Volatility Adaptive Classifier System

Ruolin Jia, Yun Sing Koh(✉), and Gillian Dobbie

The University of Auckland, Auckland, New Zealand
rjia477@aucklanduni.ac.nz, {ykoh,gill}@cs.auckland.ac.nz

Abstract. A data stream's concept may evolve over time, which is known as the concept drift. Concept drifts affect the prediction accuracy of the learning model and are required to be handled to maintain the model quality. In most cases, there is a trade-off between maintaining prediction quality and learning efficiency. We present a novel framework known as the Volatility-Adaptive Classifier System (VACS) to balance the trade-off. The system contains an adaptive classifier and a non-adaptive classifier. The former can maintain a higher prediction quality but requires additional computational overhead, and the latter requires less computational overhead but its prediction quality may be susceptible to concept drifts. VACS automatically applies the adaptive classifier when the concept drifts are frequent, and switches to the non-adaptive classifier when drifts are infrequent. As a result, VACS can maintain a relatively low computational cost while still maintaining a high enough overall prediction quality. To the best of our knowledge, this is the first data stream mining framework that applies different learners to reduce the learning overheads.

Keywords: Data stream · Concept drift · Stream volatility

1 Introduction

Data streams are sequences of unbounded data arriving in real time. For example, electricity usage records produced by a power station, online tweets generated in a region, transactions recorded in a stock market can all be presented as data streams. Such real-world data are generated in order and are considered to be infinite. The task of data stream mining is to find valuable information from these unbounded streams of data. Data stream's properties raise various requirements when designing data stream algorithms. Instances in a stream can arrive very fast, allowing only limited time and memory for the algorithm to learn its underlying concepts. Moreover, a data stream may evolve over time such that the underlying concepts in a stream may change. Consequently, the learning model loses prediction accuracy over time. This is known as concept drifts. To maintain the quality of a learning model, stream learning algorithms are expected to detect changes and update their models to overcome these concept drifts. The frequency of concept drifts is known as stream volatility [6]. High volatility means a high frequency of concept drifts.

J. Kim et al. (Eds.): PAKDD 2017, Part I, LNAI 10234, pp. 122–134, 2017.
DOI: 10.1007/978-3-319-57454-7_10

Some data stream learners can overcome a concept drift by adjusting their models to generalise the new concept and maintain a high prediction quality during the drift. These learning models can be classified as adaptive learners. Other data stream learners that cannot adjust their models are known as non-adaptive learners. Model adaptations come with a large computational cost. Thus, there is a trade-off between the model quality and the learning efficiency. It is also known that stream volatility may change in a stream over time [6]. For example, in stock market transactions, an anomalous event can result in an increasing number of concept drifts over a short period. One way to balance the trade-off between model quality and efficiency is to apply the model adaptation only when the stream volatility is high to maintain a stable prediction quality. When the volatility becomes low, we disable the model adaptation to save cost. We are addressing this problem by creating a new learning framework containing both adaptive and non-adaptive learners.

We designed a framework called Volatility Adaptive Classifier System (VACS). VACS has lower computational cost than the state-of-art adaptive learner while maintaining a similar prediction quality in a stream with volatility changes. VACS is composed of both adaptive and non-adaptive classifiers. VACS uses stream volatility [6] as the criterion to switch between classifiers. In particular, when the volatility is high, VACS applies the adaptive learner to maintain a better prediction quality. When volatility is low, it is deemed to be unnecessary to spend large overheads to handle infrequent concept drifts, so it switches to the non-adaptive classifier. As a result, VACS will maintain a sufficiently high prediction accuracy with relatively low overheads. Our contributions are as follows: (1) We proposed a Volatility Adaptive Classifier System (VACS), which is able to choose between the adaptive classifier and the non-adaptive classifier given different levels of stream volatility. (2) We show that the accuracy of VACS is comparable to state-of-the-art techniques, while maintaining low computational cost. To the best of our knowledge, this is the first data stream learning technique that uses stream volatility to adjust model adaptation behaviour to reduce computational cost while maintaining high model quality.

In the next section, we discuss the related work in the area. In Sect. 3, we illustrate how VACS works. In Sect. 4, we discuss the experimental results. Lastly, we conclude our paper in Sect. 5.

2 Related Work

There are various methods, derived from traditional learning methods, for mining data series including ensemble based methods [10] and neural network based methods [8]. In our research, we focus on tree based models: VFDT [4] and HAT-ADWIN [2] which have different expected properties.

VFDT is a decision tree classifier that is specifically designed for learning data streams. Because a data stream is infinite, VFDT splits an inner node using confident enough instances from a stream rather than seeing all instances. CVFDT [7] is a variant of VFDT. CVFDT maintains a sliding window storing

recent instances learned, and it adjusts its tree model to be consistent with the instances in the window.

Hoeffding Adaptive Tree using ADWIN (HAT-ADWIN) was proposed by Bifet et al. [2]. This algorithm installs the drift detector ADWIN [1] on each node of the VFDT decision tree. The ADWIN drift detector monitors the attribute-class statistics on its host node. If a change in the attribute-class statistics at that node is detected, it starts to grow an alternative tree rooted at that node. When the alternative sub-tree has a better prediction accuracy, the current sub-tree is replaced by the alternative one.

We can categorise those two tree algorithms into two classes: adaptive learner and non-adaptive learner. VFDT is considered to be a non-adaptive learner. It does not have the ability to adjust its tree model to new concepts in an evolving data stream. Instead, VFDT with drift detector ADWIN (VFDT-ADWIN) can rebuild its tree model when a concept drift is detected. However, a severe prediction accuracy drop can be experienced during model rebuilding. In contrast, HAT-ADWIN and CVFDT are adaptive learners. Adaptive learners are able to partially update their models to fit the new concept in an evolving stream such that they can maintain a stable accuracy when encountering concept drifts. However, adaptive learners such as HAT-ADWIN have larger overhead than non-adaptive learners in terms of training time and memory. This is because adaptive algorithms need additional computation and storage to perform model adaptation.

Recurring Concept Drift (RCD) framework [5] is similar to our proposed system (VACS) in that they both use more than one learner to mine data streams. However, RCD is designed for improving the prediction quality, while VACS is designed to reduce learning overheads.

3 Volatility Adaptive Classifier Systems

We propose the Volatility Adaptive Classifier System (VACS). Intuitively, when mining a stream, VACS automatically applies the adaptive classifier in high volatility periods, and it switches to the non-adaptive classifier in low volatility periods. It aims to reduce learning overheads while maintaining high prediction quality. VACS is composed of several modules: Volatility Measurement Window, Double Reservoirs Classifier Selector, a drift detector and two component learners. Figure 1 presents an overview of VACS. In particular, we use VFDT with

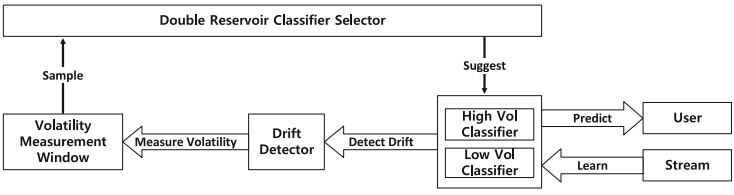

Fig. 1. VACS Overview

ADWIN (VFDT-ADWIN) in low volatility periods and we use HAT-ADWIN in high volatility periods. We use ADWIN as the drift detector.

3.1 Volatility Measurement Window

One task of VACS is to measure the volatility level of a stream such that it can switch classifiers based on different volatility levels. Huang et al. [6] calculate a stream's current volatility by calculating time intervals between each drift point in a buffer. In other words, their method measures the time differences among a fixed number of drifts. Small time differences denote high volatility while large differences denote low. Their method is appropriate to calculate relative volatility shift in a stream. However, it introduces a volatility measurement delay because it needs to wait for the next drift in order to calculate the new volatility level. The delay problem can be severe if the volatility drops from high to low. This is because the time difference between the next drift and the recent drift increases, and it needs to wait a longer period for the next drift to appear in order to update the measurement.

We develop a new method to measure the level of volatility using a sliding window with fixed size T. It contains indices of the most recent T instances learned from the stream. When a new instance's index is inserted into the window, the oldest instance's index is removed. The window maintains a value γ, which is the number of concept drifts detected in the most recent T instances. Then γ can represent the level of the current volatility. In the case when the level of volatility decreases, it does not need to wait until the next drift occurs. Instead, the window constantly updates γ over time. This new method mitigates the delay problem.

3.2 Double Reservoirs Classifier Selector

Double Reservoirs Classifier Selector (DRCS) is another module in VACS. DRCS uses the reservoir sampling technique [9] to sample and approximate both high and low levels of volatility of a stream while learning. We do not want to lose information about high volatility periods of a stream when sampling at low volatility. Similarly, we do not want to lose information about low volatility periods when sampling at high volatility. A single reservoir will not satisfy this requirement because reservoir sampling removes a random element when inserting a new element. DRCS separately samples the volatility levels from low and high volatility periods in a stream using two independent reservoirs.

In particular, DRCS has two functions: sampling and suggesting. The "sampling" function is called by VACS constantly when learning a stream. The sampling function of DRCS maintains two reservoirs named *High Reservoir* and *Low Reservoir*. Those two reservoirs sample each input γ value (volatility level) from the volatility measurement window using the reservoir sampling method [9]. The first γ is inserted into the low reservoir for initialisation. After initiation, when a new measured stream volatility γ arrives, it compares γ with the mean of the elements in the two reservoirs. If the value is lower than the mean, it stores this

value into the *Low Reservoir*. If the value is greater than the mean, it stores the value into the *High Reservoir*. We specify a means' difference threshold λ. If the difference between two reservoirs' means is greater than λ, the DRCS is set to be active. VACS can only switch a classifier when DRCS is active. This setting can prevent two undesirable behaviours. Firstly, it prevents VACS from switching classifiers when there are rare volatility changes in the stream. Secondly, if there are volatility changes in the stream but these changes only appear in a later period, it prevents VACS from switching classifiers at the early stage in which a changing volatility has not been measured yet. Intuitively, λ is used to indicate the size of the volatility change that matters to the user. If there are volatility differences greater than this threshold in a stream, we can treat the stream as a volatility-changing stream and activate our system, otherwise, we use the single learner to handle the stream. λ is also related to the volatility measurement window size T. A larger T value can result in larger measured numbers of drifts in the window. Thus, larger γ (volatility level) can be obtained and inserted into reservoirs. So λ should increase with T. However, if λ is overly large, VACS may never be activated.

The second function of DRCS is a "suggesting" function. This is used when VACS queries DRCS. When VACS queries DRCS, it compares the most recent input γ with the mean of all other γ values in two reservoirs. If the recent γ is greater than the reservoirs' mean, it returns the high volatility classifier suggestion (adaptive learner). Otherwise, it returns the low volatility classifier suggestion (non-adaptive learner).

3.3 Pseudocode

In this section, we compose each module discussed in the previous sections into the complete Volatility Adaptive Classifier System (VACS). The pseudocode can be seen in Algorithm 1. The algorithm firstly initiates the Double Reservoirs Classifier Selector (DRCS) and a drift detector. By default, we use ADWIN for the drift detector. It also initiates the volatility measurement window counting concept drifts detected in the recent T instances. We provide two classifiers for VACS. One classifier is considered to be suitable in high volatility periods (adaptive learner) while the other is deemed to be appropriate in low volatility periods (non-adaptive learner). In our implementation, we use HAT-ADWIN as the adaptive learner and VFDT-ADWIN as the non-adaptive learner. In VACS, only one classifier is active at any time to perform the learning task. The user decides which classifier should be active at the start when no volatility level has been detected. When the algorithm starts, it takes each arriving instance from the stream and classifies it with the active classifier. If the classifier correctly classifies the instance, we input 0 into the drift detector. Otherwise, we input 1 into the drift detector. The drift detector is modified such that it signals a drift only if the prediction error is increasing. Next, it updates the volatility measurement window and γ (volatility level), which is the number of drifts detected in the recent T examples. If the number of classified instances since the last volatility level measurement reaches a user-specified count τ, the algorithm measures

and inputs γ into DRCS and then queries DRCS. The reason for adding an interval τ between two consecutive γ measurements is because it is not likely to measure a change on γ if two measurements are close. Next, DRCS returns one classifier option from the two that are suggested to be used, best suited to the current level of volatility. If the suggested classifier is not consistent with the one active in VACS, it switches the current active classifier to the suggested one. It then re-initiates the new classifier. In the case of a decision tree, it resets the decision tree to a one-node tree without learning examples. When the user wants to make a classification with an instance with the unknown class, VACS will use the currently active classifier to make the prediction.

4 Results and Evaluation

We implemented VACS in the Massive Online Analysis (MOA) Framework [3]. In our experiments, we measure the performance of HAT-ADWIN, VFDT-ADWIN and VACS by evaluating total training time (Time), mean memory usage (Mem), maximal memory usage (Max Mem), the mean prediction accuracy (Acc) and the mean prediction accuracy when concept drifts occur (dAcc) on each algorithm. We compare the measurement results of those algorithms and contrast the differences among them.

Beyond those measurements, we also explore whether VACS switches between classifiers as expected. We introduce a new measurement called Percentage of Instances Classified by the Expected Classifier (PICEC). It denotes whether VACS applies the correct classifier accurately given a volatility level. Our synthetic datasets' volatility fluctuates between low and high volatility periods. VACS has two classifier options: high volatility classifier (adaptive) and low volatility classifier (non-adaptive). We obtain PICEC using the following calculation: we count the number of instances from the high volatility period in a stream classified by the high volatility classifier, and the number of instances from the low volatility period in a stream classified by the low volatility classifier. We divide the sum of these two numbers by the total number of instances in the streaming dataset.

Here we specify all parameters for VACS.

Each reservoir in DRCS has size 200, volatility measurement window size (T) is 300000, means' difference threshold (λ) is 15, the interval length between each volatility level measurement (τ) is 10000, the drift detector of VACS is ADWIN, the high volatility classifier is HAT-ADWIN, and the low volatility classifier is VFDT with ADWIN as the external drift detector (VFDT-ADWIN). The default starting classifier is VFDT-ADWIN. ADWIN uses the Hoeffding Bound whereby the δ value [1] is set to 0.002.

4.1 Experiments on Mutating Random Tree Generator Datasets

We show the evaluation results run on data generated by the mutating random tree generator. We intend to evaluate whether VACS has a reduction of

Algorithm 1. VACS: Volatility Adaptive Classifier System

```
input : S: Stream of examples
        τ: Interval length between each volatility level measurement.
        T: Size of sliding window for measuring current volatility level.
        HighVolClassifier: The classifier used in high volatility period.
        LowVolClassifier: The classifier used in low volatility period.
        StartingClassifier: The classifier chosen at the start.
        DT: Drift Detector.
1  begin
2      Initiate drift detector DT;
3      Initiate Double Reservoir Classifier Selector DRCS;
4      Initiate Volatility measurement window W;
5      Let ActiveClassifier = StartingClassifier;
6      Let γ = Number of drifts detected when classifying the most T instances;
7      foreach example(x, y_k) ∈ S do
8          ActiveClassifier classifies (x, y_k);
9          if ActiveClassifier correctly classified (x, y_k) then
10             Let e = 0;
11         else
12             Let e = 1;
13         Input e into DT;
14         Update W and γ;
15         Let i = number of instances that has been classified;
16         if i%τ = 0 then
17             Input γ in DRCS (Call DRCS Sampling);
18             Query DRCS for the suggestion (Call DRCS Suggesting);
19             Let SuggestedClassifier = Suggested Classifier of DRCS;
20             if SuggestedClassifier is not null AND is not CurrentClassifier
               then
21                 if SuggestedClassifier is HighVolClassifier then
22                     Set ActiveClassifier = HighVolClassifier;
23                 else
24                     Set ActiveClassifier = LowVolClassifier;
25                 Re-initiate ActiveClassifier;
26         Train ActiveClassifier with (x, y_k);
27     end for
28 end
```

computational cost compared with the adaptive learner (HAT-ADWIN). We also inspect if VACS switches classifiers as expected.

We developed the mutating random tree generator based on the random tree generator presented in [4]. Our generator randomly chooses a branch of the tree and rebuilds it when we want to add a concept drift in the synthetic stream.

The synthetic data has 10 attributes and 2 classes. The maximal depth of the random tree is 5. We add 5% noise to the data. The synthetic data stream is made up of 28 blocks. Each block has 1 million instances. We have two types

of blocks: high volatility blocks containing 50 concept drifts, and low volatility block containing 5 concept drifts. We interleave high volatility blocks and low volatility blocks such that the stream fluctuates between high volatility and low volatility over time. We generated three types of datasets. (1) balanced volatility periods streams: they contain equal numbers of high and low volatility blocks (2) majority of low volatility periods streams: they contain 8 high volatility blocks and 20 low volatility blocks (3) majority of high volatility periods stream: they contain 20 high volatility blocks and 8 low volatility blocks. For each pattern, we generate 20 stream samples with different random seeds, and we run experiments on each of them.

Experimental results of 20 sample streams are shown in Table 1. The bold font denotes the worst performance mean among the three algorithms. Generally, all experiments show that the prediction accuracy of VACS is close to the prediction accuracy of HAT-ADWIN in drifting periods. So it has similar stability to HAT-ADWIN when drifts occur. However, compared with HAT-ADWIN, VACS effectively saves on training time and memory usage. Results show that VACS has a better average prediction accuracy than VFDT-ADWIN and a slightly worse prediction accuracy than HAT-ADWIN. This is the expected result since

Table 1. Performance on mutating random tree generator datasets

Dataset	Balanced		Majority low vol		Majority high vol	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
VACS						
Acc %	84.86	0.28	86.71	0.456	83.43	0.38
dAcc %	81.2	0.41	81.85	0.49	80.9	0.55
Mem (B)	413178.93	30237.52	515455.88	48911.55	338391.59	22452.62
Max Mem (B)	2200813.6	540997.42	2372156	430484.32	2108248	480220.7
Time (s)	230.31	5.83	190.78	2.75	256.93	6.11
VFDT-ADWIN						
Acc %	**83.5**	0.24	**85.62**	0.5	**81.78**	0.4
dAcc %	**78.85**	0.37	**79.45**	0.6	**78.85**	0.37
Mem (B)	275669.66	29343.87	392995.25	55100	194866.07	24074.3
Max Mem (B)	2068718.4	519383.28	2274702	485691.92	1987230.4	469304.53
Time (s)	123.72	2.04	126.31	2.62	116.47	2.34
HAT-ADWIN						
Acc %	85.77	0.25	87.68	0.41	84.06	0.29
dAcc %	81.8	0.41	82.35	0.49	81.3	0.47
Mem (B)	**775754.48**	86571.52	**1115474.7**	158771.26	**520137.73**	77229.29
Max Mem (B)	**5139417.6**	1143583.34	**5754436**	1218915.82	**4763293.2**	959179.97
Time (se)	**364.69**	6.55	**392.04**	11.84	**333.41**	8.87

VACS's prediction accuracy is produced by the hybrid system composed of HAT-ADWIN and VFDT-ADWIN.

4.2 Evaluation of the Classifier Switch Quality

The objective of these experiments is to evaluate whether VACS applies appropriate learners as expected. We test VACS with the three different types of datasets from the mutating tree generator. The results are shown in Table 2. Results show decent PICECs for all types of data. PICECs for all datasets' experiments reach around 90%. The percentage of applying the low volatility learner and the high volatility learner are also consistent with the type of data. For example, in streams with a majority of low volatility periods, 72% of the instances are classified by the low volatility classifier (VFDT-ADWIN) and 28% of the instances are classified by the high volatility classifier (HAT-ADWIN).

Table 2. VACS behaviour on different streams

Dataset	Balanced		Majority low vol		Majority high vol	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
PICEC %	89	1.8	94	1.5	88	2.4
Low vol learner usage %	52	2.1	72	1.3	37	1.9
High vol learner usage %	48	2.1	28	1.3	63	1.9

4.3 Experiments on Different Volatility Measurement Window Size

The objective of these experiments is to inspect the influences of different Volatility Measurement Window Size T on the learner selection quality of VACS. In previous experiments, we use 300,000 as the default value for T. In this experiment, we vary T by both increasing and decreasing from its default value. We run VACS on 20 samples of the balanced volatility stream. The experimental results are shown in Table 3.

We can summarise that classifier selection quality of VACS is influenced by the volatility measurement window size T. Setting the value T to either too small or too large may cause low PICEC, which means a poor classifier selection quality. From our experiments, we found that T should be large enough such

Table 3. VACS with different volatility measurement window size

T	PICEC % (Mean)	PICEC % (Std dev.)
30000	50	0
100000	88	1.6
300000	89	1.8
3000000	29	4.3

that the measurement is not strongly influenced by the volatility fluctuation of randomness. Also, T should not be too large such that VACS can always measure one volatility level at a time in a stream with volatility changing between different levels. One suggestion for selecting T is to run a test on the stream before starting VACS. A user can start from a small T value and gradually increase T meanwhile evaluating whether measurements are susceptible to volatility fluctuations caused by noise. When VACS performs as expected, stop increasing T and use this value. We assume that the T value obtained in the pre-experiment period from a continuous stream is also appropriate in the upcoming period in that stream. This assumption holds true in all of our experiments.

4.4 Experiments on SEA Generator Datasets

We use SEA Generator as our second data generator from MOA. The SEA generator has 3 attributes and 2 classes by default. We generate the same three types of volatility-changing data streams as the previous experiments. The experimental results can be viewed in Table 4. The bold font denotes the worst performance metrics among three algorithms. Results show a similar conclusion for VACS reducing training time. VACS uses less training time than HAT-ADWIN.

Table 4. Performance on SEA generator datasets

Dataset	Balanced		Majority of low vol		Majority of high vol	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
VACS						
Acc %	92.14	0.08	92.62	0.07	91.69	0.19
dAcc %	90.85	0.37	90.9	0.31	90.75	0.44
Mem (B)	**173849.41**	3951.98	172856.88	2138.16	**163834.01**	2091.83
Max Mem (B)	572069.6	56281.64	576905.2	58845.55	532899.6	57047.26
Time (s)	127.33	4.33	105.41	2.39	143.82	3.3
VFDT-ADWIN						
Acc %	**92.09**	0.08	**92.52**	0.07	**91.67**	0.16
dAcc %	**90.85**	0.37	**90.8**	0.41	**90.65**	0.49
Mem (B)	56843.41	1871.21	63692.83	1191.53	41673.5	1123.55
Max Mem (B)	186323.2	15153.59	177403.6	9168.92	175032	5283.07
Time (s)	64.33	1.64	65.06	1.04	62.41	1.26
HAT-ADWIN						
Acc %	92.2	0.08	92.66	0.07	91.71	0.19
dAcc %	90.85	0.37	90.9	0.31	90.75	0.44
Mem (B)	161413.3	4963.52	**201208.75**	3811.19	112874.55	4215.42
Max Mem (B)	**666700**	71667.62	**688492.4**	88055.91	**637625.2**	67176.21
Time (s)	**188.22**	3.38	**209.84**	3.4	**171.86**	2.82

Table 5. VACS behaviour on different streams (SEA)

	Balanced		Majority of low vol		Majority of high vol	
	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
PICEC %	90	2.6	94	1.6	91	2.7
Low vol learner usage %	49	2.9	71	1.4	33	1.8
High vol learner usage %	51	2.9	29	1.4	67	1.8

Moreover, in streams composed of a majority of low volatility periods, VACS has the most effective reduction in time compared with HAT-ADWIN.

We also evaluate VACS's classifier switch quality in these datasets. We show the results in Table 5. VACS in all three types of datasets return around 90% values, which is similar to earlier experiments in mutating random tree datasets. This tells us the classifier switch quality of VACS is also decent on SEA generated datasets.

5 Experiments on Real-World Data

The aim of this set of experiments is to evaluate whether VACS can achieve the cost reduction compared with the adaptive learner on real-world datasets. We chose Poker Hand, Forest Cover Type, and Airlines real world datasets available on MOA, which contain volatility changes. After testing, we chose volatility measurement window size $T = 20,000$ and we set λ to 3 for VACS. Table 6

Table 6. Performance with real-world datasets

Dataset	Poker Hand	Forest Cover Type	Airline
VACS			
Acc %	72.0	82.14	64.98
Mem (B)	**124684.11**	**176295.35**	2642422.83
Max Mem (B)	**163392**	**357528**	10468520
Time (s)	4.29	11.62	10.14
VFDT-ADWIN			
Acc %	69.68	81.77	65.28
Mem (bytes)	28676.98	67016.01	1832277.51
Max Mem (B)	40192	131256	7855680
Time (s)	2.36	5.94	9.05
HAT-ADWIN			
Acc %	**66.42**	**81.42**	**63.37**
Mem (B)	23741.62	98633.23	**7024953.36**
Max Mem (B)	83600	304024	**13914984**
Time (s)	**5.12**	**14.37**	**15.82**

shows the experiment's results. We also plot the measured volatility level γ and the mean of double reservoirs in Fig. 2 to demonstrate the classifier switching behaviour of VACS. When γ is greater than the reservoirs' mean, VACS applies the high volatility classifier (HAT-ADWIN). When γ is lower than the mean, VACS uses the low volatility classifier (VFDT-ADWIN).

In all experiments, VACS achieves a training time reduction compared with HAT-ADWIN. In experiments with Poker Hand and Forest Cover Type, VACS has the highest prediction accuracy.

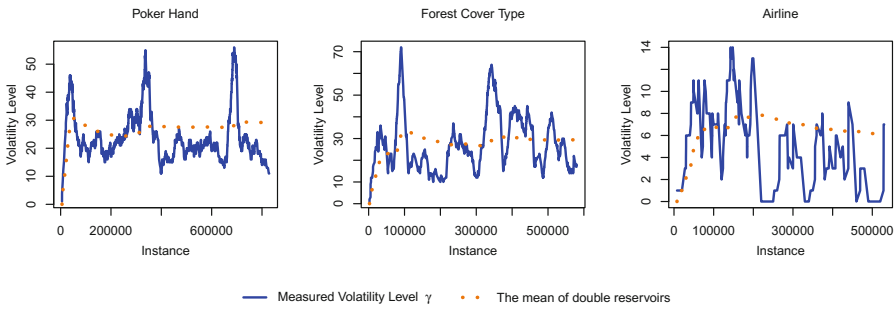

Fig. 2. Volatility measurements in the real world datasets

6 Conclusions and Future Work

We developed a system, called VACS, that can automatically choose the most suitable classifier between adaptive and non-adaptive algorithms in real time when mining a stream with changing volatility. The system applies the adaptive learner when the volatility is high and the non-adaptive learner when the volatility is low. It aims to reduce the learning costs while maintaining high enough prediction accuracy. We tested VACS on both synthetic and real-world data with changing volatility. In all of our experiments, VACS reduces the training time compared with the state-of-art adaptive learner, and VACS's prediction accuracy is also close or better than the adaptive learner. Through the experiments, we have shown that VACS is an effective approach to balance the trade-off between model prediction quality and efficiency.

One possible improvement is to enable VACS to adjust its volatility measurement window size T automatically when mining. The window is expected to shrink its size when the volatility changes notably and enlarge its size when the volatility is stable. When the volatility changes notably, it can remove the outdated elements by shrinking the window size such that it can react quickly to the changed volatility level. It can improve the accuracy of volatility measurement.

References

1. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: SDM, vol. 7, pp. 2007. SIAM (2007)
2. Bifet, A., Gavaldà, R.: Adaptive learning from evolving data streams. In: Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (eds.) IDA 2009. LNCS, vol. 5772, pp. 249–260. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03915-7_22
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**(May), 1601–1604 (2010)
4. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80. ACM (2000)
5. Gonçalves, P.M., de Barros, R.S.M.: RCD: a recurring concept drift framework. Pattern Recogn. Lett. **34**(9), 1018–1025 (2013)
6. Huang, D.T.J., Koh, Y.S., Dobbie, G., Pears, R.: Detecting volatility shift in data streams. In: 2014 IEEE International Conference on Data Mining, pp. 863–868. IEEE (2014)
7. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 97–106. ACM (2001)
8. Ng, H.T., Goh, W.B., Low, K.L.: Feature selection, perceptron learning, and a usability case study for text categorization. In: ACM SIGIR Forum, vol. 31, pp. 67–73. ACM (1997)
9. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. (TOMS) **11**(1), 37–57 (1985)
10. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 226–235. ACM (2003)