International Journal of Innovative Computing, Information and Control
Volume **8**, Number **7(B)**, July **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 4995–5010

ADAPTIVE DATA REUSE FOR CLASSIFYING IMBALANCED AND CONCEPT-DRIFTING DATA STREAMS

Hien M. Nguyen[1], Eric W. Cooper[2] and Katsuari Kamei[2]

[1]Graduate School of Science and Engineering
[2]College of Information Science and Engineering
Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577, Japan
hiennm@spice.ci.ritsumei.ac.jp; cooper@is.ritsumei.ac.jp; kamei@ci.ritsumei.ac.jp

Received May 2011; revised September 2011

Abstract. *Mining data streams has recently been the subject of extensive research efforts. However, most of the works conducted in this field assume a balanced class distribution underlying data streams. In this paper, therefore, we propose a new method for learning from imbalanced data streams. To deal with the problem of class imbalance, we select and reuse past data to improve the representation of the minority class. Different from previous methods, our method has the ability to automatically adapt data selection for concept drift. A data stream may experience a complicated concept drift, making data selection more difficult. Therefore, we consider several different candidate solutions of data selection, each of which is possibly more appropriate for certain data streaming conditions. In other words, no one of them is the best at all times. We make comparisons and identify the best candidate solution by cross-validation on the most recent training data. By experimental evaluations on simulated and real-world data streams, we show that our method achieves better performance than previous methods, especially when concept drift occurs.*

Keywords: Adaptive data reuse, Data selection, Class imbalance, Concept drift, Data stream, Ensemble learning

1. **Introduction.** Mining data streams has recently been the subject of extensive research efforts [1]. Some examples include detection of fraudulent credit card transactions [2] and web filtering [3]. Basically, there are two approaches to learning from data streams. The first is incremental learning in which a single model is built from an initial training set, and then is continuously updated with new training instances upon their arrival [4]. The second is ensemble learning in which a series of base models is trained on consecutive chunks of a training data stream, and forms an ensemble to classify previously unseen instances [5].

One of the biggest challenges facing data stream learning is to deal with "concept drift", i.e., the concept to learn is changing over time. For example, in a task of online news filtering, some user might expand his/her reading interests with a new topic such as "nuclear safety" after the Fukushima nuclear accident occurred in Japan. In this example, the concept to learn (or target concept) is the reading interests of a specific user. Concept drift could cause a classification model to perform poorly if this model is not updated appropriately to reflect changing data distributions. To deal with concept drift, one solution is to regularly retrain the classification model on a sliding window covering only the most recent training instances [6]. Another solution is to use ensemble learning in which the base models are weighted using their classification accuracy on the newest

training data chunk [2]. Both of these solutions are based on an implicit assumption that new data best reflects the distribution of forthcoming unseen instances.

While most of the works conducted in data stream learning assume a balanced class distribution underlying data streams, many real-world applications deal with the problem of class imbalance, in which one class (minority class) is heavily outnumbered by another (majority class). For example, in the task of online news filtering, those articles interesting a specific user usually account for a very small portion of the total articles arriving online. Another example is the detection and removal of undesirable posts from a stream of online discussions in social networks. In many cases, those undesirable posts, including off-topic posts, personal insults, and rude language, are very much fewer than normal ones. The problem of class imbalance could result in degradation of conventional learning methods in classifying the minority class, which is usually of more interest. The reason is that the minority class, with a small size, has a less significant impact on overall accuracy of a learned model, and therefore, this class may largely be ignored during a learning process.

To deal with the problem of class imbalance in data streams, we can consider reuse of past data to improve the representation of the minority class. As reflected by its name, the instances of the minority class may occur very sparsely in a data stream. To collect a certain amount of past data of this class, we may have to accept some data that is very old, and with a high possibility of concept drift. Therefore, data reuse could be more challenging in imbalanced data streams. However, existing methods have no effective mechanism to counter concept drift in performing data selection. For example, one method makes reuse of all past data available, while another uses a user-defined class imbalance ratio as an input to decide how many past instances should be selected. Clearly, such methods are not capable of adapting to concept drift. They could only improve performance in cases where the data stream is not significantly concept-drifting.

In this paper, we propose a new method for classifying imbalanced data streams by selecting past data consistent with a drifting target concept, in an effort to reduce the problem of class imbalance. The main advantage of our method is its ability to automatically adapt data selection for concept drift, while previous methods do not. A data stream may experience a complicated concept drift, making data selection more difficult. Therefore, we consider several different candidate solutions of data selection, each of which is possibly more appropriate for certain data streaming conditions. In other words, no one of them is the best at all times. We make comparisons and identify the best candidate solution by cross-validation on the most recent training data. By experimental evaluations on simulated and real-world data streams, we show that our method achieves better performance than previous methods, especially when concept drift occurs.

Because real-world data streams are possibly arriving at high speed and in large quantities, data stream learning imposes high requirements of reducing consumption of system resources, such as time and memory. To improve efficiency in learning and make our method more practical for applications, we simplify aspects of our method, such as keeping of only a limited quantity of past training instances in memory, and use of modest settings for various parameters.

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, we first introduce some assumptions of concept drift imposed to develop our method. Then, we analyze different candidate solutions of data reuse for improving the representation of the minority class. We summarize our proposed method at the end of this section. The description of experiments and results are presented in Sections 4 and 5, respectively. Section 6 concludes the paper.

2. **Related Work.** The problem of class imbalance has been solved with two main approaches. First, existing learning algorithms can be revised to learn the minority class instances more accurately. One way is to assign a higher cost to the minority class instances so that it is more difficult for the learning algorithm to ignore them when making a decision [7]. Second, sampling techniques can be applied to balance the training set [8, 9]. This approach is perhaps the most popular solution for dealing with class imbalance. Indeed, we also focus on sampling methods in this paper.

There so far has been very little work conducted on data streams with imbalanced class distributions. Some studies approach the problem of class imbalance for each data chunk isolatedly. In other words, each data chunk of a stream is treated as a conventional imbalanced data set, and thus can be handled by an existing technique, such as over-sampling by generating synthetic minority class instances [10] or under-sampling by clustering the majority class [11]. One drawback of such methods is that past data is not taken into account for improving the representation of the minority class.

Other methods have been proposed to exploit past data. Gao et al. [12] proposed a method in which all minority class instances from previous data chunks are incorporated into the current data chunk. This method is problematic because past data may become out-of-date due to concept drift. Spyromitros et al. [13] used two separate sliding windows, one for the minority class and the other for the majority class. The window sizes are determined so as to obtain a certain class distribution ratio, such as 40:60. This approach is essentially similar to Gao's method.

Another method, proposed in [14], selects a number of past minority class instances which are top-ranked, based on the number of minority class nearest neighbors in the current data chunk. We make two remarks on this method. First, because the current chunk is imbalanced, most nearest neighbors would possibly come from the majority class, and fewer from the minority class. In other words, past instances would share a small pool of rankings, which may make ranking of them ineffective. Second, the number of past instances selected depends on a desired degree of class imbalance given by a user, but not the concept drift in the data stream. Therefore, this method should not be seen as an adaptive method for concept-drifting data streams.

3. **Proposed Method for Imbalanced and Concept-Drifting Data Streams.** The proposed method builds and retrains a classification model for every time a new training chunk is received from the data stream. To deal with class imbalance, we consider reuse of past data for improving the representation of the minority class. Different from previous methods, our method has the ability to automatically adapt data selection for concept drift. Basically, we achieve this ability by approximating the target concept with a "balanced" model on the current chunk, which is trained after an under-sampling to balance the training set. This model, therefore, allows us to more reliably select past data which is consistent with the target concept. We rely on the classification of that model to decide which past instances are to be selected, instead of an input given by a user, such as a desired class imbalance ratio, which is itself not related to concept drift. To further improve performance of our method, we also consider additional solutions of data reuse for cases where the data stream is relatively stable or unstable.

In the following subsections, we first introduce some reasonable assumptions of concept drift under which the proposed method is developed. Then, we analyze problems of data reuse in the presence of concept drift, through which we give a specific design of our method for imbalanced and concept-drifting data streams.

3.1. **Assumptions of concept drift.** Concept drift is a phenomenon in which the concept to learn, or the data distribution which generates a data stream, is changing over time. If this change is progressing arbitrarily, we may not be able to design a good algorithm for learning data streams. Therefore, we should impose some reasonable assumptions of concept drift under which effective stream learning methods can be developed.

A data distribution can be represented by a joint probability, $P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(y|\boldsymbol{x})$, where $\boldsymbol{x}$ is an instance and y is the corresponding class label. Three different types of concept drift can be drawn from this expression [15]: (1) *feature drift*, i.e., the change of the probability $P(\boldsymbol{x})$ to observe an instance with feature vector $\boldsymbol{x}$, (2) *label drift*, i.e., the change of the conditional probability $P(y|\boldsymbol{x})$ to assign class label y to instance $\boldsymbol{x}$, and (3) *dual drift*, including both feature drift and label drift. The second type of concept drift is more difficult to learn than the first because the class label of instances may change to another, resulting in noise in training data. In reality, we usually cannot collect sufficient training data and the distribution of the sampled training set may differ from the underlying true distribution. Therefore, the feature drift seems to occur all the time, even when there is no actual change in $P(\boldsymbol{x})$. Consequently, it is more often to encounter data streams with concept drift of the first and third types.

A reasonable assumption of concept drift, as suggested in [15], is that the change of data distribution should be progressing to the extent that a model built on the training set makes sense, i.e., it has higher performance on forthcoming unknown data than random guess of class labels. This assumption can be understood as asserting that concept drift should be gradual rather than sudden. A further assumption is that the most recent training data best reflects the distribution of forthcoming unseen instances. This assumption is also reasonable because those data collected in periods close together seem to have more similar distribution.

3.2. **Concept drift and data reuse.** We consider three candidate solutions of data reuse, each of which is more appropriate for a certain situation of concept drift in data streams. The purpose of data reuse is to improve the representation of the minority class in an imbalanced training set. The best candidate solution, which changes from time to time, is identified by application of the k-fold cross-validation on the current (newest) data chunk.

3.2.1. *First candidate solution of data reuse: Train an ensemble model on the current data chunk, then use it to select past data.* We want to select some past data which is still consistent with the target concept. However, the target concept is unknown, and therefore, we attempt to build a "good" model from the current data chunk only in order to approximate this concept. Past instances of the minority class that are correctly classified by this model will be selected. Other instances can be considered no longer consistent with the target concept due to concept drift, and will be kept out of the training set. Note that we do not reuse past instances of the majority class because this class is much larger and is better represented than the minority class. Furthermore, the addition of data to the majority class increases the class imbalance, making learning more difficult. Here we suppose the training data of the majority class is already sufficient.

For now, we need to train a "good" model on the current data chunk to approximate the target concept. The current chunk D can be considered a conventional imbalanced data set. Therefore, random under-sampling can be applied to the majority class of this set to create a balanced training subset. However, under-sampling may lead to loss of useful information. To allow use of more data of the majority class, and thus reducing information loss, we randomly draw without replacement m subsets S_i from the majority class N, which have the same size as the minority class P, i.e., $|S_i| = |P|$. As a result,

TABLE 1. Ensemble training procedure for an imbalanced training set

Function: *TrainEnsemble*(D)
Input: Imbalanced training set $D = P \cup N$, where P is the minority class, and N is the majority class
Output: Ensemble $E = \{f_i | i = 1, \ldots, m\}$, where f_i are base models
begin
  $E \leftarrow \varnothing$
  for $i \leftarrow 1$ **to** m **do**
    $S_i \leftarrow$ *DrawSubset*(N, $|P|$) /* draw $|P|$ instances from N */
    $T_i \leftarrow P \cup S_i$
    $f_i \leftarrow$ *TrainBaseModel*(T_i)
    $E \leftarrow E \cup \{f_i\}$
  end for
  return E
end

there are m balanced training subsets, $T_i = P \cup S_i$, from which m base models f_i are built. All these base models are combined to form an ensemble E that classifies unknown instances by averaging component outputs, as in Equation (1):

$$E(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} f_i(\boldsymbol{x}) \tag{1}$$

In the literature of imbalanced data learning, random under-sampling has been shown to be a simple but effective method for dealing with class imbalance [16]. An ensemble learning approach combining multiple under-samplings was also investigated by Liu et al. [17]. Table 1 presents our implementation of an ensemble training procedure on an imbalanced training set. Note that if the majority class is not greater than m times the minority class, the actual number of majority class subsets S_i, and thus base models f_i, is less than m. However, this case is not explicitly shown in Table 1 for brevity.

Up to this point, one may wonder if data reuse in the first candidate solution always helps improve performance. The answer is "it depends". The ensemble E built on the current chunk to approximate the target concept may be reasonably good but not perfect for certain, and in fact, we are trying to improve it. Therefore, among those past instances selected by this model, some are consistent with the target concept, while others are not. Suppose consistent instances help increase performance by an amount δ_{con}, while inconsistent ones decrease performance by an amount δ_{inc}. We expect in most cases that $\delta_{con} > \delta_{inc}$, and therefore, data reuse helps improve performance. However, under disadvantageous conditions that may happen (such as stronger concept drift, higher class imbalance and overlap, and largely insufficient training data), the ensemble E may be not sufficiently accurate for data selection, in which case it may happen that $\delta_{con} < \delta_{inc}$, i.e., data reuse does not help.

3.2.2. *Second candidate solution of data reuse: Reuse all past data kept in a pool.* This solution may provide better results in periods when the data stream is stable without or with insignificant concept drift. It allows reuse of more past data that may be ignored by an "imperfect" ensemble model E in the first candidate solution.

3.2.3. *Third candidate solution of data reuse: Reuse no past data.* This solution may be good if the data stream is undergoing a period of stronger concept drift, which makes past data have a high risk of inconsistency with the target concept. Not reusing past data

may also be the solution for the situations in which the ensemble model E in the first candidate solution is not sufficiently accurate for data selection.

3.2.4. *Cross-validation for finding the best candidate solution of data reuse.* We have so far presented three candidate solutions of data reuse for improving the representation of the minority class. The best candidate solution of data reuse depends on the degree of concept drift as well as other data properties in different periods. We use the k-fold cross-validation to find out which candidate solution is the best. Specifically, the current data chunk D is randomly split into k equal-size subsets $S_1, S_2, \ldots, S_k$. The cross-validation includes k iterations, each of which takes a different subset S_i as a validation set, $V_i = S_i$, and combines all remaining ones into a training set, $T_i = \bigcup_{j|j\neq i} S_j$. For the j-th candidate solution ($j = 1, 2, 3$), we combine the corresponding set Z_j of past minority class instances selected with T_i to form a new training set, $T_{ij} = T_i \cup Z_j$, which should have a better representation of the minority class. Remind that Z_1 includes past minority class instances that are correctly classified by an ensemble model built on the current chunk D only, Z_2 includes all past minority class instances kept in a pool, and Z_3 is an empty set. Next, we train an ensemble model, as in Table 1, on each of three sets T_{ij} ($j = 1, 2, 3$), and then use it to classify the validation set V_i. After all iterations of the cross-validation finish, we compute overall classification performances for the three candidate solutions of data reuse. Finally, we select the candidate solution which has the highest cross-validated performance.

3.3. **Proposed method.** Table 2 presents our proposed method for learning from imbalanced and concept-drifting data streams. In Steps 1-3, we consider three candidate solutions of data reuse, as analyzed in Section 3.2, for improving the representation of

TABLE 2. Proposed method for imbalanced and concept-drifting data streams

Function: *TrainAdaptiveEnsemble*(D, Q)
Input: Current chunk $D = P \cup N$, and pool Q containing past minority class instances
Output: Ensemble $E_{adapt} = \{f_i | i = 1, \ldots, m\}$, where f_i are base models
begin
Step 1: First candidate solution of data reuse
$E \leftarrow$ *TrainEnsemble*(D)
$Z_1 \leftarrow$ *Classify*(E, Q)

Step 2: Second candidate solution reuses all past data kept in the pool
$Z_2 \leftarrow Q$

Step 3: Third candidate solution does not reuse past data
$Z_3 \leftarrow \varnothing$

Step 4: Find the optimal solution by cross-validation on the current chunk
$Z_{opt} \leftarrow$ *DoCrossValidation*(D, Z_1, Z_2, Z_3)

Step 5: Add past instances selected to the current chunk, then train an ensemble
$T \leftarrow D \cup Z_{opt}$
$E_{adapt} \leftarrow$ *TrainEnsemble*(T)

Step 6: Update the pool of past minority class instances
$Q \leftarrow Z_{opt} \cup P$
CheckPoolSize(Q)

return E_{adapt}
end

the minority class in the current data chunk. Here $Classify(E, Q)$ means that we use the ensemble E to classify the pool Q of minority class instances, and then return a set of past minority class instances correctly classified. In Step 4, $Z_{opt} \in \{Z_1, Z_2, Z_3\}$ is the past data set selected by the best candidate solution. Although the reuse of past data helps increase the minority class, the class imbalance may still exist in highly imbalanced training sets. Therefore, in Step 5, we continue training of an ensemble model, in which the majority class is under-sampled to create balanced training subsets (See Table 1). This ensemble model is then used to classify unknown instances, which will be arriving in the future. In Step 6, we update the pool Q of minority class instances, by first removing all instances except those selected by the best candidate solution (i.e., those in Z_{opt}), and then adding the minority class instances of the current chunk.

To make our method more efficient in computation as well as reducing storage space, we do not keep all past instances of the minority class in the pool Q. Instead, we set the maximum size of this pool to $|N| - |P|$, where N is the majority class and P is the minority class of the current data chunk. That is, we keep an amount of past data just enough to balance the training set. When the pool is already full, the oldest instances are removed to make room for new ones. $CheckPoolSize(Q)$ in Step 6 exactly performs this task.

4. **Experiments.**

4.1. **Evaluation measure.**

For evaluation of learning methods on imbalanced data, overall accuracy is not a suitable measure because it is dominated by the majority class. For example, if 99% of the data is from the majority class, but only 1% from the minority class, a classifier can classify every data as majority, and thus achieves 99% accuracy at the cost of no minority class instances correctly classified. Therefore, instead of the overall accuracy, we use *G-mean* [18], which is a more suitable evaluation measure, and defined by

$$G\text{-}mean = \sqrt{Acc^+ \times Acc^-}$$

where Acc^+ and Acc^- are the classification accuracies on the minority and majority classes, respectively. The *G-mean* balances accuracies between the two classes without depending on the size of each class. For the example above, we have $G\text{-}mean = \sqrt{Acc^+ \times Acc^-} = \sqrt{0\% \times 100\%} = 0\%$. Therefore, a certain learning algorithm that largely ignores the minority class would receive a low *G-mean*.

Note that for all mentions of classification performance in Section 3, which describes our proposed method, we use the *G-mean* measure.

4.2. **Data sets.**

This section describes data sets for experimental study. We used some UCI data sets to simulate data streams in a predefined scenario of concept drift. We also conducted experiments on the Slashdot data set, which represents a real-world data stream. There may exist a certain concept drift in the Slashdot, but we do not know how concept drift is progressing in reality.

4.2.1. *UCI data sets.*

We used the following six UCI data sets [19] for simulation of data streams:

- *Abalone*: Predict the age of an abalone based on physical measures. We replaced the first feature, which is categorized with three different values (male, female, infant), by three binary features representing presence or absence of each categorical value.
- *Chess*: Predict the outcome of a chess board state. There are six features indicating column and row positions of three chess pieces on the board. We replaced each

TABLE 3. Imbalanced three-class UCI data sets

Data set	Features	Two classes	Class I	Class II	Class III	Class I+II+III	Imbalance ratio
Abalone1	10	6, 7	259	391	3,527	4,177	1:9.7 ∼ 1:15.1
Abalone2	10	6, 12	259	267	3,651	4,177	1:14.6 ∼ 1:15.1
Abalone3	10	7, 12	391	267	3,519	4,177	1:9.7 ∼ 1:14.6
Chess1	48	5, 6	471	592	26,993	28,056	1:46.4 ∼ 1:58.6
Chess2	48	5, 16	471	390	27,195	28,056	1:58.6 ∼ 1:70.9
Chess3	48	6, 16	592	390	27,074	28,056	1:46.4 ∼ 1:70.9
Covtype1	54	5, 6	2,396	2,160	15,444	20,000	1:7.3 ∼ 1:8.3
Covtype2	54	5, 7	2,396	2,160	15,444	20,000	1:7.3 ∼ 1:8.3
Covtype3	54	6, 7	2,160	2,160	15,680	20,000	1:8.3 ∼ 1:8.3
Letter1	16	1, 2	789	766	18,445	20,000	1:24.3 ∼ 1:25.1
Letter2	16	1, 3	789	736	18,475	20,000	1:24.3 ∼ 1:26.2
Letter3	16	2, 3	766	736	18,498	20,000	1:25.1 ∼ 1:26.2
Wall	24	2, 4	826	328	4,302	5,456	1:5.6 ∼ 1:15.6
Wine	11	4, 8	216	193	6,088	6,497	1:29.1 ∼ 1:32.7

of those features by eight binary features corresponding to eight columns or rows, leading to the total 48 binary features.

- *Covtype*: Predict forest cover types based on cartographic measures. The original Covtype data set has 581,012 instances. However, we used only the first 20,000 instances in our experiments.
- *Letter*: Identify upper-case letters from pixel images distorted in different fonts.
- *Wall*: Predict the action of a robot when navigating around a wall, based on signals received from 24 ultrasound sensors on the robot.
- *Wine*: Predict wine quality from different measures. We combined separate data sets for red and white wines to create a larger data set.

For each UCI data set, we created several imbalanced versions, each of which includes two small classes (Class I and Class II) and one large class (Class III). The usage of such imbalanced three-class data sets for simulation of data streams will be explained shortly. We give an example with the Abalone data set, which has 28 different classes. We selected two small classes, 6 and 7, as Class I and Class II respectively, and then combined all the remaining 26 classes into Class III. We name this new imbalanced data set Abalone1. Table 3 shows some information of Abalone1, such as the number of features, two classes selected as Class I and Class II, the size of Classes I, II, and III, and the size of the whole data set. Similarly, we also created imbalanced data sets from other UCI data sets. In total, we have 14 imbalanced data sets, as described in Table 3.

We simulated concept-drifting data streams based on a scenario of online news filtering [6] in which the reading interest of a user is gradually changing from one topic to another. We illustrate this scenario in Figure 1, in which an imbalanced three-class data set in Table 3 is randomly split into 20 equal-size data chunks, and the target concept is gradually changing from Class I to Class II. In this scenario, instances from Classes I and II can belong to the minority or majority class, but those from Class III always belong to the majority class. For example, in Figure 1, the relevant level of Class I at Chunk 9 is 0.8, indicating that 80% of Class I in this chunk have the label of "minority" while the remaining 20% have the label of "majority". Formally, we use the notation $X \equiv \alpha\% \times \text{Class I} + \beta\% \times \text{Class II} + \gamma\% \times \text{Class III}$ to indicate that set X is made of $\alpha\%$ of Class I, $\beta\%$ of Class II, and $\gamma\%$ of Class III. Then, a data chunk D_i $(i = 1, 2, \ldots, 20)$ is made of the minority class $P_i \equiv \alpha_i\% \times \text{Class I} + \beta_i\% \times \text{Class II}$, and the majority class $N_i \equiv (1 - \alpha_i\%) \times \text{Class I} + (1 - \beta_i\%) \times \text{Class II} + 100\% \times \text{Class III}$. In Figure 1, relevant level $\alpha_i\%$ of Class I is gradually decreasing from 1 down to 0, while relevant level

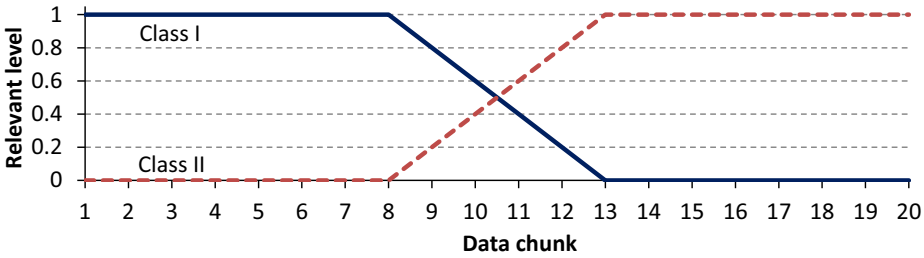

FIGURE 1. Scenario of a gradually concept-drifting data stream

$\beta_i\%$ of Class II is gradually increasing from 0 up to 1. At Chunk 13, the target concept completely changes from Class I to Class II.

We can see that the streaming scenario in Figure 1 includes the first and third types of concept drift, as given in Section 3.1. In the periods of Chunks 1-8 and Chunks 14-20, there is only feature drift. However, in the period of Chunks 9-13, there is dual concept drift which includes both feature drift and label drift.

A final remark in this section is that the last column in Table 3 provides a reference to the degree of class imbalance in simulated data streams. For example, a string of "1:9.7 $\sim$ 1:15.1" means that the imbalance ratio of the minority to majority classes varies from 1:9.7 to 1:15.1.

4.2.2. *Slashdot data set.* The Slashdot data set[1] includes about 140,000 user posts commenting on news articles dealing with politics. These posts had been collected for one year, from 28 August 2005 to 30 August 2006. The average length of posts is 99 words. We sorted these posts by the date and time at which they were published. Therefore, the Slashdot represents a real-world data stream in which posts are arriving sequentially. The learning task here is to detect undesirable posts, such as off-topic posts, personal insults, and rude language. In the Slashdot data set, each post was given an evaluation score in $\{-1, 0, 1, 2, 3, 4, 5\}$ by a meta-moderation system. We selected posts with the score of –1 as undesirable, while those with the score of 0 to 5 as normal. As a result, we have an imbalanced Slashdot data set in which the minority class includes 2,466 undesirable posts and the majority class includes 138,311 normal posts. The class imbalance ratio is $2,466 : 138,311 \approx 1 : 56$.

One remarkable point in the Slashdot data set is that posts belong to different threads, each of which starts with a post of a news article conveying the topic of discussion. For those undesirable posts marked with "off-topic", they should be considered in relation to the first post of the same thread. Therefore, we made each instance in the Slashdot data stream by appending one post to the corresponding first post, which indicates its context.

For text preprocessing, we split posts into words based on whitespace characters and punctuation marks. We also removed stop words (such as "a", "an" and "the") which convey little information, as well as stemming words using the Porter stemming algorithm (e.g., the two words "computation" and "computing" have the same stem "comput"). The total number of distinct words remaining is 88,098. Next, we performed feature selection based on document frequency, i.e., the number of posts in which a word appears. This feature selection technique is the simplest and fast to run, but still competitive with more complex other techniques [20]. We experimented with different numbers of features, including 500, 1000, 2000, ... , 6000 features with the highest document frequencies. In this experiment, we used the Support Vector Machine learning algorithm as the base learner. However, the results show insignificant difference between those numbers of

[1]Slashdot data set can be downloaded at http://caw2.barcelonamedia.org.

features. It is possible that many additional features are redundant. Therefore, for all experiments following, we used only 500 features. We represented each instance by a binary vector of 500 dimensions, each of which indicates if the corresponding feature appears in this instance. It was shown in [21] that this binary representation can provide high accuracy in text categorization despite its simplicity.

4.3. **Compared methods and experimental settings.** We compared our method with several previous methods, as listed below:

- *Single model*: Train a single model on the current chunk of a data stream. We ran this method to show that the performance of a conventional learning algorithm would possibly be very poor on imbalanced data streams if the training data is left as it is, without any preprocessing to balance it.
- *Full data*: Incorporate all past data of the minority class into the current chunk, and then train an ensemble model, as in Table 1. This approach of reusing all past data was proposed in [12].
- *Ranking*: Rank past data based on the number of minority class nearest neighbors in the current chunk, then select a number of top-ranked instances, and finally train an ensemble model, as in Table 1. This approach of ranking past data was proposed in [14]. We set parameters as in the original paper. Specifically, the number of nearest neighbors was set to ten, and past training instances were selected with a quantity so as to obtain a class imbalance ratio of up to 1:2.
- *Adaptive*: This is our proposed method, which allows an adaptive data reuse in the presence of concept drift. This method is presented in Table 2.

We used two base learning algorithms, Support Vector Machines (SVM) [22] and Naive Bayes (NB) [23], for training classification models in the compared methods. For training of SVM classifiers, we ran the LIBSVM software [24] with the Gaussian RBF kernel, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2)$, on the UCI data sets, and the linear kernel, $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \boldsymbol{x}_j$, for text categorization on the Slashdot data set. Because the selection of optimal parameters is quite expensive for data stream learning, we simply set the parameters to a reasonable value. Specifically, the trade-off parameter C of the SVM learning algorithm was set to 100, and the width γ of the RBF kernel was set to 1. For the NB learning algorithm, we assumed a normal distribution for continuous features in data.

Some other experimental settings were also made with modest options. The k in the k-fold cross-validation used in our method (See Section 3.2.4.) was set to three. The number of base models m in the ensemble training procedure in Table 1 was set to ten.

We evaluated the compared methods by alternating the training of a model M_i on a data chunk D_i with the classification of the next data chunk D_{i+1}. In other words, at time step i, D_i is the training set, while D_{i+1} is the test set. When training model M_i on data chunk D_i, we only know the label of instances in D_i, but know nothing about D_{i+1}, which is future data and may have a different distribution due to concept drift. However, at time step $i+1$, D_{i+1} becomes a new training set, while D_{i+2} is used as a new test set. This process continues until the end of the data stream. Note that the last chunk is only used for testing. This evaluation approach allows us to trace the performance of learning methods along the data stream.

5. Results.

5.1. **Results on UCI data sets.** There are some factors that may influence the relative performances of the compared methods, such as the instance order in a data stream and random sampling of data in some learning methods. Therefore, instead of a single run,

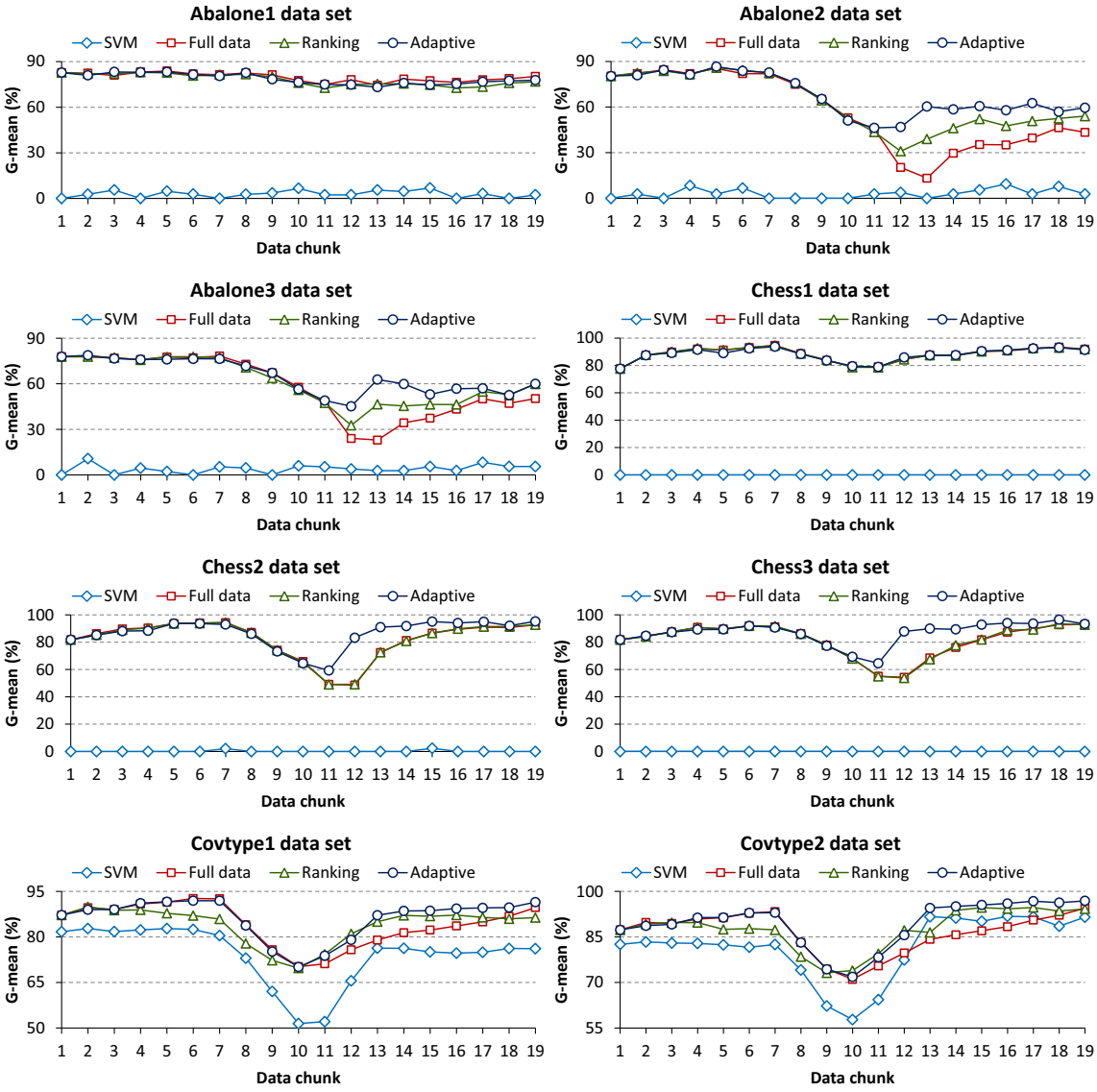

FIGURE 2. *G-mean* performance on UCI data sets using Support Vector Machines (continued in Figure 3)

we repeated experiments ten times for each imbalanced UCI data set in Table 3, and then computed the average results.

Figures 2 and 3 show the results in the *G-mean* measure when using SVM as the base learner. In this case, the first compared method (i.e., "single model") simply trains a standard SVM classifier on each data chunk, and thus is referred to as "SVM" in the figures. The data streaming scenario includes an actual concept drift, which progresses from Chunk 9 to Chunk 13. Therefore, the performance of the methods starts decreasing at Chunk 8 (the model on Chunk 8 is evaluated on Chunk 9), but recovers at Chunk 13 (the model on Chunk 13 is evaluated on Chunk 14). Some remarkable points that can be drawn from the results are as follows:

- The SVM method is always the worst, with a 0% performance at many times, because SVM alone cannot deal with the problem of class imbalance.
- In the first period when the data stream is stable (before Chunk 9), the three methods of data reuse (including "full data", "ranking" and "adaptive") have similar performance. The reason is that all past data is consistent with the target concept.

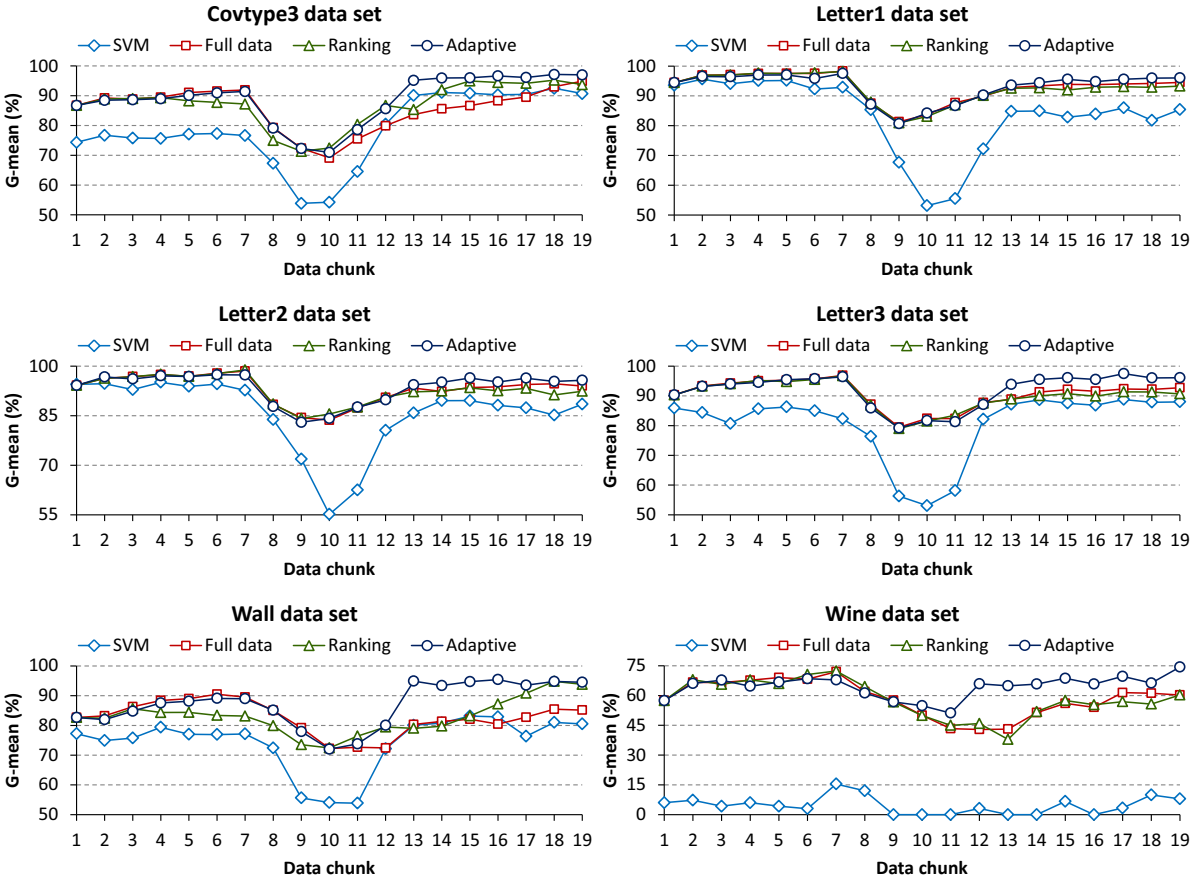

FIGURE 3. *G-mean* performance on UCI data sets using Support Vector Machines (continued from Figure 2)

- In the remaining period with concept drift (from Chunk 9), an amount of past data becomes out-of-date, leading to the degradation of the data reuse methods. However, they gradually recover when the data stream gets stable again. In addition, the adaptive method is faster to recover, and remains a higher performance than the other methods.

The results when using NB as the base learner are shown in Figures 4 and 5. These results in general are consistent with those in the case of using SVM. It can be seen that the adaptive method with use of NB improves performance even more. For the Covtype1 and Covtype2 data sets, the adaptive method performs better than the other methods for all data chunks of the data stream.

In summary, the results on the UCI data sets, with use of SVM and NB as the base learners, show that the proposed adaptive method is overall better than the other compared methods, especially since concept drift occurs. As described in Section 3, the proposed adaptive method makes use of three candidate solutions of data reuse. At each time step, corresponding to a data chunk, the best candidate solution identified by the k-fold cross-validation is used to select past data. To see the contribution of candidate solutions, Table 4 shows the rate [%] of times in which each of them is the best.

5.2. **Results on Slashdot data set.** We split the Slashdot data stream into consecutive data chunks, each of which has the same size and contains a varying number of minority class instances. In this split, we ignored some instances remaining at the end of the

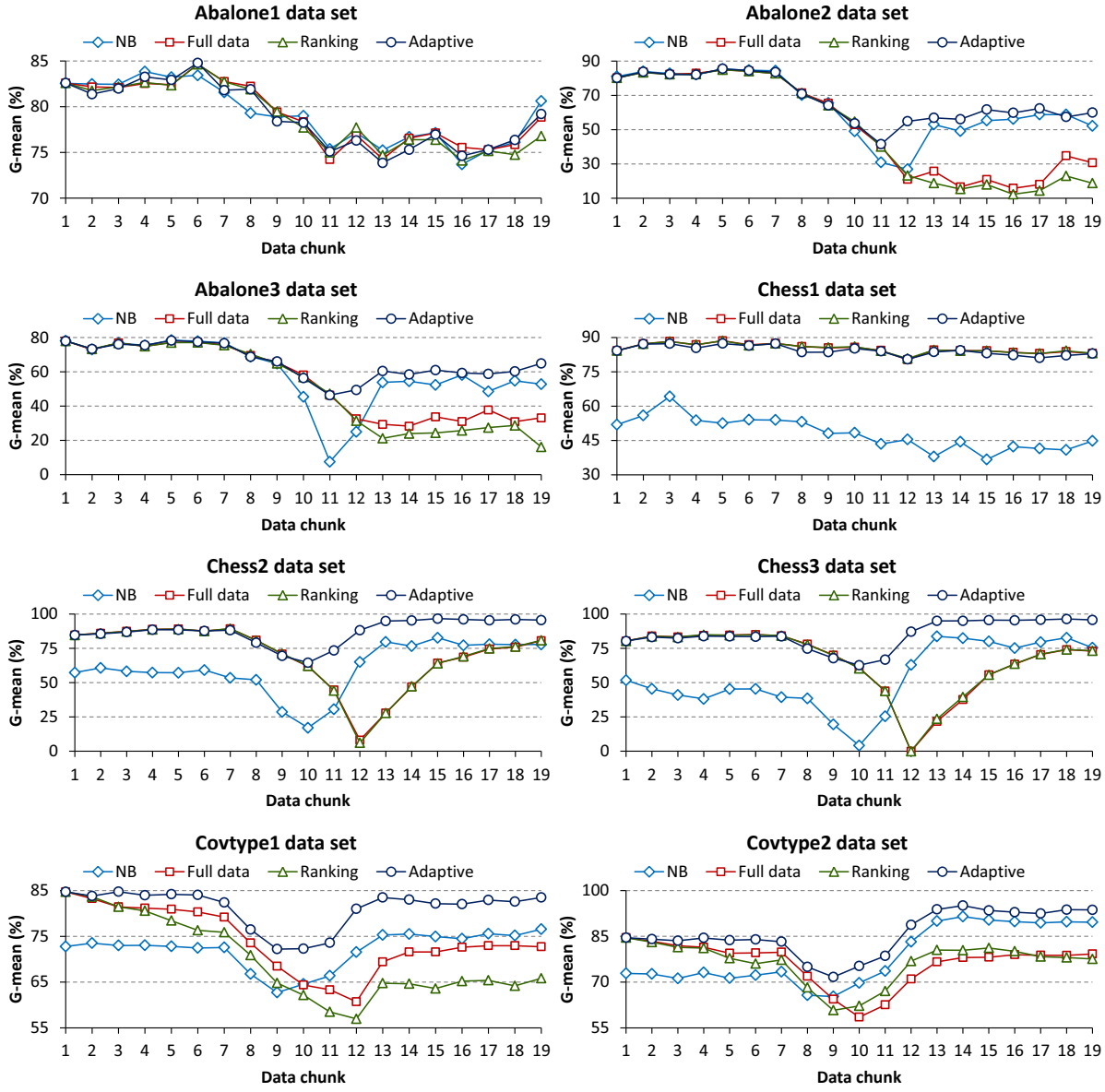

FIGURE 4. *G-mean* performance on UCI data sets using Naive Bayes (continued in Figure 5)

data stream, which are insufficient to form a data chunk. Table 5 shows the split of the Slashdot data stream with different chunk sizes.

We ran experiments five times, and report, in Figure 6, overall *G-mean* performance on the entire Slashdot data stream when varying chunk size. The left plot is for the case of using SVM as the base learner, while the right is for the case of using NB. The adaptive method performs best among the compared methods. The other two methods of data reuse, "full data" and "ranking", are inferior to our method and have a quite similar performance.

We also report the detailed results for the case where the Slashdot data stream is split into data chunks of 5000 instances, as shown in Figure 7. In general, the adaptive method is better than the others, especially when we use SVM as the base learner.

6. **Conclusions.** In this paper, we proposed an effective method for learning from imbalanced and concept-drifting data streams. We select and reuse past data to improve the

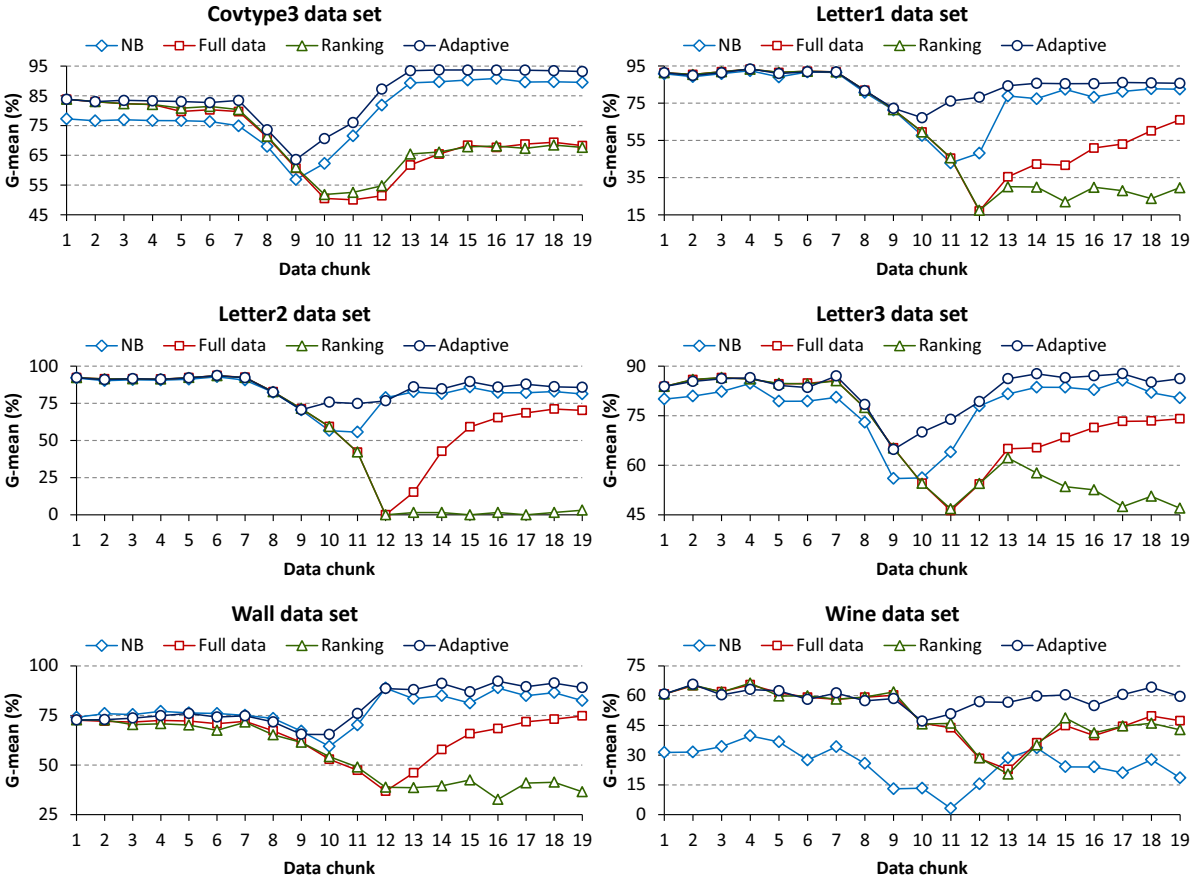

FIGURE 5. *G-mean* performance on UCI data sets using Naive Bayes (continued from Figure 4)

TABLE 4. The rate [%] of times in which each candidate solution of data reuse in the proposed adaptive method is the best (Note that there are total 18 times, corresponding to Chunks 2-19, in which data reuse solutions are considered)

Data set	Support Vector Machines			Naive Bayes		
	Solution 1	Solution 2	Solution 3	Solution 1	Solution 2	Solution 3
Abalone1	38.9	40.6	20.6	43.9	31.7	24.4
Abalone2	34.4	46.1	19.4	46.7	27.2	26.1
Abalone3	26.7	45.6	27.8	38.9	36.1	25.0
Chess1	0.6	97.8	1.7	31.7	57.8	10.6
Chess2	3.3	88.9	7.8	52.8	37.8	9.4
Chess3	0.6	91.1	8.3	45.6	44.4	10.0
Covtype1	16.1	76.1	7.8	20.0	22.8	57.2
Covtype2	11.1	81.1	7.8	19.4	29.4	51.1
Covtype3	14.4	79.4	6.1	23.3	29.4	47.2
Letter1	41.1	56.7	2.2	24.4	42.2	33.3
Letter2	35.6	63.9	0.6	29.4	37.8	32.8
Letter3	41.7	56.7	1.7	19.4	43.3	37.2
Wall	29.4	61.7	8.9	35.0	33.3	31.7
Wine	34.4	47.2	18.3	29.4	43.3	27.2

TABLE 5. Split of Slashdot data stream with different chunk sizes

Split no.	Chunk size	Number of chunks	Size of minority class in a chunk	
			from	to
1	2000	70	16	60
2	3000	46	27	83
3	4000	35	46	94
4	5000	28	65	115
5	6000	23	76	137

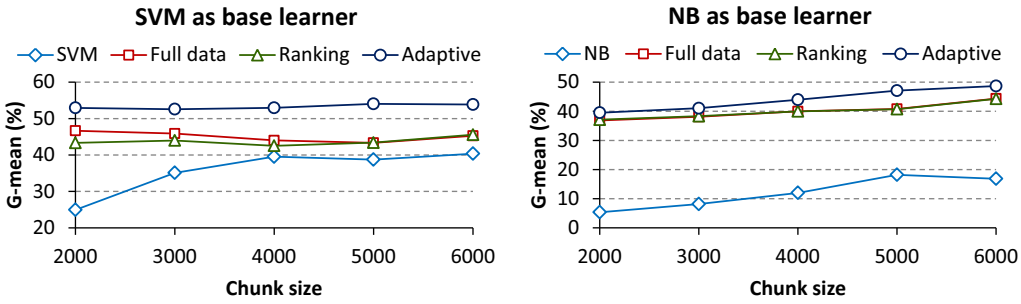

FIGURE 6. Overall *G-mean* performance on the entire Slashdot data stream when varying chunk size

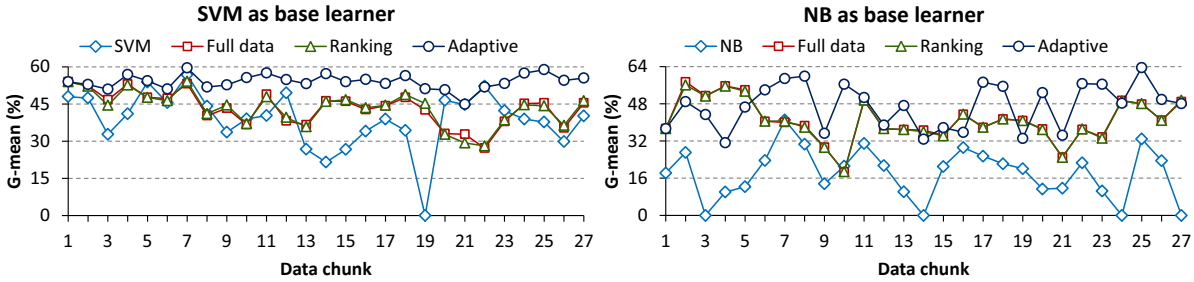

FIGURE 7. *G-mean* performance on Slashdot data stream with chunk size of 5000 instances

representation of the minority class. Different from previous methods, our method has the ability to automatically adapt data selection for concept drift. Basically, we achieve this ability by approximating the target concept with a "balanced" model on the current data chunk, which is trained after an under-sampling to balance the training set. This model, therefore, allows us to more reliably select past data which is consistent with the target concept. To further improve performance of our method, we also consider additional solutions of data reuse for cases where the data stream is relatively stable or unstable. The experimental results on simulated and real-world data streams showed that our method achieves better performance than previous methods, especially when concept drift occurs.

REFERENCES

[1] C. C. Aggarwal, *Data Streams: Models and Algorithms*, 1st Edition, Springer, 2007.

[2] H. Wang, W. Fan, P. S. Yu and J. Han, Mining concept-drifting data streams using ensemble classifiers, *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Washington D.C., USA, pp.226-235, 2003.

[3] A. Du and B. Fang, Novel approach for web filtering based on user interest focusing degree, *International Journal Innovative Computing, Information and Control*, vol.4, no.6, pp.1325-1334, 2008.

[4] P. Domingos and G. Hulten, Mining high-speed data streams, *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, USA, pp.71-80, 2000.
[5] W. N. Street and Y. S. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, *Proc. of the 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Francisco, USA, pp.377-382, 2001.
[6] R. Klinkenberg and T. Joachims, Detecting concept drift with support vector machines, *Proc. of the 17th Int. Conf. on Machine Learning*, Stanford, USA, pp.487-494, 2000.
[7] K. Veropoulos, C. Campbell and N. Cristianini, Controlling the sensitivity of support vector machines, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence*, Stockholm, Sweden, pp.55-60, 1999.
[8] G. E. A. P. A. Batista, R. C. Prati and M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, *ACM SIGKDD Explorations Newsletter*, vol.6, no.1, pp.20-29, 2004.
[9] H. M. Nguyen, E. W. Cooper and K. Kamei, Borderline over-sampling for imbalanced data classification, *Int. J. Knowledge Engineering and Soft Data Paradigms*, vol.3, no.1, pp.4-21, 2011.
[10] G. Ditzler, R. Polikar and N. Chawla, An incremental learning algorithm for non-stationary environments and class imbalance, *Proc. of the 20th Int. Conf. on Pattern Recognition*, Istanbul, Turkey, pp.2997-3000, 2010.
[11] Y. Wang, Y. Zhang and Y. Wang, Mining data streams with skewed distribution by static classifier ensemble, *Studies in Computational Intelligence*, vol.214, pp.65-71, 2009.
[12] J. Gao, B. Ding, W. Fan, J. Han and P. S. Yu, Classifying data streams with skewed class distributions and concept drifts, *IEEE Internet Computing*, vol.12, no.6, pp.37-49, 2008.
[13] E. Spyromitros, M. Spiliopoulou, G. Tsoumakas and I. Vlahavas, Dealing with concept drift and class imbalance in multi-label stream classification, *Proc. of the 22th Int. Joint Conf. on Artificial Intelligence*, Barcelona, Spain, pp.1583-1588, 2011.
[14] S. Chen and H. He, Towards incremental learning of nonstationary imbalanced data stream: A multiple selectively recursive approach, *Evolving Systems*, vol.2, no.1, pp.35-50, 2011.
[15] J. Gao, W. Fan and J. Han, On appropriate assumptions to mine data streams: Analysis and practice, *Proc. of the 7th IEEE Int. Conf. on Data Mining*, Omaha, USA, pp.143-152, 2007.
[16] C. Drummond and R. C. Holte, C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling, *Proc. of ICML Workshop on Learning from Imbalanced Data Sets*, Washington D.C., USA, pp.1-8, 2003.
[17] X. Y. Liu, J. Wu and Z. H. Zhou, Exploratory under-sampling for class-imbalance learning, *IEEE Trans. on Systems, Man and Cybernetics – Part B*, vol.39, no.2, pp.539-550, 2009.
[18] M. Kubat and S. Matwin, Addressing the curse of imbalanced training sets: One-sided selection, *Proc. of the 14th Int. Conf. on Machine Learning*, Nashville, TN, USA, pp.179-186, 1997.
[19] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, School of Information and Computer Science, University of California, Irvine, USA, 2010.
[20] Y. Yang and J. O. Pedersen, A comparative study on feature selection in text categorization, *Proc. of the 14th Int. Conf. on Machine Learning*, Nashville, TN, USA, pp.412-420, 1997.
[21] S. T. Dumais, J. C. Platt, D. Heckerman and M. Sahami, Inductive learning algorithms and representations for text categorization, *Proc. of ACM Int. Conf. on Information and Knowledge Management*, Bethesda, USA, pp.148-155, 1998.
[22] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, vol.2, pp.121-167, 1998.
[23] P. Domingos and M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss, *Machine Learning*, vol.29, no.2-3, pp.103-130, 1997.
[24] C. C. Chang and C. J. Lin, LIBSVM: A library for support vector machines, *ACM Trans. on Intelligent Systems and Technology*, vol.2, no.3, pp.27:1-27:27, 2011.