


# Classification of high-dimensional evolving data streams via a resource-efficient online ensemble

Tingting Zhai<sup>1</sup> · Yang Gao<sup>1</sup>  ·  
Hao Wang<sup>1</sup> · Longbing Cao<sup>2</sup>

Received: 30 March 2016 / Accepted: 27 February 2017 / Published online: 23 March 2017  
© The Author(s) 2017

**Abstract** A novel online ensemble strategy, ensemble BPegasos (EBPegasos), is proposed to solve the problems simultaneously caused by concept drifting and the curse of dimensionality in classifying high-dimensional evolving data streams, which has not been addressed in the literature. First, EBPegasos uses BPegasos, an online kernelized SVM-based algorithm, as the component classifier to address the scalability and sparsity of high-dimensional data. Second, EBPegasos takes full advantage of the characteristics of BPegasos to cope with various types of concept drifts. Specifically, EBPegasos constructs diverse component classifiers by controlling the budget size of BPegasos; it also equips each component with a drift detector to monitor and evaluate its performance, and modifies the ensemble structure only when large performance degradation occurs. Such conditional structural modification strategy makes EBPegasos strike a good balance between exploiting and forgetting old knowledge. Lastly, we first prove experimentally that EBPegasos is more effective and resource-efficient than the tree ensembles on high-dimensional data. Then comprehensive experiments

---

Responsible editor: Thomas Gärtner, Mirco Nanni, Andrea Passerini and Céline Robardet.

---

✉ Yang Gao  
gaoy@nju.edu.cn  
Tingting Zhai  
zhht\_nju@sina.com  
Hao Wang  
wanghao@nju.edu.cn  
Longbing Cao  
longbing.cao@uts.edu.au

<sup>1</sup> State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

<sup>2</sup> Advanced Analytics Institute, University of Technology Sydney, Sydney, Australia

on synthetic and real-life datasets also show that EBPegasos can cope with various types of concept drifts significantly better than the state-of-the-art ensemble frameworks when all ensembles use BPegasos as the base learner.

**Keywords** High dimensionality · Concept drift · Data stream classification · Online ensemble

## 1 Introduction

Mining evolving data streams has attracted numerous research attention recently (Zliobaite et al. 2015; Kreml et al. 2014; Zliobaite and Gabrys 2014; Zhang et al. 2014). In particular, mining high-dimensional evolving data streams is a challenging task, which aims to capture the latest functional relation between the observed variables and the target variable and then accurately predict the labels of the upcoming observed values. Compared with static data, streaming data has the following characteristics:

- *Infinite amount* Data is potentially endless, thus storing all data for multi-pass processing is impractical. To process infinite data, classifiers should work in resource-limited environments. Typically, the space cost for saving the models should be independent of the number of observations.
- *High velocity* Data arrives so quickly that real-time response is required. Algorithms relying on retraining new classifiers on the latest data from scratch may not be useful in such environments.

In addition to the above general characteristics, data considered in this work also possesses the following special characteristics:

- *Concept drifting* Data is not distributed independently and identically, but has evolving distributions (Gama et al. 2013). The relation between the observed variables and the target variable is expected to change with time unpredictably, while the distribution of the observed variables may either change or remain constant. Concept drifts affect decision-making; to handle them well, classifiers should be able to forget the old data and be consistent with the most recent data.
- *High dimensionality* Data is described by hundreds or thousands of attributes, such as in texts, images, sounds, videos, hyper-spectral data, financial data, etc. With so many attributes the challenge is well known as *the curse of dimensionality*. One effect is related to the scalability. Some algorithms become intractable in both time and space with increasing dimensionality. Another effect is connected with the scarcity of the available data because the number of instances required to represent any distribution grows exponentially with the number of attributes (Tomasev et al. 2014), while each concept in the stream may not contain so huge data in many applications. With few instances in high-dimensional space, many classification models are very likely to overfit the data, thereby leading to poor generalization performance (Pappu and Pardalos 2014).

The above characteristics appear in many real applications. For example, in sentiment classification on twitter streams (Bifet and Frank 2010), since millions of users utter their opinions and make comments now and then, the messages that need to be

processed flow infinitely and fast, meanwhile the vocabularies used to convey positive and negative feelings tend to evolve with different contexts. In visual object tracking (Liu and Zhou 2014), surveillance cameras produce a stream of images at a rate of many frames per second. With the variation of illumination or background in adjacent frames, both the features used to describe target and non-target objects change, the same as the class boundary. In a personalized email filtering system (Katakis et al. 2009), a user receives a stream of emails over time. Whether an email is a spam depends on the user's interest, which may drift from time to time. In all the above scenarios, data is related to high dimensionality.

Although many methods have been proposed for data stream classification, they generally focus on dealing with either concept drifting or high dimensionality. Few works consider both of them. Methods handling only concept drifting of streaming data can be classified as single-model and multi-model approaches (Hosseini et al. 2015). Single-model approaches are mainly incremental counterparts of conventional batch learning methods, equipped with certain strategies for coping with concept drifts. For example, methods based on decision trees (Gama et al. 2006; Bifet et al. 2010c; Rutkowski et al. 2013),  $k$ -nearest neighbors (Bifet et al. 2013), and support vector machines (SVM) (Wang et al. 2012) have been respectively explored. Multi-model approaches are namely adaptive ensembles which maintain multiple incremental models in memory for making combined decisions. Compared with single models, ensemble methods are more popular since they can easily forget outdated data and adapt to the most recent data simply by removing outdated component classifiers and creating new ones. Moreover, they generally have stronger generalization ability. There are mainly two categories of such methods (Brzezinski and Stefanowski 2014a). The first category consists of block-based ensembles such as Learn++.NSE (Elwell and Polikar 2011), AUE1 (Brzeziński and Stefanowski 2011), AUE2 (Brzezinski and Stefanowski 2014b), etc., which are designed for situations where data arrives in batches or blocks. The second category includes online ensembles which process each instance once on its arrival without the requirement for storage and reprocessing. Representative methods are ASHTBag and ADWINBag (Bifet et al. 2009), LevBag (Bifet et al. 2010b), OAUE (Brzezinski and Stefanowski 2014a), DDD (Minku and Yao 2012), etc. Among the ensembles mentioned above, Hoeffding trees (Domingos and Hulten 2000) are popular base learners due to the fact that, given sufficient samples, they are guaranteed to asymptotically approach the trees generated by batch learners. However, with few samples in each concept of the high-dimensional data streams, Hoeffding tree, as an incremental decision tree induction algorithm, tend to overfit.

There is much less work on handling high dimensionality of data streams. Aggarwal and Yu (2008) developed an instance-centered subspace classification approach, which constructs discriminative subspaces specific to the locality of the particular test instance and makes decisions by combining the results of different subspace samples. Wang et al. (2014) proposed a framework of sparse online classification, which overcomes the scalability problem by introducing sparsity in learnt weights. McCallum et al. (1998) considered two different naive bayes models for document classification. These methods, however, are not designed to handle concept drifts.

In addition to the above methods, some improved random forests are reported to perform well on very high-dimensional data (Do et al. 2010; Ye et al. 2013). These

methods, however, cannot easily be modified to be online methods. There are also some works focusing on building the online version of random forests. Some of them, e.g., (Denil et al. 2013; Lakshminarayanan et al. 2014), are not designed to handle concept drifts. Some others are aware of concept drifts, but rely on certain prior knowledge, e.g., the range of each attribute (Abdulsalam et al. 2007; Saffari et al. 2009; Abdulsalam et al. 2011), which do not fit our online environment requirement that the information related to the data needs to be calculated or evaluated online rather than be known in advance. Moreover, whether these methods are robust to very high-dimensional data are also unknown.

In this paper, we simultaneously tackle concept drifting and high dimensionality in mining streaming data. First, in order to weaken the effect caused by the curse of dimensionality, we apply BPegasos (Wang et al. 2012), an online kernelized SVM method, as the component classifier of ensemble. Second, to better handle various types of concept drifts, a new specialized ensemble strategy is designed for BPegasos. BPegasos is selected owing to the following reasons:

1. BPegasos has good scalability. It works in  $O(Bd)$  space and its time cost for processing one instance is  $O(Bd)$ , where  $d$  is the number of attributes, and  $B$ , of which the value is independent of  $d$ , is the budget for controlling the number of support vectors. By introducing the predefined parameter  $B$ , BPegasos can control the resource consumption according to the actual requirement, which makes it memory-efficient.
2. BPegasos inherits the theoretical advantages of SVM so that it can generalize well on small data in high-dimensional space by capacity control through margin maximization (Abe 2005; Pappu and Pardalos 2014). The high dimensionality of the feature spaces induced by Mercer kernels is even considered as a bonus since it allows to build more powerful classifiers.

While single BPegasos possesses the above merits, it is limited in handling various types of concept drifts. Meanwhile, existing ensemble strategies have mainly been designed for the tree classifiers, which are very different from BPegasos in stability and adaptability. Hence, directly applying these strategies to BPegasos may produce unsatisfactory results. Therefore, we design a new ensemble method for BPegasos to compensate for the above constraints.

Our main contributions are threefold:

1. We formalize the problem of online classification on data streams embedded with both high dimensionality and concept drifts. To the best of our knowledge, this problem has not been addressed in the literature.
2. A novel solution for the above problem is developed. The solution uses BPegasos as the online component classifier, injects diversity among components by controlling their budget sizes, and modifies the ensemble structure by taking the data characteristics into account.
3. Extensive experiments are conducted and demonstrate that our solution is superior to the typical Hoeffding tree-based ensembles in addressing the proposed problem. The superiority lies not only in accuracy but also in the resource-efficiency in terms of time and space. When the competitor ensembles also use BPegasos as

**Table 1** Notation

Symbols	Description
$\mathbf{x}_t \in \mathbb{R}^d, y_t \in \mathbb{R}$	A sample and its label at time $t$
$c_t: \mathbb{R}^d \rightarrow \mathbb{R}$	The true concept at time $t$
$P_t$	A fixed distribution at time $t$

base learners, our solution still shows the best capability to deal with various types of drifts.

The organization of this paper is as follows. Section 2 presents the problem definition and related work. Section 3 introduces BPegasos. Our proposed ensemble strategy is presented in detail in Sect. 4. Experiments are conducted and the corresponding results are analyzed in Sect. 5. Finally, Sect. 6 draws conclusions and discusses future work.

## 2 Problem definition and related work

### 2.1 Problem definition

Here, we define the problem of classifying high-dimensional evolving data streams. For readability, Table 1 summarizes the main notation used in this paper. Assume that only one instance is available at one time and data arrives sequentially in the following way:  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t), \dots\}$ , where  $\mathbf{x}_t$  is drawn from distribution  $P_t$  and  $y_t = c_t(\mathbf{x}_t)$ . In our case, the dimensionality of variables may vary from a few dozen to many thousands. If the concept  $c_t$  keeps constant for the whole stream, the stream is said to be *stationary*; Otherwise, it contains concept drifts. Relying on how the time-adjacent concepts change, concept drifts are often categorized as sudden, gradual, and incremental drifts (Gama et al. 2014; Brzezinski and Stefanowski 2014a). Sudden drifts happen when the transition between concepts take place within a very limited amount of time. For gradual drifts, the transition may last longer and during this time, the probability that examples come from the first concept decreases with time, but the probability that examples from the second concept increases with time. Incremental drifting refers to the case where the concept changes slightly but continuously; the change is often unnoticeable until sufficient difference is accumulated. When the data flows in, up to time  $t$ , the available instance sequence is  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_t, y_t)\}$  and the corresponding concept sequence is  $\{c_1, c_2, \dots, c_t\}$ . Using the available instances, the task of the learner at time  $t$  is to obtain a good estimate  $\hat{c}_{t+1}$  of the true concept  $c_{t+1}$  with the *time constraint* that computing  $\hat{c}_{t+1}$  and using it for prediction must be completed before the true label  $y_{t+1}$  comes and the *space constraint* that the learner can only use limited memory resource. In the long run, the learner hopes to minimize the number of wrong predictions despite of the type of drifts.

**Table 2** A summary of online ensemble methods

Methods	①	②	③	④	⑤	⑥
OzaBag	Any OL	Resampling	—	M	×	×
DWM	Any OL	Periodically create members	ED	WM	×	✓
ASHTBag	Hoeffding tree	Resampling + control tree size	EWMA	WM	×	×
ADWINBag	Any OL	Resampling	ADWIN	M	✓	✓
LevBag	Any OL	Resampling using Poisson( $\lambda$ )	ADWIN	M	✓	✓
OAUE	Any OL	Periodically create members	MSE	WM	×	✓
DDD	Any ensemble	Control resampling weights	PA	SWM	✓	×

“OL” = “online learner”; “—” = “not required”; “ED” = “exponentially decreasing”; “EWMA” = “exponentially weighted moving average”; “MSE” = “mean square error”; “PA” = “prequential accuracy”; “M” represents such voting mechanism:  $\hat{y} \leftarrow \arg \max_{y \in Y} \sum_{i=1}^k f_{i,y}$ , where  $f_{i,y}$  is the decision of  $i$ -th base learner on the class  $y$  and  $f_{i,y} \in \{0, 1\}$  or  $f_{i,y} \in [0, 1]$ ; “WM” represents:  $\hat{y} \leftarrow \arg \max_{y \in Y} \sum_{i=1}^k a_i f_{i,y}$ , where  $a_i$  is the weight assigned to the  $i$ -th base learner; “SWM” represents that not all the components, but some are used in the “WM”

In the literature, there exist other definitions of concept drifts which are slightly different from ours. In Gama et al. (2014), Gama et al. define a concept as a joint probability distribution  $p(\mathbf{X}, y)$ , where  $\mathbf{X}$  is the observed variable set and  $y$  is the target variable. A drift happens between time  $t_0$  and  $t_1$  if  $\exists \mathbf{X} : p_{t_0}(\mathbf{X}, y) \neq p_{t_1}(\mathbf{X}, y)$ . In particular, they distinguish two different types of drifts: real concept drifts and virtual ones. The former refer to the changes of  $p(y|\mathbf{X})$  either with or without changes in  $p(\mathbf{X})$  and the latter happen if only  $p(\mathbf{X})$  changes. It can be seen that our definition corresponds to real concept drifts.

## 2.2 Related work

Online ensembles proposed in recent years are summarized in Table 2. They are described from the following aspects: ① component classifiers wrapped, ② diversity injection methods, ③ evaluation strategy for components, ④ aggregation strategy for prediction, ⑤ whether to use drift detection mechanism and ⑥ whether to modify the ensemble structure (i.e., add new components and discard the outdated ones).

*OzaBag* (Oza 2005) is an online version of bagging, which simulates offline bootstrap sampling by presenting each available instance  $k$  times to each component, where  $k \sim \text{Poisson}(1)$ . Such a practice is based on the observation that the probability that an instance is chosen for a replicate in bootstrap sampling tends to observe Poisson(1) distribution. Anytime decision is made either by a simple majority voting when each component outputs a unique class label, or by finding the maximum class support when each component outputs an estimate of class posterior probabilities. For the lack of structural modification, it is difficult for OzaBag to adapt to fast drifts.

*DWM* (Kolter and Maloof 2007) maintains a dynamic weighted pool of components. It adjusts the weights of components, creates new components and removes

poorly-performing ones after processing every  $p$  instances. Each component has an initial weight 1. Whenever a component makes a wrong prediction, its weight will be discounted by  $\beta \in [0, 1)$ . When a component's weight falls below a threshold  $\theta$ , it will be removed from the ensemble. Since each component's weight is exponentially decreasing (ED), DWM is susceptible to noise and may discard some useful components incorrectly.

*ASHTBag* (Bifet et al. 2009) is an ensemble tailored for Hoeffding trees, which adds diversity among components by bagging Hoeffding trees with different sizes. It uses exponentially weighted moving average (EWMA) to estimate the error of each tree and sets the weight of each tree inversely proportional to the square of its error. Like OzaBag, ASHTBag also suffers from fast drifts.

*ADWINBag* (Bifet et al. 2009) incorporates the ADWIN drift detector (Bifet and Gavalda 2007) into OzaBag for detecting drifts and evaluating performance for structural modification. *LevBag* (Bifet et al. 2010b) further enhances ADWINBag by increasing resampling weight using Poisson( $\lambda$ ) distribution. A typical value for  $\lambda$  is 6. However, this strategy makes LevBag vulnerable to noise, since the noisy instances are also learnt many times via resampling.

*OAUE* (Brzezinski and Stefanowski 2014a) simulates the block-based ensembles by creating new members and discarding obsolete ones periodically. Performance evaluation is done by incrementally calculating the mean squared error (MSE) of each component classifier and a random prediction classifier on the last  $w$  examples. Like block-based ensembles, OAUE may encounter the *stability-and-plasticity dilemma*. On one hand, larger blocks can produce stable classifiers that react slowly to concept drifts. On the other hand, smaller blocks are beneficial to react to drifts quickly but bring unstable performance.

*DDD* (Minku and Yao 2012) maintains four ensembles with different diversity levels. Each ensemble resamples using different Poisson ( $\lambda$ ) distributions. The idea is motivated by the analysis of the impact of diversity on online ensemble learning in the presence of concept drifts in Minku et al. (2010). DDD uses some drift detection technique for monitoring the state of the stream so as to select different ensembles for prediction at different periods. The performance of each ensemble is evaluated using the prequential accuracy (PA) from the last moment of detecting a drift to current moment. Different from the above-mentioned ensemble methods, DDD is an ensemble of ensembles, thus its time and space costs are larger.

No work has been reported on classifying high-dimensional data streams with concept drifting, to the best of our knowledge.

### 3 BPegasos

BPegasos (Wang et al. 2012) is an improvement over kernelized Pegasos (Shalev-Shwartz et al. 2011). In this section, we give a brief introduction to Pegasos and BPegasos both in the multi-class setting.

Given a data stream  $\{(\mathbf{x}_t, y_t), t = 1, 2, \dots\}$ , where  $\mathbf{x}_t \in \mathbb{R}^d$ ,  $y_t \in Y = \{1, 2, \dots, C\}$ , the multi-class decision function at time  $t$  is

$$f_t(\mathbf{x}) = \arg \max_{i \in Y} \mathbf{w}_t^{(i)\top} \mathbf{x}, \quad (1)$$

where  $\mathbf{w}_t^{(i)}$  is the  $i$ -th class-specific weight vector at time  $t$ .

Pegasos uses *stochastic gradient descent* (SGD) to solve the following primal formulation of the SVM problem

$$\min_{\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \dots, \mathbf{w}^{(C)}} \frac{\lambda}{2} \sum_{i \in Y} \|\mathbf{w}^{(i)}\|^2 + \frac{1}{T} \sum_{t=1}^T \max \left\{ 0, 1 + \mathbf{w}^{(r_t)\top} \mathbf{x}_t - \mathbf{w}^{(y_t)\top} \mathbf{x}_t \right\},$$

where  $r_t = \arg \max_{i \in Y, i \neq y_t} \mathbf{w}^{(i)\top} \mathbf{x}_t$  and  $\lambda$  is the parameter of the regularization term controlling the complexity of the model. At  $t$ -th round, when  $\mathbf{x}_t$  is available, Pegasos uses  $\mathbf{w}_t^{(i)}$  learned before to predict its label by Eq. (1). After receiving the true label  $y_t$  of  $\mathbf{x}_t$ ,  $\mathbf{w}_t^{(i)}$  is updated as

$$\mathbf{w}_{t+1}^{(i)} = \mathbf{w}_t^{(i)} - \eta_t \nabla_t^{(i)}, \quad i = 1, 2, \dots, C, \quad (2)$$

where  $\eta_t = 1/(\lambda t)$  is the learning step, and  $\nabla_t^{(i)}$  belongs to the sub-gradient of the instantaneous objective function  $P_t$  with respect to  $\mathbf{w}_t^{(i)}$ .  $P_t$  is defined as

$$\frac{\lambda}{2} \sum_{i \in Y} \|\mathbf{w}_t^{(i)}\|^2 + \max \left\{ 0, 1 + \mathbf{w}_t^{(r_t)\top} \mathbf{x}_t - \mathbf{w}_t^{(y_t)\top} \mathbf{x}_t \right\},$$

where  $r_t = \arg \max_{i \in Y, i \neq y_t} \mathbf{w}_t^{(i)\top} \mathbf{x}_t$ . Plugging into Eq. (2) the expression of  $\nabla_t^{(i)}$ , the update rule becomes

$$\mathbf{w}_{t+1}^{(i)} = (1 - \lambda \eta_t) \mathbf{w}_t^{(i)} + \beta_t^{(i)} \mathbf{x}_t, \quad i = 1, 2, \dots, C,$$

where

$$\beta_t^{(i)} = \begin{cases} \eta_t & \text{if } i = y_t, \\ -\eta_t & \text{if } i = r_t, \\ 0 & \text{otherwise.} \end{cases}$$

Through derivation,  $\mathbf{w}_t^{(i)}$  can eventually be represented as

$$\mathbf{w}_t^{(i)} = \sum_{j=1}^{t-1} \alpha_j^{(i)} \mathbf{x}_j,$$

where

$$\alpha_j^{(i)} = \beta_j^{(i)} \prod_{k=j+1}^{t-1} (1 - \lambda \eta_k).$$



$\alpha_j^{(i)}$  is the contribution factor of the  $j$ -th instance to the  $i$ -th class. If  $\alpha_j^{(i)}$  is non-zero,  $\mathbf{x}_j$  is called a support vector (SV).

When used for non-linear classification, a non-linear mapping  $\phi$  can be introduced to map the original examples to a new higher dimensional feature space.  $\mathbf{w}_t^{(i)}$  can be rewritten as

$$\mathbf{w}_t^{(i)} = \sum_{j \in I_t^{(i)}} \alpha_j^{(i)} \phi(\mathbf{x}_j), \quad (3)$$

where  $I_t^{(i)}$  is the set of indices of all SVs in  $\mathbf{w}_t^{(i)}$ . Since  $\phi$  may be infinite dimensional,  $\mathbf{w}_t^{(i)}$  cannot be represented explicitly by a vector just like in linear situations. Instead, an SV set  $S_t = \{(\alpha_j^{(i)}, \mathbf{x}_j), j \in I_t^{(i)}\}$  needs to be maintained. By inserting Eq. (3) into Eq. (1), the decision function can be rewritten as

$$f_t(\mathbf{x}) = \arg \max_{i \in Y} \sum_{j \in I_t^{(i)}} \alpha_j^{(i)} \phi(\mathbf{x}_j)^T \phi(\mathbf{x}). \quad (4)$$

In Eq. (4), the inner product can be efficiently computed as  $\phi(\mathbf{x}_j)^T \phi(\mathbf{x}) = \mathcal{K}(\mathbf{x}_j, \mathbf{x})$ , where  $\mathcal{K}(\cdot)$  is a Mercer kernel derived from  $\phi$ . As shown in Eq. (4), the time cost for prediction and the space cost for saving the model are both proportional to the number of SVs. The main problem of kernelized Pegasos is that the number of SVs may grows unboundedly with the number of data processed, which restricts its usage in resource-limited streaming environments. BPegasos is proposed to solve the problem.

BPegasos keeps  $\mathbf{w}_t^{(i)}$  spanned by at most  $B$  SVs, where  $B$  is a user-defined parameter. Denote  $I_t = \bigcup_{i=1}^C I_t^{(i)}$ ; when  $|I_t| > B$ , a budget maintenance step is triggered and achieved as

$$\mathbf{w}_t^{(i)} \leftarrow \mathbf{w}_t^{(i)} - \Delta_t^{(i)}, i = 1, 2, \dots, C. \quad (5)$$

where  $\Delta_t^{(i)}$  is the weight degradation of  $\mathbf{w}_t^{(i)}$  before and after the budget maintenance.

Wang et al. (2012) proved that a good budget maintenance strategy should minimize the total degradation  $\sum_{i=1}^C \|\Delta_t^{(i)}\|^2$ . Specifically, they proposed three strategies: (i) deleting one SV, (ii) projecting one SV to the remaining ones, and (iii) merging two SVs into a newly created one. Among them, the merging strategy is the most appealing, since it has time and space complexity of  $O(Bd)$  per iteration and can produce smaller total weight degradation. The basic idea of merging is to choose two SVs and then merge them into a new one. The details are in Wang et al. (2012).

Algorithm 1 presents the pseudocode of BPegasos. In the remaining sections, when it comes to BPegasos, we refer to the BPegasos with the merging strategy. By default, the popular RBF kernel with width  $\delta$  is used (Hsu et al. 2003):

$$\mathcal{K}(\mathbf{x}_j, \mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}_j - \mathbf{x}\|^2}{2\delta^2}\right).$$

**Algorithm 1:** BPegasos

**Input:** Data stream  $\mathbf{D} = \{(\mathbf{x}_t, y_t), t = 1, 2, \dots\}$ ; budget  $B$ ; regularization parameter  $\lambda$ ; kernel function  $\mathcal{K}(\cdot)$

**Output:**  $f_t(\mathbf{x})$

```

1 for  $t = 1, 2, \dots$  do
2   Receive  $\mathbf{x}_t$ ;
3   Predict its label using Eq. (4);
4   Receive the true label  $y_t$ ;
5   Update  $\mathbf{w}_t^{(i)}$  using SGD;
6   if  $|I_t| > B$  then
7     Do the budget maintenance using Eq. (5);

```

**4 Ensemble BPegasos (EBPegasos)**

In order to improve the adaptability of BPegasos to various types of drifts, a novel online ensemble method, called EBPegasos, is proposed. In the following, three critical parts in EBPegasos are presented respectively:

1. *Diversity injection*, which produces a component set as accurately and diversely as possible;
2. *Component management*, which monitors the performance status of components, and makes accurate predictions by aggregating the outputs of the components.
3. *Structural modification*, discards the components at the right time, with the objective of striking a good balance between exploiting and forgetting old knowledge.

**4.1 Diversity injection**

It is known that *diversity* plays an important role in learning data streams with concept drifts (Minku et al. 2010). For unstable learners such as decision trees, small changes in data can bring about large variations in the classification. Accordingly, *resampling* is effective for increasing diversity among unstable learners. Unfortunately, as an SVM-based method, BPegasos is relatively stable and thus resampling cannot help much.

Instead of using resampling, EBPegasos creates diversity by allowing different budget sizes of BPegasos components. This is due to the fact that varying the budget size greatly affects the performance of BPegasos (Wang et al. 2012). In general, a larger budget means more stable performance since the margin information is better kept by more SVs. This is preferable when the data stream is stable, as BPegasos may thus reduce the number of false alarms of concept drifting. However, the stability becomes an impediment to BPegasos when the underlying concept in the data stream is actually drifting, as it usually takes more time for BPegasos to forget that many obsolete SVs. In contrast, a smaller budget means higher instability of performance but more sensitivity to changes, which thus may cause more false alarms when the data stream is stable but contribute to the early detection of concept drifts when they indeed happen. Considering this, we design EBPegasos to include multiple BPegasos

classifiers with different budget sizes, so that it can achieve good performance in both stable and unstable periods of the data stream.

## 4.2 Component management

EBPegasos predicts by weighted majority voting. In particular, when making a prediction  $\hat{y}_t$  for the data sample  $\mathbf{x}_t$ , EBPegasos calculates a weighted sum of the predictions of the components,

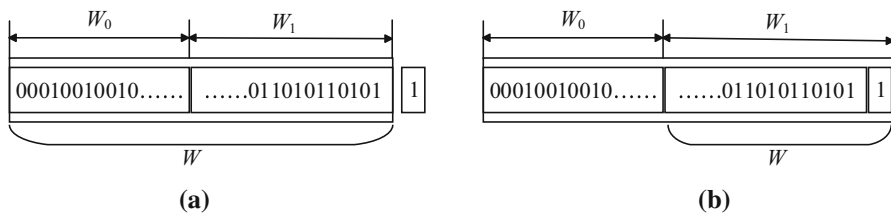
$$\hat{y}_t \leftarrow \arg \max_{y \in Y} \sum_{i=1}^k a_t^i \cdot \mathbb{I}(f_t^i(\mathbf{x}_t) = y), \quad (6)$$

where  $a_t^i$  is the weight assigned to the  $i$ -th component, and  $\mathbb{I}(\cdot)$  is an indicator function which outputs 1 if the Boolean expression in the brackets is true, or 0 otherwise.

To obtain  $a_t^i$ , EBPegasos equips each of its components with an ADWIN (Bifet and Gavalda 2007) to monitor and evaluate its performance. ADWIN is originally designed as a *drift detector*; Fig. 1 presents its working process. ADWIN is originally designed as a *drift detector*; Fig. 1 presents its working process. Every time a component makes a prediction  $\hat{y}_t$ , a single bit  $b$  is generated (via comparing  $\hat{y}_t$  to  $y_t$ , the true class label of  $\mathbf{x}_t$ ) and fed into ADWIN, indicating whether the prediction  $\hat{y}_t$  is correct ( $b = 0$ ) or wrong ( $b = 1$ ). As shown in Fig. 1a, ADWIN tracks a window  $W$  of such bits, which consists of two large enough sub-windows  $W_0$  and  $W_1$ . The basic idea of ADWIN is that, when there is no concept drift in the data stream, the average of the elements (i.e., the error rate) in  $W_0$  should be statistically consistent with the average in  $W_1$ . Put in other words, whenever the difference between the error rates in  $W_0$  and  $W_1$  is statistically significant, ADWIN alarms a concept drift, in which case ADWIN drops the sub-window  $W_0$  and makes  $W_1$  a new  $W$  (Fig. 1b). Note that the overall error rate in  $W$  (i.e., the number of 1's in  $W$  divided by the length of  $W$ ),  $\text{Err}_t^i$ , can be used as a performance measure of the corresponding component. Hence, the weight  $a_t^i$  can be computed as

$$a_t^i = \frac{1 - \text{Err}_t^i}{\sum_{i=1}^k (1 - \text{Err}_t^i)}, \quad i = 1, 2, \dots, k,$$

where  $k$  is the number of ensemble members.



**Fig. 1** The working process of ADWIN. **a**  $W$  before a drift, **b**  $W$  after a drift

As mentioned in Sect. 4.1, there are component BPegasos classifiers with various budgets in EBPegasos. Hence, another benefit of such a practice is that the ADWIN detectors associated with different budgeted classifiers can issue alarms at different times, which helps to reduce the false negative (missing alarms) rate of the drift detection.

### 4.3 Conditional structural modification

Structural modification involves when to create new components to substitute the outdated ones. To handle various types of drifts well, especially for fast incremental drifts, structural modification is essential in ensemble methods.

Existing ensemble methods such as ADWINBag and LevBag modify the structure once a drift is detected. However, since these methods are designed for component classifiers without forgetting mechanisms, their timing of making the structural modification is not suitable for BPegasos. BPegasos is very different from other component classifiers used in other ensemble solutions in that it naturally has slow forgetting capability due to its budget maintenance. Specifically, when drifts occur, BPegasos has two ways to establish the new decision margin: (i) discarding the previous model learnt and retraining a new one on the most recent data; or (ii) keeping updating the previous model incrementally. Which way is better depends on the intrinsic characteristics of data. Intuitively, the first way is better for severe drifts that bring large performance degradation of classifiers; while for slight drifts that bring small degradation and that maybe are caused by noise, the second way is better.

In order to identify the severity of drifting, we define  $r$ , the *model degradation rate* (MDR), as follows:

$$r = \frac{f_1 - f_0}{|W_1|}, \quad (7)$$

where  $f_0$  and  $f_1$  are the errors in the sub-windows  $W_0$  and  $W_1$  in Fig. 1b respectively.  $f_1 - f_0$  reflects how much a model degrades when the concept drifts, and  $|W_1|$  indicates how long it takes for the accumulated degradation to be detected.

For severe drifts, one may expect a large  $r$  since performance degrades greatly in relatively short time. In contrast, for slight drifts,  $r$  is small since getting a significant degradation  $f_1 - f_0$  needs a larger window  $W_1$ . Intuitively, with a large  $r$ , it is better to forget old knowledge rapidly by retraining a new classifier from scratch; and, on the contrary, with a small  $r$ , exploiting fully the old knowledge by incrementally updating the old classifiers is often preferable. Therefore, we define a threshold  $\theta$  in EBPegasos for determining when to use which way. Specifically, when a drift alarm is issued, if  $r$  is larger than  $\theta$ , a new classifier is retrained; otherwise, the old classifier is retained and keeps updating itself. In this way, EBPeegasos strikes a good balance between reusing and fast forgetting previously learned knowledge.

#### 4.4 Algorithm pseudocode

Algorithm 2 presents the pseudocode of EBPegasos.

---

##### Algorithm 2: EBPegasos

---

**Input:** Data stream  $\mathbf{D} = \{(\mathbf{x}_i, y_i), i = 1, 2, \dots\}$ , budget parameter  $B$ , the number of component classifiers  $k$ , regularization parameter  $\lambda$ , RBF kernel parameter  $\delta^2$ , and severity threshold  $\theta$

**Output:**  $\hat{y}_t$

```

1 Initialize  $k$  components and their drift detectors  $DT_i, i = 1, 2, \dots, k$ ; The  $i$ -th component has a
  budget of  $i \cdot B$ ;
2 newStart  $\leftarrow -1$ ;
3 for  $t = 1, 2, \dots$  do
4   Receive  $\mathbf{x}_t$ ;
5   Predict  $\hat{y}_t$  using Eq. (6);
6   Receive  $y_t$ ;
7   Update  $f_t^i$  using  $(\mathbf{x}_t, y_t), i = 1, 2, \dots, k$ ;
8   Modify  $\leftarrow false$ ;
9   for  $i = 1, 2, \dots, k$  do
10    Input  $f_t^i(\mathbf{x}_t)$  into  $DT_i$ ;
11    if  $DT_i$  issues a drift alarm then
12      Compute  $r$  using Eq. (7);
13      if  $r > \theta$  then
14        Modify  $\leftarrow true$ ;
15  if Modify is true then
16     $j \leftarrow \max_{i=1 \dots k, i \neq \text{newStart}} \text{Err}_t^i$ ;
17    Reset the  $j$ -th member and its detector  $DT_j$ ;
18    newStart  $\leftarrow j$ ;
```

---

During initialization,  $k$  components are created, each is equipped with a drift detector for monitoring its performance, and the  $i$ -th component is allocated a budget of  $i \cdot B$  (Line 1). Then, EBPegasos strictly follows the procedure of online learning: Whenever a new example  $\mathbf{x}_t$  arrives, EBPegasos uses each component's current decision function  $f_t^i(\cdot)$  to predict its label and calls a weighted majority voting (Lines 4, 5). After receiving the true label  $y_t$ , each component will be incrementally updated using  $(\mathbf{x}_t, y_t)$  (Lines 6, 7). When the performance of some component degrades severely, it triggers a drift alarm (Lines 9–14), and EBPegasos thus resets the worst-performing component as well as its detector (Lines 15–18). Note that a newly-created classifier typically requires certain amount of training data to grow stable. Hence, to prevent a new classifier from being dropped due to some performance perturbation in its very early stage, it is offered an exemption such that it may survive the first structural modification after its creation.

#### 4.5 Complexity analysis

For the  $i$ -th component, the main time cost is in prediction and budget maintenance, thus the time complexity of the  $i$ -th component is  $O(i \cdot Bd)$ . In addition, the  $i$ -th

component stores  $i \cdot B$  SVs, and thus requires  $O(i \cdot Bd)$  space. Each ADWIN drift detector needs  $O(\log |W|)$  time and  $O(\log |W|)$  memory for processing an input bit, where  $W$  is the maximum stable window (Bifet and Gavalda 2007). Hence, checking whether it is necessary to do the structural modification takes  $O(k \log |W|)$  time, dominating the time cost,  $O(k)$ , for actually performing a structural modification. Therefore, to sum up, EBPegasos works in  $O(k^2 Bd + k \log |W|)$  space and costs  $O(k^2 Bd + k \log |W|)$  time per iteration.

It is easy to see that (i) the space cost of EBPegasos remains constant after its initial configuration, and (ii) the time cost for processing one data instance is irrelevant to the length of the data stream. Hence, EBPegasos is very suitable for resource-limited situations.

## 5 Experiments

In this section, four sets of experiments are presented.

1. *Accuracy and efficiency analysis on high-dimensional data.* This set of experiments demonstrates the advantages of EBPegasos over the state-of-the-art tree ensembles on high-dimensional data streams.
2. *Adaptivity analysis.* This set of experiments compares the ability of EBPegasos to cope with various types of concept drifts with that of the state-of-the-art ensemble frameworks, showing that the superiority of EBPegasos comes from not only the choice of the components but also its novel component ensemble strategy.
3. *Comparative analysis on low-dimensional data.* This set of experiments observes the performance of EBPegasos compared to the tree ensembles on low-dimensional datasets.
4. *Sensitivity analysis.* This set of experiments analyzes the parameter sensitivity of EBPegasos.

All the experiments were conducted in the Massive Online Analysis (MOA) framework (Bifet et al. 2010a), which is a dedicated software platform for running experiments of evolving data stream mining, containing most of the competitor algorithms considered in our experiments. All experiments were performed on a machine with 2.60GHz Intel(R) Xeon(R) CPU E5-2650 v2 and 64GB 1866MHz DDR3 memory.

*Prequential accuracy* (Gama et al. 2014, 2013) is used to evaluate the method, which is computed at every moment  $t$  as soon as the prediction  $\hat{y}_t$  is made

$$Acc(t) = \frac{1}{t} \sum_{i=1}^t L(y_i, \hat{y}_i),$$

where  $L(y_i, \hat{y}_i) = 1$  if  $y_i \neq \hat{y}_i$  and 0 otherwise. For comparing two methods, *average prequential accuracy* over all the moments is reported. Since we handle high-dimensional data in resource-limited scenarios, space and time costs are also of essential importance in evaluating an algorithm. To measure them, we use the tools

**Table 3** The characteristics of the datasets and the parameter settings of EBPegasos

Name	Datasets			EBPegasos				
	#inst	#Attr	#Classes	$k$	$\lambda$	$\delta^2$	$B$	$\theta$
Spam_data	9324	500	2	5	1e-4	10	4	1e-5
Gisette	6000	5000	2	5	1e-4	5	80	1e-5
NewsGroup4	12,000	26,214	2	5	1e-6	60	4	1e-5
Spam_corpus	9324	39,916	2	5	1e-4	70	4	1e-5

provided in MOA. In addition, to verify whether the performance difference is statistically significant, Wilcoxon signed-rank test (Demšar 2006) is used, which is regarded as the best statistical measure for evaluating two classifiers on multiple datasets.

### 5.1 Accuracy and efficiency analysis on high-dimensional data

This section aims to demonstrate the accuracy and efficiency of EBPegasos in face of high-dimensional data, compared with the state-of-the-art tree ensembles. Four real-world very high-dimensional datasets are used and their characteristics are shown in Table 3. Spam\_corpus is from SpamAssasin data collection (Katakis et al. 2009), containing 9,324 emails, in which around 20% are spams. Each email is represented as a 39,916-dimensional vector, where each binary component indicates whether or not a corresponding term appears in this email. Spam\_data is the dataset after applying feature selection with the  $\chi^2$  measure on Spam\_corpus (Katakis et al. 2010). The task of Gisette<sup>1</sup> data is to distinguish the highly confusable handwritten “4” and “9”. Each record in the dataset describes such a digit. NewsGroup4, created from 20newsGroup,<sup>2</sup> simulates the evolution of a particular user’s interest in four news groups over time. The parameter settings of EBPegasos on the above datasets are chosen by using the first 20% of data and are displayed in Table 3. The chosen settings are then used on the whole stream.

The compared methods include OzaBag (Oza) (Oza 2005), ASHTBag (ASHT) (Bifet et al. 2009), ADWINBag (ADWIN) (Bifet et al. 2009), LevBag (Lev) (Bifet et al. 2010b), DWM (Kolter and Maloof 2007), OAUE (Brzezinski and Stefanowski 2014a) and multinomial naive bayes (MNB) (McCallum et al. 1998). The tree ensemble methods use 10 component classifiers, each of which is the Hoeffding Naive Bayes tree (HNBT) (Holmes et al. 2005) that adds Naive Bayes prediction at each leaf so as to adaptively choose at each leaf between majority class prediction and Naive Bayes prediction. In addition, we also consider one possible random forests (RF) implementation, which uses as the component classifier random HNBT that selects  $\sqrt{d}$  attributes randomly ( $d$  is the number of attributes) and only uses these attributes when building a tree and making prediction, as Bifet et al. did in their experiments (Bifet

<sup>1</sup> <https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#gisette>.

<sup>2</sup> <http://qwone.com/~jason/20Newsgroups/>.

et al. 2010b). RF combines 25 such trees using LevBag ensemble framework. Given that using of Hoeffding bound hinders trees from growing which may degrade their performance, we consider two groups of parameter settings of trees. In the first setting, the split confidence is 0.01 and the tie threshold is 0.05. Such setting does not allow attributes to split until high quasi-certainty about which attribute test should be used is obtained. The second setting using the tie threshold as 1.0 turns a tree into a regular tree that introduces an attribute splitting after seeing a relatively small and fixed number of instances. The grace period parameter in both settings is chosen from the interval  $[20, 30, \dots, 200]$ . For DWM, the factor for discounting weights and the threshold for deleting component classifiers are 0.5 and 0.01 respectively, as suggested in Kolter and Maloof (2007). But the period parameter in DWM for controlling component removal, creation and weight update and the window size in OAUE for structural modification and component evaluation are both data-specific and are chosen from the interval  $[100, 200, \dots, 1000]$  so that they achieves the best performance. The results of the tree ensembles for the two different parameter settings, including the average prequential accuracy, total processing time consisting of training and testing time, and maximum space cost have been reported respectively in Table 4, in which, “-” represents that the results cannot be obtained since the corresponding algorithm runs out of the available memory and interrupts its performing. As a specialized document classifier, MNB is not suitable for the Gisette data; thus its result on the dataset is shown as “-” in the Table 4. The results of EBPegasos are presented in the last column of Table 4.

As we can see, for the tree ensembles, the second parameter setting generally leads to better average prequential accuracy than the first one does, but costs more time and space resources. Among the tree ensembles, LevBag tends to use much greater time and space resources, which limits its performance on very high dimensional datasets. RF is not always more resource-efficient than the other tree ensembles, but achieves significantly better accuracy in most cases. As a simple document classifier and also a single model, MNB is very resource-efficient. However, for lacking of the mechanism to handle concept drifts, its performance suffers greatly on NewsGroup4. Compared with all the methods on the four datasets, EBPegasos always achieves the best average prequential accuracy, meanwhile its memory and time cost, although larger than MNB, is smaller than the tree ensembles in most cases.

## 5.2 Adaptability analysis

In this section, we want to compare the adaptability of EBPegasos to that of state-of-the-art ensemble methods mentioned above on datasets with various types of concept drifts. All the ensemble algorithms use BPEGasos as the component learners, thus the base learners in the tree-based ensembles are substituted with BPEGasos.<sup>3</sup>

<sup>3</sup> ASHTBag is excluded here since its ensemble strategy is dedicated to Hoeffding trees.



**Table 4** The average prequential accuracy (%), processing time (s) and space cost (MB) on the very high-dimensional datasets

Datasets		Oza	ASHT	ADWIN	Lev	DWM	OAUE	RF	MNB	EBP
Spam_data	(%)	92.04	92.11	92.62	96.58	93.09	90.47	95.44	95.77	<b>96.90</b>
	(s)	95.01	94.69	95.29	96.44	93.35	90.89	95.65		
	(MB)	16.30	17.49	19.16	52.55	19.02	13.37	16.72	<b>1.65</b>	11.52
Gisette	(%)	30.09	27.72	29.82	78.90	26.65	19.86	20.01		
	(s)	28.58	17.46	17.61	113.85	2.24	16.21	35.17	<b>0.22</b>	0.60
	(MB)	76.69	36.54	52.76	202.40	3.29	42.23	46.80		
NewsGroup4	(%)	61.23	59.24	61.23	83.43	62.39	65.35	85.41	–	<b>87.53</b>
	(s)	77.46	64.42	77.46	76.29	83.29	75.73	83.71		
	(MB)	286.33	353.42	342.71	708.64	88.38	132.68	141.04	–	2829.79
Spam_corpus	(%)	489.77	548.52	544.21	1671.01	<b>88.05</b>	418.90	311.21		
	(s)	230.32	158.12	230.34	1159.15	48.17	84.34	102.76	–	<b>46.52</b>
	(MB)	988.31	461.64	988.32	1524.49	129.29	577.52	290.80		
Spam_corpus	(%)	52.70	52.03	53.82	62.95	57.17	51.48	76.60	75.95	<b>90.49</b>
	(s)	63.56	60.04	62.78	–	67.18	57.65	79.94		
	(MB)	2098.85	2505.74	3082.90	6285.23	859.53	1335.98	26,422.80	<b>2.68</b>	661.61
Spam_corpus	(%)	4088.70	4260.18	4577.57	–	3301.34	4120.99	92,935.80		
	(s)	1769.04	1875.44	1209.25	2031.06	521.20	1388.44	1919.89	<b>2.97</b>	15.59
	(MB)	1992.99	1916.95	1759.07	–	1908.73	1991.94	1460.86		
Spam_corpus	(%)	87.52	87.34	89.69	94.41	90.32	87.05	92.84	94.65	<b>96.85</b>
	(s)	94.22	92.44	94.16	–	91.51	90.02	95.39		
	(MB)	1395.10	1570.95	1467.71	4180.36	2085.22	1159.50	22,143.60	<b>185.41</b>	902.10
Spam_corpus	(%)	5779.12	5613.11	5706.43	–	4573.86	3493.74	64,750.30		
	(s)	1848.25	1229.42	1153.07	1483.98	131.88	1249.95	1241.78	<b>26.15</b>	45.36
	(MB)	1841.36	1916.07	958.46	–	278.48	1658.07	1919.35		

The best result on each dataset is given in bold

**Table 5** The characteristics of the datasets

Datasets	#inst	#Attr	#Classes	Noise	#Drift	Drift type
Hy_f	1 M	50	2	10%	–	Incremental
Hy_s	1 M	50	2	10%	–	Incremental
RBF_f	100 k	100	10	–	–	Incremental
RBF_s	100 k	100	10	–	–	Incremental
LED_s	400 k	24	10	10–20–15–10%	3	Sudden
LED_g	400 k	24	10	10–20–15–10%	3	Gradual
SEA_s	400 k	3	2	10–20–15–10%	3	Sudden
SEA_g	400 k	3	2	10–20–15–10%	3	Gradual
Tree_s	400 k	30	10	–	3	Sudden
Tree_g	400 k	30	10	–	3	Gradual
Waveform_s	400 k	40	3	–	3	Sudden
Waveform_g	400 k	40	3	–	3	Gradual
Coverttype	581 k	54	7	–	–	Unknown
Kddcup99	494 k	121	23	–	–	Unknown

Noise is the label noise in each concept

We first use synthetic datasets to analyze the performance of the algorithms. For constructing datasets with sudden, gradual and incremental drifts,<sup>4</sup> we use the data stream generators available in MOA, including the *Rotating hyperplane*, *Random RBF*, *LED*, *SEA*, *Random Tree* and *Waveform* generators, which have been used frequently in the context of concept drifts (Bifet et al. 2010b; Brzeziński and Stefanowski 2011; Brzezinski and Stefanowski 2014a, b). Table 5 presents the characteristics of these datasets.

Besides the synthetic datasets, two real datasets are also considered. The first one is the Coverttype<sup>5</sup> which contains the forest cover type obtained from US Forest Service Region 2 Resource Information System data. The second one is the kddcup99,<sup>6</sup> which includes a wide range of intrusions simulated in a military network and whose task is to distinguish between bad connections and normal connections. We have transformed the original nominal attributes in Kddcup99 to binary attributes that are then treated as numeric.

To perform fair comparisons, all methods use 5 BPegasos components and take the same total budget space  $\mathcal{B}$  (in terms of the number of SVs) on each dataset. With such a resource limit, EBPegasos combines BPegasos components with budgets  $B$ ,  $2B$ ,  $3B$ ,  $4B$ ,  $5B$  to create diversity among components (Sect. 4.1), and guarantees  $15B = \mathcal{B}$ . The other methods use BPegasos components with the same budget  $B'$ , and ensures  $5B' = \mathcal{B}$ . To observe how much improvement EBPegasos can make

<sup>4</sup> The scripts for generating these datasets are available at <http://cs.nju.edu.cn/rl/people/zhaait/datasets/GenerateScript.txt>.

<sup>5</sup> It can be downloaded from <http://moa.cms.waikato.ac.nz/datasets/>.

<sup>6</sup> It can be downloaded from <http://www.cse.fau.edu/~xqzhu/stream.html>.

**Table 6** The average prequential accuracy using BPegasos as the base learner (%)

Datasets	Oza	ADWIN	Lev	DWM	OAUE	BP	EBP
Hy_f	84.66	84.66	84.26	76.66	84.87	84.71	<b>85.13</b>
Hy_s	85.06	85.06	84.80	76.90	85.15	85.04	<b>85.43</b>
RBF_f	21.97	51.52	<b>57.33</b>	52.61	46.49	35.15	55.27
RBF_s	48.01	<b>82.51</b>	75.32	75.11	52.38	57.18	79.86
LED_s	65.18	64.85	62.55	53.94	60.66	64.81	<b>65.66</b>
LED_g	63.58	62.82	58.88	50.70	60.46	63.44	<b>63.91</b>
SEA_s	85.03	84.65	83.41	81.72	84.52	84.78	<b>85.10</b>
SEA_g	84.38	84.04	82.57	81.29	83.90	84.19	<b>84.53</b>
Tree_s	67.24	70.02	70.69	52.48	65.39	67.83	<b>72.73</b>
Tree_g	62.96	62.97	60.74	47.20	59.34	64.11	<b>65.28</b>
Waveform_s	84.89	84.83	83.56	80.26	84.42	83.65	<b>85.54</b>
Waveform_g	84.16	83.81	82.44	79.75	83.99	83.16	<b>84.22</b>
Coverttype	89.91	90.43	91.90	92.32	91.30	91.41	<b>92.32</b>
Kddcup99	23.58	98.38	98.35	96.30	80.45	26.87	<b>98.38</b>
Wilcoxon $R^+$	105	91.5	95	105	105	105	
$R^-$	0	13.5	10	0	0	0	

The best result on each dataset is given in bold

over its base learners, we compare EBPEGASOS with its largest component: “BP” with the budget  $5B$ . We use the first 20% of each dataset to seek for each algorithm the best parameter setting and then apply that setting to the whole stream. The average prequential accuracy and processing time of these methods are reported in Tables 6 and 7 respectively. The last row of both tables also presents the results of Wilcoxon signed-rank test between EBPEGASOS and each of the other algorithms. In order to demonstrate the importance of the conditional structural modification in EBPEGASOS (Sect. 4.3), we also present the results of varying the threshold  $\theta$  from 0 to  $1e-1$  but keeping the other parameters’ values constant in Table 8.

As we can see from Table 6, OzaBag and BP perform poorly on RBF\_f, RBF\_s and Kddcup99, which may be due to the fact that concepts in these datasets change faster. Actually, this is expected according to Table 8, as preventing the structural modification (by setting a larger  $\theta$ ) in EBPEGASOS degrades the performance greatly. On the same datasets, DWM performs better than OAUE, since the component evaluation strategy of DWM encourages removing and creating components more frequently, while the periodical component evaluation and substitution of OAUE provides too slow reaction to fast changes. Comparing ADWINBag and LevBag on the noisy datasets, i.e., Hy\_f, Hy\_s, LED\_s, LED\_g, SEA\_s, and SEA\_g, we find that LevBag is always worse than ADWINBag, which results from the fact that increasing resampling weight makes LevBag more susceptible to noise.

On the datasets except RBF\_f, RBF\_s and Kddcup99, the base learner BP can achieve acceptable performance whenever drifts happen suddenly or gradually and no matter how many drifts occur. This is owing to its slow forgetting mechanism. Ignoring of the characteristic of BPegasos, ADWINBag, LevBag, DWM and OAUE

**Table 7** The total processing time in second on each dataset, including both the training and the testing time when using BPegasos as the base learner

Datasets	Oza	ADWIN	Lev	DWM	OAUE	BP	EBP
Hy_f	1174.78	1327.49	1903.08	1739.7	2067.46	<b>632.763</b>	1976.16
Hy_s	1199.27	1307.59	1930.51	1685.65	2091.53	<b>623.617</b>	1958.15
RBF_f	2965.60	1537.0	1874.31	2127.24	2932.46	<b>1239.37</b>	2724.67
RBF_s	2832.94	1388.12	1093.24	1702.79	4169.06	<b>1032.45</b>	2361.08
LED_s	1336.73	1223.16	1657.17	2045.43	2384.13	<b>844.78</b>	2255.99
LED_g	1404.38	1290.68	1790.18	2140.01	2529.25	<b>889.15</b>	2580.63
SEA_s	353.71	353.00	659.03	1669.82	616.28	<b>174.21</b>	575.65
SEA_g	359.10	359.16	650.92	1666.93	633.83	<b>178.58</b>	593.64
Tree_s	1660.54	1425.31	1699.37	2206.34	2798.93	<b>671.76</b>	1740.63
Tree_g	1794.4	1609.78	2172.84	2355.32	2917.6	<b>980.18</b>	2324.46
Waveform_s	440.74	390.36	496.32	403.96	916.92	<b>193.68</b>	1083.61
Waveform_g	394.84	364.31	497.76	976.05	705.42	<b>208.12</b>	1175.66
Coverttype	443.42	511.42	607.57	539.27	545.22	<b>190.41</b>	565.292
Kddcup99	307.21	1693.14	2465.44	1283.15	1240.11	<b>155.66</b>	2502.26
Wilcoxon $R^+$	11	0	16	35	69	0	
$R^-$	94	105	89	70	36	105	

The best result on each dataset is given in bold

**Table 8** The average prequential accuracy of EBPegasos when the threshold  $\theta$  varies from 0 to  $1e-1$ 

Datasets	0	$1e-6$	$1e-5$	$1e-4$	$1e-3$	$1e-2$	$1e-1$
Hy_f	76.14	76.14	76.67	<b>85.13</b>	85.13	85.13	85.13
Hy_s	76.29	76.29	81.38	<b>85.43</b>	85.43	85.43	85.43
RBF_f	55.27	<b>55.27</b>	55.24	54.92	52.31	17.14	17.14
RBF_s	79.22	79.23	<b>79.86</b>	52.38	52.38	52.38	52.38
LED_s	61.35	61.35	64.55	65.29	<b>65.66</b>	65.27	65.27
LED_g	59.18	59.18	63.49	<b>63.91</b>	63.91	63.91	63.91
SEA_s	81.67	81.67	83.55	<b>85.10</b>	85.08	85.08	85.08
SEA_g	81.10	81.10	84.53	<b>84.53</b>	84.53	84.53	84.53
Tree_s	72.38	72.54	72.73	<b>72.73</b>	72.70	69.92	69.92
Tree_g	62.37	62.63	64.82	<b>65.28</b>	65.28	65.28	65.28
Waveform_s	84.29	84.29	84.24	<b>85.54</b>	69.99	69.99	69.99
Waveform_g	84.00	<b>84.22</b>	82.44	69.55	69.60	69.60	69.60
Coverttype	<b>92.32</b>	92.27	92.22	92.15	91.63	90.75	90.75
Kddcup99	98.38	98.38	98.37	<b>98.39</b>	98.38	23.25	23.25

The best result on each dataset is given in bold

cannot outperform BPegasos in many cases, e.g. on LED\_g, SEA\_s, SEA\_g and Tree\_g. In contrast, EBPegasos, considering the characteristic, can always obtain better performance than BPegasos on all datasets.

For the Wilcoxon signed-rank test, the null hypothesis is that the two algorithms being compared can achieve equal performance. When the number of datasets is 14, if  $\min\{R^+, R^-\} \leq 21$ , the null hypothesis can be rejected with a confidence level 0.05. Seen from the last row of Table 6, we can conclude that EBPegasos is statistically better than the other methods in terms of averaged prequential accuracy. In terms of the total processing time, Bpegasos, as a single learner, costs the least time on all datasets. Among the ensemble methods, OzaBag, ADWINBag and LevBag cost significantly less time than EBPegasos.

From Table 8, we can see clearly that the function of conditional structural modification in EBPegasos. On Hy\_f, Hy\_s, LED\_s, LED\_g, SEA\_s, SEA\_g and Tree\_g, exploiting old knowledge fully by setting a larger  $\theta$  is more beneficial, while on RBF\_f, RBF\_s, Tree\_s, Waveform\_s, Waveform\_g, Covertype and Kddcup99, promoting structural modification by setting a smaller  $\theta$  (thus learning new knowledge quickly) is better. From the above analysis, we can conclude that the *conditional structural modification* strategies (Sect. 4.3) makes EBPegasos strike a good balance between exploiting old knowledge fully and learning new knowledge. The appropriate value for  $\theta$  may depend on the specific data characteristics.

### 5.3 Comparative analysis on low-dimensional data

It is also very interesting to observe how EBPegasos behaves on the low-dimensional data streams compared with the tree ensembles. Thus, we perform the comparative experiments on the datasets displayed in Table 5. EBPegasos uses the same parameter setting as in Sect. 5.2. All the tree ensembles use the first parameter settings described in Sect. 5.1. The average prequential accuracy and the corresponding Wilcoxon signed-rank test results between EBPegasos and each of the other algorithms are displayed in Table 9. In addition, EBPegasos costs more time but less memory than the tree ensembles on these datasets, but the results are not displayed due to the space constraint. From Table 9, we can conclude that EBPegasos cannot outperform significantly the other tree ensembles except RF on these low-dimensional datasets.

### 5.4 Sensitivity analysis

In this section, we analyze the effect of the parameters except the threshold  $\theta$  in EBPegasos. The regularization parameter  $\lambda$  and RBF kernel width  $\delta$ , are inherent to SVM with the RBF kernel. It is well known that they will affect the performance of SVM greatly, so we do not discuss about them. We assume that the two parameters and the threshold  $\theta$  have been suitably selected and discuss the effects of the numbers of components  $k$  and initial budget  $B$ .

Due to the space constraint, we do not display the performance change when these parameters are set as different values here, but give concluding remarks. The parameters  $k$  and  $B$  are related to the sufficiency of the representation of the decision margin. For datasets with linear decision margins, such as Hy\_f, Hy\_s, Spam\_data and Spam\_corpus datasets, small  $k$  and  $B$  are enough for obtaining competing results, and further increasing  $k$  and  $B$  will not improve the performance significantly. How-

**Table 9** The average prequential accuracy of EBPegasos and the tree ensembles on low-dimensional datasets (%)

Datasets	Oza	ASHT	ADWIN	Lev	DWM	OAUE	RF	EBP
Hy_f	75.33	77.65	77.40	72.12	77.00	79.26	69.85	<b>85.13</b>
Hy_s	76.01	78.17	77.94	72.57	77.01	79.47	70.78	<b>85.43</b>
RBF_f	23.03	22.81	34.13	36.70	32.68	19.89	18.15	<b>55.27</b>
RBF_s	56.08	54.73	55.25	65.38	53.52	47.79	57.09	<b>79.86</b>
LED_s	62.52	65.60	<b>66.51</b>	66.02	66.36	65.76	63.42	65.66
LED_g	62.23	64.20	<b>64.52</b>	63.76	64.58	63.94	61.34	63.91
SEA_s	84.39	85.12	85.28	<b>85.74</b>	85.28	84.66	84.97	85.10
SEA_g	84.18	84.64	84.81	<b>85.35</b>	84.94	83.81	84.53	84.53
Tree_s	77.78	82.79	87.40	<b>92.41</b>	89.61	89.68	68.86	72.73
Tree_g	73.97	77.68	84.02	<b>87.20</b>	82.05	81.79	67.59	65.28
Waveform_s	84.32	83.10	84.52	84.41	82.88	83.48	82.91	<b>85.54</b>
Waveform_g	84.12	82.49	84.16	84.09	81.91	83.23	82.50	<b>84.22</b>
Coverttype	89.01	86.42	88.58	<b>93.25</b>	88.49	89.17	89.72	92.32
Kddcup99	99.33	99.33	99.33	<b>99.58</b>	99.06	98.03	99.64	98.38
Wilcoxon $R^+$	82	69	62	52	67	79	95	
$R^-$	23	36	43	53	38	26	9	

The best result on each dataset is given in bold

ever, on datasets with nonlinear decision margins, e.g., RBF\_f, RBF\_s, Kddcup99 datasets, increasing  $k$  and  $B$  will first improve the performance considerably, then the improvement becomes less and less and finally becomes constant. Clearly, the more complex a margin is, the more components and budgets are needed for representing it. However, when the margin has been sufficiently represented, further increasing  $k$  and  $B$  will not be necessary.

## 6 Conclusions

High-dimensional evolving streaming data is very common but it is very challenging to handle it well. In this paper, we have proposed a novel online ensemble method, called EBPegasos, to cope with the challenges simultaneously posed by high dimensionality and concept drifting in streaming data. In particular, EBPegasos maintains a group of BPegasos classifiers with different budget sizes, monitors and evaluates their performances using ADWIN drift detectors, and modifies the structure of ensemble only when drifts that may produce severe effects are detected. The novelty of EBPegasos originates from the following three aspects: (1) it uses as the base learner an online SVM classifier, which endows it with good capability to work on high-dimensional data; (2) it injects diversity among components by using the advantages of different budget sizes; (3) it uses conditional structural modification, which makes it strike a good balance between exploiting and forgetting the old knowledge. Experiments on four real-world datasets show that EBPegasos has its advantage on high-dimensional

data streams, in terms of not only the average prequential accuracy but also the resource usage, compared with the state-of-the-art tree ensembles. We also show that when all comparison methods use BPegasos as the base learner, EBPegasos can still possess the best capability to deal with various types of concept drifts. We are further working on deriving some theoretical bound for explaining the good adaptability of EBPegasos. The bound may show the maximum drifting amount between time-adjacent concepts that EBPegasos can tolerate.

**Acknowledgements** This work is supported by the National NSF of China (Nos. 61432008, 61503178), NSF and Primary R&D Plan of Jiangsu Province, China (Nos. BE2015213, BK20150587), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

## References

- Abdulsalam H, Skillicorn DB, Martin P (2007) Streaming random forests. In: 11th international database engineering and applications symposium, pp 225–232
- Abdulsalam H, Skillicorn DB, Martin P (2011) Classification using streaming random forests. *IEEE Trans Knowl Data Eng* 23(1):22–36
- Abe S (2005) Support vector machines for pattern classification. Springer, London
- Aggarwal CC, Yu PS (2008) Locust: an online analytical processing framework for high dimensional classification of data streams. In: Proceedings of the 24th IEEE international conference on data engineering, pp 426–435
- Bifet A, Frank E (2010) Sentiment knowledge discovery in twitter streaming data. In: International conference on discovery science, pp 1–15
- Bifet A, Gavalda R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the 7th SIAM international conference on data mining, pp 443–448
- Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavalda R (2009) New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, pp 139–148
- Bifet A, Holmes G, Kirkby R, Pfahringer B (2010a) Moa: massive online analysis. *J Mach Learn Res* 11:1601–1604
- Bifet A, Holmes G, Pfahringer B (2010b) Leveraging bagging for evolving data streams. In: Joint European conference on machine learning and knowledge discovery in databases, pp 135–150
- Bifet A, Holmes G, Pfahringer B, Frank E (2010c) Fast perceptron decision tree learning from evolving data streams. In: Pacific-Asia conference on knowledge discovery and data mining, pp 299–310
- Bifet A, Pfahringer B, Read J, Holmes G (2013) Efficient data stream classification via probabilistic adaptive windows. In: Proceedings of the 28th annual ACM symposium on applied computing, pp 801–806
- Brzeziński D, Stefanowski J (2011) Accuracy updated ensemble for data streams with concept drift. In: International conference on hybrid artificial intelligence systems, pp 155–163
- Brzezinski D, Stefanowski J (2014a) Combining block-based and online methods in learning ensembles from concept drifting data streams. *Inf Sci* 265:50–67
- Brzezinski D, Stefanowski J (2014b) Reacting to different types of concept drift: the accuracy updated ensemble algorithm. *IEEE Trans Neural Netw Learn Syst* 25(1):81–94
- Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7:1–30
- Denil M, Matheson D, De Freitas N (2013) Consistency of online random forests. In: Proceedings of the 30th international conference on machine learning, pp 1256–1264
- Do TN, Lenca P, Lallich S, Pham NK (2010) Classifying very-high-dimensional data with random forests of oblique decision trees. In: Guillet F, Ritschard G, Zighed DA, Briand H (eds) *Advances in knowledge discovery and management*. Springer, Berlin, Heidelberg, pp 39–55
- Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the 6th ACM SIGKDD international conference on knowledge discovery and data mining, pp 71–80
- Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. *IEEE Trans Neural Netw* 22(10):1517–1531
- Gama J, Fernandes R, Rocha R (2006) Decision trees for mining data streams. *Intell Data Anal* 10(1):23–45

- Gama J, Sebastiao R, Rodrigues PP (2013) On evaluating stream learning algorithms. *Mach Learn* 90(3):317–346
- Gama J, Zliobaite I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. *ACM Comput Surv* 46(4):44
- Holmes G, Kirkby R, Pfahringer B (2005) Stress-testing hoeffding trees. In: European conference on principles of data mining and knowledge discovery, pp 495–502
- Hosseini MJ, Gholipour A, Beigy H (2015) An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. *Knowl Inf Syst* 46:1–31
- Hsu CW, Chang CC, Lin CJ, et al (2003) A practical guide to support vector classification. <https://www.cs.fu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf>
- Katakis I, Tsoumakas G, Banos E, Bassiliades N, Vlahavas I (2009) An adaptive personalized news dissemination system. *J Intell Inf Syst* 32(2):191–212
- Katakis I, Tsoumakas G, Vlahavas I (2010) Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowl Inf Syst* 22(3):371–391
- Kolter JZ, Maloof MA (2007) Dynamic weighted majority: an ensemble method for drifting concepts. *J Mach Learn Res* 8:2755–2790
- Krempel G, Zliobaite I, Brzeziński D, Hüllermeier E, Last M, Lemaire V, Noack T, Shaker A, Sievi S, Spiliopoulou M, Stefanowski J (2014) Open challenges for data stream mining research. *SIGKDD Explor* 16(1):1–10
- Lakshminarayanan B, Roy DM, Teh YW (2014) Mondrian forests: efficient online random forests. In: Advances in neural information processing systems 27: annual conference on neural information processing systems 2014, Montreal, Quebec, Canada, pp 3140–3148
- Liu Y, Zhou Y (2014) Online detection of concept drift in visual tracking. In: International conference on neural information processing, pp 159–166
- McCallum A, Nigam K et al (1998) A comparison of event models for naive bayes text classification. In: AAAI-98 workshop on learning for text categorization, vol 752, pp 41–48
- Minku LL, Yao X (2012) Ddd: A new ensemble approach for dealing with concept drift. *IEEE Trans Knowl Data Eng* 24(4):619–633
- Minku LL, White AP, Yao X (2010) The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Trans Knowl Data Eng* 22(5):730–742
- Oza NC (2005) Online bagging and boosting. In: 2005 IEEE international conference on systems, man and cybernetics, vol 3, pp 2340–2345
- Pappu V, Pardalos PM (2014) High-dimensional data classification. In: Aleskerov F, Goldengorin B, Pardalos PM (eds) Clusters, orders, and trees: methods and applications. Springer, New York, pp 119–150
- Rutkowski L, Pietruczuk L, Duda P, Jaworski M (2013) Decision trees for mining data streams based on the McDiarmid's bound. *IEEE Trans Knowl Data Eng* 25(6):1272–1279
- Saffari A, Leistner C, Santner J, Godec M, Bischof H (2009) On-line random forests. In: 2009 IEEE 12th international conference on computer vision workshops, pp 1393–1400
- Shalev-Shwartz S, Singer Y, Srebro N, Cotter A (2011) Pegasos: primal estimated sub-gradient solver for SVM. *Math Program* 127(1):3–30
- Tomasev N, Radovanovic M, Mladenec D, Ivanovic M (2014) The role of hubness in clustering high-dimensional data. *IEEE Trans Knowl Data Eng* 26(3):739–751
- Wang Z, Crammer K, Vucetic S (2012) Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale SVM training. *J Mach Learn Res* 13(1):3103–3131
- Wang D, Wu P, Zhao P, Wu Y, Miao C, Hoi SC (2014) High-dimensional data stream classification via sparse online learning. In: 2014 IEEE international conference on data mining, pp 1007–1012
- Ye Y, Wu Q, Huang JZ, Ng MK, Li X (2013) Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognit* 46(3):769–787
- Zhang X, Furtlehner C, Germain-Renaud C, Sebag M (2014) Data stream clustering with affinity propagation. *IEEE Trans Knowl Data Eng* 26(7):1644–1656
- Zliobaite I, Gabrys B (2014) Adaptive preprocessing for streaming data. *IEEE Trans Knowl Data Eng* 26(2):309–321
- Zliobaite I, Bifet A, Read J, Pfahringer B, Holmes G (2015) Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Mach Learn* 98(3):455–482