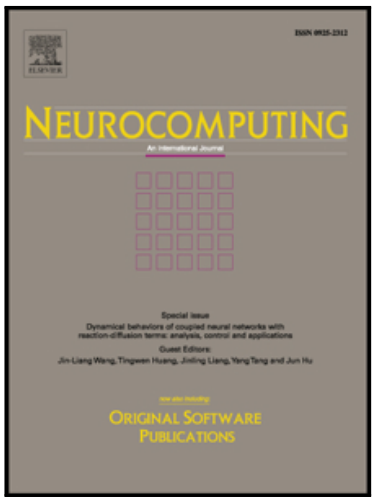


Adaptive Online Extreme Learning Machine by Regulating Forgetting Factor by Concept Drift Map

Hualong Yu , Geoffrey I. Webb

PII: S0925-2312(19)30157-2
DOI: https://doi.org/10.1016/j.neucom.2018.11.098
Reference: NEUCOM 20428

To appear in: *Neurocomputing*

Received date: 6 October 2017
Revised date: 2 September 2018
Accepted date: 29 November 2018

Please cite this article as: Hualong Yu , Geoffrey I. Webb , Adaptive Online Extreme Learning Machine by Regulating Forgetting Factor by Concept Drift Map, *Neurocomputing* (2019), doi: https://doi.org/10.1016/j.neucom.2018.11.098

Adaptive Online Extreme Learning Machine by Regulating Forgetting Factor by Concept Drift Map

Hualong Yu[1, 2,*], Geoffrey I. Webb[2]

1. School of Computer, Jiangsu University of Science and Technology, Zhenjiang Jiangsu, China

2. Faculty of Information Technology, Monash University, VIC 3800, Australia

Abstract. In online-learning, the data is incrementally received and the distributions from which it is drawn may keep changing over time. This phenomenon is widely known as *concept drift*. Such changes may affect the generalization of a learned model to future data. This problem may be exacerbated by the form of the drift itself changing over time. Quantitative measures to describe and analyze the concept drift have been proposed in previous work. A description composed from these measures is called a *concept drift map*. We believe that these maps could be useful for guiding how much knowledge in the old model should be forgotten. Therefore, this paper presents an adaptive online learning model that uses a concept drift map to regulate the forgetting factor of an extreme learning machine. Specifically, when a batch of new instances are labeled, the distribution of each class on each attribute is firstly estimated, and then it is compared with the distribution estimated in the previous batch to calculate the magnitude of concept drift, which is further used to regulate the forgetting factor and to update the learning model. Therefore, the novelty of this paper lies in that a quantitative distance metric between two distributions constructed on continuous attribute space is presented to construct *concept drift map* which can be further associated with the forgetting factor to make the learning model adapt the concept drift. Experimental results on several benchmark stream data sets show the proposed model is generally superior to several previous algorithms when classifying a variety of data streams subject to drift, indicating its effectiveness and feasibility.

Keywords. Online learning, Extreme learning Machine, Concept drift map, Online extreme learning machine, Forgetting factor

1. Introduction

Recent years have witnessed ever increasing interest in the topic of online learning. Meanwhile, the online deployment of learned models gives increasing urgency to the development of effective and efficient mechanisms to address learning in the context of nonstationary distributions, or as it is commonly called *concept drift* [1-2]. Many real-world applications associate with nonstationary data stream, including network security, market basket analysis, industrial procedure control, intelligent user interface, weather forest and credit card fraud detection [3-5].

Unlike batch learning which constructs models in a static scenario, online learning faces two significant challenges. First, the learner must process each training example or each batch of training instances once "on arrival", without the use of storage or reprocessing [6]. Second, there might be *concept drift*, i.e., the data distributions may change over time. Thus, the learning machine deployed in a nonstationary stream environment must learn examples by means of *one pass* while adapting to dynamic variation in data distributions. The first term could be satisfied by either modifying the traditional learning approaches to incrementally tune the model parameters [7-9], or using ensemble learning algorithms which train a new learner on each newly received batch of instances [10-18]. As for the second term, it is significantly more complex considering there might exist many concept drift types [1], e.g., abrupt drift, gradual drift and recurring drift. When one or multiple drift types appear in the data stream, the change detection mechanisms [19-23] and forgetting mechanisms [24-27] might be required.

We note that in recent years, online learning models with forgetting mechanisms have been widely proposed, and the improved classification results have been presented. However, we also note that the forgetting factors adopted in most models are always constant, and only a few methods make the forgetting factors adapt to the magnitude or rate of distribution drift. This might be regarded as an inherent defect of most existing models. We argue that to make the learner adapt effectively, it is important to match the forgetting factors to the rate of drift. That means the rate of concept drift should be monitored in the data stream, and used to tune the forgetting factor. In a recent work, Webb *et al.*, [28] presented a method called a *concept drift map*, to quantitatively describe and analyze the rate of concept drift. Specifically, a concept drift map corresponds to a curve, in which each point indicates the variation of two marginal distributions defined on two

continuous batches of training instances acquired before and after that time point, respectively. That means a concept drift map can reflect the real-time rate of concept drift, thereby we believe it should be a useful tool for providing guidance to update models.

In this paper, we integrate concept drift maps into the online extreme learning machine (ELM) algorithmic framework, and use it to regulate the forgetting factor adaptively. Considering most data stream applications hold numeric attributes, thus we modify the metric of drift magnitude in the traditional concept drift map [28]. In particular, we provide a strong hypothesis to assume during each time interval, for each attribute, the instances belonging to a category complies with the Gaussian distribution. Then the distance between two Gaussian distributions can be measured to reflect the drift magnitude. In addition, an online ELM algorithm with forgetting mechanism [25] has been adopted. Instead of designating an empirical & constant forgetting factor, we design a metric associated with the real-time drift magnitude to regulate it, which guarantees the learning model be adaptively updated with the variation of knowledge. Therefore, the originalities of the paper are two folds: 1) a quantitative distance metric between two distributions constructed on continuous attribute space is presented, which can promote the description quality of the concept drift map and 2) the distance reflecting concept drift magnitude is associated with the forgetting factor to guide the online neural network model to adapt the drift data stream. We compare the proposed model with several previous works in terms of 0-1 loss, and the results indicate the effectiveness and superiority of our learning model.

The remainder of this paper is organized as follows. Section 2 briefly reviews some related work and highlights several important methods. Section 3 presents the proposed methods, including how to design the concept drift map on continuous attribute space, how to establish associations between the drift metric and the forgetting factor, and the description of the proposed learning model. In Section 4, we present the experimental results and analysis. Finally, Section 5 concludes the paper.

2. Related work

2.1 Extreme learning machine

Extreme learning machine (ELM) is a fast and accurate algorithm for training a single

hidden-layer feedback network (SLFN) [29-32]. The basic network structure of SLFN can be observed in Fig.1. In contrast to the traditional back-propagation algorithm [33], the weights and biases between the input layer and hidden layer in ELM are fixed and do not need to be adjusted. Therefore, ELM can directly solve the weight matrix between the hidden layer and the output layer by the least-squares approach. ELM has been applied in many real-world applications due to its fast training speed and strong generalization performance.

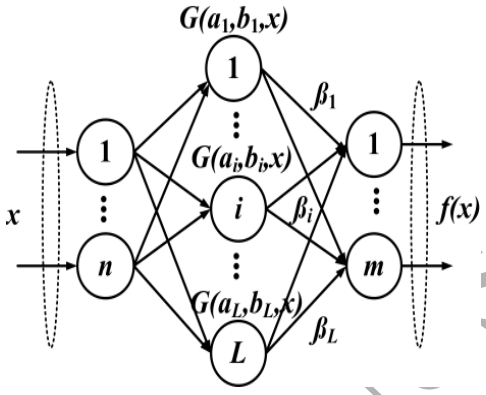


Fig.1 The basic network structure of SLFN.

Considering a classification problem with N training instances to distinguishing m classes, then the i^{th} training instance can be represented as (x_i, t_i), where x_i is a $n\times1$ input vector, while t_i is the corresponding $m\times1$ output vector. If an SLFN can approximate these N examples with zero error, it then implies that there exist β_j, a_j, b_j such that:

$$f(x_i)=\sum_{j=1}^{L}\beta_j G(a_j,b_j,x_i)=t_i,\ i=1,2,...,N \tag{1}$$

where a_j and b_j respectively denote the weight and bias belonging to the j^{th} hidden layer node, β_j represents the weight vector connecting the j^{th} hidden layer node and the output layer nodes, and G is the activation function of neural network. Then Eq.(1) can be written compactly as:

$$\mathrm{H}\beta=\mathrm{T} \tag{2}$$

where

$$\mathrm{H}=\begin{bmatrix} G(a_1,b_1,x_1) & \cdots & G(a_L,b_L,x_1) \\ \vdots & \ddots & \vdots \\ G(a_1,b_1,x_N) & \cdots & G(a_L,b_L,x_N) \end{bmatrix} \tag{3}$$

$$\beta = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \vdots & \ddots & \vdots \\ \beta_{L1} & \cdots & \beta_{Lm} \end{bmatrix} \tag{4}$$

and

$$\mathrm{T} = \begin{bmatrix} t_{11} & \cdots & t_{N1} \\ \vdots & \ddots & \vdots \\ t_{1m} & \cdots & t_{Nm} \end{bmatrix} \tag{5}$$

It is clear that in Eq.(2), only β is unknown, hence we can adopt the least square algorithm to acquire its solution, which could be described as follows.

$$\beta = \mathrm{H}^{\dagger}\mathrm{T} = \begin{cases} \mathrm{H}^{\mathrm{T}}(\mathrm{H}\mathrm{H}^{\mathrm{T}})^{-1}\mathrm{T}, \text{ when } N \le L \\ (\mathrm{H}\mathrm{H}^{\mathrm{T}})^{-1}\mathrm{H}^{\mathrm{T}}\mathrm{T}, \text{ when } N > L \end{cases} \tag{6}$$

where $\mathrm{H}^{\dagger}$ denotes the Moore-Penrose generalized inverse of the hidden layer output matrix H, which can guarantee the solution is the least-norm least-square solution of Eq. (2).

2.2 Online sequential Extreme learning machine

In recent years, ELM has also been widely adopted to tackle the data stream classification problem. Liang *et al.*, [34] presented an online sequential ELM (OS-ELM) that can incrementally learn data one-by-one or chunk-by-chunk without reserving the old training instances. OS-ELM has already been proven to acquire the same classification performance as ELM trained on the complete set of data [34].

In OS-ELM, the update rule of the output layer weight matrix β can be represented as:

$$\beta^{(k+1)} = \beta^{(k)} + \mathrm{P}_{k+1}\mathrm{H}_{k+1}^{\mathrm{T}}(\mathrm{T}_{k+1} - \mathrm{H}_{k+1}\beta^{(k)}) \tag{7}$$

where H_{k+1} and T_{k+1} correspond the hidden layer output matrix and the target matrix of the new observations in the $(k+1)^{\text{th}}$ chunks, while $\beta^{(k)}$ and $\beta^{(k+1)}$ denote the output layer weight matrix β after receiving the k^{th} and $(k+1)^{\text{th}}$ chunks, respectively. As for P_{k+1}, it can be calculated by the following formula.

$$\mathrm{P}_{k+1} = \mathrm{P}_k - \mathrm{P}_k\mathrm{H}_{k+1}^{\mathrm{T}}(\mathrm{I} + \mathrm{H}_{k+1}\mathrm{P}_k\mathrm{H}_{k+1}^{\mathrm{T}})^{-1}\mathrm{H}_{k+1}\mathrm{P}_k \tag{8}$$

i.e., P_k can be also iteratively updated, and the initial P_0 can be represented as:

$$\mathrm{P}_0 = (\mathrm{H}_0^{\mathrm{T}}\mathrm{H}_0)^{-1} \tag{9}$$

where H_0 indicates the initial hidden layer output matrix. Let's return to Eq.(7), as H_{k+1}, T_{k+1} and P_{k+1} all can be calculated only by the new received instances, thereby the output layer weight matrix β can be adjusted to adapt both old and new instances, but not need to be recalculated with all training examples. It is not difficult to observe that the main merit of OS-ELM algorithm is to decrease the time-complexity to a large extent, when we can only acquire the training instances dynamically [34]. In addition, it has been proven that OS-ELM can acquire the totally same classification performance as ELM trained on the complete set of data [34].

Singh *et al.*, [35] applied OS-ELM in a network intrusion detection task by combining an ensemble of Filtered, Correlation and Consistency based feature selection techniques. Considering the randomness of ELM, Lan *et al.*, [36] combined OS-ELM with ensemble learning to present EOS-ELM model, resulting in improved performance in comparison to OS-ELM.

2.3 Online sequential Extreme learning machine with forgetting mechanism

OS-ELM can effectively tackle online learning problem, however, it might lose efficacy when the data stream is nonstationary, i.e., there exists *concept drift*. That is to say, OS-ELM pays the same attention for both old and new knowledge, causing the model must adapt the concept drift slowly. Therefore, Zhang and Wang [25] presented a selective forgetting extreme learning machine algorithm (SF-ELM).

Suppose the learning model has learned the k^{th} batch of instances, then when it receives the $(k+1)^{\text{th}}$ batch H_{k+1} and its label set T_{k+1}, according to Eq.(6), the new output layer weight matrix $\beta^{(k+1)}$ can be represented as:

$$\beta^{(k+1)} = \left(\begin{bmatrix} \mathrm{H}_k \\ \mathrm{H}_{k+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathrm{H}_k \\ \mathrm{H}_{k+1} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathrm{H}_k \\ \mathrm{H}_{k+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \mathrm{T}_k \\ \mathrm{T}_{k+1} \end{bmatrix} \tag{10}$$

which can be further expressed as follows

$$\beta^{(k+1)} = (\mathrm{H}_k^{\mathrm{T}}\mathrm{H}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1})^{-1}(\mathrm{H}_k^{\mathrm{T}}\mathrm{T}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{T}_{k+1}) \tag{11}$$

Considering both $\mathrm{H}_k^{\mathrm{T}}\mathrm{H}_k$ and $\mathrm{H}_k^{\mathrm{T}}\mathrm{T}_k$ are only related with the old instances, hence a forgetting factor η ($0 \le \eta \le 1$) can be inserted into these two terms, then Eq.(11) can be rewritten as:

$$\beta^{(k+1)} = ((1\text{-}\eta)\mathrm{H}_k^{\mathrm{T}}\mathrm{H}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1})^{-1}((1-\eta)\mathrm{H}_k^{\mathrm{T}}\mathrm{T}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{T}_{k+1}) \tag{12}$$

Specifically, the forgetting factor η associates with the level of forgetting old knowledge. The larger η is, the more old knowledge will be forgotten. We also note that there are two specific cases that when η=1, the model would discard all old experience but merely use the newly received batch of training instances, while when η=0, it completely degrades into OS-ELM model.

To incrementally solve $\beta^{(k+1)}$, we let,

$$\mathrm{P}_{k+1} = ((1\text{-}\eta)\mathrm{H}_k^{\mathrm{T}}\mathrm{H}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1})^{-1} \tag{13}$$

then taking the inversion on both sides in Eq.(13), we have,

$$\mathrm{P}_{k+1}^{-1} = (1\text{-}\eta)\mathrm{P}_k^{-1} + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1} \tag{14}$$

Taking Eq.(14) into Eq.(12), then $\beta^{(k+1)}$ can be represented as:

$$\begin{aligned}\beta^{(k+1)} &= \mathrm{P}_{k+1}((1-\eta)\mathrm{H}_k^{\mathrm{T}}\mathrm{T}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{T}_{k+1}) \\ &= \mathrm{P}_{k+1}((1-\eta)\mathrm{P}_k^{-1}(\mathrm{P}_k\mathrm{H}_k^{\mathrm{T}}\mathrm{T}_k) + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{T}_{k+1}) \\ &= \mathrm{P}_{k+1}((1-\eta)\mathrm{P}_k^{-1}\beta^{(k)} + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{T}_{k+1}) \\ &= \mathrm{P}_{k+1}((\mathrm{P}_{k+1}^{-1} - \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1})\beta^{(k)} + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{T}_{k+1}) \\ &= \beta^{(k)} + \mathrm{P}_{k+1}\mathrm{H}_{k+1}^{\mathrm{T}}(\mathrm{T}_{k+1} - \mathrm{H}_{k+1}\beta^{(k)})\end{aligned} \tag{15}$$

When the learning model receives data one-by-one, then according to Sherman-Morrison formula [40], we have,

$$\mathrm{P}_{k+1} = \frac{\mathrm{P}_k}{(1-\eta)} - \frac{Q_k Q_k^{\mathrm{T}}}{(1-\eta)(1-\eta+\mathrm{H}_{k+1}Q_k)} \tag{16}$$

where

$$Q_k = \mathrm{P}_k\mathrm{H}_{k+1}^{\mathrm{T}} \tag{17}$$

While if the model learns data in the mode of chunk-by-chunk, it must update P_{k+1} by,

$$\mathrm{P}_{k+1} = ((1-\eta)\mathrm{P}_k + \mathrm{H}_{k+1}^{\mathrm{T}}\mathrm{H}_{k+1})^{-1} \tag{18}$$

As P_k can be incrementally updated, thus when the forgetting factor η is designated, the output layer weight matrix $\beta^{(k+1)}$ can be deduced by $\beta^{(k)}$ directly. In Ref. [25], the model learns new data one-by-one and designates a constant forgetting factor η=0.01, but does not consider the influence of distribution drift.

Specifically, the model could adaptively decide whether to forget a part of old knowledge by comparing the training error on the new received instance with a previously defined threshold. However, the forgetting factor in this model is still a constant empirical value.

Profiting from the idea of EOS-ELM, Zhao *et al.*, [24] also proposed an ensemble online sequential ELM with forgetting mechanism which is called FOS-ELM. This algorithm integrated both incremental and decremental mechanisms into the learning procedure, as well adopted a sliding window to learn from each new batch of training instances and discard the oldest one. That means at any time point, the learning model only reserves the knowledge of several recent training chunks. Xu and Wang [37] presented a dynamic extreme learning machine (DELM) model to deal with concept drift problem. DELM employs structure learning, i.e., when the error rate estimated by the learner is higher than a given threshold, it assesses that concept drift has occurred, and then adds hidden-layer nodes to improve its approximation capability. In addition, we note that DELM adopts double hidden-layers to promote the classification performance of the online learner. Furthermore, the possibility of combining online learning with class imbalance learning has also been investigated [38-39]. Meta-cognitive online sequential extreme learning machine (MOS-ELM) incorporates OS-ELM and weighted extreme learning machine (WELM) algorithms to promote classification performance of an online learner in a skewed data stream [38]. Taking into account the robustness of ensemble learning, Mirza *et al.*, [39] also adopted ensemble classifiers to deal with imbalanced data streams. Here, a change detection mechanism is used to capture distribution drift, after which a new classifier is trained from only newly received instances and then added into the ensemble classifier's pool.

2.4 Variable forgetting strategies with change detection mechanism

In addition to the methods listed above, there are also several algorithms considering to use change detection strategies to regulate the forgetting factors. Two representative work are Floating Rough Approximation (FLORA) series algorithms [41] and Exponentially Weighted Moving Average (EWMA) algorithm [42].

The FLORA series algorithms is constructed on rough set theory. The algorithms adopt three variables (the low and high coverage of the set containing the description items matching only positive instances, and the acceptable predictive accuracy) to monitor the concept drift and how

much drift has happened. Then the drift magnitude is used to dynamically control the size of sliding window which decide how many instances are used to construct the newest predictive model. The algorithms have two limitations as below, 1) it can be only used to deal with the data stream only having discrete attributes, 2) it can only adopt rule-based classifiers. Therefore, it can not be integrating into our experimental framework.

The EWMA algorithm computes a recent estimate of the error rate, μ_t, by progressively down-weighting older data: $Z_0 = \mu_0$ and $Z_t = (1-\lambda)Z_{t-1} + \lambda e_t, t > 0$, where e_t is the error at the current instance. It can be shown that, independently of the distribution of X_t, the mean and standard deviation of Z_t are $\mu z_t = \mu_t$ and $\sigma z_t = \sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2t}}\sigma_E$. Assume that before the change point that $\mu_t = \mu_0$. The EWMA estimator Z_t fluctuates around this value. When a change occurs, the value of μ_t and Z_t will react to this by diverging from μ_0 towards μ_c. We can signal a change when $Z_t > \mu_0 + L\sigma z_t$. The control limit parameter *L* determines how far from Z_t must diverge from μ_0 before a change is flagged. The algorithm is independent of specific classification models, however, we also note that when a change happened, the classifier must be reset. A limitation of this algorithm lies in that it can be used only for the binary-class problem.

In this paper, we focus on investigating how to take advantage of a concept drift map to adaptively decide the forgetting factor, in the context of a single ELM model. Therefore, in our subsequent experiments, we only compared our proposed algorithm with OS-ELM, FOS-ELM, SF-ELM and EWMA-ELM algorithms. While these mechanisms could further be utilized for structure learning or class imbalance learning, that lies outside the scope of the current paper.

3. Methods

3.1 Concept drift map

Concept drift map [28] is a visual and quantitative description tool for presenting concept drift in entire data stream. Fig.2 shows an intuitionistic example of drift map collecting from the noaa data set which records the climate data over 50 years. The example is expected to help readers to

be simply familiar with the concept drift map before introducing its calculation rule. Specifically, in the figure, the vertical axis variation gives an idea of the size of the concept drift and is calculated based on the rule proposed in section 3.2. In this simple example, we map the drift in the climate distribution between every two consecutive years. The map reveals that there exists a relatively dramatic variation from the 7^{th} to 8^{th}, 8^{th} to 9^{th} and the 9^{th} to 10^{th} years in our record. Then we can refer to the other relevant history climate information to determine the causes of this extended period of rapid change.

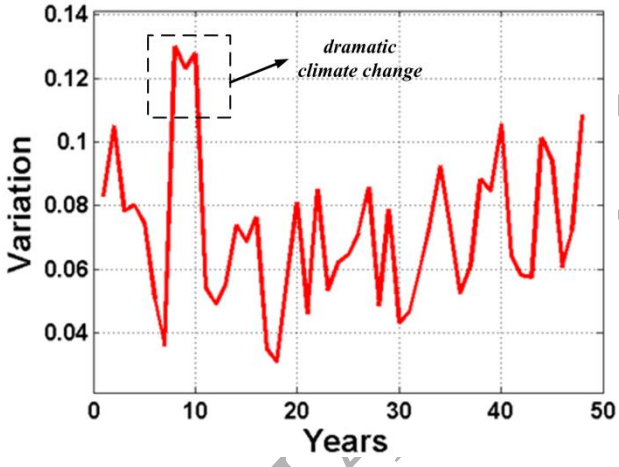


Fig.2 An example of concept drift map for the noaa data.

The example tells us that the concept drift map is a useful tool to describe the drift magnitude in an online learning scenario and to help the decision maker to understand and analyze concept drift by historical data. Furthermore, we emphasize if a dynamic online learner could adapt concept drift in real-time, then it could minimize decreases in classification performance caused by the concept drift. In order to get to this goal, in this paper we try to associate the forgetting mechanism of online learning model with the concept drift map.

The concept drift map is a quantitative tool to track the drift magnitude. The technique can be applied with any measure for calculating the distance between two different distributions. In the work presented by Webb et al.,[28], to simplify the complexity of calculation, each numeric attribute has been previously discretized using 5 bin equal frequency discretization. That means all values on each attribute should be firstly sorted, and then the sorted sequence is averaged divided into five bins where each bin uses a specific discrete value to replace the original continuous attribute values. On the transformed discrete attribute space, the Total Variation Distance [43] is adopted to calculate the distance between two continuous distributions. However, discretization

might result in information loss, and meanwhile, to guarantee to acquire an exact discretization result, all data chunks are required to be previously prepared, which can't satisfy the requirement of the practical online learning applications. If the lower and upper boundaries of each bin are calculated by using only the initial data chunk, then it is foreseeable that the new discretized results must be seriously inaccurate when concept drift happens. Therefore, it is necessary to find an alternative of the original variation metric to measure concept drift in the continuous feature space. To achieve this goal, in this paper, we provide an assumption about the form of distribution, i.e., each data chunk conforms to normal distribution, which can simplify the calculation of drift magnitude directly from the numeric values, and further, provide a relatively accurate result when the real data does not deviate the normal distribution obviously. As we know, many real world data approximately conform to the normal distribution [44-47], thus we can say that the assumption presented in this paper has a low risk.

For a given data chunk $\Phi=\{(x_i, y_i), x_i \in \Re^n, y_i \in \{1,2,...,m\}\}$, then for any class j, where $j \in \{1,2,,...m\}$, we can extract all instances belonging to the j^{th} class in Φ, and for each attribute k, calculate its mean μ_j^k and variance σ_j^k which fully specify a specific Gaussian distribution $N(\mu_j^k,(\sigma_j^k)^2)$. For two consecutive time points t and u, we can receive two different data chunks Φ_t and Φ_u, then two Gaussian distributions $X_t \sim N_t(\mu_t,\sigma_t^2)$ and $X_u \sim N_u(\mu_u,\sigma_u^2)$ can be obtained. When we consider the distance or variation $\upsilon_{t,u}$ between these two distributions, the overlapping area of two corresponding Gaussian curves which indicates the similarity $s_{t,u}$ of these two distributions could be used (see the diagram form in Fig.3).

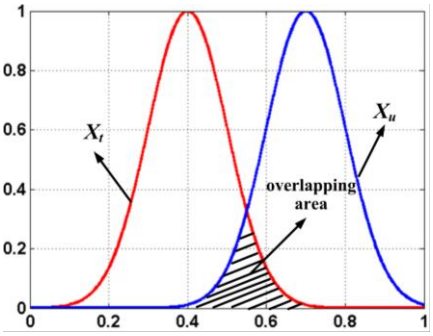


Fig.3 Diagram illustrating the similarity $s_{t,u}$ calculation between two Gaussian distributions, where the area of overlap is considered their similarity.

Let c denote the point of intersection where two Gaussian curves meet, then the area of the overlapping part, i.e., their similarity can be simply calculated as follows,

$$\begin{aligned} s_{t,u} &= P(X_t > c) + P(X_u < c) \\ &= 1\text{-}F_t(c)+F_u(c) \\ &= 1\text{-}\frac{1}{2}\mathrm{erf}(\frac{c-\mu_t}{\sqrt{2}\sigma_t})+\frac{1}{2}\mathrm{erf}(\frac{c-\mu_u}{\sqrt{2}\sigma_u}) \end{aligned} \tag{19}$$

where the point of intersection c can be calculated as follows,

$$c = \frac{\mu_u\sigma_t^2 - \sigma_u\left(\mu_t\sigma_u + \sigma_t\sqrt{(\mu_t-\mu_u)^2 + 2(\sigma_t^2-\sigma_u^2)\log(\frac{\sigma_t}{\sigma_u})}\right)}{\sigma_t^2-\sigma_u^2} \tag{20}$$

Note that in the calculation procedure above, on x-axis, the curve corresponding to the distribution X_t must be guaranteed to appear on the left side of the curve corresponding to X_u. If these two curves arrange in the reverse order, then Eq.(19) would be changed to $S_{t,u}=P(X_t<c)+P(X_u>c)$.

Next, the distance or variation $\upsilon_{t,u}$ can be calculated as,

$$\upsilon_{t,u} = 1 - s_{t,u} \tag{21}$$

We note a phenomenon that the more similar two distributions are, the smaller the distance between these two distributions, which conforms to our anticipation. Several basic natures to define a distance can be also satisfied by observing several functions above. Due to for any two distributions α and β, there exists a sole point of intersection c, that means $s_{\alpha,\beta}=s_{\beta,\alpha}$ according to Eq.(19), then $v_{\alpha,\beta}=v_{\beta,\alpha}$ according to Eq.(21), indicating the proposed distance or variance metric satisfies the conditions of the identity and symmetry. Meanwhile, we observe that the proposed variance metric is merely correlated with the area of the overlapping part between two Gaussian curves, thereby when and only when two distributions are totally same, the variance equals to be 0, otherwise, the distance will be larger than 0. Based on this boundary condition, it is not difficult to observe that the proposed variance metric satisfies the condition of the non-negativity. Moreover, it is intuitively to observe that the proposed distance metric satisfy the condition of the triangle inequality by drawing any three normal distributions. However, it may be difficult to strictly prove this theory in mathematics.

In addition, we also note that for each attribute, there exists a variation, thus to present the integrated variation on a certain category, we use the mean value over all attributes that can be

calculated as follows,

$$\upsilon_{t,u}(j)=\frac{1}{n}\sum_{l=1}^{n}(1-s_{t,u}(l,j)) \tag{22}$$

where $s_{t,u}(l,j)$ denotes the similarity of two distributions in time window *t* and *u* which only constructed on the instances belonging to the j^{th} class and the l^{th} attribute, while $\upsilon_{t,u}(j)$ denotes the variation of the j^{th} class instances between these two distributions.

Furthermore, we define two different ways to present the total variation between two data chunks over all classes. One extracts the maximal variation, and the other calculates the average variation on all classes. These two ways can be implemented by,

$$\upsilon_{t,u}=\arg\max_{j\in\{1,2,\ldots,m\}}\upsilon_{t,u}(j) \tag{23}$$

$$\upsilon_{t,u}=\frac{1}{m}\sum_{j=1}^{m}\upsilon_{t,u}(j) \tag{24}$$

It is clear that the $\upsilon_{t,u}$ can be used to show the distribution variable magnitude between two continuous time windows, and further to be used to generate the concept drift map.

3.2 Dynamic regulation strategy of forgetting factor

Next, we need to associate the concept drift map with the forgetting factor in SF-ELM. There is a basic principle to design the association, that is the larger the drift variation υ is, the larger the forgetting factor η also should be as the historical data will be less relevant to the current distribution. Based on this principle, we design the association function as follows,

$$\eta=1-\lambda-(1-\lambda)\left(1\Big/(\xi\exp(1))^{\upsilon}\right) \tag{25}$$

where λ is a constant between [0,1] to denote the fixedly reserved percentage of old knowledge, ξ is a user-defined constant to regulate the old knowledge magnitude that should be forgotten, and $\exp(1)$ denotes the natural logarithm e. In this paper, ξ is empirically designated as 10. By observing Eq.(25), we note that η is positive respect with υ, which is consistent with our anticipation. We also note that λ will be an important parameter to influence the performance of the learner. Intuitively speaking, if the data chunk is too small, there might exist a large expected bias for distribution estimation, i.e., the estimation of υ would be extremely inaccurate, then we should designate a relatively large value for λ, and vice verse. The discussion about the impact

of λ will be carried out in Section 5 in detail.

3.3 Description of the proposed algorithm

We call the proposed adaptive online ELM algorithm with regulating forgetting factor by concept drift map as AO-ELM. The flow path of this algorithm is described in Fig.4.

globals
L: the number of hidden layer nodes
sig: the type of activation function
λ: the fixedly reserved percentage of old knowledge
ξ: the user-defined constant to regulate the old knowledge magnitude that should be forget
procedure Θ=AO-ELM($\Omega=(\Phi_0, \Phi_1,..., \Phi_z)$, *L*, *sig*, λ, ξ)
1. **collect** the first data chunk Φ_0;
2. **generate** the random weight and bias matrices *a* and *b* which are associated with *L* and *sig*;
3. **generate** H_0 by *a*, *b* and Φ_0;
4. **calculate** $P_0= H_0^T H_0$;
5. **calculate** $\beta^{(0)}$ by Eq.(6);
6. **let** i=0;
7. **while** there are more data chunks, do the following:
8. $i=i+1$**;**
9. **receive** the i^{th} data chunk Φ_i;
10. **generate** H_i by *a*, *b* and Φ_i;
11. **acquire** predictive label subset $\Theta_i=H_i\beta^{(i-1)}$;
12. **calculate** the variation υ_i by Eq.(23) for the maximal variation or Eq.(24) for the average variation;
13. **calculate** η_i with υ_i, λ and ξ by Eq.(25);
14. **calculate** P_i with P_{i-1}, H_i and η_i by Eq.(18);
15. **update** $\beta^{(i)}$ with H_i, P_i and $\beta^{(i-1)}$ by Eq.(15);
16. **end while**

Fig.4 Flow path description of AO-ELM algorithm.

In Fig.4, *z* represents the number of data chunks. In addition, in the 12th and 13th lines, the variation υ and the forgetting factor η need to be delay calculated, i.e., for the i^{th} received data chunk, its corresponding variation υ_t and the forgetting factor η_t should be calculated by the $(i\text{-}2)^{th}$ and $(i\text{-}1)^{th}$ chunks.

4. Experiments

4.1 Data sets

We use four public real-world stream data sets, Powersupply and HyperPlane from the Stream

Data Mining Repository[1], electricity from MOA website[2], and noaa from National Centers for Environmental Information[3], as well as four synthetic stream data sets generated by data stream generators, called SEA, spiral, Gaussian and checkerboard, respectively. Due to the limitation of the space, the details about how to generate these synthetic data sets, please refer the Ref. [48]. Note that each data set above involves concept drift to some extent. Specifically, except for the electricity data set, all other data sets merely include continuous attributes. As for the electricity data set, it owns 7 continuous attributes and 1 discrete attribute. Also, to investigate the impact of data chunk size (received number of instances in one time stamp) to classification performance, for each data set, four different chunk magnitudes are provided. In particular, for several real-world data streams, the choices of the chunk magnitude are meaningful and can match the natural cycles, e.g., Powersupply collects data stream with an interval of one hour, thus 72-168-240-720 corresponds to 3 days-one week-10 days-one month, while noaa acquires instances with an interval of one day, thereby 15-30-90-365 indicates half a month-one month-one season-one year. Table 1 gives the detailed description of these data sets.

Table 1. Description about the used data sets. Except for electricity, which has one discrete attribute, all other attributes are continuous.

Data set	Number of attributes	Number of instances	Number of classes	Data chunk magnitude
Powersupply	2	29928	24	72-168-240-720
HyperPlane	10	100000	5	500-800-1000-1500
electricity	8	45312	2	15-30-90-180
noaa	8	18159	2	15-30-90-365
SEA	3	10000	2	30-50-80-100
spiral	2	80000	2	200-400-600-800
Gaussian	2	80000	2	400-800-1200-1600
checkerboard	2	20000	2	50-100-150-200

4.2 Initial results and analysis

In the experiments, we compare the proposed AO-ELM algorithm with OS-ELM [34], FOS-ELM [24], SF-ELM [25] and EWMA-ELM [42] algorithms. For FOS-ELM algorithm, only three recent chunks are reserved. In SF-ELM, an empirical forgetting factor η =0.2 is designated. In EWMA-ELM algorithm, all parameters adopt the default parameter settings recommended in Ref.[42], and only when reset conditions are satisfied, the old ELM classifier can be abandoned,

[1] http://www.cse.fau.edu/~xqzhu/stream.html

[2] http://moa.cms.waika to.ac.nz/datasets/

[3] ftp://ftp.ncdc.noaa.gov/pub/data/gsod

then the new learner model are constructed by the recent batch of instances. Also, EWMA-ELM algorithm only run on 6 binary-class data sets. AO-ELM algorithm includes two versions, AO-ELM$_{mean}$ and AO-ELM$_{max}$ which correspond to two different ways to calculate v. In these two algorithms, the parameter λ is empirically assigned as 0.7, 0.5, 0.3, 0.1 with the increase of chunk magnitude, and it has been fixed on all data sets but not selected the optimal one on each data set in order to maintain the fairness of the compared results. For all algorithms, the parameter L is previously determined by grid search by the initial received chunk, where $L \in \{5,10,15,...,100\}$.

All algorithms are implemented in the Matlab 2013a environment, and experiments are conducted on Intel(R) Core(TM) i7 6700HQ 8 cores CPU (main frequency: 2.60GHZ for each core) and 16GB RAM.

The classification performance of each algorithm is evaluated by 0-1 loss, i.e., error rate. Considering the randomness of ELM model, the experimental results might be unstable, hence we adopt 10 times random runs to calculate the average result for each algorithm. The results are provided in the form of mean±standard deviation.

Table 2~Table 9 present respectively the 0-1 loss of various algorithms on 8 stream data sets. A result for OS-ELM, FOS-ELM, SF-ELM or EWMA-ELM has been highlighted in boldface if it is less than the minimum of the two AO-ELM approaches in the same row. A result for an AO-ELM approach has been highlighted in boldface if it is less than the minimum of the results for the other four alternatives. Moreover, to show whether the best AO-ELM algorithm performs significantly better/worse than the comparing algorithms, the pairwise *t*-test at 5% significance level has been conducted. Accordingly, a win/loss was counted and a marker ▼/▲ was shown in the tables whenever the best AO-ELM algorithm achieves significantly superior/inferior performance on one data set. Otherwise, a tie was counted and no marker was given.

Table 2. 0-1 loss on Powersupply data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
72	0.8385±0.0006(1.45×10^{-6})▼	0.8191±0.0017(1.24×10^{-3})▼	0.8188±0.0026(9.57×10^{-3})▼	-	**0.8073±0.0028**	**0.8096±0.0021**
168	0.8389±0.0006(7.99×10^{-6})▼	0.8239±0.0028(6.19×10^{-3})▼	0.8259±0.0014(9.87×10^{-4})▼	-	**0.8155±0.0016**	**0.8119±0.0026**
240	0.8393±0.0009(6.22×10^{-5})▼	0.8301±0.0020(9.63×10^{-3})▼	0.8285±0.0017(1.48×10^{-2})▼	-	**0.8227±0.0017**	**0.8183±0.0018**
720	0.8414±0.0007(8.19×10^{-2})	0.8510±0.0005(2.76×10^{-5})▼	0.8411±0.0006(6.98×10^{-2})	-	**0.8397±0.0009**	**0.8361±0.0011**

Table 3. 0-1 loss on Hyperplane data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
500	0.5190±0.0053(1.55×10^{-4})▼	0.4785±0.0037(7.72×10^{-1})	0.4874±0.0055(4.94×10^{-2})▼	-	**0.4762±0.0042**	**0.4777±0.0050**
800	0.5204±0.0062(5.28×10^{-5})▼	0.4845±0.0049(4.76×10^{-2})▼	0.4828±0.0049(2.97×10^{-1})	-	**0.4810±0.0061**	**0.4774±0.0051**
1000	0.5159±0.0039(3.18×10^{-6})▼	0.4876±0.0040(7.11×10^{-2})	0.4857±0.0046(2.34×10^{-1})	-	**0.4827±0.0073**	**0.4798±0.0040**
1500	0.5183±0.0064(7.47×10^{-5})▼	0.4915±0.0047(4.36×10^{-1})	0.4954±0.0039(2.09×10^{-2})▼	-	0.4941±0.0032	**0.4878±0.0038**

Table 4. 0-1 loss on electricity data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
15	0.2444±0.0055(2.38×10^{-6})▼	0.3187±0.0031(1.30×10^{-9})▼	0.2147±0.0028(8.44×10^{-4})▼	0.2371±0.0025(5.92×10^{-6})▼	**0.1943±0.0034**	**0.1931±0.0041**
30	0.2475±0.0058(7.33×10^{-5})▼	0.2798±0.0031(2.91×10^{-6})▼	0.2230±0.0025(9.47×10^{-3})▼	0.2346±0.0051(1.12×10^{-3})▼	**0.2107±0.0026**	**0.2113±0.0026**
90	0.2490±0.0052(1.67×10^{-2})▼	0.2440±0.0023(1.94×10^{-2})▼	0.2312±0.0030(1.97×10^{-1})	0.2290±0.0033(4.86×10^{-1})	**0.2289±0.0039**	**0.2275±0.0030**
180	0.2553±0.0036(2.11×10^{-2})▼	0.2509±0.0027(2.96×10^{-2})▼	0.2332±0.0039(9.89×10^{-1})	0.2377±0.0041(8.72×10^{-2})	**0.2330±0.0039**	0.2363±0.0034

Table 5. 0-1 loss on noaa data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
15	**0.2067±0.0059**(2.37×10^{-2})▲	0.2583±0.0034(4.95×10^{-4})▼	0.2187±0.0030(2.09×10^{-1})	0.2219±0.0036(7.18×10^{-2})	0.2162±0.0028	0.2188±0.0029
30	**0.2049±0.0030**(1.28×10^{-1})	0.2468±0.0053(8.79×10^{-3})▼	0.2091±0.0051(1.33×10^{-1})	0.2148±0.0032(5.14×10^{-2})	0.2082±0.0044	0.2116±0.0045
90	0.2078±0.0040(5.30×10^{-2})	0.2043±0.0040(1.74×10^{-1})	0.2003±0.0029(3.62×10^{-1})	0.2021±0.0043(2.55×10^{-1})	**0.1989±0.0043**	0.2006±0.0027
365	0.2078±0.0036(3.76×10^{-2})▼	0.1990±0.0051(8.46×10^{-1})	0.1995±0.0047(7.44×10^{-1})	0.1997±0.0037(5.63×10^{-1})	**0.1982±0.0036**	**0.1987±0.0045**

Table 6. 0-1 loss on SEA data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
30	0.1214±0.0010(4.95×10^{-7})▼	0.0880±0.0019(3.15×10^{-3})▼	0.0881±0.0008(1.86×10^{-3})▼	0.0981±0.0022(7.14×10^{-4})▼	**0.0783±0.0024**	**0.0775±0.0016**
50	0.1206±0.0015(5.14×10^{-7})▼	0.0803±0.0013(9.56×10^{-2})	0.0792±0.0007(7.52×10^{-2})	0.1018±0.0019(8.15×10^{-5})▼	**0.0767±0.0006**	**0.0775±0.0016**
80	0.1215±0.0011(6.11×10^{-7})▼	0.0794±0.0013(1.42×10^{-1})	0.0786±0.0019(4.91×10^{-1})	0.0897±0.0011(8.09×10^{-3})▼	**0.0778±0.0018**	0.0786±0.0016
100	0.1209±0.0017(1.66×10^{-6})▼	0.0804±0.0017(2.33×10^{-1})	0.0852±0.0018(6.51×10^{-2})	0.0942±0.0015(7.37×10^{-3})▼	**0.0783±0.0015**	**0.0780±0.0014**

Table 7. 0-1 loss on spiral data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
200	0.5314±0.0043(3.03×10^{-8})▼	0.4722±0.0041(4.12×10^{-4})▼	0.4476±0.0051(2.15×10^{-2})▼	0.4752±0.0046(5.75×10^{-4})▼	**0.4300±0.0071**	**0.4282±0.0065**
400	0.5242±0.0032(4.69×10^{-4})▼	**0.4090±0.0019**(5.55×10^{-7})▲	**0.4107±0.0019**(7.94×10^{-7})▲	0.4960±0.0028(9.08×10^{-3})▼	0.4913±0.0028	0.4851±0.0030
600	0.5280±0.0028(1.93×10^{-6})▼	**0.4173±0.0079**(2.21×10^{-2})▲	**0.4277±0.0057**(2.73×10^{-1})	**0.4179±0.0066**(2.84×10^{-2})▲	0.4322±0.0015	0.4303±0.0059
800	0.5275±0.0013(8.78×10^{-2})	**0.4154±0.0015**(6.59×10^{-8})▲	**0.4361±0.0012**(3.58×10^{-6})▲	**0.4228±0.0039**(3.76×10^{-6})▲	0.5193±0.0046	0.5158±0.0040

Table 8. 0-1 loss on Gaussian data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
400	0.1438±0.0025(4.67×10^{-7})▼	**0.0490±0.0010**(7.74×10^{-1})	0.0496±0.0005(9.07×10^{-1})	0.0872±0.0007(3.19×10^{-6})▼	0.0511±0.0015	0.0494±0.0005
800	0.1470±0.0093(3.58×10^{-6})▼	**0.0500±0.0007**(3.42×10^{-2})▲	**0.0520±0.0019**(8.96×10^{-2})	0.0819±0.0006(7.84×10^{-5})▼	0.0635±0.0004	0.0565±0.0007
1200	0.1450±0.0024(5.27×10^{-6})▼	**0.0539±0.0017**(1.44×10^{-2})▲	**0.0589±0.0004**(6.19×10^{-2})	0.0642±0.0011(2.75×10^{-1})	0.0760±0.0010	0.0632±0.0010
1600	0.1471±0.0026(1.98×10^{-4})▼	**0.0577±0.0011**(9.89×10^{-3})▲	**0.0632±0.0012**(8.66×10^{-2})	0.0759±0.0019(7.44×10^{-2})	0.0831±0.0018	0.0685±0.0020

Table 9. 0-1 loss on checkerboard data set, where the value in each bracket denotes the *p*-value of *t*-test belonging to the corresponding algorithm compared with the best AO-ELM algorithm.

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
50	0.1236±0.0006(6.27×10^{-8})▼	0.0811±0.0015(9.04×10^{-2})	0.0799±0.0013(1.23×10^{-1})	0.1099±0.0019(3.16×10^{-4})▼	**0.0785±0.0012**	**0.0776±0.0010**
100	0.1241±0.0007(1.18×10^{-7})▼	0.0780±0.0006(2.32×10^{-1})	0.0787±0.0012(2.01×10^{-1})	0.0914±0.0011(6.92×10^{-5})▼	0.0784±0.0013	**0.0772±0.0013**
150	0.1237±0.0004(2.15×10^{-9})▼	0.0806±0.0011(1.96×10^{-1})	0.0849±0.0016(7.92×10^{-2})	0.0892±0.0018(5.24×10^{-2})	0.0842±0.0009	**0.0794±0.0016**
200	0.1241±0.0006(9.35×10^{-8})▼	**0.0811±0.0012**(7.44×10^{-3})	0.0891±0.0014(1.97×10^{-1})	0.0943±0.0022(6.24×10^{-2})	0.0905±0.0010	0.0866±0.0007

The results in Table 2~Table 9 show as follows:

1) Forgetting mechanism is a valuable strategy to reduce the error rate of online learning model when there exists concept drift in data stream, as on most data sets, the five algorithms with forgetting mechanism have presented lower 0-1 loss than that of OS-ELM. The results are consistent with the conclusions in previous work [24-25,42] and our anticipation.
2) In contrast with SF-ELM, FOS-ELM is generally more dependent on the size of data chunk. That means when the data chunk is extremely small, FOS-ELM has a risk to provide significantly worse performance than several other models, e.g., the results on electricity and noaa data sets. However, if the data chunk is large enough, it always presents an approximate classification results to the others. It is not difficult to explain this phenomenon as FOS-ELM only reserves several recent data chunks to train classification model but forgets all other old knowledge, hence if the data chunk is extremely small, the number of training instances might be too few to construct an excellent enough model. We believe the problem could be adaptively solved by regulating the number of reserved data chunks.
3) One similar phenomenon has also been observed from the results of EWMA-ELM algorithm, i.e., when the size of the data chunk is extremely small, it has a risk to present the worse performance than the proposed AO-ELM algorithm, but with the increase of the data chunk size, the performance may improve gradually. We believe it is related with the reset mechanism of this algorithm. That is to say, when a large enough change is detected, EWMA only adopts the reserved instances in the window to retrain a classifier whose quality might be closely related with the number of training instances.
4) On all natural datasets and several of the synthetic ones, our proposed AO-ELM algorithms have shown an improved or at least comparable performance in comparison with the SF-ELM algorithms. That means it is an effective strategy to regulate the forgetting factor by receiving the feedback information from a concept drift map. It also indicates that it is

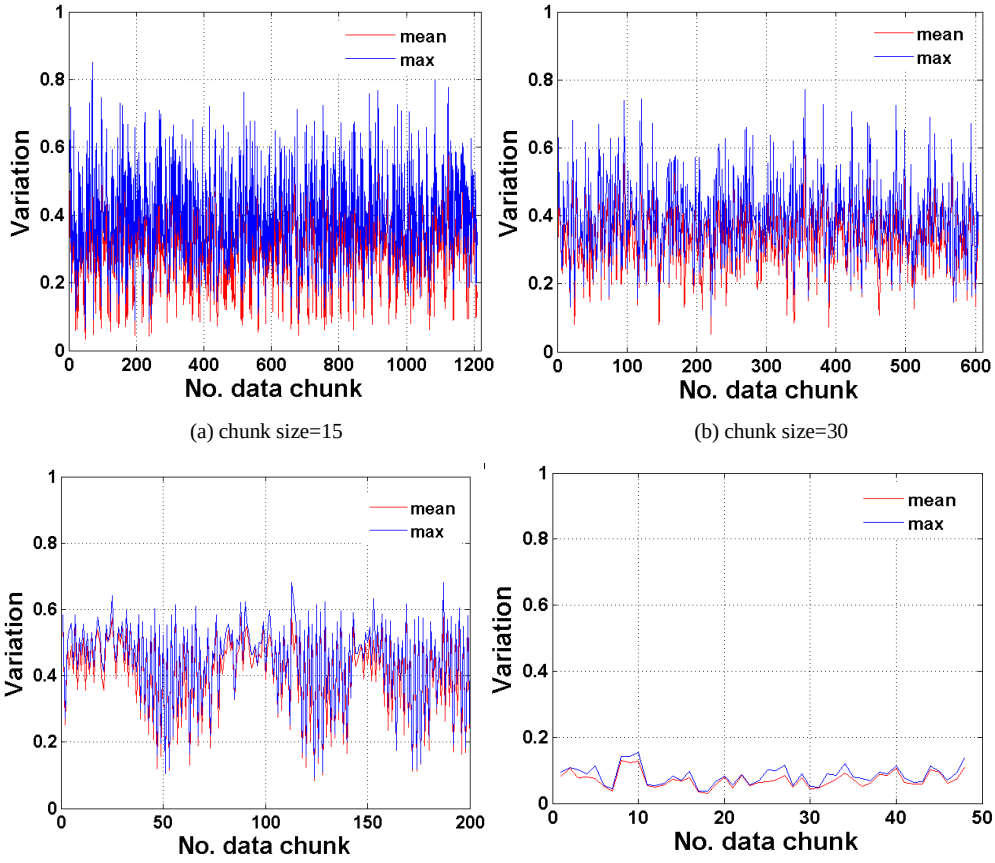
inappropriate to designate a constant forgetting factor for an online learning model as concept drift is dynamic and the drift magnitude is variable. When two consecutive data chunks have similar distributions, more old knowledge should be reserved, but if the distributions between them are extremely different, then more old experience should be removed. A variable forgetting factor regulated by a concept drift map meets this requirement.

5) In comparison with EWMA-ELM algorithm, our proposed AO-ELM algorithm can generally acquire better or similar classification performance by observing the their pairwise *t*-test results. Actually, on one half experimental results, this superiority is significant, and on the other half results, although our proposed algorithm is not statistically better, but shows similar performance with EWMA-ELM algorithm. The results also indicates that our proposed AO-ELM algorithm is generally robust.

6) For different drift types except for recurring drift, AO-ELM algorithms always has strong adaption. If the drift direction is approximately monotonous, no matter the drift speed is fast or slow, our algorithm can make the model adapt the new received instances. However, it seems not to deal with the recurring drift well, as the old concept can be only monotonously forgot. This conclusion can be safely observed from the results in noaa data set which hold both gradual and recurring drift types. On this data set, when the data chunk size is not large enough, OS-ELM algorithm which saves all old knowledge has acquired better classification accuracy than several other algorithms. Therefore, forgetting mechanism is not an excellent strategy to deal with recurring drift.

7) As for the two different algorithms proposed in this paper, they seem to have similar performance. Generally speaking, AO-ELM$_{max}$ performs a little better than AO-ELM$_{mean}$ in most cases, thus we prudently recommend users to adopt AO-ELM$_{max}$ model in their practical applications. Actually, in contrast with AO-ELM$_{mean}$, AO-ELM$_{max}$ is a more radical regulation strategy as it only considers the variation corresponding to the category that holds the maximal distribution drift.

8) Another interesting phenomenon has also been observed from the results in these tables, i.e., the results on Powersupply, Hyperplane and spiral data sets are significantly worse than that on some other data sets. We note that both Powersupply and Hyperplane data sets have multiple classes, where the Powersupply holds 24 classes, and Hyperplane owns five classes.



(a) chunk size=15 (b) chunk size=30

(c) chunk size=90 (d) chunk size=365

Fig.5 Concept drift map on noaa data set with different chunk sizes ((a):15; (b):30; (c): 90; (d): 365)

As we know, for a binary-class problem, random guess can generally produce an approximately 50% classification accuracy, while for multiple class problem, the accuracy of the random guess will constantly decrease with the addition of the number of classes. Therefore, the results on these two data sets can be accepted. As for the spiral data set, we know the gradual drift happens on each round and the drift magnitude, i.e., the variation, is always large enough [48], causing in each iteration, most old knowledge are forgotten, but the new knowledge are not enough to rebuild an excellent enough model because of the limitation of the size of a data chunk. It explains why the awful results can be produced on the spiral data set as well.

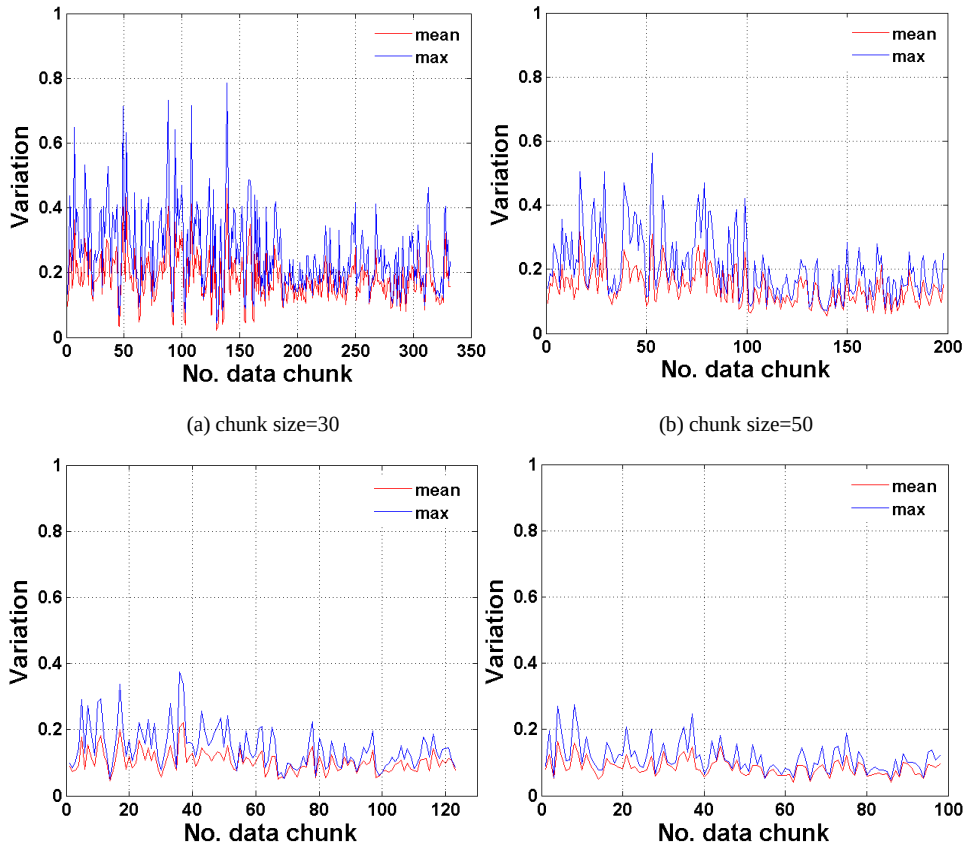
We also note an interesting phenomenon that on the noaa data set, only when the chunk size is set to be a year, those advanced algorithms with forgetting mechanism could present better classification performance than OS-ELM. To elucidate this issue, a concept drift map based on four different data chunk sizes is presented in Fig.5.

In Fig.5, we observe that when the data chunk size is 15 or 30, both max variation and mean variation vary rapidly, indicating concept drift happens frequently. This may result when the chunks do not correspond to the natural intervals in the data, as, for example, if hourly data was chunked into chunks of 20 observations, whereby midnight to 8pm of the first day would be compared to 8pm of the first day through to 4pm of the next day, and so on. When incommensurable intervals are compared it should be expected that there will be substantial drift even if such drift is not apparent if the natural chunking size is employed. Considering in noaa data sets, each example records one day's meteorological indicators, hence when the chunk size equals to 15, it means the model is updated every half a month, while if the chunk size equals to 30, it means the model updates every month. We know weather data approximates a cycling drift, where two different years might be similar, but within one year, the distribution drift may be drastic due to differences between seasons. That is to say, it is reasonable to use last year's data to predict this year, but that adopting the historical data of last month or season to predict this month or season might be misguided. Therefore, we can say for this type of data, yearly chunking might be more effective, and in general it is critical to match chunk sizes to natural cycles in the data. In real-world applications, the users are also encouraged to discreetly determine the chunk size.

4.3 Further discussions about the parameters

As we know, there is an important parameter λ, which denotes the percentage of fixedly reserved old knowledge that is not associated with distribution variation, in our proposed algorithms. Next, we investigate the association between the parameter λ and the size of data chunk. In Fig.5, we have observed a certain tendency that with the amplification of data chunk, the distribution variation would be smaller and smaller, although it is not obvious in this figure. Therefore, we show a clearer example in Fig.6 that describes the concept drift map with different data chunk magnitudes on the SEA data set.

Fig.6 shows a clearer example to present the relation between the size of data chunk and distribution variation, which verifies the observation in our initial experiment. As mentioned in Section 4, we believe the reason lies in that in small data chunk, there are not enough instances belonging to each category to accurately estimate the data distribution, i.e., the distribution estimation holds a high randomness or a large bias. Then it further means the calculation of distribution drift υ will be extremely inaccurate. Based on the analysis above, it is easy to understand why the phenomenon arises. Next, we expect to make clear how to take advantage of this to previously select an appropriate value for parameter λ. Fig.7 presents the variation of 0-1 loss of AO-ELM$_{max}$ algorithm based on different λ values and different sizes of data chunk on noaa and SEA data sets, respectively. Specifically, λ varies from 0 to 1 with an increment of 0.1.



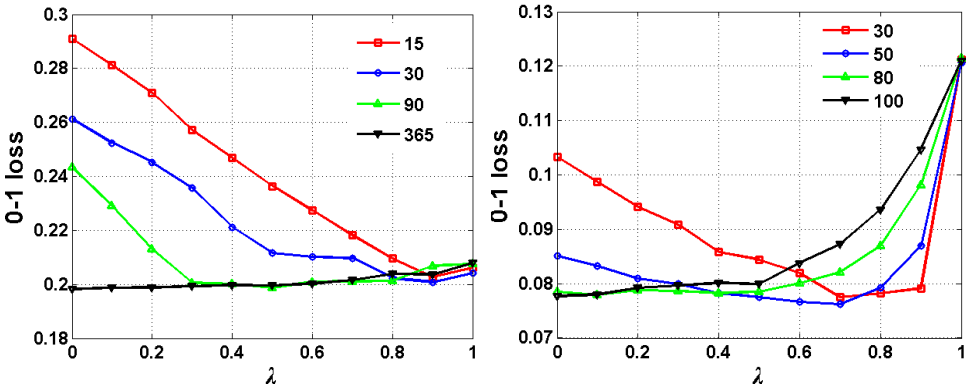
(a) chunk size=30 (b) chunk size=50

(c) chunk size=80 (d) chunk size=100

Fig.6 Concept drift map on SEA data set with different chunk sizes ((a):30; (b):50; (c): 80; (d): 100)

In Fig.7, we observe that although there are some fluctuations, it still shows a clear association between the data chunk size and the parameter λ. When the data chunk is small, a large λ should be designated to produce excellent enough classification performance, while with the

increase of chunk magnitude, a smaller and smaller λ is needed to promote the quality of the online learner. It is easy to analyze the reason of this phenomenon that for a small chunk, the estimation for distribution is excessively inaccurate, causing a high variation could be received, thus it is necessary to retain more old knowledge by assigning a large λ. While when the data chunk is large enough, the estimation for distribution would become accurate, then the variation between two data continuous chunks tends to decrease, thereby reducing the value of λ becomes the only way to promote classification performance. The rule is also useful for helping users to select an appropriate λ based to the data chunk magnitude in their practical data stream applications.



(a) noaa (b) SEA

Fig.7 Variation of 0-1 loss of AO-ELM$_{max}$ algorithm based on different λ values and different sizes of data chunk on noaa data set (sub-graph (a)) and SEA data set (sub-graph (b))

4.4 Comparison of running time

Considering the real time requirement in online learning scenario, we should focus on the time-complexity of various algorithms, too. Table 10~Table 17 present the average running time of various algorithms on eight different stream data sets, respectively.

Table 10. Running time (s) on Powersupply data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
72	0.7972	0.9578	0.8784	2.1319	5.1543	5.0981
168	0.6802	0.7894	0.7176	1.2237	2.7643	2.7550
240	0.6880	0.7114	0.7238	1.5175	2.4149	2.4913
720	0.4789	0.4945	0.5249	1.0844	1.5116	1.5428

Table 11. Running time (s) on Hyperplane data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
500	2.6395	2.7534	2.7709	4.1997	6.5411	6.2681
800	2.0857	2.2152	2.1341	3.1966	4.8485	4.8485
1000	2.0218	2.0717	2.1296	3.7928	4.5427	4.5162
1500	1.9391	2.0093	2.0297	3.0946	4.0576	4.1028

Table 12. Running time (s) on electricity data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
15	3.8283	4.7768	3.9141	9.2852	11.4630	11.2196
30	2.2698	2.6177	2.3965	5.1876	6.9623	6.7970
90	1.1450	1.2542	1.1918	3.2159	3.3509	3.3805
180	0.9485	0.9968	0.9719	2.4648	2.2792	2.3041

Table 13. Running time (s) on noaa data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
15	1.4024	1.8471	1.5819	4.9966	5.2836	5.3103
30	0.8018	0.9812	0.8408	3.5287	3.2698	3.2745
90	0.4477	0.4867	0.4618	1.3644	1.4165	1.3853
365	0.3115	0.3619	0.3370	0.6150	0.8128	0.7722

Table 14. Running time (s) on SEA data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
30	0.4306	0.4774	0.4534	0.8837	0.9984	0.9734
50	0.3058	0.3682	0.3295	0.5762	0.7176	0.7051
80	0.2434	0.2683	0.2658	0.4991	0.5803	0.5413
100	0.2184	0.2683	0.2322	0.4220	0.5850	0.5600

Table 15. Running time (s) on spiral data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
200	2.1138	2.1606	2.2293	2.5680	3.4211	3.4570
400	1.8455	2.2854	2.1981	2.7799	3.1887	3.2167
600	1.6521	1.7394	1.7956	2.2152	2.8829	2.8018
800	1.5475	1.7004	1.6739	2.1215	2.7004	2.7004

Table 16. Running time (s) on Gaussian data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
400	2.0436	2.0467	2.0748	3.5039	3.4055	3.3275
800	1.8330	1.8455	1.9269	3.1012	3.2542	3.2152
1200	1.4352	1.5350	1.6054	2.1742	3.0576	3.0389
1600	1.6661	1.4586	1.7210	2.2298	2.8486	2.8533

Table 17. Running time (s) on checkerboard data set

chunk size	OS-ELM	FOS-ELM	SF-ELM	EWMA-ELM	AO-ELM$_{mean}$	AO-ELM$_{max}$
50	0.7363	0.8611	0.7597	1.3257	1.4737	1.4815
100	0.5366	0.5741	0.5592	0.8891	1.0918	1.0418
150	0.4243	0.4774	0.4637	0.6573	0.9625	0.9438
200	0.4212	0.4165	0.4524	0.6223	1.2730	1.2152

In Table 10~Table 17, we note that on almost all data sets, OS-ELM is the fastest learning model as it has the simplest update rule. In contrast with OS-ELM, both FOS-ELM and SF-ELM models always consume a little more running time. We believe the reason lies in that FOS-ELM introduces two different operation processes during model updating, incremental and decremental, while SF-ELM adopts more complex update rule to calculate output layer weight matrix β than OS-ELM. In comparison with these three algorithms, EWMA-ELM is generally more time-consuming, as change detection will be conducted after received a chunk of new instances, and once change is detected, a new classifier will be trained, too. Two proposed models are obviously more time-consuming models, as the concept drift map needs to be drawn synchronously. In fact, the time-complexity of the proposed algorithms are associated with two key factors, including the number of attributes and the number of categories, because for each attribute on each class, the distribution needs to be estimated and the variation needs to be calculated. The results on Powersupply, HyperPlane, electricity and noaa data sets, which either hold multiple attributes or include multiple classes, provide a consistent conclusion with our analysis. We argue, although the proposed algorithms are more time-consuming, in most cases, the increase of the time consumption are still relatively limited and acceptable. In addition, we also observe that chunk magnitude has a close relationship with running time. That is to say, with the increase of chunk magnitude, the time consumption of each model would decrease monotonically. It is not difficult to understand this phenomenon as the smaller the data chunk is, more frequent update the models will be.

5. Concluding remarks

Online learning from nonstationary data streams is a challenging task in machine learning and data mining. In this scenario, concept drift might happen frequently, as well for each drift, the magnitude may differ. An excellent online learner should adapt to both drift type and magnitude by forgetting old knowledge to some extent. In this paper, we present an adaptive online learning model with forgetting mechanism in the context of extreme learning machine, which includes two main novelties as follows:

1) Based on a strong assumption, a quantitative measure of distance between two distributions

on continuous attribute space is designed, further it can be used to generate the concept drift map;

2) The association between the forgetting factor and the concept drift map is built, further it can be adopted to make the learning model adapt the variation of data distributions.

Our experiment demonstrate that the proposed algorithm is an effective and efficient solution for constructing online learning models in nonstationary environments.

Our possible direction of future research is to explore more effective and efficient strategies to quantificationally estimate the difference between two continuous distributions, to make the model adapt non-Gaussian data stream and decrease the risk brought by the strong assumption in this paper. In addition, we also wish to investigate the strategy that could automatically determine the optimal parameters.

Acknowledgements

The work was supported in part by National Natural Science Foundation of China under grants No.61305058 and No.61572242, Natural Science Foundation of Jiangsu Province of China under grant No.BK20130471, and China Postdoctoral Science Foundation under grants No.2013M540404 and No.2015T80481. This material is also based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-17-1-4033.

References

[1] G.I. Webb, R. Hyde, H. Cao, H.L. Nguyen, F. Petitjean, Characterizing concept drift, Data Mining and Knowledge Discovery, 30 (4) (2016) 964-994.

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation. ACM Computing Surveys, 46 (4) (2014) 44.

[3] T. Becker, T. Enders, A. Delgado, Dynamic neural networks as a tool for the online optimization of industrial fermentation, Bioprocess and Biosystems Engineering, 24 (6) (2002) 347-354.

[4] W. Hu, J. Gao, Y. Wang, O. Wu, S. Maybank, Online adaboost-based parameterized methods for dynamic distributed network intrusion detection, IEEE Transactions on Cybernetics, 44 (1) (2014) 66-82.

[5] W. Wei, J. Li, L. Cao, Y. Ou, J. Chen, Effective detection of sophisticated online banking fraud on extremely imbalanced data, World Wide Web, 16 (4) (2013) 449-475.

[6] L.L. Minku, X. Yao, DDD: A new ensemble approach for dealing with concept drift, IEEE Transactions on Knowledge and Data Engineering, 24 (4) (2012) 619-633.

[7] W.Y. Cheng, C.F. Juang, A fuzzy model with online incremental SVM and margin-selective gradient descent learning for classification problems, IEEE Transactions on Fuzzy Systems, 22 (2) (2014) 324-337.

[8] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, Advances in neural information processing systems, (2001) 409-415.

[9] N. Kasabov, Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 31 (6) (2001) 902-918.

[10] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, IEEE Transactions on Neural Networks, 22 (12) (2011) 1901-1914.

[11] H.M. Gomes, J.P. Barddal, F. Enembreck, A. Bifet, A Survey on Ensemble Learning for Data Stream Classification, ACM Computing Surveys, 50 (2) (2017) 23.

[12] J.Z. Kolter, M.A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, Journal of Machine Learning Research, 8 (12) (2007) 2755-2790.

[13] B. Krawczyk, L.L. Minku, J. Gama, J. Stefanowski, M. Woźniak, Ensemble learning for data stream analysis: a survey, Information Fusion, 37 (2017) 132-156.

[14] L.L. Minku, A.P. White, X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Transactions on Knowledge and Data Engineering, 22 (5) (2010) 730-742.

[15] M.D. Muhlbaier, A. Topalis, R. Polikar, Learn ++. NC: Combining Ensemble of Classifiers With Dynamically Weighted Consult-and-Vote for Efficient Incremental Learning of New Classes, IEEE Transactions on Neural Networks, 20 (1) (2009) 152-168.

[16] R. Polikar, L. Upda, S.S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Transactions on Systems, Man, and Cybernetics, part C (Applications and Reviews), 31 (4) (2001) 497-508.

[17] C. Pagano, E. Granger, R. Sabourin, G.L. Marcialis, F. Roli, Adaptive ensembles for face recognition in changing video surveillance environments, Information Sciences, 286 (2014) 75-101.

[18] R. Razavi-Far, P. Baraldi, E. Zio, Dynamic weighting ensembles for incremental learning and diagnosing new concept class faults in nuclear power systems, IEEE Transactions on Nuclear Science, 59 (5) (2012) 2520-2530.

[19] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: Proceeding of 2007 SIAM International Conference on Data Mining (SDM), 2007, pp.443-448.

[20] F. Desobry, M. Davy, C. Doncarli, An online kernel change detection algorithm, IEEE Transactions on Signal Processing, 53 (8) (2005) 2961-2974.

[21] R. Pears, S. Sakthithasan, Y.S. Koh, Detecting concept change in dynamic data streams, Machine Learning, 97 (3) (2014) 259-293.

[22] L.I. Kuncheva, Change detection in streaming multivariate data using likelihood detectors, IEEE Transactions on Knowledge and Data Engineering, 25 (5) (2013) 1175-1180.

[23] A.A. Qahtan, B. Alharbi, S. Wang, X. Zhang, A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp.935-944.

[24] J. Zhao, Z. Wang, D.S. Park, Online sequential extreme learning machine with forgetting mechanism, Neurocomputing, 87 (2012) 79-89.

[25] X. Zhang, H.L. Wang, Selective forgetting extreme learning machine and its application to time series prediction, Acta Physica Sinica, 60 (8) (2011) 013.

[26] Y. Jiang, Q. Zhao, Y. Lu, Ensemble based data stream mining with recalling and forgetting mechanisms, in: Proceedings of the 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), IEEE, 2014, pp.430-435.

[27] M. Zimmermann, E. Ntoutsi, M. Spiliopoulou, Adaptive semi supervised opinion classifier with forgetting mechanism, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp.

805-812.

[28] G.I. Webb, L.K. Lee, F. Petitjean, B. Goethals, Understanding Concept Drift, 2017, arXiv preprint arXiv:1704.00362.

[29] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing, 70 (1) (2006) 489-501.

[30] G.B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42 (2) (2012) 513-529.

[31] G.B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, International journal of machine learning and cybernetics, 2 (2) (2011) 107-122.

[32] G. Huang, G.B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Networks, 61 (2015) 32-48.

[33] D.R. Williams, G. Hinton, Learning representations by back-propagating errors. Nature, 323 (6088) (1986) 533-538.

[34] N.Y. Liang, G.B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks, 17 (6) (2006) 1411-1423.

[35] R. Singh, H. Kumar, R.K. Singla, An intrusion detection system using network traffic profiling and online sequential extreme learning machine, Expert Systems with Applications, 42 (22) (2015) 8609-8624.

[36] Y. Lan, Y.C. Soh, G.B. Huang, Ensemble of online sequential extreme learning machine, Neurocomputing, 72 (13) (2009) 3391-3395.

[37] S. Xu, J. Wang, Dynamic extreme learning machine for data stream classification, Neurocomputing, 238 (2017) 433-449.

[38] B. Mirza, Z. Lin, Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification, Neural Networks, 80 (2016) 79-94.

[39] B. Mirza, Z. Lin, N. Liu, Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift, Neurocomputing, 149 (2015) 316-329.

[40] J. Sherman, W.J. Morrison, Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix, Annals of Mathematical Statistics, 21 (1) (1950)124-127.

[41] G.Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning, 23(1) (1996), 69-101.

[42] G. J. Ross, N. M. Adams, D. K. Tasoulis, D. J. Hand, Exponentially weighted moving average charts for

detecting concept drift, Pattern Recognition Letters, 33(2012) 191-198.

[43] D.A. Levin, Y. Peres, E.L. Wilmer, Markov chains and mixing times. American Mathematical Soc., 2009.

[44] E. Singer, D. A. Reynolds, Domain mismatch compensation for speaker recognition using a library of whiteners, IEEE Signal Processing Letters, 22(2015) 2000-2003.

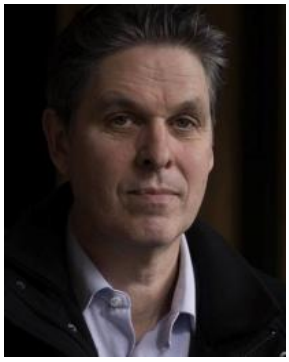
[45] D.L. Ruderman, T.W. Cronin, C.C. Chiao, Statistics of cone responses to natural images: implications for visual coding, Journal of the Optical Society of America-Part A, 15(1998) 2036-2045.

[46] S. S. Kelly, The effect of age on neuromuscular transmission, Journal of Physiology, 274(1) (1978) 51-62.

[47] Y. Wang, W. Fu, D. P. Agrawal, Gaussian versus uniform distribution for intrusion detection in wireless sensor networks, IEEE Transactions on Parallel and Distributed systems, 24(2) (2013) 342-355.

[48] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Transactions on Knowledge and Data Engineering, 25 (10) (2013) 2283-2301.

Hualong Yu received the B.S. degree in computer science from Heilongjiang University, Harbin, China, in 2005, and received M.S. and Ph.D. degrees in computer science from Harbin Engineering University, Harbin, China, in 2008 and 2010, respectively. Since 2010, he has been an Associate Professor in School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China. From 2013 to 2017, he was a Post-Doctoral Fellow in School of Automation, Southeast University, Nanjing, China. Since 2017, he has been a senior visiting scholar in Faculty of Information Technology, Monash University, Melbourne, Australia. He has authored or co-authored more than 60 international journal and conference papers, and 4 monographs. His research interests include machine learning and data mining. Dr. Yu is an active reviewer for more than 20 high-quality international journals, including *IEEE TNNLS, TCYB, TKDE, TCBB, Knowledge-Based Systems and Neurocomputing* etc., and the member in the organizing committee of several international conferences. He is also the member of ACM, China Computer Federation (CCF), Youth Committee of the Chinese Association of Automation (CAA) and Jiangsu Artificial Intelligent Federation (JSAI).



Geoffrey I. Webb is Director of the Monash University Center for Data Science. He was editor in chief of Data Mining and Knowledge Discovery from 2005 to 2014. He has been Program Committee Chair of both ACM SIGKDD and IEEE ICDM, as well as General Chair of ICDM. He is a Technical Advisor to BigML Inc, who incorporate his best of class association discovery software, Magnum Opus, into their cloud based Machine Learning service. He developed many of the key mechanisms of support-confidence association discovery in the 1980s. His OPUS search algorithm remains the state-of-the-art in rule search. He pioneered multiple research areas as diverse as black-box user modeling, interactive data analytics and statistically-sound pattern discovery. He has developed many useful machine learning algorithms that are widely deployed. He received the 2013 IEEE Outstanding Service Award, a 2014 Australian Research Council Discovery Outstanding Researcher Award and is an IEEE Fellow since 2015.