illustrated in Fig. 3(c).

Note that all aggregated drifting CDAVs might distribute among several attributes and SCRIPT will check if they are in the same path in the original tree before the correctness. If two aggregated CDAVs are in the same path, the one locates in the highest level will be reserved and the other will be deleted.

3.4. *SCRIPT Algorithm*

Combine Section 3.1 to 3.3, we summary our Sensitive Concept Drift Probing Decision Tree (SCRIPT) algorithm in this section. Based on the variation of CDAV (Class Distribution on the Attribute Value), SCRIPT aims to apply to large scale and high speed applications which also require the sensitiveness to handle the drifting concepts. SCRIPT cuts the data stream into sequential data blocks. When the test threshold is set as α, each training data in the block should be read no more than once and processed in minimal constant time while concepts are stable. While concepts are instable, SCRIPT would detect concept drift, and then build the alternate tree to correct previously built tree. The pseudo code of SCRIPT is shown in Fig. 4. Giving the size of data block $N(1000$ as the default) and the significance level $\alpha(0.05$ as the default), SCRIPT calculates the CDAVs in data block B_0 in Line 4 as the initial reference. Note that, N can be set larger in a high speed environment or smaller for the real time application; however, f_{ijk} must be larger than 5 which is a basic requirement in X^2 statistics test. Similarly, α can be set smaller if the detection of concept drift is very important and larger otherwise. The default significant level α in SCRIPT is set as 0.05 since this value is widely used as the default in the statistic. It is not hard to imagine that SCRIPT will be more sensitive to the concept drift but may require more computational cost if we use a larger significant level α; on the contrary, SCRIPT will be more tolerant to the noise data with a smaller α. The CDAV of new coming block B_{t+1} is calculated in Line 6 to 10. The CDAVs of two data blocks B_t and B_{t+1} are compared in Line 11 to 14. All drifting CDAVs are then aggregated in Line 16 for the purpose of efficiently correcting the decision model in Line 22 to 31. Finally, the recorded information is updated in Line 32.

```
Inputs: N: the size of the block (1000 as the default);
        B_t: the data block in time step t;
        α: the level of significance (0.05 as the default);
Procedure SCRIPT (N, α, B_t)
/* Initialization */
 1.  t = 0;
 2.  Build the original decision tree DT_0 by B_0;
 3.  Record the set of split attributes of DT_0 in Splitatt [];
 4.  Count the CDAVs in B_0 and record them in RCDAV[];
/* Detect the concept drift and correct the original decision tree */
 5.  t = t + 1
 6.  For the new coming data block B_t in time step t
 7.    For each instance (a_i, c_k) in B_t in S
 8.      For each value a_ij of each attribute a_i ∈ A
 9.        For each class c_k
10.          Count CDAV_kjk in RCDAV[];
11.  For each CDV_ij in the new data block
12.    If | CDAV_{t-1→t}(i, j)| > ε
13.      Record this CDAV in DCDAV[];
14.    End if
15.  If DCDAV[] is not empty
16.    Aggregate the recorded CDAVs;
17.    For all aggregated CDAVs
18.      If two aggregated CDAVs are in the same path in the original decision tree
19.        Reserve the one locates in the highest level in ACDAV[];
20.      End if
21.  Update ;
22.  For all aggregated CDAVs belonged to attribute a_i in ACDAV[]
23.    If attribute a_i is not in Splitatt []
24.      Build an alternative tree rooted at this node by using B_t;
25.    Elseif attribute a_i is in Splitatt [] and all aggregated CDAVs in a_i group in an
       interval a_ij
26.      Remove the subtree rooted at this attribute value a_ij from DT_{t-1};
27.      Use the new data block B_t to build the alternative tree rooted at a_ij;
28.    Else
29.      Remove the subtree rooted at this attribute a_i from DT_{t-1};
30.      Use the new data block B_t to build the alternative tree rooted at a_j;
31.    End if
32.  Update Splitatt [] , RCDAV[], and the recorded decision tree;
33.  End If
34.  Return.
```

Fig. 4. The pseudo code of SCRIPT algorithm.

3.5. *The Comparison of System Cost Among SCRIPT, DNW, and CVFDT*

In this section, we compare the system cost of SCRIPT to that of two state-of-the-art window-based approaches: DNW and CVFDT. Assumed a data block has i attributes, k target classes, each attribute has j attribute values, since SCRIPT records the referred

CDAVs, it has a small memory cost $O(ijk)$. SCRIPT also needs to record a decision tree and the split attributes in this tree, however, the memory cost is $O(n)$ and can be ignored, where n is the number of nodes of the recorded decision tree. For CVFDT, since it has to record the counting in each node of the recorded decision tree, the memory cost needed will be $O(nijk)$. DNW, which is a sliding window approach, might have a worst record cost since it record instances instead counting and new data blocks might have to be mixed into old ones. If the maintained data is w times as much as that of a new data block, then the needed memory cost will be $O(wNi)$, where N is the number of instances in the block .

In the aspect of computational cost, while the concept is stable, SCRIPT only needs to carry out $i \times j$ times of comparisons to see if there are any drifting CDAVs. Therefore, the required computational cost of $O(ij)$. If we use CVFDT, we have to check the information gain for each attribute in each node of decision tree when a new data block is given. If the tree has n nodes, the computational cost needed would be $O(nijk)$ (Klinkenberg and Renz, 1998). For DNW, the computational cost would be $O(nwNi)$ since the tree is rebuilt from scratch. When the concept drifts, DNW and CVFDT have a similar computational cost to that in stable stream. Since SCRIPT directly correcting some sub-trees by checking the drifting CDAVs, the computational cost of the rebuilding is $O(ij) + O(n'ijk)$, where $n' \leqslant n$ and $O(ij)$ is responsible for the comparison of CDAVs and $O(n'ijk)$ is the computational cost for the rebuilding of sub-trees. Comparisons of system cost among SCRIPT, DNW, and CVFDT in stable and drifting data stream are summarized in Table 4. In summary, SCRIPT has the smallest memory requirement and computational cost when concept is stable. When concept drifts, SCRIPT still requires the smallest memory cost and a better or comparable computational cost.

4. Empirical Analysis

In this section, two the-state-of-art data stream mining algorithms, DNW and CVFDT, are implemented to compare with our SCRIPT. We run all experiments on a PC equipped with Windows XP professional operating system, Pentium III 1GHz CPU and 512mb Sdram memory. For the preset of parameters in DNW, we refer to (Klinkenberg and Renz, 1998) and set $\alpha = 5.0, \beta = 0.25$, and $\gamma = 0.50$.

Table 4
The Comparisons of system cost among SCRIPT, DNW, and CVFDT

Algorithm	Concept Stable		Concept Drift	
	Memory Cost	Computational Cost	Memory Cost	Computational Cost
DNW	$O(wNi)$	$O(nwNi)$	$O(wNi)$	$O(nwNi)$
CVFDT	$O(nijk)$	$O(nijk)$	$O(nijk)$	$O(nijk)$
SCRIPT	$O(ijk)$	$O(ij)$	$O(ijk)$	$O(n'ijk)$

4.1. *Experimental Datasets*

We select three datasets from UCI databases in which the number of instances is larger than 4000 as our experimental datasets. UCI database (UCI Repository of Machining Learning Database) is a repository of several kinds of datasets and these datasets are widely used by the machine learning community for the empirical analysis. The summary of the four real datasets is shown in Table 5. To simulate a data stream, for each UCI dataset we first divide it into three data blocks B_0, B_1, and B_2 to simulate a stable data stream. Then, we code a program to generate consecutively data blocks contain drifting concepts by modifying B_2. As described in Introduction and Section 2, one purpose of SCRIPT is to detect the drifting concepts more sensitively to make it suitable for the applications in which a small part of drifting concepts would cause large damage. To evaluate the sensitiveness about the detection of concept drift, we set the drifting ratio as 1%; that is, there are $N \times (t-2)\%$ drifting instances in the data block B_t in time step t ($t \geqslant 2$), where N is the number of instances in B_0. This program works as follows. First, it randomly picks up one instance S in data block B_t and randomly selects attributes a_m ($1 \leqslant m \leqslant 10$) for reference. The target classes of instances, which have the same values to that of S in all attributes a_m, are then changed. We limit the number of referable attributes less than 10 since drifting concepts should be caused by some but not a lot attribute values. For example, age and salary may influence the application of credit cards but weight and height will not; IP address and the number of sending packages may be the main basis to find a PC which sends virus package; fraudulent credit card transactions can be detected by the payment amount and location. If the number of drifting instances is less than the requirement, the program goes on next loop to get more drifting instances. On the contrary, if there are more instances satisfy the requirement, $N\%$ instances are randomly picked up as drifting ones. Consequently, for each UCI dataset, we generate 13 data blocks ($B_0 \sim B_{13}$) and each data block is divided into 10 parts of which nine parts are used as training dataset to calculate the execution time and the remaining one as the testing dataset to count the accuracy. Each generated data stream would correspond to our assumption described in Introduction and Section 3.3 that drifting instances should gather in some specific areas in the dimensional space of attributes, otherwise they can be regarded as noise. Finally, to get an objective result, all experiments were repeated 100 times to obtain the average.

Table 5

The summary of three experimental UCI datasets

Dataset	Number of instances	Number of attributes	Number of classes
satimage	6435	36	6
thy	7200	21	3
spambase	4601	57	2

4.2. *The Comparison of Accuracy*

Figs. 5–7 illustrates the comparison of accuracy among DNW, CVFDT, and SCRIPT on t. In Figs. 5–7, we can find that SCRIPT has a similar accuracy to that of DNW and CVFDT in stable data blocks B_0 to B_2. It follows the Proposition mentioned in Section 3.2 that when concept is stable, the accuracy of the SCRIPT is similar to that of classifier which is rebuilt. When there are drifting concepts from data blocks B_3 to B_{12}, we can find SCRIPT is much more sensitive than DNW and CVFDT. That is, SCRIPT always keeps a high accuracy but DNW and CVFDT can detect the changes until the number of drifting instances reaches a threshold to cause obvious difference in accuracy or information gain. Take the satimage dataset in Fig. 5 for example, DNW averagely recognizes the drifting concepts in blocks B_6, B_7 and B_{12} and therefore reduce the window size to build an up-

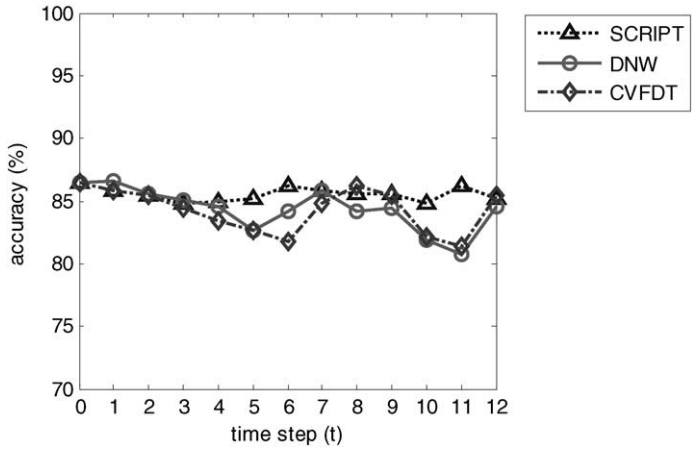

Fig. 5. The comparison of accuracy on satimage.

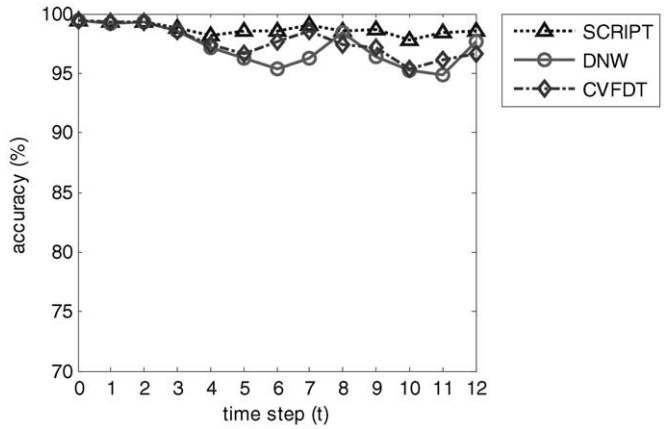

Fig. 6. The comparison of accuracy on thy.

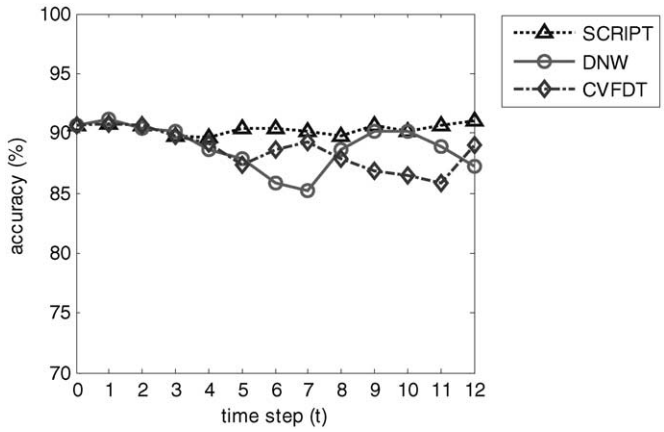

Fig. 7. The comparison of accuracy on spambase.

to-date decision tree; CVFDT averagely uses the alternate subtree to correct the original classification model to raise the accuracy in time step 7, 8 and 12. Similar conditions occur in thy and spambase dataset: in thy, DNW usually discovers the changes in time step 7, 8, and 12, and in time step 8 and 9 in spambase; CVFDT averagely recognizes the drifting concepts in time step 6, 7, and 11in thy and spambase dataset.

4.3. *The Comparison of Execution Time*

In the aspect of comparison of execution time in Figs. 8–10, SCRIPT, DNW, and CVFDT have a similar execution time for the building of the first decision tree in data block B_0. However, SCRIPT and CVFDT requires a little more execution time in the initial step since SCRIPT needs to calculate the CDAVs and CVFDT must record the counts in each node. After the beginning step, SCRIPT is much more efficient than DNW and CVFDT

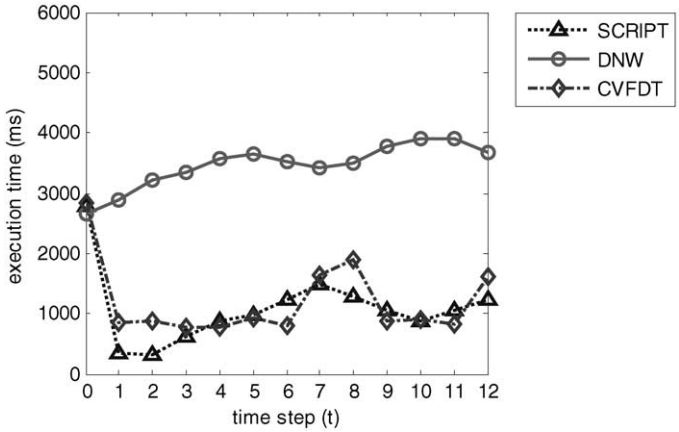

Fig. 8. The comparison of execution time on satimage.

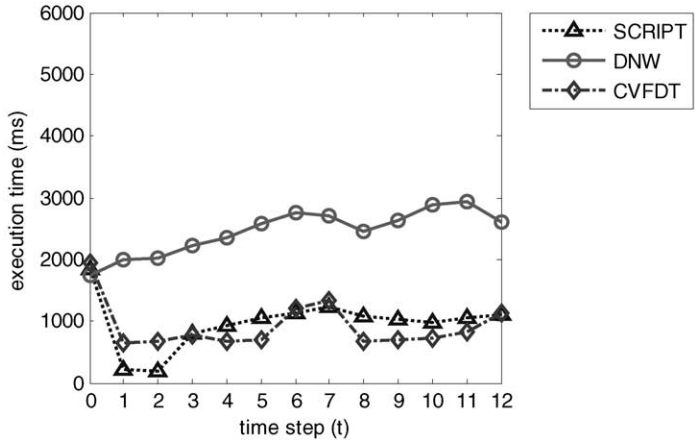

Fig. 9. The comparison of execution time on thy.

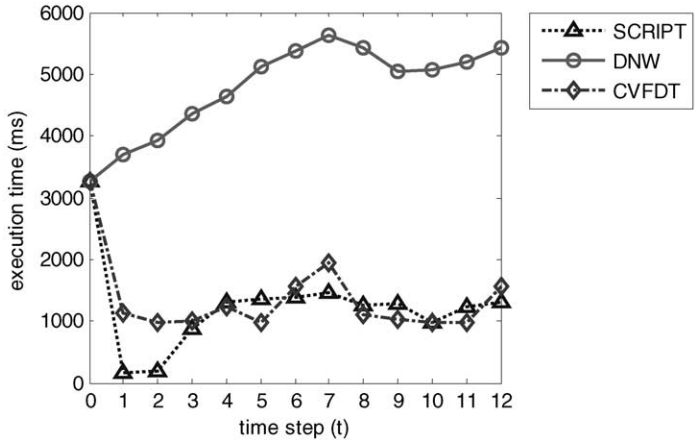

Fig. 10. The comparison of execution time on spambase.

when concept is stable in time step 1 and 2. When concept drifts, the execution time of SCRIPT is worse than that of CVFDT in most cases. However, this is caused by the fact that SCRIPT recognizes the drifting concepts and therefore needs more execution time to correct the original decision tree in these time steps. It is worth to note that when both SCRIPT and CVFDT detect the drifting instances and correct the decision tree, e.g., in time step 7, 8 and 12 in Fig. 8, SCRIPT is more efficient than CVFDT. The reason is SCRIPT can immediately know which sub-trees should be amended by checking the drifting CDAVs but CVFDT have to check the variation of information gain node by node from the root. Similar condition can be found in time step 6 and 7 in thy and spambase dataset. DNW, which builds a new classifier in each time step, always has the worst computational cost. The computational is much worse as time goes by since DNW does not recognize the concept drift and therefore mixes the new data block into the old one.

4.4. *The Analysis of Significance Level* α

Finally, in this section, we use the three drifting datasets generated in Section 4.1 to evaluate our SCRIPT on different levels of significance. Due to the content limit and the similar results on three datasets, we only illustrate the result of satimage in Fig. 11 and 12. As inferred in Section 3.4, SCRIPT will be more sensitive to the concept drift but should require more computational cost if we use a larger significant level α; on the contrary, SCRIPT will be more tolerant to the noise data with a smaller α. The experimental results in the three datasets correspond to our inference. In Fig. 11, SCRIPT with a higher significance level (0.1 in this example) can on average reach a higher accuracy. The reason is that the hypothesis which assumes the class distribution in two data block

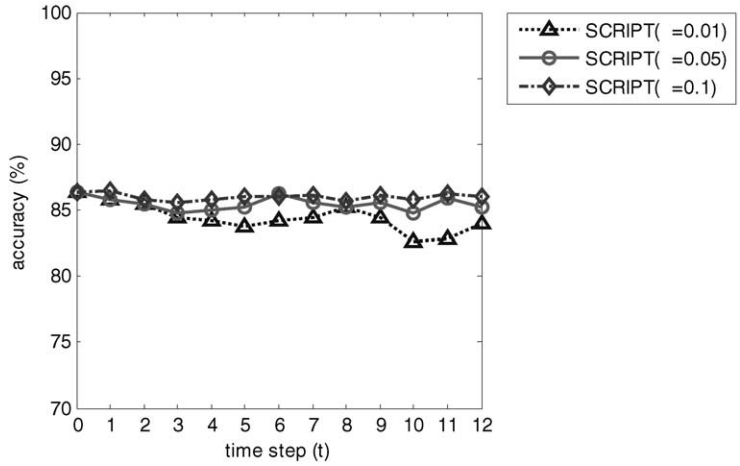

Fig. 11. The comparison of accuracy of SCRIPT with different levels of significance on satimage.

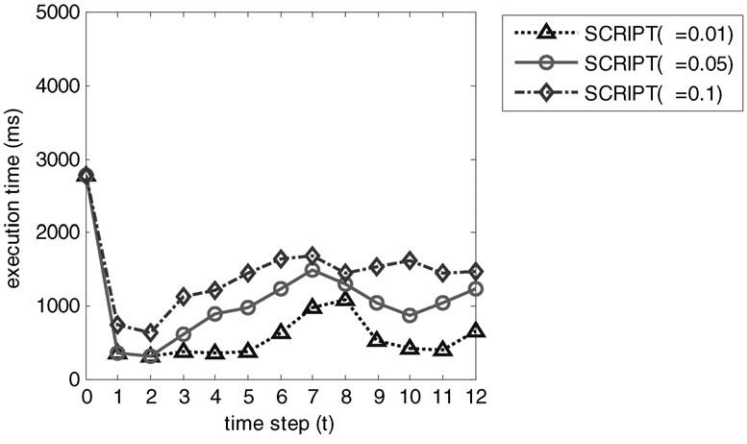

Fig. 12. The comparison of execution time of SCRIPT with different levels of significance on satimage.

is identical will be rejected more easily when the significance level is larger. Therefore, a small number of drifting instances will make SCRIPT to correct the original decision tree. Relatively, as illustrated in Fig. 12, SCRIPT with a higher significance level averagely needs more execution time to implement the correct mechanism. It is also worth to note that in the stable data blocks B_1 and B_2, SCRIPT with significant level 0.01 still correct the decision tree constructed by B_0. This is occurred by the fact that we randomly split the original dataset into three subsets to simulate a stable data stream and the three data blocks might contain some inconsistency caused by the random sampling.

5. Conclusions

Mining drifting concepts on data stream has become a novel research topic of growing interest in knowledge discovery. However, while the concept is stationary, the proposed methods spend a lot unnecessary system cost, including computational cost to rebuild the decision tree or storage cost to record similar data blocks. Moreover, they generally are not sensitive enough to the concept drift problem. In this paper, we propose a Sensitive Concept Drift Probing Decision Tree algorithm, called SCRIPT, to handle this problem. The Proposition in Section 3.2 verifies the validity for SCRIPT to determine if concept drift exists between two data blocks. The experiments in Section 4 also show that SCRIPT can sensitively, accurately, and efficiently handle the drifting concepts on data stream.

Acknowledgments

This research was partially supported by the National Science Council of Republic of China under Grant No. NSC 96-2520-S-024-002.

References

Agrawal, R., S. Ghosh, T. Imielinski, B. Iyer and A. Swami (1992). An interval classifier for database mining applications. In *Proceedings of the 18th International Conference on Very Large Databases*. pp. 560–573.

Blum, A. (1997). Empirical support for winnow and weighted-majority algorithms: results on a calendar scheduling domain. *Machine Learning*, **26**, 5–23.

Cohen, W. (1996). Learning rules that classify e-mail. In *Proceedings of the AAAI Spring Symposium on Machine Learning in Information Access*. Menlo Park, CA. AAAI Press, Technical Report SS96-05. pp. 18–25.

Chawla, N.V., L.O. Hall, K.W. Bowyer, T.E. Moore and W.P. Kegelmeyer (2002). Distributed pasting of small Votes. In *Multiple Classifier Systems*. pp. 52–61.

Domingos, P., and G. Hulten (2000). Mining high-speed data streams. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*. Boston, MA. ACM Press. pp. 71–80.

Fan, W. (2004). Systematic data selection to mine concept-drifting data streams. In *Proceedings of the10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 128–137.

Gehrke, J., R. Ramakrishnan and V. Ganti (2000). RainForest: a framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery*, **4**(2/3), 127–162.

Han, J., and M. Kamber (2001). *Data Mining*: *Concepts and Techniques*. Morgan Kaufmann Publisher.

Harries, M.B., C. Sammut and K. Horn (1998). Extracting hidden context. *Machine Learning*, **32**(2), 101–126.

Hsieh, C.H. (2004). *An Efficient Approach for Mining Concept-Drifting Data Streams*. Master thesis. Institute of Information Education, National University of Tainan, Taiwan, R.O.C.

Hulten, G.., L. Spencer and P. Ddmingos (2001). Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco. pp. 97–106.

Jegelevičius, D., A. Lukoševičius, A. Paunksnis and V. Barzdziukas (2002). Application of data mining technique for diagnosis of posterior uveal melanoma. *Informatica*, **13**(4), 455–464.

Jin, R., and G. Agrawa (2003). Efficient decision tree construction on streaming data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC. pp. 571–576.

Japkowicz, N., and S. Stephen (2002). The class imbalance problem: a systematic study. *Intelligent Data Analysis*, **6**(5), 429–450.

Koychev, I. (2000). Gradual forgetting for adaptation to concept drift. In *Proceedings of ECAI 2000 Workshop in Spatio-Temporal Reasoning*. Berlin, Germany.

Kifer, D., S. Ben-David and J. Gehrke (2004). Detecting change in data streams. In *Proceedings of the 30th International Conference on Very Large Databases*. Toronto, Canada. pp. 180–191.

Kolter, J.Z., and M.A. Maloof (2003). Dynamic weighted majority: a new ensemble method for tracking concept drift. In *Proceedings of the 3rd International IEEE Conference on Data Mining*. Melbourne, FL. pp. 123–130.

Kuh, A., T. Petsche and R. L. Rivest (1991). Learning time-varying concepts. In *In Advances in Neural Information Processing Systems 3*, vol. 3. Morgan Kaufmann, San Francisco, CA. pp. 183–189.

Klinkenberg, R., and I. Renz (1998). Adaptive information filtering: learning in the presence of concept drifts. In *Proceedings of International Conference on Machine Learning*. Menlo Park, California. pp. 33–40.

Maloof, M.A., and R.S. Michalski (2000). Selecting examples for partial memory learning. *Machine Learning*, **41**(1), 27–52.

Maloof, M. (2003). Incremental rule learning with partial instance memory for changing concepts. In *Proceedings of the International Joint Conference on Neural Networks*. IEEE Press, Los Alamitos, CA.

Mehta, M., R. Agrawal and J. Rissanen (1996). SLIQ: a fast scalable classifier for data mining. In *Proceedings of the 5th International Conference on Extending Database Technology*. *Lecture Notes in Computer Science*. Springer-Verlag. pp. 18–32.

Misevicius, A. (2006). Experiments with hybrid genetic algorithm for the grey pattern problem. *Informatica*, **17**(2), 237–258.

Quinlan, J.R. (1993). *C4.5*: *Program for Machine Learning*. Morgen Kaufmann Publisher, San Mateo, CA.

Rastogi, R., and K. Shim (1998). PUBLIC: a decision tree classifier that integrates building and pruning. In *Proceedings of the 24th International Conference on Very Large Databases*. pp. 404–415.

emeikis, N., I. Skučas, V. Melninkaitė (2004).Text categorization using neural networks initialized with decision trees. *Informatica*, **15**(4), 551–564.

Shafer, J.C., R. Agrawal and M. Mehta (1996). SPRINT: a scalable parallel classifier for data mining. In *Proceedings of the 22th International Conference on Very Large Databases*. pp. 544–555.

Schlimmer, J.C., and D.H. Fisher (1986). A case study of incremental concept induction. In *Proceedings of the 5th International Conference on Artificial Intelligence*. Philadelphia, PA. pp. 496–501.

Schlimmer, J., and R. Granger (1986). Beyond incremental processing: tracking concept drift. In *Proceedings of 5th National Conference on Artificial Intelligence*. Philadelphia, PA. pp. 502–507.

Street, W., and Y. Kim (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of 7th International Conference on Knowledge Discovery and Data Mining*. New York. pp. 377–382.

Utgoff, P.E. (1989). Incremental induction of decision trees. *Machine Learning*, **4**(2), 161–186.

Wang, H., W. Fan, P.S. Yu and J. Han (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC. pp. 226–235.

Widmer, G., and M. Kubat (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, **23**(1), 69–101.

C.-J. Tsai was born in Tainan, Taiwan, Republic of China, 1973. He received the BS degree in mathematics and science education from National Ping Tung University of Education in 1995 and M.S. degrees in information education from National University of Tainan in 2000. Currently, he is a PhD student in the Department of Computer Science in National Chiao Tung University, Hsinchu, Taiwan, Republic of China. His current research interests include data mining, database theory and application, information retrieval, e-learning, and information system.

C.-I Lee was born in Taipei, Taiwan, Republic of China, 1965. He received the BS degree in computer science from Feng Chia University in 1987 and MS degree in applied mathematic from National Sun Yat-Sen University in 1993. He received the PhD degree in computer science from National Chiao Tung University in June 1997 and then joined the Graduate Institute of Computer Science and Information Education, National University of Tainan, Tainan, Taiwan, Republic of China. He is currently an associate professor and his research interests include data mining, e-learning, multimedia databases and information retrieval.

W.-P. Yan was born on May 17, 1950 in Hualien, Taiwan, Republic of China. He received the BS degree in mathematics from National Taiwan Normal University in 1974, and the MS and PhD degrees from the National Chiao Tung University in 1979 and 1984, respectively, both in computer engineering. Since August 1979, he has been on the faculty of the Department of Computer Science and Information Engineering at National Chiao Tung University, Hsinchu, Taiwan. In the academic year 1985–1986, he was awarded the National Postdoctoral Research Fellowship and was a visiting scholar at Harvard University. From 1986 to 1987, he was the director of the Computer Center of National Chiao Tung University. In August 1988, he joined the Department of Computer and Information Science at National Chiao Tung University, and acted as the head of the Department for one year. Then he went to IBM Almaden Research Center in San Jose, California for another one year as visiting scientist. From 1990 to 1992, he was the head of the Department of Computer and Information Science again. He was the visiting scholar of the University of Washington in Seattle for one year in 1996–97. He was the director of University Library from 1988–2004. Since 2004 he has transferred to National Dong Hwa University at Hualien and acted as the head of the Department of Information Management. Now he is the dean of School of Management. His research interests include database theory and application, information retrieval, data miming, digital library, and digital museum. Dr. Yang is also a senior member of IEEE, and a member of ACM.

Efektyvus ir patikimas sprendimų medžio metodas kintamos koncepcijos duomenų gavybai

Cheng-Jung TSAI, Chien-I LEE, Wei-Pang YANG

Duomenų srautų duomenų gavyba yra žinių technologija, kuriai pastaruoju metu skiriama nemažai dėmesio. Dauguma duomenų srautų tyrimo algoritmų remiasi prielaida, kad analizuojami informacijos blokai yra atsitiktiniai, o jų skirstiniai stacionarūs. Tačiau daugelio duomenų bazių atveju ši prielaida nepasitvirtina. Tai, kad laikui bėgant kai kurių klasių įrašų turinys gali kisti, yra apibūdinama koncepcijos kitimo sąvoka. Šiame darbe mes siūlome patikimą koncepcijos kitimo zondavimo sprendimų medžio (Sensitive Concept Drift Probing Decision Tree (SCRIPT)) algoritmą, pagrįstą statistinio X2 testo taikymu koncepcijos kitimui nagrinėti. Palyginus su žinomais metodais SCRIPT pasižymi tokiais privalumais: a) padeda išvengti kaštų, susijusių su stacionariais duomenų srautais; b) padeda greitai ir efektyviai koreguoti pradinius klasifikatorius, kai duomenų srautai yra nestabilūs; c) yra ypač efektyvi tais atvejais, kai reikalinga patikimai įvertinti koncepcijos pokyčius.