TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/11 pp379–388
DOI: 10.26599/TST.2018.9010119
Volume 24, Number 4, August 2019

A Classifier Using Online Bagging Ensemble Method for Big Data Stream Learning

Yanxia Lv, Sancheng Peng*, Ying Yuan, Cong Wang, Pengfei Yin, Jiemin Liu, and Cuirong Wang

Abstract: By combining multiple weak learners with concept drift in the classification of big data stream learning, the ensemble learning can achieve better generalization performance than the single learning approach. In this paper, we present an efficient classifier using the online bagging ensemble method for big data stream learning. In this classifier, we introduce an efficient online resampling mechanism on the training instances, and use a robust coding method based on error-correcting output codes. This is done in order to reduce the effects of correlations between the classifiers and increase the diversity of the ensemble. A dynamic updating model based on classification performance is adopted to reduce the unnecessary updating operations and improve the efficiency of learning. We implement a parallel version of EoBag, which runs faster than the serial version, and results indicate that the classification performance is almost the same as the serial one. Finally, we compare the performance of classification and the usage of resources with other state-of-the-art algorithms using the artificial and the actual data sets, respectively. Results show that the proposed algorithm can obtain better accuracy and more feasible usage of resources for the classification of big data stream.

Key words: big data stream; classification; online bagging; ensemble learning; concept drift

1 Introduction

At present, the influence of big data[1] on our daily lives has become more pervasive. In order to improve their performance, many types of big data require real-time processing, in order to obtain useful knowledge or require better insights to react quickly. The classical applications for big data include face recognition, advertisement recommendation, automatic driving, stock market predictions, cancer detection, and so on. Thus, learning on big data stream has attracted increasing research attention in the past years.

Current thinking suggests that big data refer to datasets whose sizes are beyond the ability of typical tools to capture, manage, and analyze[2]. In relation to this, several challenges for the processing of big data streams in the existing learning algorithms have been identified. Some of the challenges include the limited resource (time and memory) requirements and the required speed of execution. Another challenge is the evolution of underlying data distribution (known as "concept drift") resulting from the dynamic changes in user behaviors and network environments. The final challenge is the occurrence of several issues (e.g., overfitting, missing values, class imbalance, etc.) in batch learning and data stream leaning.

Ensemble methods have been extensively studied

• Yanxia Lv, Ying Yuan, Cong Wang, Jiemin Liu, and Cuirong Wang are with School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China. E-mail: lyx@neuq.edu.cn; yuanying@neuq.edu.cn; wangcong@neuq.edu.cn; ljm@neuq.edu.cn; wcr@neuq.edu.cn.
• Sancheng Peng is with Laboratory of Language Engineering and Computing, and also with School of Cyber Security, Guangdong University of Foreign Studies, Guangzhou 510006, China. E-mail: psc346@aliyun.com.
• Pengfei Yin is with School of Information Science and Engineering, Central South University, Changsha 410083, China. E-mail: pppypf@163.com.
* To whom correspondence should be addressed.
Manuscript received: 2018-07-07; accepted: 2018-09-01

and used in practical applications, which complete learning tasks by building multiple learners. This is the case in the big data stream environments[3]. Aside from exploiting weak learners and improving the accuracy[4], they can also be used to solve concept drifts[5,6], new class detection[7], feature drifts[8], and so on. Bagging[9]-, boosting[10]-, and Random Forest (RF)[11]-based algorithms are classical, frequently used ensemble methods.

Boosting-based algorithms combine multiple weak learners to obtain strong learners and mainly concern with reducing deviations, which can improve generalization ability. Bagging is a parallel ensemble learning method, in which the base learners are generated in parallel. Bagging is mainly concerned with reducing variance and obtaining good generalization ability. Meanwhile, RF is an extended variant of bagging which uses the decision trees to build the bagging ensemble. In the context of evolving data streams, there are fewer RF models than those based on the bagging and boosting methods.

In the evolving data stream setting, the bagging-based algorithms are more optimal than the boosting-based algorithms[12,13], the reason for which is also an open issue. In the current paper, we present a bagging-based online classification algorithm for the evolving big data streams. Our contributions are summarized as follows:

- We exploit the excellent performance of bagging and introduce different levels of diversity, in order to change the input space of single classifiers and to increase the diversity of the ensemble. We also tune the resampling weights with higher or lower diversity to obtain the most optimal accuracy for the current type of concept drift.
- We construct a change detection module for adapting to the concept drift. This is used to train a backup base classifier when a warning is alarmed, and replace the current model when the drift occurs instead of simply resetting the base classifiers simply.
- We design a robust method for voting the outputs. We use fuzzy output codes as a form of voting on the base classifiers instead of deterministic codes. This can be used to reduce the correlations among the base classifiers and to increase the ability of fault tolerance and the diversity of the ensemble.
- We implement a parallel version for our proposed ensemble classifier according to the Random Access Memory (RAM)-Hours and Central Processing Unit (CPU) time. This improves the efficiency of the algorithm, but does not affect the classification performance of the algorithm.

The remainder of this paper is organized into sections. We describe the related work in Section 2, and improve the online bagging algorithm in Section 3. In Section 4, we conduct the experimental evaluation. Finally, in Section 5, we conclude this paper and suggest possible directions for future studies.

2 Related Work

In online learning, ensembles of classifiers have been used to improve the performance of single classifiers successfully. Besides, the ensemble can add, remove, update, and reset the members dynamicly confronting concet drifts in data stream learning. The bagging- and boosting-based algorithms are the most frequently used ensemble methods for data stream leaning. Studies have shown that the performance of online bagging is better than that of the online boosting in most cases[14].

Breiman[9] proposed bagging as an aggregating method for classification, this method uses bootstrap aggregating[15] to improve accuracy. Fern and Givan[16] proposed an online bagging, which takes a fixed value as the weight of the upcoming instances to update the base classifier. However, experiment results show that its performance is not so good. The probability of each instance to be sampled for a training subset is approximately according to a Poisson distribution with different λ values, for example, $\lambda=1$[12], $\lambda=6$[17], and a changing λ value[18]. All these could change the diversity of the ensemble, while higher diversity corresponds to a lower average Q statistic, which indicates better accuracy[14]. Q statistic, proposed by Yule and Pearson[19], is widely used in diversity measurement in many fields. The range of Q statistic is -1 and 1. On the basis of an analysis from ten measures, they recommended the use of the Q statistic for the purpose of improving the accuracy of ensembles[14]. For two classifiers, D_i and D_j, the Q statistic can be calculated as

$$Q_{i,j}=\frac{N^{11}N^{00}-N^{01}N^{10}}{N^{11}N^{00}+N^{01}N^{10}} \tag{1}$$

where $N^{a,b}$ represents the amount of training instances that a and b come from classifiers D_i and D_j, respectively. When the classification of D is correct, the value of $Q_{i,j}$ is 1; otherwise, the value is 0. In this paper, the high or low average Q statistic corresponds to low or high diversity, respectively.

This paper introduces different diversities by

changing the λ values. As shown in Fig. 1, when $\lambda = 1$, 37% of the values that represent the probability of instances being sampled calculated by the Poisson distribution are 0, 37% are 1, and 26% are bigger than 1. The leveraging bagging algorithm[17] uses a λ value equal to 1 and the Diversity for Dealing with Drifts (DDD) algorithm[18] uses a changing λ value. We use a different λ value before and after a drift in order to improve the generalization ability on the old or new concepts, resulting in the finding that diversity by itself is conducive in reducing the initial increase in error caused by a drift.

In evolving data stream environments, ensembles use several methods for the detection of concept drift problems. For example, Bifet et al.[17] adopted the ADaptive WINdowing (ADWIN) algorithm and Minku and Yao[18] adopted the Early Drift Detection Method (EDDM) for the detection of concept drifts. An ensemble of classifiers, which is another method to cope with concept drifts, constantly resets the lower performance base classifiers with new ones when concept drift appears[20]. Our algorithm sets a warning and a drift threshold in each base classifier. Then, a backup base classifier is trained based on a current window when a waning is detected, and the current base classifier is replaced with the backup base classifier when a drift is detected. Meanwhile, the value of λ is reset according to the current drift type.

Dietterich and Bakiri[21] proposed the error-correcting code technique to improve the performance of multiclass problems by only using a new binary classifier. Their proposed method maps the instances category from original space to {0, 1}. Schapire[22] presented a boosting-based ensemble method using the variation of error correcting output codes, and Bifet et al.[17] also used a variation of error-correcting output codes to introduce the randomization. This approach can increase the diversity of ensemble and reduce the impact of correlations among the base classifiers. In the current work, we use fuzzy output codes as a voting strategy on the output of the ensemble, which can reduce the correlation among the base classifiers in the ensemble and increase the diversity of the classifiers.

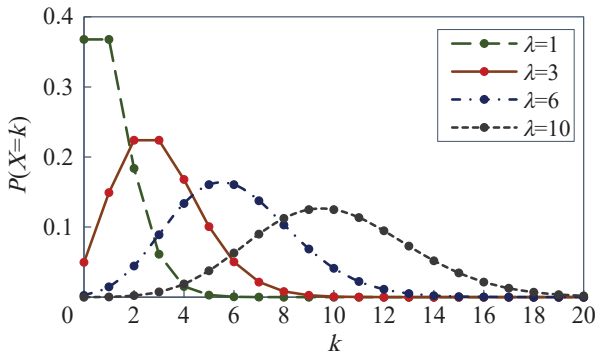

Fig. 1 Poisson distribution.

3 An Efficient Online Bagging Ensemble

In data stream classification, we train a model to predict a class label y for an unlabeled new instance x, which is a vector of d features. We assume that the actual class label of a new upcoming instance is available before the next instance arrives, so that it can be used for training immediately after it is used for testing.

3.1 Diversity for training space

Online bagging is a widely used ensemble learning algorithm in evolving data stream, not just because of its higher performance than single classifiers, i.e., there is no need to adjust complex parameters and easy parallelizing, but also because of its abilities to add, remove, and update base classifiers when drifts occur. The pseudo code of the original online bagging algorithm is shown in Algorithm 1. We use S to denote the data stream, Y to denote the set of class labels, M to denote the number of base model, and x to denote the features vector of instances. The learning environments can be explained as the number of instances tending to infinity in batch setting. At this point, the frequency w of each training instance appearing in each base classifier h_m approximately obeys the distribution of Poisson, wherein λ equals 1. When an instance is used to training, it will be used w times. The overall prediction class label for a new upcoming instance is given below.

$$h_o(x) = \text{argmax}_{y \in Y} \sum_{m=1}^{M} I(h_m(x) = y) \qquad (2)$$

where $I(\cdot)$ is the indicator function.

We use different λ values instead of 1 in Poisson distribution before and after concept drift to encourage different levels of diversity and obtain better accuracy.

Algorithm 1 Online bagging

Input: OnlineBagging(M, S, Y)
Output: predicted calss label(Majority vote)
Initialize M base classifier: $h_1, h_2, \ldots, h_M$;
while HasNext(S) do
  $(x, y) \leftarrow$ NextInstance(S);
  for $m \leftarrow 1, \ldots, M$ do
    $w \leftarrow Possion(1)$;
    Train h_m with instance using weight w;
return the overall class label through majority vote;

λ value is not a fixed value. In this way, higher or lower λ values are correlated to higher or lower average Q statistics[23], respectively. In turn, higher or lower average Q statistics, represents lower or higher diversity, respectively. This can modify the space of the training sets for sub-classifiers inside the ensemble.

3.2 Solution for the concept drift

In the evolving data stream setting, the algorithm should not only be accurate, but also be able to deal with concept drift. It is known to all that early instances correspond to outdated concepts, whereas new instances are associated with the latest concept in stream setting. The traditional way is to reset the worst classifier immediately when the drift is detected. The process of retraining a new classifier reduces the classification performance of the ensemble. Given that the new classifier has not been trained on any existing instances, predicting the new concept very well is impossible. Thus, instead of resetting the classifier when drifts occur, we use a more lenient threshold to represent the occurrence of a warning and train a backup classifier on the most recent instances alongside the ensemble without influencing the overall decisions. As shown in Fig. 2, whenever the drift is detected for a classifier, then it is replaced by its backup classifier. This approach has at least two advantages. First, it spends less time in positively influencing the overall ensemble decision because of the trained backup classifier. Second, the backup classifier is superior to existing classifiers as it is trained based on the latest instances.

The pseudo-code of our proposed algorithm is shown in Algorithm 2. We use a broad threshold θ_w to detect warnings and a narrow threshold θ_d to detect drift from lines 12 to 16. When a warning is detected, a backup classifier is trained and used to replace the bad one when drift is detected.

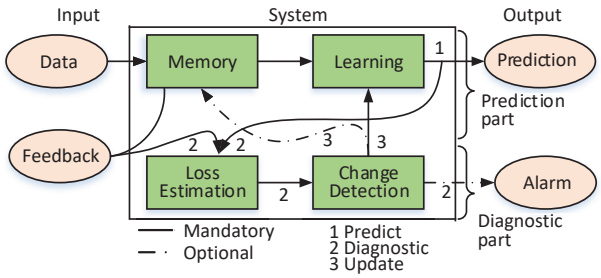

Fig. 2 A generic schema for an online adaptive learning algorithm.

Algorithm 2 Efficient online bagging

Input: EoBag$(M, S, \theta_w, \theta_d, Y)$
Output: predicted calss label
1 Initialize M classifier: $h_1, h_2, \ldots, h_M$ and weights of classifier: W_m;
2 Compute coloring $\mu_m : y \leftarrow \{0, 1\}$;
3 while HasNext(S) do
4 $(x, y) \leftarrow$ NextInstance(S);
5 for $m \leftarrow 1, \ldots, M$ do
6 $\hat{y} \leftarrow$ predict(h_m, x);
7 update class $\mu_m(y)$ and $W(h_m)$ with y and $\hat{y}$;
8 $w \leftarrow Possion(\lambda)$;
9 while $w > 0$ do
10 Train h_m with instance using weight w;
11 $w = w - 1$;
12 if $\theta_w <$ detect$(h_m, x, y) < \theta_d$ then
13 $b_m \leftarrow$ CreateBackClassifier();
14 if detect$(h_m, x, y) > \theta_d$ then
15 $h_m = b_m$;
16 update λ value with a greater one
17 return the overall class label through weighted majority vote;

3.3 Fuzzy output codes

In the traditional ensemble method, all the classifiers try to predict the same result. However, we use the error correction code to make each classifier with a different prediction function. This reduces the correlation among the individual classifiers in the ensemble and increases the diversity of the classifiers. In data stream classification setting, the error-correcting output codes are inspired by the error-correcting codes presented in Shannon's information theory[24].

We introduce coding ideas into class resolution and raise fuzzy output codes based on error-correcting output codes. This is expected to be a fault-tolerant approach in the decoding process. In detail, we assign a binary string of a specific length to each class label, each bit of the string is implemented in the following way: we select a binary value $\mu_m(y)$ for each classifier m and each class label y uniformly and randomly in line 2 of Algorithm 2.

The sequence of bit assignments for each of the class labels can be viewed as a "code word". A good code word for a k-class problem should meet two criteria. As shown in Table 1 for an ensemble of six classifiers in a classification task of three class labels, the first criterion is column independence, wherein each code word should be differentiated according to the Hamming distance; the other criteria is that f_i should be uncorrelated with f_j.

In Table 1, the binary string of Class A is 001110 and the first 0 corresponds to the code result of classifier 1 and so forth. Each classifier has a different

Table 1 Instance matrix of fuzzy output codes for 3 classes and 6 classifiers.

Classifier	Class A	Class B	Class C
f_1	0	0	1
f_2	0	1	1
f_3	1	0	0
f_4	1	1	0
f_5	1	0	1
f_6	0	1	0

prediction function. When a new instance is arriving, each classifier outputs a value of 0 or 1 and then the ensemble gets a binary string. Then, we compute the Hamming distance between this predicted string and all class label's code words, resulting in the identification of the closest class label. For a prediction instance x, if its real class label is C, the prediction result from f_1 to f_6 is 0, 1, 1, 0, 0, 0, respectively, among which f_5 incorrectly predicts 1 to 0. Next, we calculate the Hamming distance between 011000 and the code word of each class label. The closest one is Class C and the prediction is correct. In this paper, this is called "fuzzy voting", which uses the similarity degree of string to predict the classification result. We can see that this method possesses a certain ability of fault tolerance. The process is realized in lines 6 to 7 of Algorithm 2.

According to the efficient online bagging (hereafter referred to as EoBag), the overall prediction class label for a new coming instance is given below.

$$h_o(x) = \text{argmax}_{y \in Y} \sum_{m=1}^{M} W_m I(h_m(x) = \mu_m(y)) \quad (3)$$

3.4 Parallel implementation

In ensemble classification, the training of the base classifiers is usually the most time-consuming task, unless greedy learners are used in the ensemble. In big data stream scenarios, the base learners are repeatedly used for multiple tasks, such as tracking drifts and updating data structures that represent their weights.

In EoBag, the training of the base classifier includes the updating of drift detection parameters and the evaluation of the prequential accuracy on the base classifier, and the training of a new backup classifier if the drift warning appears. Each base classifier performs these operations independently, allowing it to design multi-threaded concurrent executions. In order to verify the advantages of this approach, we compare the parallel and serial versions of EoBag. In the experimental part, the comparison results indicate that the parallel version execution speed is significantly improved. At the same time, we simply parallelize the parts that are executed independently, so the classification performance is not affected.

4 Experimental Analysis

In this section, we evaluate the performance of the EoBag in the following three aspects: classification accuracy, resource usage (memory, time), and running time with experiments. Experiments comprise the comparison of our proposed algorithm against three other state-of-the-art ensemble classifiers and a most commonly used single classifier. To assess classification performance, we perform a 10-fold cross-validation prequential evaluation[25]. Experiments focusing on resource usage were run individually and repeated 10 times to eliminate fluctuations on the results. The number of base classifiers in all ensembles is set to 50, and the configuration of the base classifier for all ensembles is the same. We use four artificial and three real datasets to conduct our experimental analysis.

All experiments were configured and executed within the Massive Online Analysis (MOA) framework[26]. All algorithms evaluated in this paper were implemented in the Java programming language by extending the MOA software. The experiments were performed on a 1.9 GHz Core 16 Intel(R) Xeon(R) CPU E7-4820 machine with 32 GB RAM.

4.1 Experimental setting

Artificial data has several advantages: they are easier to reproduce and there is little cost in terms of storage and transmission. We consider four of the most likely used in the literature. The University of California Irvine (UCI) machine learning repository[27] contains some real-world benchmark data for evaluating machine learning techniques. We also consider three of the largest that have been thoroughly used in the literature to evaluate the classification performance of data stream classifiers: Forest Covertype, Electricity, and Poker Hand. Table 2 presents an overview of the data sets used in the current work.

4.2 Experimental results

We start our experimental evaluations by comparing variations of EoBag to evaluate its sensitivity to various parameters (e.g., drift and warning threshold, ensemble size, and subspace size) and variations of the algorithm

Table 2 Data sets.

Data set	Number of instances	Number of features	Type	Number of classes
LED	500 000	24	Artificial	10
SEA	500 000	3	Artificial	2
Hyperplane	500 000	10	Artificial	2
Random RBF	500 000	10	Artificial	5
Covertype	581 012	54	Real	7
Electricity	45 312	8	Real	2
PokerHand	1 000 000	10	Real	10

that deactivates some of its characteristics (e.g., drift detection and warning detection). The second set of experiments concerns the evaluation of computational resources usage (CPU time and RAM-Hours). Finally, we present experiments comparing the classification accuracy of efficient online bagging and other state-of-the-art ensemble classifiers.

4.2.1 EoBag variations

The first experiment is a comparison among six variations of the efficient online bagging algorithms (the following is expressed in EOBag), each of which "removes" some characteristics from EOBag (e.g., drift detection). We performed this comparison to illustrate the benefits of using EOBag, and to discuss how each strategy included contributes to the overall classification performance. Each variation configuration is presented below.

(1) **EoBag$_{\mathbf{m}}$** (m represents moderate) uses a more tolerant detection threshold to ADWIN, which brings fewer drifts or warnings ($\theta_w = 0.0001$ and $\theta_d = 0.000\,01$).

(2) **EoBag$_{\mathbf{f}}$** (f represents fast) uses a more stringent detection threshold to ADWIN, which brings more drifts or warnings (θ_w=0.01 and θ_d=0.001).

(3) **EoBag$_{\mathbf{nbk}}$** (nbk represents no backup classifier). There is no warning detection and backup classifier, that is, it will reset the corresponding classifiers immediately as soon as the drifts are detected.

(4) **EoBag$_{\mathbf{std}}$** (std represents standard). This is an original online bagging version as it does not detect a concept drift; thus, instead of resetting the classifiers, it uses majority vote strategies.

(5) **EoBag$_{\mathbf{maj}}$** (maj represents majority). This is consistent with EOBag$_{\mathrm{m}}$, the only difference is the use of the majority vote instead of the weighted majority.

As shown in Table 3, **EoBag$_{\mathbf{f}}$** is slightly better than **EoBag$_{\mathbf{m}}$** because the former can detect drift faster and train the backup classifier earlier to cope with concept drift. By comparing **EoBag$_{\mathbf{m}}$** with **EoBag$_{\mathbf{nbk}}$**, there

Table 3 Accuracy for EoBag variations. (%)

	Data set	EoBag$_{\mathrm{m}}$	EoBag$_{\mathrm{f}}$	EoBag$_{\mathrm{nbk}}$	EoBag$_{\mathrm{std}}$	EoBag$_{\mathrm{maj}}$
Artificial	LED	73.90	73.94	73.91	67.51	73.92
	SEA	89.66	89.67	89.66	87.80	89.66
	Hyperplane	90.26	90.56	90.53	83.68	90.26
	Random RBF	89.70	89.88	89.76	78.76	89.67
	Avg	85.88	86.01	85.97	79.44	85.88
Real	Covertype	92.32	91.86	92.35	88.20	92.31
	Electricity	88.92	89.45	88.95	86.83	88.80
	PokerHand	85.27	87.50	86.49	85.10	85.26
	Avg	88.84	89.60	89.26	86.71	88.79
	Overall_Avg	87.15	87.55	87.38	82.55	87.13

is no obvious difference in terms of the effect of using a backup classifier. This is because, in a large ensemble, it will not have a great impact on the overall results while only a few members have drift. However, compared with **EoBag$_{\mathbf{nbk}}$** which uses a completely new classifier to obtain a prediction, we can see that the performance of **EoBag** (**EoBag$_{\mathbf{m}}$** and **EoBag$_{\mathbf{f}}$**) with a backup classifier is slightly better. We can also see that the performance of an original bagging **EoBag$_{\mathbf{std}}$** without drift detection is relatively poorer compared with others. The reason is that using the classifier, which has been drifting, can have a negative impact on the classification result. Therefore, when the concept drift occurs, we either discard the classifier directly or replace the classifier with the backup classifier.

However, when the concept drift affects more members of the ensemble, the method of direct abandonment will reduce the scale of the ensemble and affect the classification performance. Finally, we observe that the weighted majority can obtain better classification performance than the simple majority voting in most cases. This is because the weights depend on the accuracy of the classification as the previous reset is more effective. However, it can be observed in some datasets that **EoBag$_{\mathbf{maj}}$** shows better performance than others. This is because, when the weight is overestimated or undervalued, it tends to have a bad effect on the classification results. Thus, the selection of the weight function should be as reasonable as possible. In general, the weighted voting is better.

4.2.2 Comparison between the parallel and serial versions

We compared the performance of the EoBag's parallel version and serial version in terms of resource usage. The sizes of the ensemble are 5, 10, 15, and 30. Figures 3a and 3b give the comparison of the experimental

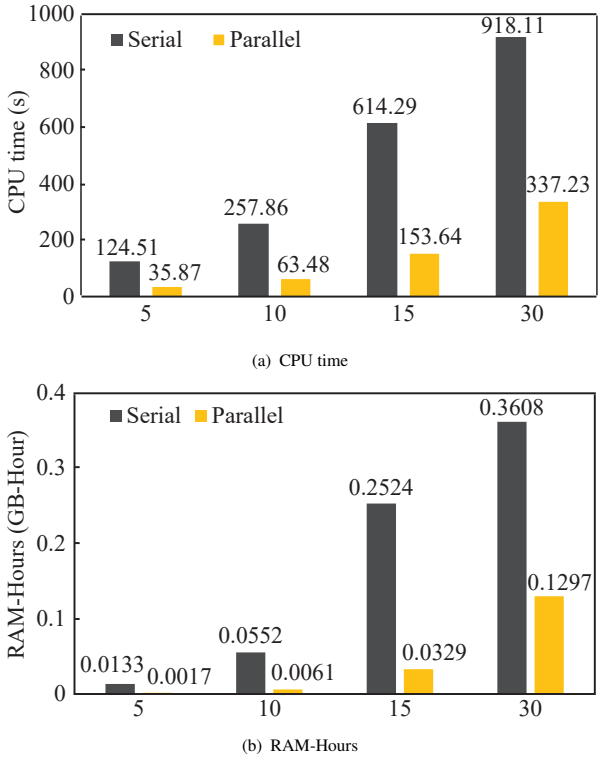

Fig. 3 Comparison between the paralell and serial versions in terms of CPU time and RAM-Hours, for 5, 10, 15, and 30 learners.

results in CPU time and RAM-Hours, respectively. The parallel version requires more memory than serial version, but it executes faster with an average RAM-Hours value that is much lower than that of the serial version. Ideally, under the circumstances of the parallel execution of independent tasks, our algorithm refers to the training of the base classifiers, and the number of parallel tasks can be scaled linearly according to the number of threads assuming that there are enough available threads. Nevertheless, some factors can impede the scalability of the EoBag's parallel implementation, such as the number of available threads and some operations that have not been parallelized (such as merging vote). In Figs. 3a and 3b, we can see that the gains are more prominent when the number of trees is equal to or greater than 15 (the number of available processors is 16), This is consistent with the expected result as all the base classifiers can be trained simultaneously.

4.2.3 Comparison between the EoBag and other classifiers

Next, we compare the overall classification performances in terms of accuracy of EoBag, Hoeffding tree, and other widely used ensemble algorithms. EoBag uses ADWIN for warning and drift detection with the parameter $\theta_w = 0.01$ and $\theta_w = 0.001$, respectively. Experiments were carried out on both artificial datasets and real datasets with concept drifts.

Figures 4–7 show the classification accuracy of EoBag and other algorithms on artificial datasets of the LED, SEA, Hyperplane, and Random RBF,

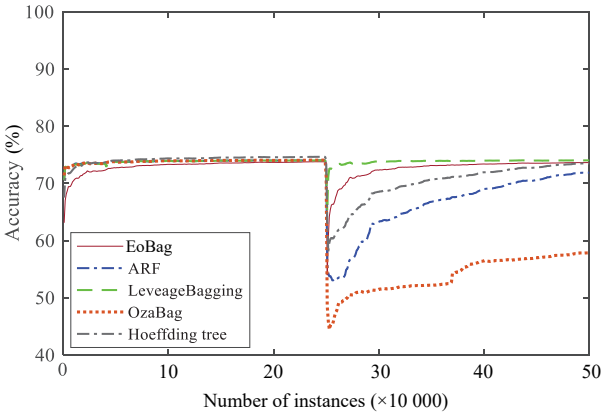

Fig. 4 Accuracy of the data set LED.

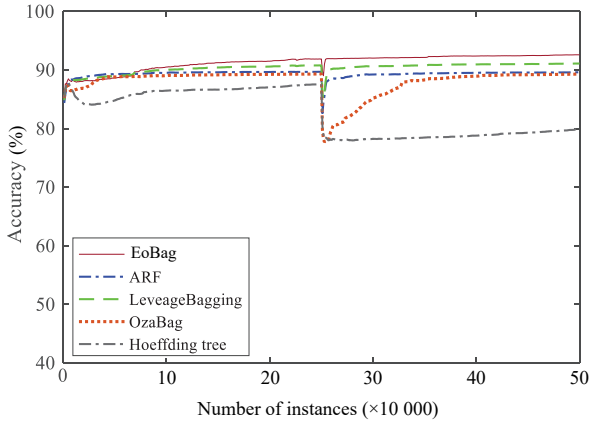

Fig. 5 Accuracy of the data set SEA.

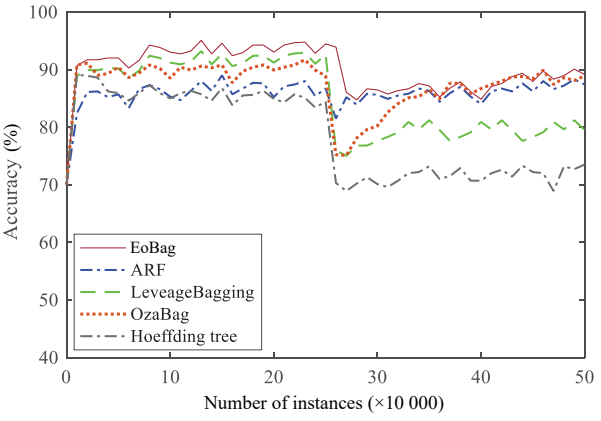

Fig. 6 Accuracy of the data set Hyperplane.

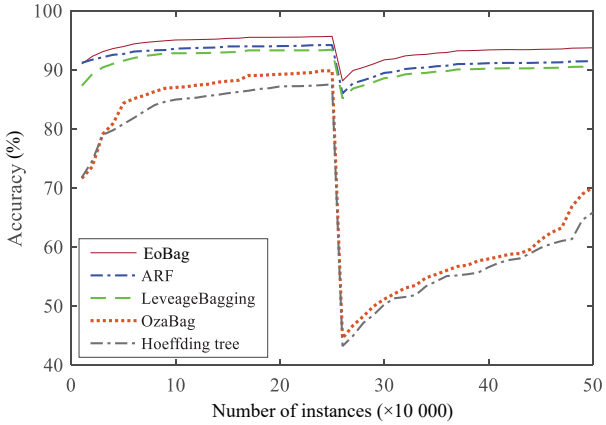

Fig. 7 Accuracy of the data set Random RBF.

respectively. The results show that the ensemble model has better performance than the single model in terms of classification accuracy and the ability to adapt to concept drift. In Fig. 4, before the concept drift, all algorithms perform almost the same, but after that, EoBag can recover quickly and reach the previous level. In Fig. 5, EoBag is comparable to ARF and LeverageBagging in the SEA dataset in terms of accuracy, but shows better performs than OzaBag in terms of adapting to the abrupt concept drifts. Figure 6 indicates that EoBag shows better accuracy than other algorithms, but ARF is less influenced by the concept drift and has the highest accuracy after concept drift. EoBag, ARF, and LeverageBagging obtain similar performances when used on the Random RBF dataset (Fig. 7), and all three perform much better than OzaBag and Hoeffding Tree when concept drifts occur.

In the real-world datasets, EoBag consistently performs the best when used on the Poker Hand dataset, as shown in Fig. 8 and shows almost the same performance as the ARF on the Electricity dataset (Fig. 9) and the LeverageBagging on the Forest

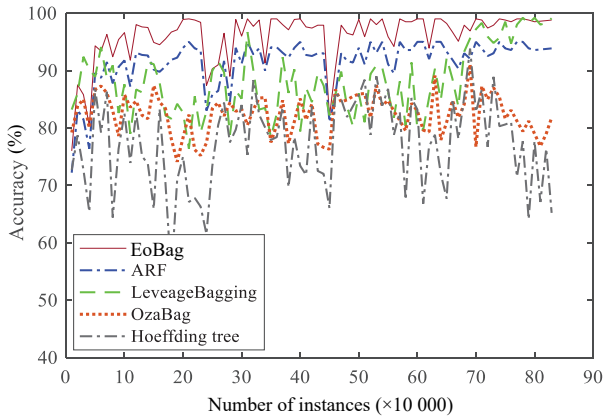

Fig. 8 Accuracy of the data set Poker Hand.

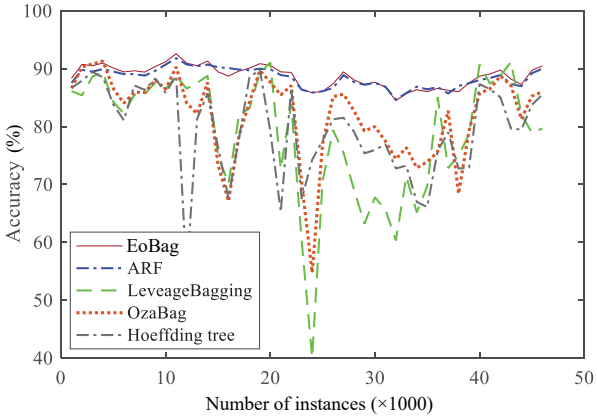

Fig. 9 Accuracy of the data set Electricity.

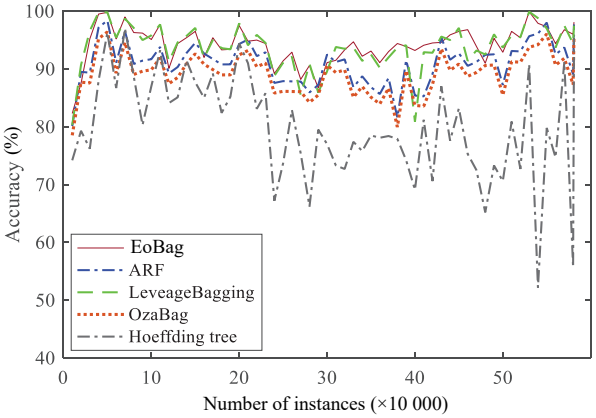

Fig. 10 Accuracy of the data set Forest Covertype.

Covertype dataset (Fig. 10). Compared with all the other algorithms, EoBag obtains the best performance and maintains stability when concept drifts occur.

In addition, we compared EoBag against some widely used ensemble classifiers in terms of CPU time and RAM-Hours with 50 base models in Tables 4 and 5, respectively. We can see that EoBag outperforms LeverageBagging and is close to OzaBag and ARF

Table 4 Compared EoBag against some widely used ensemble classifiers in terms of CPU time with 50 base models. (s)

	Data set	EoBag	Leveage-Bagging	OzaBag	ARF
Artificial	LED	**908.25**	3817.06	1088.65	1056.67
	SEA	1243.18	4687.76	1263.78	**1187.24**
	HyperPlane	**1198.57**	4798.76	1293.58	1978.91
	Random RBF	**2376.57**	7013.65	2478.87	2869.77
	Avg	**1431.64**	5079.31	1531.22	1773.15
Real	Covertype	**1783.07**	3365.78	2087.38	2034.04
	Electricity	398.42	589.66	**231.02**	467.18
	PokerHand	**899.97**	1264.86	1596.31	977.89
	Avg	**1027.15**	1740.10	1304.90	1159.70
	Overall_Avg	**1229.40**	3409.70	1418.06	1466.43

Table 5 Compared EoBag against some widely used ensemble classifiers in terms of RAM-Hours with 50 base modes. (GB-Hour)

	Data set	EoBag	Leveage-Bagging	OzaBag	ARF
Artificial	LED	**0.399**	0.412	1.765	0.467
	SEA	1.784	5.391	**0.418**	1.972
	HyperPlane	**1.031**	7.912	3.219	1.129
	Random RBF	**0.014**	0.476	0.601	0.019
	Avg	**0.087**	3.548	1.501	0.897
Real	Covertype	0.107	0.986	3.121	**0.028**
	Electricity	**0.001**	0.007	0.002	0.001
	PokerHand	**0.001**	0.002	0.001	0.001
	Avg	0.036	0.332	1.041	**0.010**
	Overall_Avg	**0.421**	1.940	1.271	0.453

algorithms in terms of CPU Time. Furthermore, EoBag is more efficient than OzaBag and LeverageBagging, and is similar to ARF in terms of RAM-Hours.

5 Conclusion and Future Work

Data stream classification has raised some research issues as it uses limited resources to adapt to a constantly evolving stream with data arriving at high speeds. Ensemble-based methods are appropriate choices for the evolving data streams, as they often obtain high performance of classification and adapt to some drifts. In this paper, we explored the problem of classification for big data stream. Different from previous modeling methods, we presented an improved online bagging algorithm for the classification of the evolving big data stream. The proposed method provides effective solutions, such as better accuracy and more feasible usage of resources, to the problem of big data stream classification. As for our further work, we will focus on improving the execution efficiency of the algorithm. In addition, we aim to propose a semi-supervised, weak-label classification strategy to deal with the real-world scenario.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Nos. 61702089, 61876205, and 61501102), the Science and Technology Plan Project of Guangzhou (No. 201804010433), and the Bidding Project of Laboratory of Language Engineering and Computing (No. LEC2017ZBKT001).

References

[1] S. C. Peng, G. J. Wang, and D. Q. Xie, Social influence analysis in social networking big data: Opportunities and challenges, *IEEE Network*, vol. 31, no. 1, pp. 11–17, 2017.

[2] J. L. Torrecilla and J. Romo, Data learning from big data, *Statistics and Probability Letters*, vol. 136, pp. 15–19, 2018.

[3] Z. Q. Wang, J. C. Xin, H. X. Yang, S. Tian, G. Yu, C. R. Xu, and Y. D. Yao, Distributed and weighted extreme learning machine for imbalanced big data learning, *Tsinghua Science and Technology*, vol. 22, no. 2, pp. 160–173, 2017.

[4] A. O. M. Abuassba, Y. Zhang, X. Luo, D. Z. Zhang, and W. Aziguli, A heterogeneous ensemble of extreme learning machines with correntropy and negative correlation, *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 691–701, 2017.

[5] H. M. Gomes and F. Enembreck, SAE2: Advances on the social adaptive ensemble classifier for data streams, in *Proc. 29th Annu. ACM Symp. Applied Computing*, Gyeongju, Republic of Korea, 2014, pp. 798–804.

[6] T. T. T. Nguyen, T. T. Nguyen, A. W. C. Liew, and S. L. Wang, Variational inference based bayes online classifiers with concept drift adaptation, *Pattern Recognition*, vol. 81, pp. 280–293, 2018.

[7] B. S. Parker and L. Khan, Detecting and tracking concept class drift and emergence in non-stationary fast data streams, in *Proc. 29th AAAI Conf. Artificial Intelligence*, Austin, TX, USA, 2015.

[8] J. P. Barddal, H. M. Gomes, F. Enembreck, and B. Pfahringer, A survey on feature drift adaptation: Definition, benchmark, challenges and future directions, *Journal of Systems and Software*, vol. 127, pp. 278–294, 2017.

[9] L. Breiman, Bagging predictors, *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] Y. Freund and R. E. Schapire, Experiments with a new boosting algorithm, in *Proc. 13th Int. Conf. Machine Learning*, Bari, Italy, 1996, pp. 148–156.

[11] L. Breiman, Random forests, *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] A. B. Owen and D. Eckles, Bootstrapping data arrays of arbitrary order, *Annals of Applied Statistics*, vol. 6, no. 3, pp. 895–927, 2012.

[13] N. C. Oza and S. Russell, Experimental comparisons of online and batch versions of bagging and boosting, in *Proc. 7th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Ming*, San Francisco, CA, USA, 2001, pp. 359–364.

[14] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet, A survey on ensemble learning for data stream classification, *ACM Computing Surveys*, vol. 50, no. 2, p. 23, 2017.

[15] S. D. Grimshaw, An introduction to the bootstrap, *Technometrics*, vol. 37, no. 3, pp. 340–341, 1995.

[16] A. Fern and R. Givan, Online ensemble learning: An empirical study, *Machine Learning*, vol. 53, pp. 71–109, 2003.

[17] A. Bifet, G. Holmes, and B. Pfahringer, Leveraging bagging for evolving data streams, in *Machine Learning and Knowledge Discovery in Databases*, J. L. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, eds. Springer, 2010, pp. 135–150.

[18] L. L. Minku and X. Yao, DDD: A new ensemble approach for dealing with concept drift, *IEEE Transactions on*

Knowledge and Data Engineering, vol. 24, no. 4, pp. 619–633, 2012.

[19] G. U. Yule and K. Pearson, On the association of attributes in statistics: With illustrations from the material of the childhood society, *Philosophical Transactions of the Royal Society of London*, vol. 194, nos. 252–261, pp. 257–319, 1900.

[20] D. Brzezinski and J. Stefanowski, Combining block-based and online methods in learning ensembles from concept drifting data streams, *Information Sciences*, vol. 265, pp. 50–67, 2014.

[21] T. G. Dietterich and G. Bakiri, Solving multiclass learning problems via error-correcting output codes, *Journal of Artificial Intelligence Research*, vol. 2, no. 1, pp. 263–286, 1994.

[22] R. E. Schapire, Using output codes to boost multiclass learning problems, in *Proc. 14th Int. Conf. Machine Learning*, San Francisco, CA, USA, 1997, pp. 313–321.

[23] L. L. Minku, A. P. White, and X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010.

[24] C. E. Shannon, A mathematical theory of communication, *Bell System Technical Journal*, vol. 27, pp. 623–656, 1948.

[25] A. Bifet, G. de Francisci, Morales, J. Read, G. Holmes, and B. Pfahringer, Efficient online evaluation of big data stream classifiers, in *Proc. 21th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Sydney, Australia, 2015, pp. 59–68.

[26] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, MOA: Massive online analysis, a framework for stream classification and clustering, *Journal of Machine Learning Research*, vol. 11, pp. 44–50, 2010.

[27] D. Newman, S. Hettich, C. Blake, C. Merz, and D. Aha, UCI repository of machine learning databases, http://archive.ics.uci.edu/ml/datasets.html, 2017.

Yanxia Lv received the BS and MS degrees from Yanshan University, China, in 2005 and 2009, respectively. She is currently working toward the PhD degree in the School of Computer Science and Engneering, Northeastern University, Shenyang, China. Her research interests include online learning, big data analysis, and machine learning.

Sancheng Peng received the PhD degree from Central South University, China, in 2010. He is a full professor with Laboratory of Language Engineering and Computing, and also with School of Cyber Security, Guangdong University of Foreign Studies, Guangzhou, China. His professional interests include social networking big data analysis and information security.

Ying Yuan received the PhD degree from Northeastern University, China, in 2015. She is an associate professor of the School of Computer Science and Engineering at Northeastern University. Her interests include data center networking analysis and machine learning.

Cong Wang received the PhD degree from Northeastern University, China, in 2011. He is an associate professor of the School of Computer Science and Engineering at Northeastern University. His interests include cloud computing, virtual network mapping, big data analysis, and machine learning.

Pengfei Yin received the BS degree from Jishou University in 2002, and MS degree from East China Normal University in 2008. He is currently working toward the PhD at the School of Information Science and Engineering, Central South University, Changsha, China. His research interests include information retrieval and recommender system.

Jiemin Liu is a full professor of the School of Computer Science and Engineering at Northeastern University. He received the PhD degree from Northeastern University, China, in 2008. His interests include next-generation network, machine learning, and big data analysis.

Cuirong Wang is a full professor of the School of Computer Science and Engineering at Northeastern University. She received the PhD degree from Northeastern University, China, in 2006. She is a senior member of CCF. Her interests include cloud computing, big data analysis, and distributed computing.