Research Article

Adaptive Ensemble with Human Memorizing Characteristics for Data Stream Mining

Yanhuang Jiang,^{1,2} Qiangli Zhao,^{1,3} and Yutong Lu^{1,2}

¹State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China

²School of Computer Science, National University of Defense Technology, Changsha 410073, China

³School of Computer and Information Engineering, Hunan University of Commerce, Changsha 410205, China

Correspondence should be addressed to Qiangli Zhao; zhao-qiangli@163.com

Received 9 January 2015; Accepted 18 May 2015

Academic Editor: Haibo He

Copyright © 2015 Yanhuang Jiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Combining several classifiers on sequential chunks of training instances is a popular strategy for data stream mining with concept drifts. This paper introduces human recalling and forgetting mechanisms into a data stream mining system and proposes a Memorizing Based Data Stream Mining (MDSM) model. In this model, each component classifier is regarded as a piece of knowledge that a human obtains through learning some materials and has a memory retention value reflecting its usefulness in the history. The classifiers with high memory retention values are reserved in a “knowledge repository.” When a new data chunk comes, most useful classifiers will be selected (recalled) from the repository and compose the current target ensemble. Based on MDSM, we put forward a new algorithm, MAE (Memorizing Based Adaptive Ensemble), which uses Ebbinghaus forgetting curve as the forgetting mechanism and adopts ensemble pruning as the recalling mechanism. Compared with four popular data stream mining approaches on the datasets with different concept drifts, the experimental results show that MAE achieves high and stable predicting accuracy, especially for the applications with recurring or complex concept drifts. The results also prove the effectiveness of MDSM model.

1. Introduction

Classification is one of the main applications of machine learning. Traditional classification methods are devoted to static environment where the whole training data is available to a learning system. However, recently new applications require that the learning systems work in dynamic environments, where data comes continuously with high speed as data streams [1]. Examples include social media mining, web log analysis, spam categorization, and network stream monitor. These data streams are often characterized by huge volumes of instances, rapid arrival rate, and drifting concept. The learning system must adapt to recent data in order to provide continuous high predicting performance. Compared with static environment, data stream mining is subject to the following constraints [2, 3]: (1) inspect an input instance at most once; (2) use a limited amount of memory; (3) work in a limited amount of time; (4) be ready to predict at any time.

Concept drifting is a hot issue of data stream mining, which can be categorized into sudden, gradual, and recurring drifts. These different concept drifts are often mixed with one another in a real application. A good learning system should adapt to different concept drifts, especially for the complex drifts in real applications. In recent years, many approaches have been proposed to handle data streams with concept drifting, which include sliding window approaches [4, 5], drift detecting techniques [6–8], and adaptive ensembles [2, 9–14].

Sliding window approaches use traditional batch algorithms to produce stream classifiers through sliding window techniques [4, 5]. The sliding window limits the number of instances to the most recent ones, and a batch algorithm is adopted to generate a classifier for the instances in each window. In sliding window approaches, only the most recent classifier will be used for prediction. The classifier built on a small window will react quickly to changes but may lose on

accuracy in stable periods. While the classifier built on a large window will fail to adapt to rapidly changing concepts. To deal with this problem, researchers proposed some heuristic methods which can adjust the window size dynamically [5].

Drift detecting techniques use a drift detector to test whether the class distribution remains constant or not over time [6–8]. If a drift is detected and reached to a warning level, the current classifier will be dropped and a new classifier will be generated from the instances stored in a separated “warning” window. Drift detecting approaches are suitable for the applications with sudden drifts. While for the gradually changing concepts, the approaches may not detect the changes since the drifts cannot trigger the warning level.

Adaptive ensembles generate component classifiers sequentially from fixed-size blocks of training instances called data chunks. When a new data chunk arrives, existing components are evaluated and a new classifier is generated. To predict a new instance, the predicting results of all reserved classifiers will be combined to give a final result. Most adaptive ensembles, such as SEA [2], AWE [9], and ACE [10], only reserve a limited number of classifiers and use all the reserved classifiers for the next prediction. The evaluation value of each classifier contains no history importance of the classifier since it is only related with the most recent data chunk. The classifiers with the lowest evaluation value will be removed from the reserved ensemble when the ensemble is oversized, which makes these algorithms sensitive to sudden concept drifts. Learn++.NSE [11] and Bagging++ [12] are the other kind of chunk-based ensemble algorithms in which no pruning is used to limit the number of component classifiers. This makes them requiring much memory and testing time. AUE [13, 14] uses incremental algorithms to create component classifiers instead of static batch learners, so it can well adapt to gradual drifts as well as sudden drifts. While the capability of incremental learning limits the usable algorithms for component classifier generation, the weight of each classifier in AUE only gives the temporal importance of the classifier as the other ensemble approaches.

This paper studies new techniques for the ensemble-based data stream mining, in which a batch learner is used to generate component classifiers. Inspired by human recalling and forgetting mechanism, we propose a new model, MDSM (Memorizing based Data Stream Mining), which introduces human memorizing characteristics into data stream mining. In this model, we look upon each component classifier as a piece of knowledge stored in human memory. Each component classifier is associated with a value of memory retention, which reflects the usefulness of the component classifier in its history. MDSM provides a knowledge repository where the classifiers whose memory retentions are high enough will be reserved. For prediction, only the most useful classifiers are selected (recalled) from the repository and work as an ensemble, which makes the ensemble adapt to concept drifts quickly. In MDSM model, a component classifier which has low accuracy for current data trunk can still be reserved in knowledge repository if its memory retention is high enough. This prevents useful classifiers from being discarded when sudden concept drifts occur and improves the stability of data stream mining. Based on MDSM, a new algorithm, MAE

(Memorizing based Adaptive Ensemble), is put forward. The algorithm uses Ebbinghaus forgetting curve as the forgetting mechanism and ensemble pruning as the recalling mechanism. Experiments results show that, compared with other four popular approaches, MAE achieves better predicting accuracy.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 proposes MDSM model. Section 4 presents our algorithm, MAE, in detail. Section 5 gives our experimental setup. Experimental results and analysis are shown in Section 6. Finally, we draw conclusions and discuss future works.

2. Related Work

Most adaptive ensembles for data stream mining generate one classifier for each data chunk by using a batch learner. The system maintains a limit-sized ensemble, which is used for prediction process and updated after each new classifier is generated. SEA (Stream Ensemble Algorithm) [2] is the first method for creating adaptive ensembles, which uses C4.5 to generate component classifiers. When an ensemble reaches a given size, the new classifier will replace the worst component of the ensemble. SEA uses majority voting for ensemble prediction. AWE (Accuracy Weighted Ensemble) [9] is also among the most representative methods, in which each component is associated with a weight, and the component with the lowest weight is discarded when component replacement is needed. AWE adopts weighted voting for ensemble prediction, which improves the adaptability for concept drifts. ACE (Adaptive Classifiers Ensemble) [10] introduces a drift detector in its learning system, and new classifiers will be generated when concept drifts are detected. To achieve high accuracy, no pruning or replacement mechanism is provided in ACE, which always makes the ensemble clumsy after several periods of learning.

For convenient description, we abstract a formal description for above algorithms. Let DB be the current data chunk of a data stream. When the data chunk DB comes, a learning algorithm will generate a new classifier c from DB . The learned classifier c is put into a classifier set ES :

$$\begin{aligned} c &= \text{Learn}(DB), \\ ES &= ES \cup \{c\}. \end{aligned} \quad (1)$$

Then, all of the classifiers in ES are evaluated according to the current data chunk DB :

$$\mathbf{W} = \text{Evaluate}(ES, DB) \quad (2)$$

$\mathbf{W}$ is a vector where each classifier c_i has a corresponding value w_i in the vector. It is noticeable that the evaluation vector is only related to the current ES and DB . It has no relation with former evaluate values (or the history importance of the classifiers). The classifier with the lowest evaluation value will be removed from ES when the size of ES is over a given size k . That is, the ES satisfies the following condition:

$$|ES| \leq k. \quad (3)$$

When a prediction task comes, ES is returned as an ensemble for instance prediction. For an unknown instance X , its predicting result is computed as follows:

$$P(X) = \text{Ensemble}(ES, \mathbf{W}, X). \quad (4)$$

For SEA, the $\mathbf{W}$ has no use for prediction since majority voting is used to derive the final result. For the other adaptive ensembles, each component in $\mathbf{W}$ is the voting weight of the corresponding classifier.

These traditional ensemble approaches for data stream mining have the following limitations: (1) unstable for the tasks with complex random or sudden concept drifts: the evaluation and replacement of component classifiers is on the basis of the most recent data chunk, while the weakest component for current data chunk may not mean it is useless for the future. If the data chunk includes random drifts (such as noises) and frequently happening sudden drifts, removing the weakest component for current data chunk is not a good idea, especially when the component has been proved to be useful by many prior data chunks; (2) slowly respond to concept drifts: some components in ES may have no contributions or even worse effects on current prediction task, which will degrade the performance and efficiency of the ensembles. In the context of concept drifts, using these “bad” components for prediction may result in low accuracy and slow response to concept drifts.

In most real data stream applications, concept drifts are very complex and happen frequently, which mixed with different type of drifts, such as sudden, gradual, recurring, short-time, long-time drifts, and even noise drifts. To mine data streams with complex concept drifts, algorithms with high adaptability and stability are required.

3. MDSM: A New Data Stream Mining Model

To overcome the limitations of the existing ensemble approaches for data stream mining, we put forward a new learning model to mine chunk-based data streams.

3.1. Human Recalling and Forgetting Mechanisms. As we know, recalling and forgetting mechanisms play an important role in human learning. For a piece of knowledge, the more it is recalled, the more it is apt to be used in future. Similarly, once learned, the less it is used, the more possible it will be forgotten.

Hermann Ebbinghaus is a German psychologist who pioneered the experimental study of human memory. In 1885, he conducted experiments on himself to understand how long the human mind retains information over time [15]. The experimental results can be plotted on a graph what is now known as “Ebbinghaus forgetting curve” (see Figure 1). Ebbinghaus discovered that a human being is apt to rapidly forget the knowledge he has just learned from some material, but review of the material over time makes the knowledge more stable in memory. Apparently, reviewing the material will recall the corresponding knowledge at the same time. This extrapolated the hypothesis of the exponential nature of

FIGURE 1: Ebbinghaus forgetting curve.

human forgetting and the relationship between recalling and forgetting mechanisms.

The forgetting curve shows some important characteristics of human learning: (1) the memory retention of a piece of knowledge is declined in time when there is no attempt to recall it; (2) recalling a piece of knowledge will strengthen the knowledge in the memory; (3) the more stable is a piece of knowledge in memory, the longer period of time that a person is able to recall it.

3.2. MDSM Model. Inspired by the characteristics of human recalling and forgetting, we proposed a new model, MDSM (Memorizing based Data Stream Mining), for data stream mining. The main innovation of MDSM model is that it looks on data stream mining as human memorizing process, which includes the following contributions: (1) *knowledge*: MDSM looks upon a data chunk as an event or material that can be learned by people, and each component classifier generated from a data chunk as a piece of learned knowledge; (2) *memory retention and knowledge repository*: each classifier has a corresponding memory retention value; the higher the memory retention value, the more important the corresponding classifier is; the classifiers with high memory retention will be kept in a “knowledge repository”; that means they are memorized by the system; the number of component classifiers in the repository is limited by a specific “memory capacity”; (3) *recalling mechanism*: when each new data chunk comes, the classifiers in the “knowledge repository” will be tested for recalling; the recalled classifiers will be used for next prediction tasks and their memory retention will be increased; the rest classifiers will not participate in next predictions, and their memory retention will be declined; but they still have the chance to be recalled by the following data chunks since they are in the knowledge repository; (4) *forgetting mechanism*: when the number of classifiers in the knowledge repository reaches the memory capacity of the system, the classifier with the lowest memory retention value will be discarded and removed from the repository; the discarded classifiers are forgotten by the system and will never be recalled any more.

In MDSM model, when a new data chunk DB comes, the learning system generates a classifiers c from DB and puts it into the knowledge repository MS ;

$$\begin{aligned} c &= \text{Learn}(DB), \\ MS &= MS \cup \{c\}. \end{aligned} \quad (5)$$

Then, the most useful classifiers in MS which get the best predicting result for DB are recalled:

$$ES = \text{Recall}(MS, DB, k). \quad (6)$$

In (6), k is the maximum number of classifiers that can be recalled; that is,

$$|ES| \leq k. \quad (7)$$

According to the recalled results ES , the memory retention of each classifier in the MS is updated as follows:

$$\mathbf{W} = \text{Evaluate}(MS, \mathbf{H}), \quad (8)$$

where $\mathbf{W}$ is a memory retention vector where each classifier in MS has a corresponding memory retention value. $\mathbf{H}$ contains the history information for each corresponding classifier. $\mathbf{H}$ is updated according to the current history information $\mathbf{H}'$ and the recalled result ES ; that is,

$$\mathbf{H} = \text{Update}(ES, \mathbf{H}'). \quad (9)$$

Equations (8) and (9) tell us that the memory retention value of each classifier is decided by its recalled results in its history.

After the evaluation process, the system checks whether the number of classifiers in MS is oversized. If it is, the classifier with the lowest memory retention will be discarded (forgotten) in order to satisfy

$$|MS| \leq m, \quad (10)$$

where m is the “memory capacity.” It is obvious that $k \leq m$.

When a prediction task comes, ES is returned as current target ensemble for instance prediction. For an unknown instance X , its predicted result $P(X)$ is computed as follows:

$$P(X) = \text{Ensemble}(ES, X). \quad (11)$$

Figure 2 depicts the structure of a learning system based on MDSM model. There are four main parts in the learning system, where data collection and preprocessing, evaluation and optimization, and prediction and application are the same as other learning systems. We mainly describe the part of Data Stream Mining.

The part of MDSM based data stream mining is responsible for learning classifiers from data chunks, and managing the classifiers. There are three units in this part. The unit of “Classifier generation” builds classifiers from data chunks by using some kind of learning algorithm, such as decision tree, neural network, and SVM. This unit corresponds to the “learning knowledge” ability of a human being. The unit of “Classifier management” corresponds to the “recalling

FIGURE 2: Structure of a MDSM based learning system.

and forgetting mechanisms” of human beings. The forgetting mechanism is a classifier evaluation process, which is responsible for updating memory retention of the reserved classifiers, and forgetting the knowledge with the lowest memory retention. The recalling mechanism is a classifier selection process, which recalls knowledge through selecting the most useful classifiers from the “knowledge repository” for incoming data chunks. The “knowledge repository” is a classifier set which keeps the classifiers with high memory retention. That is, these classifiers are memorized by the system. To predict new instances, only the current recalled classifiers compose the target ensemble. So, just like human, the MDSM model has the functions of learning knowledge, recalling knowledge, and forgetting knowledge.

The memory retention value of each classifier indicates the history importance of the classifier. It is not a temporal importance of the classifier as traditional adaptive ensembles. In MDSM model, the memory retention value of each component classifiers is only used to decide whether the classifier should be reserved (memorized) in the knowledge repository or discarded (forgotten) from the repository.

Traditional adaptive ensembles, such as SEA, AWE, and ACE, can be looked as special examples of MDSM model where $ES = MS$ and the memory retention values of component classifiers only indicates the temporal importance of the classifiers.

4. MAE Algorithm

Based on MDSM model, a new algorithm, MAE (Memorizing Based Adaptive Ensemble), is proposed. The main characteristics of MAE include the following. (1) It uses Ebbinghaus Forgetting Curve as the forgetting mechanism to update the memory retention values of component classifiers. (2) It adopts ensemble pruning as the recalling mechanism to select related classifiers for current data chunk. Ensemble pruning selects part of classifiers for prediction instead of combining all learned classifiers directly. It is an effective way to improve the predicting performance in machine learning field [16–18]. (3) When a prediction task is coming, majority voting is adopted to combine the predicting results of all classifiers in the target ensemble ES .

Input: S : data stream of instances
 m : memory capacity, the maximum number of component classifiers in the knowledge repository.
 k : the maximum number of classifiers that can be recalled

```

(1) Initialization:  $MS \leftarrow \emptyset$ ;  $\alpha \leftarrow 1$ ;  $t = 0$ ;
(2) for all data chunks  $DB_t \in S$  do
  (2.1)  $c \leftarrow$  a new component classifier built on  $DB_t$ ;
  (2.2) Initialize parameters for classifier  $c$ :
         $w_c \leftarrow 1$ ;  $R_c = 0$ ;  $\tau_c = t$ ;  $f_c = \alpha$ ;
  (2.3) Add  $c$  to the knowledge repository:
         $MS \leftarrow MS \cup \{c\}$ ;
  (2.4)  $ES =$  ensemble-pruning ( $MS, DB_t, k$ );
  (2.5) for all classifiers  $c_i \in ES$  do
    (2.5.1)  $\tau_{ci} = t$ ;
    (2.5.2)  $R_{ci} = R_{ci} + 1$ ;
    (2.5.3) compute the forgetting factor of  $c_i$  based on (13);
  (2.6) end for
  (2.7) for all classifiers  $c_j \in MS$  do
    (2.7.1) update the memory retention value of  $c_j$  based on (12);
  (2.8) end for
  (2.9) if  $|MS| > m$  then remove the classifier with the lowest memory retention value from  $MS$ ;
  (2.10)  $t = t + 1$ ;
(3) end for

```

ALGORITHM 1: Memorizing based Adaptive Ensemble.

As traditional chunk-based approaches, MAE also divides a data stream into equally sized blocks (data chunks). A new classifier is generated for each new block, and the evaluation of all classifiers is performed after processing all instances from the block. The memory retention of a classifier is decided by two factors: (1) the number of recalled times for the classifier; (2) the time interval from the last recall to current. In MAE algorithm, the memory retention value of a classifier c is calculated as follows:

$$w_c = e^{-f_c \cdot (t - \tau_c)}, \quad (12)$$

where w_c denotes the memory retention of classifier c , f_c is the forgetting factor of c , τ_c denotes the last recalling time of c (or the building time of c if c got no recall), and t is the current time. The forgetting factor is calculated as follows:

$$f_c = \frac{\alpha}{R_c + 1}, \quad (13)$$

where R_c is the number of recalled times for classifier c in its history and α is the initial forgetting factor for a new generated classifier. The forgetting factor of a classifier will be reduced after each recall. That is, each recall will strengthen the memory retention of the corresponding classifier and make it harder to be forgotten. In MAE algorithm, the history information for classifier c includes f_c and τ_c .

When a new data chunk comes, MAE generates a new classifier and adds it to the knowledge repository. Then an ensemble pruning process is carried out on all classifiers in the repository, which uses the new data chunk as the validation set. Ensemble pruning selects useful classifiers on the basis of the validation set. In MAE, the classifiers being selected are the knowledge recalled by the MDSM

system, and their history information, forgetting factors and last recalling time, are updated. Then the memory retention values of all classifiers in the knowledge repository are computed.

If the size of the knowledge repository is larger than the “memory capacity” m , the classifier with the lowest memory retention value is removed from the repository. That means it is forgotten by the system forever. Algorithm 1 shows the pseudocode of MAE algorithm. We set the default value of α (initial forgetting factor) be 1; while m and k are adjustable parameters whose values can be set by users.

MAE applies the current recalled classifiers (namely, the classifiers in ES) on prediction tasks and uses simple majority voting to combine the predicting results of all classifiers in ES .

5. Experimental Setup

We compared our MAE algorithm with four traditional data stream algorithms, Win, SEA, AWE and ACE on 15 large datasets. In our experiments, we uses synthetic datasets with different type of concept drifts to test the adaptability of the algorithm to specific drifts. Real datasets are used to test the usability of the compared algorithms in real applications.

5.1. Datasets. All the tests are performed on 15 datasets. SEA, Tree, Hyp, RBF, and LED are synthetic datasets and generated through MOA framework [19]. All the other 10 datasets are real datasets, where Elec dataset comes from [20] and the other 9 datasets come from UCI machine learning repository [21]. Table 1 summarizes the characteristics of each dataset used in our experiments.

Each synthetic dataset has a different type of concept drift. For the real datasets, we do not know what concrete drifts they

TABLE 1: Data sets description.

Dataset	Number of insts.	Number of attrs.	Number of cls.	Noise	Number of drifts	Drift type	Application area
SEA	1,000,000	3	4	10%	3	Sudden	Artificial data
Tree	100,000	10	6	0%	15	Recurring	Artificial data
Hyp	1,000,000	10	2	5%	1	Gradual	Artificial data
RBF	1,000,000	20	4	0%	4	Gradual	Artificial data
LED	1,000,000	24	10	10%	3	Gradual	Artificial data
Adult	48,842	14	2			Unknown	Census Income
Bank	41,188	20	2			Unknown	Bank marketing
Conn	67,557	42	3			Unknown	Connect-4
Cover	581,012	53	7			Unknown	Covertime
Elec	45,312	7	2			Unknown	Electricity market
EEG	14,980	14	2			Unknown	EEG Eye State
page	5,473	10	5			Unknown	Page blocks
Person	164,860	7	11			Unknown	Person Activity
Poker	1,025,010	10	10			Unknown	Poker game
Robot	5,456	24	4			Unknown	Robot

contain and when the drifts occur. But in most cases, a real stream dataset contains complex and time-related concept drifts which are hardly described. One concept may appear again after some period of time. Artificial datasets are useful to test the adaptability of an algorithm to a specific concept drift, while real datasets are more useful to test the usability of the algorithms.

5.2. Methodology. In our experiments, each dataset is divided into data chunks and input as a data stream. We use C4.5 decision tree to learn one classifier for each chunk [22]. To make the comparison more meaningful, the chunk size was set to the same value for all the tested algorithm. The previous research shows that 500 is a good choice for the chunk size [2, 9]. Our experiments results got the same conclusion that 500 is a good chunk size for all the tested algorithms, which achieves good balance between drift adaption and predicting accuracy. So we set chunk size to be 500 in our experiments.

Win is a simple sliding window algorithm. Its window size is set to the chunk size 500. When a new data chunk comes, it generates a decision tree for the data chunk and uses this decision tree for prediction. The previous learned decision tree is discarded when the new one is built.

SEA, AWE, ACE, and MAE are all adaptive ensemble algorithms. We set the maximum number of classifiers in the target ensemble (parameter k) to be the same value for all compared ensemble algorithms.

For ACE, its original algorithm requires too much memory space. We remove the component classifiers with the lowest weight to keep the maximum number of classifiers be k instead of keeping all classifiers as the original algorithm in reference [10]. This updating increases the computing efficiency and predicting accuracy for ACE algorithm.

For MAE, it uses MDSQ as ensemble pruning algorithm to select (*recall*) k classifiers from knowledge repository for each incoming data chunk [23]. The initial forgetting factor was set to the default value $\alpha = 1$. For memory capacity m , its value affects the work time of forgetting mechanism. Let

N be the number of accessed instances when the forgetting process is started; we have

$$N = m \cdot |DB|. \quad (14)$$

That means the forgetting mechanism works only for the datasets which have more than N instances. Considering that the number of instances in most of real datasets are not very large, we set $m = 50$ for all datasets in our experiment.

We evaluate the performance of the tested algorithms from three aspects: predicting accuracy, training time, and predicting time. In our experiments, all algorithms worked as the test-then-train paradigm for each data chunk. That is, when a new data chunk comes, it is used to test the predicting performance of the ensembles at first and then used to generate a new component classifier and update the evaluation values of component classifiers. The results of chunk predicting accuracy, chunk training time, and chunk predicting time of each dataset are the average values of the corresponding results among all data chunks.

Our previous work, LibEDM, is an open source library for Ensemble Based Data Mining [24, 25]. We implemented the library in C++ language for performance purpose. It achieves much better performance than popular JAVA based machine learning software, such as MOA [19] and WEKA [26]. To reduce computing time, we choose LibEDM as software platform and implement all the tested algorithms in C++ language. All the implementations have already been included in LibEDM library [25]. Our experiments were run on a computer equipped with two 4-core 2.2 GHZ Intel processors, 32 GB RAM, and Linux operating system.

6. Results and Discussion

In our experiments, all the tested algorithms were compared from three facets: predicting accuracy, training time, and predicting time. Before doing that, we set a test to choose a suitable value for the maximum size of target ensemble (parameter k) at first.

TABLE 2: Results of average predicting accuracy and variance on all data chunks [%].

Dataset	Win	SEA	AWE	ACE	MAE
SEA	84.58 $\pm$ 2.30	87.68 $\pm$ 2.14	87.98 $\pm$ 2.14	87.14 $\pm$ 2.89	86.54 $\pm$ 2.28
Tree	62.37 $\pm$ 9.59	71.36 $\pm$ 15.41	79.65 $\pm$ 10.65	76.71 $\pm$ 10.81	81.70 $\pm$ 8.34
Hyp	76.02 $\pm$ 3.66	84.29 $\pm$ 2.33	83.85 $\pm$ 2.39	82.01 $\pm$ 3.28	83.86 $\pm$ 2.91
RBF	83.49 $\pm$ 5.68	93.02 $\pm$ 5.80	93.59 $\pm$ 4.34	91.30 $\pm$ 4.79	91.09 $\pm$ 4.42
LED	40.36 $\pm$ 22.17	49.75 $\pm$ 23.88	50.97 $\pm$ 22.80	50.00 $\pm$ 22.93	50.80 $\pm$ 22.72
Adult	92.81 $\pm$ 2.30	94.38 $\pm$ 1.64	94.46 $\pm$ 1.65	94.29 $\pm$ 1.66	94.56 $\pm$ 1.65
Bank	87.36 $\pm$ 14.73	88.73 $\pm$ 14.23	88.62 $\pm$ 14.75	88.96 $\pm$ 13.87	88.89 $\pm$ 13.99
Conn	64.99 $\pm$ 15.91	67.67 $\pm$ 15.53	67.43 $\pm$ 17.46	68.52 $\pm$ 15.99	71.33 $\pm$ 13.05
Cover	88.79 $\pm$ 9.30	82.31 $\pm$ 10.98	84.18 $\pm$ 8.91	66.97 $\pm$ 17.01	87.92 $\pm$ 9.05
EEG	59.62 $\pm$ 31.08	54.31 $\pm$ 26.85	59.03 $\pm$ 31.07	64.19 $\pm$ 34.66	62.05 $\pm$ 30.18
Elec	73.79 $\pm$ 10.33	76.68 $\pm$ 9.48	76.08 $\pm$ 9.83	75.26 $\pm$ 11.64	77.28 $\pm$ 8.79
Page	91.56 $\pm$ 5.57	93.78 $\pm$ 4.58	93.81 $\pm$ 4.66	91.73 $\pm$ 6.10	93.78 $\pm$ 4.58
Person	30.24 $\pm$ 23.93	27.81 $\pm$ 21.66	29.19 $\pm$ 22.96	25.55 $\pm$ 24.64	34.31 $\pm$ 22.40
Poker	75.04 $\pm$ 17.34	69.74 $\pm$ 23.01	75.14 $\pm$ 19.04	69.65 $\pm$ 16.36	81.30 $\pm$ 16.89
Robot	77.50 $\pm$ 15.16	83.04 $\pm$ 15.51	84.17 $\pm$ 16.16	75.60 $\pm$ 18.26	83.04 $\pm$ 15.51
Mean	72.54 $\pm$ 12.48	75.02 $\pm$ 12.82	76.61 $\pm$ 12.48	73.88 $\pm$ 13.54	77.97 $\pm$ 11.66

FIGURE 3: Mean results of average chunk accuracies for different k .

6.1. Maximum Size of Target Ensemble. The maximum size of target ensemble (parameter k) effects the performance of adaptive ensembles greatly. To get a suitable k , we set k to be 5, 10, 20, 30, and 50, respectively, and got the mean results of the average chunk accuracies on all datasets for all tested algorithms.

Figure 3 illustrates the mean accuracy results for different k . We can see that, among all tested algorithms, MAE got the best results for all different k . For MAE and AWE, the best size of the target ensemble is $k = 10$, while for SEA and ACE, $k = 20$ get the best mean results. Considering that lower size will decrease predicting accuracy, and higher size will increase much computing time, we set $k = 10$ since it is a

good balance between computing efficiency and predicting accuracy for all the algorithms.

6.2. Predicting Accuracy. Table 2 lists the average predicting accuracy and variance on all data chunks for each dataset with $k = 10$. The last row lists the arithmetic mean of the results over all datasets. The best result for each dataset is highlighted using bold typeface.

From Table 2, we can see that MAE outperforms other algorithms on 6 out of 15 datasets on the average accuracies, including Tree, Adult, Conn, Elec, Person, and Poker. Where Tree contains the recurring concept drifts, the other five datasets are real applications with complex concept drifts. The reason is that MAE has recalling and forgetting mechanisms. The component classifiers with high predicting performance in the history will get high memory retention, so they will not be discarded from knowledge repository immediately even if they get low accuracy for current data chunk. When a former concept appears again, the reserved classifiers will be recalled from the knowledge repository, which improves the adaptability of the algorithm for recurring and complex-random concept drifts. For example, Person is a typical dataset that one concept (classes) has a time relation with another concept, and the same concept appears again after a period of time. MAE achieves much better results on this kind of dataset than the other algorithms. MAE also achieves the highest accuracy among the mean results over all datasets, and the mean variance result of MAE is the best one too.

AWE achieves the best accuracy results on 5 datasets, which are SEA, RBF, LED, Page, and Robot, respectively. Its mean result ranked the second. AWE uses weighted voting as ensemble strategy during prediction. For the applications with gradual or sudden concept drifts, this ensemble strategy achieves good predicting accuracy. The reason is that, the new generated component classifier will get the highest weight for

TABLE 3: Hypothesis and values of Bergmann-Hommel test.

Number	Versus	z	p	APV
0	Win versus MAE	3.9260	$8.638e - 005$	0.0008638
1	Win versus AWE	3.2332	0.001224	0.007346
2	ACE versus MAE	2.6558	0.007912	0.04747
3	SEA versus MAE	1.9630	0.04965	0.1986
4	Win versus SEA	1.9630	0.04965	0.1986
5	AWE versus ACE	1.9630	0.04965	0.1489
6	Win versus ACE	1.2702	0.204	0.408
7	SEA versus AWE	1.2702	0.204	0.408
8	AWE versus MAE	0.6928	0.4884	0.9768
9	SEA versus ACE	0.6928	0.4884	0.9768

FIGURE 4: Graphical representation of Bergmann-Hommel test.

gradual or sudden concept drifts, which makes AWE adapt to these drifts quickly. While the adaptability of AWE is a little worse than MAE for recurring and complex concept drifts.

SEA outperforms the other algorithms on Hyp datasets, and its mean accuracy result is the third one. SEA does not consider the history importance of component classifiers as MAE algorithm and does not set voting weights for component classifiers as AWE, so its predicting accuracy is worse than these two algorithms.

ACE uses a detector to test whether a concept drift exists in the data stream. In fact, it is impossible that a drift detector is suitable for different kinds of drifts. Our experimental results show that ACE outperforms the other four algorithms on Bank and EEG datasets, and the mean accuracy result of ACE is only better than that of Win.

Win is a simple sliding window algorithm which only uses the recently generated classifier to predict new instances. In our experiment, it only outperforms the other algorithms on Cover dataset. The mean accuracy result of Win is significantly lower than the other algorithms. This showed us that adaptive ensemble is a good strategy for data stream mining.

Bergmann-Hommel test is an exhaustive statistical procedure for testing $n \times n$ comparisons [27, 28]. Table 3 lists the result of Bergmann-Hommel test, where z is the test statistics. The z value is used to find the corresponding probability (p value) from the table of normal distribution, which is then compared with an appropriate level of significance. The significance level was set to 0.025 in our experiment.

Figure 4 shows the graphical representation of Bergmann-Hommel test. We can see that the predicting accuracy results of MAE are significantly better than SEA, ACE, and Win, while the average results of MAE and AWE are in the same group. Table 2 shows that AWE achieves

FIGURE 5: Chunk accuracy results of Elec dataset.

FIGURE 6: Chunk accuracy results of Conn dataset.

better accuracy than MAE on three artificial datasets and two real datasets, while these two real datasets, robot and page, have small number of instances, and the forgetting mechanism of MAE did not work on them. For larger real datasets, MAE achieves better results than AWE. This may show that MAE is more suitable for the real applications with large volume of continuous data.

Figures 5 and 6 illustrate the accuracy results on each data chunk for Elec and Conn datasets, respectively. Since both of them have relative small number of instances, the figures can illustrate the accuracy results clearly. From these two figures, we can see that our algorithm MAE achieves better results than the other algorithms not only on accuracy but also on stability. The range of its accuracy deviation is smaller than the other four algorithms.

6.3. Training Time. In our experimental tests, all the compared algorithms are implemented in C++ language and achieve good computing efficiency. Table 4 reports the results of average trunk training time in 10^{-3} seconds for these algorithms. Each result is the average training time of all data chunks for the corresponding dataset. The last line lists the arithmetic mean results on all datasets for each algorithm.

From Table 4, we can see that the training time results of Win are the lowest since it only learns a classifier for each

TABLE 4: Results of average chunk training time [10^{-3} second].

Dataset	Win	SEA	AWE	ACE	MAE
SEA	1.43	6.93	4.44	14.99	14.75
Tree	4.86	10.95	8.42	37.01	19.38
Hyp	6.39	12.24	9.67	42.88	20.70
RBF	16.50	22.23	19.71	77.93	30.80
LED	5.74	12.68	9.56	23.81	22.30
Adult	2.79	6.06	5.73	28.96	10.11
Bank	1.86	7.96	5.81	13.84	10.95
Conn	4.56	9.11	7.29	21.05	14.04
Cover	6.96	12.62	10.14	80.92	20.56
EEG	2.41	5.11	3.97	13.76	7.85
Elec	2.10	7.08	4.89	20.02	11.15
Page	3.02	7.15	4.78	12.09	4.73
Poker	2.19	7.90	5.40	48.55	16.38
Person	2.63	9.87	7.61	15.25	12.33
Robot	9.82	13.01	11.24	20.72	13.54
Mean	4.88	10.06	7.91	31.45	15.30

TABLE 5: Results of average chunk testing time [10^{-6} second].

Dataset	Win	SEA	AWE	ACE	MAE
SEA	0.30	2.00	2.50	4.50	2.00
Tree	0.50	3.50	3.50	21.50	3.55
Hyp	1.50	3.50	4.00	20.50	3.50
RBF	0.50	3.55	3.50	33.00	3.05
LED	0.50	4.00	4.00	5.00	3.50
Adult	0.50	2.20	3.50	4.50	3.50
Bank	1.50	17.20	19.50	153.00	23.00
Conn	0.45	14.76	23.10	33.40	22.10
Cover	0.30	0.50	0.40	1.76	0.40
EGG	1.30	3.20	4.50	5.00	4.50
Elec	22.50	41.90	33.00	92.70	33.00
Page	10.50	45.00	43.00	123.00	75.00
Person	0.30	11.50	11.70	16.50	12.30
Poker	2.50	4.70	3.10	9.10	3.30
Robot	0.30	35.30	32.50	136.00	62.50
Mean	2.90	12.85	12.79	43.96	17.01

data chunk and no other special operation is required. AWE consumes additional time to calculate the weight for each classifier in the target ensemble, so it consumes more training time than Win. The evaluation process of SEA requires more time than that of AWE, whose training time result is ranked the third. The detector of ACE is very consumable which makes its training process requiring a lot of time. In our experiments, ACE is the slowest algorithm. MAE must do an ensemble pruning process, and update the memory retention for each classifier in the knowledge repository. Its training time results are higher than those of Win, SEA, and AWE, but much less than those of ACE. For all of the datasets in our experiment, MAE took about 10~30 milliseconds to finish the training process for one data chunk, which is fast enough for most of online applications.

6.4. Predicting Time. Table 5 lists the predicting time results in 10^{-6} seconds for all compared algorithms. The result is the average time for predicting one data chunk. The last line lists the mean results on all datasets for each algorithm.

Table 5 shows that Win got the lowest predicting time. The reason is that it only uses one classifier to predict data chunks. Followed by AWE, SEA, and MAE, these three algorithms took almost the same predicting time. The predicting time results of ACE are the highest among all algorithms. For all algorithms, their chunk predicting time results are about one thousandth of the corresponding training time results.

7. Conclusions

In this paper, we proposed a new model, MDSM, for data stream mining, and put forward an algorithm, MAE, based on this model. The main contribution of MDSM model is that it introduces human recalling and forgetting mechanisms to ensemble-based data stream mining systems. The novelty of MAE algorithm is that it uses Ebbinghaus forgetting curve

and ensemble pruning technique to implement the forgetting and recalling mechanisms of the MDSM model respectively.

The proposed MAE algorithm was compared with other state-of-art algorithms, including Win, SEA, AWE, and ACE. The experimental results have shown that, MAE outperforms the other algorithms on chunk predicting accuracy for data streams with concept drifts, especially for the data streams with recurring concept drifts and the real applications with complex concept drifts. The predicting performance of MAE is also more stable than other algorithms. In conclusion, MAE is a good data stream mining algorithm with high predicting accuracy and moderate training time. The experiments also proved the effectiveness of our MDSM model.

There are a lot of things waiting us to do in the future. Firstly, we will optimize the recalling and forgetting mechanisms in MAE algorithm to achieve better performance. Secondly, we plan to implement more data stream mining algorithms in our system and do a more sufficient performance comparison. Thirdly, semisupervised learning is a hot topic in many applications with big data [29–31]. Applying MDSM model for semisupervised data stream mining is another challenge.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant nos. 61272141, 61120106005, and 60905032, the National High Technology Research and Development Program of China (863) under Grant no. 2012AA01A301, and the Open Fund from HPCL under Grant no. 201513-02.

References

- [1] J. Gama, *Knowledge Discovery from Data Streams*, Chapman & Hall, London, UK, 1st edition, 2010.
- [2] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pp. 377–382, August 2001.
- [3] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.
- [4] E. Cohen and M. J. Strauss, "Maintaining time-decaying stream aggregates," *Journal of Algorithms*, vol. 59, no. 1, pp. 19–36, 2006.
- [5] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 443–448, April 2007.
- [6] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1-2, pp. 100–115, 1954.
- [7] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence—SBIA 2004: Proceedings of the 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29–October 1, 2004*, vol. 3171 of *Lecture Notes in Computer Science*, pp. 286–295, Springer, Berlin, Germany, 2004.
- [8] M. Baena-García, J. D. Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early drift detection method," in *Proceedings of the 4th International Workshop on Knowledge Discovery from Data Streams*, pp. 1–10, 2006.
- [9] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 226–235, August 2003.
- [10] K. Nishida, K. Yamauchi, and T. Omori, "ACE: adaptive classifiers-ensemble system for concept-drifting environments," in *Proceedings of the 6th International Workshop on Multiple Classifier Systems*, pp. 176–185, June 2005.
- [11] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.
- [12] Q. Zhao, Y. Jiang, and M. Xu, "Incremental learning by heterogeneous bagging ensemble," in *Advanced Data Mining and Applications: Proceedings of the 6th International Conference, ADMA 2010, Chongqing, China, November 19–21, 2010, Part II*, vol. 6441 of *Lecture Notes in Computer Science*, pp. 1–12, Springer, Berlin, Germany, 2010.
- [13] D. Brzezinski and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *Proceedings of the 6th International Conference on Hybrid Artificial Intelligent Systems (HAIS '11), II*, pp. 155–163, May 2011.
- [14] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: the accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014.
- [15] H. Ebbinghaus, *Memory: A Contribution to Experimental Psychology*, translated by: H. A. Ruger, C. E. Bussenius, 1885, <http://nwpsych.rutgers.edu/~jose/courses/578.mem.learn/2012/readings/Ebbinghaus.1885.pdf>.
- [16] A. Lazarevic and Z. Obradovic, "Effective pruning of neural network classifier ensembles," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, pp. 796–801, July 2001.
- [17] Q.-L. Zhao, Y.-H. Jiang, and M. Xu, "A fast ensemble pruning algorithm based on pattern mining process," *Data Mining and Knowledge Discovery*, vol. 19, no. 2, pp. 277–292, 2009.
- [18] G. Martinez-Muñoz, D. Hernández-Lobato, and A. Suarez, "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 245–259, 2009.
- [19] <http://moa.cms.waikato.ac.nz/>.
- [20] M. Harries, "SPICE-2 comparative evaluation: electricity pricing," Tech. Rep. 9905, School of Computer Science and Engineering, University of New South Wales, New South Wales, Australia, 1999.
- [21] UCI Machine Learning Repository, <http://archive.ics.uci.edu/ml/>.
- [22] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.
- [23] G. Martinez-Munoz and A. Suarez, "Aggregation ordering in bagging," in *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, pp. 258–263, 2004.
- [24] Q. Zhao and Y. Jiang, "LibEDM: a platform for ensemble based data mining," in *Proceedings of the IEEE International Conference on Data Mining Workshop (ICDMW '14)*, pp. 1250–1253, IEEE, Shenzhen, China, December 2014.
- [25] <https://github.com/Qiangli-Zhao/LibEDM>.
- [26] <http://www.cs.waikato.ac.nz/ml/weka>.
- [27] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [28] S. García and F. Herrera, "An extension on 'statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.
- [29] P. Zhang, X. Zhu, and L. Guo, "Mining data streams with labeled and unlabeled training examples," in *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM '09)*, pp. 627–636, December 2009.
- [30] J. Xu, H. He, and H. Man, "DCPE co-training: co-training based on diversity of class probability estimation," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '10)*, pp. 1–7, 2010.
- [31] Z. Ahmadi and H. Beigy, "Semi-supervised ensemble learning of data streams in the presence of concept drift," in *Hybrid Artificial Intelligent Systems: 7th International Conference, HAIS 2012, Salamanca, Spain, March 28–30th, 2012. Proceedings, Part II*, vol. 7209 of *Lecture Notes in Computer Science*, pp. 526–537, Springer, Berlin, Germany, 2012.