A Self-Adaptive Sliding Window Technique for Mining Data Streams

Yi Yang[1] and Guojun Mao[2]

[1] School of Computer Science, Beijing University of Technology, Beijing, China
[2] College of Information, Central University of Finance and Economics, Beijing, China
cnadrian@126.com, maximmao@hotmail.com

Abstract. The methods of data stream mining have recently garnered a great deal of attention in the field of data mining, and the sliding window technique has been widely used during many researches on it. This paper proposes a new type of self-adaptive sliding window (SASW) model, which has self-adjusting window parameters, and the technique details are presented under the ensemble learning method of single data stream environment. Experimental result shows that the definition of evaluating SASW parameters is appropriate and the gratifying results can be obtained. This idea also can be used in many other algorithms of data stream mining.

Keywords: -data stream mining, SASW technique, ensemble learning.

1 Introduction

The data stream, different from traditional databases, is a real-time, continuous, high-volume and open-ended sequence of data items. In many real-world applications, like sensor networks, traffic analysis, network intrusion detection, data streams have become a powerful information organization form, so building proper streaming mining models for these applications is becoming an important focus.

According to the nature of data stream, the problem of memory-efficiency has been arisen in stream mining models. In fact, a data stream is potentially unbounded but memory resource is limited, thus a mining model must support to find out hidden patterns by the restricted memory from continuous data streams.

The sliding window is one of the popular techniques for data stream mining [1]. It can discover so-far patterns using finite resources through restricting the number of items or the time range to the current window from a data stream. In a general way, the shorter the size of window is defined, the smaller memory is used and the higher memory-efficiency can be gotten. Of course, the longer size of the window can consider more data in a window and so can get a higher mining accuracy.

In addition, when using sliding window model during mining process, the problem of CPU-efficiency should also be noted. When a new window is created, it is necessary to drop some out-of-date data from the old window and add some new arrival data to the new one. However, the number of dropping or adding data for window changes in sliding windows will cause the problem of CPU-efficiency. In general, the longer the dropping or adding steps, the less the times of window transformation for a data stream and the higher CPU-efficiency can be achieved. Comparably, the shorter steps of dropping or adding data can result in mining patterns in a more real-time way. Therefore, setting appropriate window size or moving step values for a sliding window model is a trade-off between mining accuracy and efficiency.

Sliding window models have widely been used in many existing researches of data stream mining [2, 3, 4, 5, 6], however, there are few studies about influence of different size or step settings in sliding windows. In fact, there should be different optimizing parameters of sliding windows for different data streams, that is, optimizing sliding windows is not a universal problem, so such optimizations should be adapted to the changes of a data stream over time. In this paper, we present a new type of self-adaptive sliding window (SASW) model, which can learn the sliding window control parameters that meet the demand given automatically when patterns are mined from data streams.

The rest of paper is organized as follows. Section 2 shows some definitions about the data stream and sliding window model. In section 3, we proposed SASW model and apply it into the ensemble learning of stream mining and give the descriptions on evaluating process of the parameters. Then an example is showed through our experiment in section 4. Finally section 5 summarizes our research.

2 Problem Statement

Given a data stream $S = < s_1, s_2, \cdots, s_t \cdots >$ where s_t is the data item that arrived at the time point *t*. As this paper will discuss the problem of classifying data streams, so a data item can be represented as $s_t = \{attrvec(s_t), class(s_t)\}$, where $attrvec$ (s_t) is the vector of condition attributes and $class$ (s_t) is the class label of s_t.

There are two basic strategies to design a sliding window model in data stream mining, i.e. count-based windows and time-based windows [1]. The former always maintains the same number of data items before and after window changes, while the latter use a fixing time interval to define the size of windows. This paper will focus on count-based sliding window, where the size of a window means the number of items that the window includes.

Let the size of a window *N*, given *k* in (0, 1], we can define a called "*k*-step" sliding window. There are two typical situations are being popularly applied for sliding windows: 1) *k*=1/*N*, removing only one data item from the previous window and supplying a new arrival item in order to create a new window [4]; 2) *k*=1, data of the old window are all removed and data of the new window are all new arrival items [2, 3, 5, 6]. Obviously, for a "*k*-step" sliding window, the number of removed items (say *step*), is *k***N*. Using the sliding window, we can collect the data in a window and the obtained data can be mined to get valuable classes by a fixing time point. Figure 1 shows a *k*-step count-based sliding window with the size of *N*. Each window collects *N* data items and the next window will be generated to replace the oldest *k***N* items by new arrival ones.

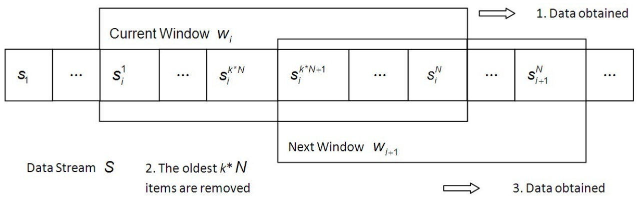

Fig. 1. Processing in a count-based & *k*-step sliding window

According to the existing researches, it can be guessed that when the *N* value is fixed, there should be different optimal *k* values for different applications and computing environments, and so dynamically optimizing *k* value for an ongoing data stream is very important. Moreover, if the value *k* can be learn and adjusted automatically with the changes of data stream, then better mining efficiencies or results may be able to be achieved.

3 A SASW-based Mining Model for Oriented to Ensemble Learning in Data Streams

As we stated, a SASW should adapt to the changes of the mined data stream, and can dynamically adjusted over time. Therefore, a better solution is that integrates adjusting SASW into learning patterns in a mining model to fit dynamic data streams.

In this section, we perform step-by-step analysis to our model, a mining model that integrates SASW technology to ensemble learning for data streams.

3.1 Model Overview

The basic processing framework of our model is described in Figure 2.

As Figure 2 shows, our model has mainly three phases. In phase 1, firstly enough new data is gathered from the data stream. Due to *k*N* items is needed for *k*-step SASW, so after collecting data, a new window can be formed by replacing k**N* data items in the previous window by new items.

In phase 2, a new classifier *NC* can be obtained. *NC* can be learnt by a number of popular classifying algorithms, and we used C4.5 to our experiments in this paper. We use the ensemble learning method in our model; this is because of its better results than single classifier for mining data streams, which have been proven by many researches [7, 8, 9]. Multiple basic classifiers, *M* classifiers at most in Figure 2, constitute an ensemble *EC*. Each classifier has a called weight between 0 and 1 to express its importance in *EC*. After a new window is created and a new classifier obtained, all classifiers in the ensemble will be evaluated and their weight can be updated. The weights of all classifiers can be dynamically changed according to the rates of correct classifying for data in the current window. Furthermore, if the number of classifiers exceeds the maximum defined number of classifiers in the ensemble (say *M* in Figure 2), the classifier with the lowest weight will be removed unless the newest classifier has the lowest weight.

Parameter description

S: the data stream;
W: a SASW (the current window);
N: the size of *W*;
k: control parameter for step of *W* (initial 1.0);
C: classifying method (algorithm, like C4.5);
EC: ensemble (all classifiers in *EC* are with method *C*);
EV: evaluation value of SASW with a fixing *k*;
α and β: the allowed lower and upper bound of *EV*s;
r: the adjustment factor of *k*(between 0 and 1)*;*
ET: evaluation time of SASW in a fixing *k*(by number of windows, initial $ET_{1.0}$ is set by user);
M: the maximum number of classifiers in *EC*.

Model description

Phase 1.Create a new window
1-1. Collect *k*N* new arrival items from *S*;
1-2. Remove the earliest *k*N* data items from *W*;
1-3. Add the new *k*N* items to *W*.

Phase 2. Learn a new classifier and update the ensemble
2-1. Learn a new classifier *NC* in *W*;
2-2. Recalculate weights of all classifiers in *EC* by *W*;
2-3. If the number of classifiers in *EC* is less than *M*, insert *NC* into *EC*; Otherwise, insert *NC* and remove the classifier with the minimum weight in *EC* unless *NC* has the lowest weight.

Phase 3. Evaluate SASW and adjust *k* value
With fixing *k*, for *ET* windows, iteratively do:
3-1. Compute the average *EV* values by the current window (say *EV*).
3-2. If $EV > \beta$ or $EV < \alpha$, then adjust *k*;
3-3. If *k* is changed, recalculate *ET*.

Fig. 2. Framework of the model

In a nutshell, for *k*-step count-based sliding windows, adjusting parameter *k* roughly respects that changing SASW to get better mining qualities, and so adjusting the control parameter *k* value according to the evaluating result to current SASW will be conducted in phase 3. More details about this phase will be given in Part B of this section.

In fact, these phases can simultaneously work. That is, when a new window is being created, pattern learning is doing by the previous window, and the evaluate process is also running and the control parameter *k* of SASW can be adjusted at the same time if necessary.

3.2 Evaluating SASW by Computing EVs

In order to test whether the current SASW still adapted well to the changes of the data stream, a series of measurements must be taken.

Definition 1. Let the current window W_i^i, and given a classifier c, R_c^i, the correct classifying rate of classifier c in W_i^i can be computed by the percentage of the correctly classified items to all data items in W_i^i

Definition 2. Let the current window W_i^i and given an ensemble $E = \{c_1, c_2, \ldots, c_m\}$, R_E^i, the correct classifying rate of ensemble E in W_i^i, is defined as the maximal rate of all classifiers in E for W_i^i:

$$R_E^i = max(R_{cj}^i \mid \text{j}=1, 2, \ldots, \text{m}) \tag{1}$$

Definition 3. Let the current window W_i^i, and given a classifying method C and its ensemble EC. If a classifier NC is generated by method C, then EV (W_i^i, C, EC), the evaluation value of W_i^i using method C in the way of ensemble learning, is defined as

$$EV_i = R_{NC}^i - R_{EC}^i \tag{2}$$

Definition 4. For a fixing k, given a series of k-step windows $w^k=\{w_1, w_2, \ldots, w_n\}$, and let a classifying method C and its ensemble EC. $EV(k, w^k, C, EC)$, the evaluation value of the k-step sliding window for data w^k using method C in the way of ensemble learning, means the average evaluation value of all windows in w^k:

$$EV(k, w^k, C, EC) = average(EV(W_i^i, C, EC) \mid i=1, 2, \ldots, n) \tag{3}$$

As far as data stream mining is concerned, the effectiveness of a k-step SASW to a data stream can be reflected by evaluation value of a set of objected k-step windows. In general, the smaller the EV value for a given k, such a k-step SASW can better match the changes of the data stream. The main process of Function EV (k, w^k, C, EC) is given in Figure 3.

```
Function EV(k, w^k, C, EC)

1.set k;
2. FOR each window w_i in w^k DO
3. compute EV(w_i, C, EC);
4.sum up EV(w_i, C, EC)  for get their average value;
5.END DO
6. RETURN the average of EVs for all windows in w^k .
```

Fig. 3. Computing process of evaluation values

3.3 Adjusting Control Parameter k in SASW

In Part A of this section we mentioned that adjusting of the control parameter *k* value according to *EV*s is an important process of our model. Here, there are two questions have to be answered: when and how parameter *k* of SASW is adjusted.

As Figure 3 shows, because the number of windows in w^k directly affect the computing efficiency and evaluating effectiveness of Function *EV* (k, w^k, C, EC), there is no need to check *EV* after every window change. Instead, we use a periodic evaluation time that is called as *ET* in Figure 2. In this paper, *ET* is directly related to the current *k* value.

Definition 5. For a fixing *k*, given an initial $ET_{1.0}$, *ET* value related to *k*, denoted as ET_k, is defined as:

$$ET_k = ET_{1.0}/k. \tag{4}$$

For Example, if the initial *k* value is 1.0, and the user sets $ET_{1.0}$=100, then when *k* value is adjusted 0.5, $ET_{0.5}$=200. As Figure 2 stated, if the current *k*=1.0, evaluation will be done and *k* can be adjusted after 100 windows disposed; if the current *k*=0.5, evaluation or adjustment for *k* can be performed after 200,so *ET* value means the number of windows before the checking of whether the current *k* value is suitable. When the *k* value is adjusted, the *ET* value is also recalculated.

Our definition of *ET* based on some basic considerations about stream mining: when the ensemble construction is processing, the value *EV* (w_i, C, EC) is also calculated continuously, because the data stream is a real-time model, some of the *EV* values may be abnormally compared to other results. For example, after disposing many windows with *k*=1.0, most computed *EV* values are between 3.5 and 4.0, but a few of them may exceed 5.0 or lower than 2.5, if the adjustment of value *k* occurs after every window change, such abnormal *EV* values may badly affect the result. So the *EV* value we need to evaluate the *k* value should be a statistical value which computed by many *EV* (w_i, C, EC) values to eliminate ill effects. In our paper we use the average value calculated by the values collected after the number of *ET* windows with fixed value *k*. Besides, the initial $ET_{1.0}$ value should be considered seriously. First, the *ET* value shouldn't be very small based on our analysis above. Second, when *k* is adjusted, the adjusted *ET* value is getting larger than it before, if it is too large, there may be taking a long time for the processing and hurt the performances of the mining model.

Another important issue we'll describe is how to adjust the *k* value. As can be seen from Figure 2, whether *k* value is need to be adjust depends on whether its calculated *EV* value is between the lower and the upper bound(say α and β in Figure 2) defined by user. Generally, if the *EV* value is greater than β, the *k* value should be decreased, and it should be increased while *EV* is smaller than α.

The setting of α and β value depends upon the considerations about not only the demand of mining effect, but also the potential risk occurs during the mining process: if the *EV* value is very large, it can be seen that the effect of current *EC* is not very well, so a threshold value (say β in Figure 2) of *EV* that meet the user's demand is set. Besides, in Part B of this section we mentioned that: in general, the smaller the *EV* value for a given *k*, such a *k*-step SASW can better match the changes of the data stream. But when *EV* is too small, it may cause the overfitting, that is, the *EC* is overfit the items of new window. To avoid it, the decreasing of *EV* should be controlled, then a lower bound value α is defined to achieve this.

When the current *k* value is unsuitable, it needs to be adjusted as we mentioned above. An adjustment factor *r* which between 0 and 1 is defined to decide the length of *k* to adjust in each time: the increasing process of *k* is defined as $k=k*(1+r)$, while the decreasing process is $k=k*(1-r)$.

Based on the above analyses, Figure 4 gives the description of the adjusting method of the control parameter *k* in SASW.

```
Procedure of the adjusting of parameter $k$ of SASW($ET_k$, $k$, $ET_{1.0}$, $EV$, $\alpha$, $\beta$, $r$)

1.  FOR a parameter value $k$, when $ET_k$ arrived, DO:
2.  IF $EV>\beta$ BEGIN
3.      Adjust $k$: $k=k*(1-r)$;
4.      Recalculate $ET_k$: $ET_k = ET_{1.0}/k$;
5.      END
6.  ELSE IF $EV<\alpha$ BEGIN
7.      Adjust $k$: $k=k*(1+r)$;
8.      Recalculate $ET_k$: $ET_k = ET_{1.0}/k$;
9.  END
10. ELSE   Current $k$ value is suitable.
11. END DO
```

Fig. 4. The adjusting process of parameter value *k* in SASW

4 Experiment Result

To show the working process of SASW, we conduct an experiment based on the model in Figure 2. Our experiment was implemented using JAVA on a computer with Intel core 2 Q9450 2.66GHz and 3GB-RAM, the WEKA toolkit is also used in the programming.

The dataset used in our experiment is the *spambase* dataset, which has 57 continuous attributed and two type of class label. In order to simulate a data stream, we designed a stream generator with some control parameters such as flow speed etc., which can choose data items from the dataset randomly and continuously.

Some initial parameter values set by us are showed in Table 1.

Analysis of Results: In the first few rounds, because the *EV* value is much larger than the upper bound β, the *k* value has been decreasing. From round 9, the *EV* value is very close to β, and due to the lower bound value α, which is not very far from β(β-α=0.100 in our example), the *k* value is increasing and decreasing by turns during the next rounds. Finally in the 19th round, the *EV* value (say 2.389) is the first value between α and β. So its corresponding *k* value (0.409 as the result shows) is the final adjusting result.

Table 1. Initial Parameter Values in Experiment

Flow speed	100-200 items/100-200 ms
N	1000
Initial k	1.000
α and β	2.300 and 2.400
r	0.1
$ET_{1.0}$	20
C	C4.5 Revision 8
M	8

Table 2 shows a complete adjusting process of parameter k in our experiment, the ET_k and EV value calculated after ET_k windows for each k are showed in it.

Table 2. Results to Our Experiment

Round	$\boldsymbol{k}$	$\boldsymbol{EV}$	$\boldsymbol{ET_k}$
1	1.000	4.098	20
2	0.900	3.891	22
3	0.810	3.702	24
4	0.729	3.511	27
5	0.656	3.304	30
6	0.590	3.097	33
7	0.531	2.902	37
8	0.478	2.695	41
9	0.430	2.489	46
10	0.387	2.280	51
11	0.426	2.477	46
12	0.384	2.265	52
13	0.421	2.463	47
14	0.380	2.261	52
15	0.418	2.440	47
16	0.376	2.241	53
17	0.414	2.419	48
18	0.372	2.236	53
19	**0.409**	**2.389**	48

5 Conclusion

In this paper, we proposed a self-adaptive sliding window (SASW) technique for mining data streams based on our "k-step" window idea, then some technical details such as the evaluating and adjusting process of the SASW control parameter are showed through an ensemble learning model with our SASW model supported to. The SASW technique has a universal use when mining data streams, e.g. if we use SASW in the local sites when mining distributed data streams, the quality of the mining of local sites can be improved and the maintaining of global schema can be very convenient than ever before.

Acknowledgement. This work has supported by Beijing Municipal Key Laboratory of Multimedia and Intelligent Software Technology, and the National Science Foundation of China under Grants No.60873145.

References

[1] Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in Data Stream Systems. In: Proc. The 21st ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems (PODS 2002), pp. 1–16. ACM Press (June 2002), doi:10.1145/543613.543615

[2] Hulten, G., Spencer, L., Domingoes, P.: Mining Time-Changing Data Streams. In: Proc. The 7th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2001), pp. 97–106. ACM Press (August 2001), doi:10.1145/502512.502529

[3] Babcock, B., Datar, M., Motwani, R., O'Callaghan, L.: Maintaining Variance and K-Medians over Data Stream Windows. In: Proc. The 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS 2003), pp. 234–243. ACM Press (June 2003)

[4] Chang, J., Lee, W.: estWin: Online data stream mining of recent frequent itemsets by sliding window method. Journal of Information Science 31(2), 76–90 (2005), doi:10.1177/0165551505050785

[5] Chi, Y., Wang, H., Yu, P.S., Muntz, R.R.: Catch the Moment: Maintaining closed frequent tuplesets over a data stream sliding window. Knowledge and Information Systems 10(3), 265–294 (2006), doi:10.1007/s10115-006-0003-0

[6] Gibbons, B., Tirthapura, S.: Distributed streams algorithms for sliding windows. In: Proc. The 14th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 63–72. ACM Press (August 2002)

[7] Street, W., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: Proc. The 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001), pp. 377–382. ACM Press (August 2001), doi:10.1145/502512.502568

[8] Gianluigi, F., Clara, P., Giandomenico, S.: An adaptive distributed ensemble approach to mine concept-drifting data streams. In: Proc. The 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), pp. 183–187 (October 2007), doi:10.1109/ICTAI.2007.51

[9] Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: A Multi-partition Multi-chunk Ensemble Technique to Classify Concept-Drifting Data Streams. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 363–375. Springer, Heidelberg (2009), doi:10.1107/9-78-3-642-01307-2_34