Cogn Comput (2015) 7:150–160
DOI 10.1007/s12559-014-9279-7

Classification of Uncertain Data Streams Based on Extreme Learning Machine

Keyan Cao · Guoren Wang · Donghong Han ·
Jingwei Ning · Xin Zhang

Received: 22 August 2013 / Accepted: 24 May 2014 / Published online: 11 June 2014

Abstract Classification over data streams is an important task in data mining. The challenges become even larger when uncertain data are considered. An important challenge in the classification of uncertain data streams is concept drift and uncertainty of data. This paper studies the problem using extreme learning machine (ELM). We first propose weighted ensemble classifier based on ELM (WEC-ELM) algorithm, which can dynamically adjust classifier and the weight of training uncertain data to solve the problem of concept drift. Furthermore, an uncertainty classifier based on ELM (UC-ELM) algorithm is designed for the classification of uncertain data streams, which not only considers tuple value, but also its uncertainty, improving the efficiency and accuracy. Finally, the performance of our methods is verified through a large number of simulation experiments. The experimental results show that our methods are effective ways to solve the problem of classification of uncertain data streams and are able to solve the problem of concept drift, reduce the execution time and improve the efficiency.

Keywords Extreme learning machine · Classification · Uncertain data streams · Possible worlds · Single-hidden-layer feedforward neural networks · Weighted

Introduction

Along with the generation of uncertain data streams, more and more attention is paid to the mining of uncertain data streams. This need is due to the inherent uncertainties of data in many real-world applications, such as sensor network monitoring [1, 2] and moving object detection [3–5]. Since the intrinsic differences between uncertain and deterministic data streams, existing data mining algorithms on deterministic data streams cannot be applied to uncertain data streams. The mining tasks over uncertain data streams are much more complicated than mining task on deterministic data streams, because the possible world instances derived from the uncertain data stream expand exponentially.

Classification is considered as an important cognitive computation [6–10] task, which has drawn much attention. The objective is the creation of an target function, which maps each property set x to a predefined class label y. Due to inherent uncertainty and streams properties, the important challenge that classification over data streams mining is the uncertainty of data [11] and concept drift [12–16]. While the input data are uncertain, each uncertain tuple consists of several instances. These special properties of the uncertain data need to be considered in the process of classification, further complicating the classification process. Concept drift is the change of the data distribution, forcing changes in the classification model reflecting the drift. If the change in data distribution is ignored, inevitably, the accuracy of the classifier trained for the previous data will greatly decline. Therefore, the data streams classification algorithm research is focused on testing the new class and timely updating the classification model to adapt the new data distribution. Concept drift rate be loosely classified into two categories, namely emergent concept drift and gradual concept drift [17], in this paper, we only study gradual concept drift.

Motivation 1 (sensor data): Sensor networks are frequently used to monitor the surrounding environment. The

K. Cao (✉) · G. Wang · D. Han · J. Ning · X. Zhang
Northeastern University, Shenyang, Liaoning, China
e-mail: caokeyan@gmail.com

monitoring range is covered by a number of individual sensors, each reporting its measurements to a central location. In the case of environmental monitoring sensor net, measurements may include temperature, humidity and air pressure. The true measured values cannot be accurately obtained due to limitations of the measuring equipment. Instead, sensor readings are send in order to approximate the true value, and a confidence in the measurement is often estimated based on its failure rate, leading to uncertain tuples, where multiple readings collected from a sensor at this location are instances of this uncertain tuple. An example of concept drift in this context includes the change of temperature due to the changes of the seasons.

Motivation 2 (stock data): Stock data streams come every day, which requires analyzing the data online with a timely manner. However, in order to protect the users' privacy, the user's data need to be blurred. Uncertain data streams are classified online, results are feeedback in time.

In the field of uncertain data mining, the presence of uncertainty can significantly affect the results of data mining [11]. Since the input data are uncertain, we should consider both the value and uncertainty of each uncertain tuple in the classification process. To the best of our knowledge, there is no existing method taking the uncertainty into account in the classification process for data mining in uncertain data streams.

Traditional classification algorithms are unable to deal with such challenges. In this paper, we investigate classification based on extreme learning machine (ELM) [18–28] of uncertain data streams. Our main contributions are:

- We propose the weighted ensemble classifier based on ELM (WEC-ELM) algorithm, which can dynamically adjust the classifier and the weight of training data to solve the problem of concept drift.
- Based on ELM and the uncertainty calculation method, we propose a two phase classification algorithm (UC-ELM) of uncertain data streams, which is able to balance the value and uncertainty of tuples, thus improving the effectiveness.
- Because there is no real uncertain data set available, the data set we used in the experiments is synthesized from real data sets. We implement a series of experimental evaluations, showing that our methods are both effective and efficient.

The rest of the paper is organized as follows. We first introduce the ELM in "Brief of extreme learning machine" section. After that, "Problem definition" section defines our problem formally. We analyze the challenge of classification over uncertain data streams and develop the two algorithms, respectively, in "Classification algorithm" section. "Performance verification" section presents an extensive empirical study. In "Conclusions" section, we conclude this paper with directions for future work.

Brief of Extreme Learning Machine

In this section, we present a brief overview of extreme learning machine (ELM), developed by Huang et al. [18, 28–31]. ELM is based on a generalized single-hidden-layer feedforward network (SLFN). In ELM, the hidden-layer node parameters are mathematically calculated instead of being iteratively tuned, providing good generalization performance at thousands of times higher speeds than traditional popular learning algorithms for feedforward neural networks [32]. The output function of *SLFNs* with L hidden nodes can be represented by

$$f_L(x) = \sum_{i=1}^{L} \beta_i g_i(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), \quad x \in R^d, \ \beta_i \in R^m \tag{1}$$

where g_i denotes the output function $G(a_i, b_i, x)$ of the ith hidden node. For additive nodes with activation function g, g_i is defined as

$$g_i = G(a_i, b_i, x) = g(a_i \cdot x + b_i), \quad a_i \in R^d, \ b_i \in R \tag{2}$$

For radial basis function (RBF) nodes with activation function g, g_i is defined as

$$g_i = G(a_i, b_i, x) = g(b_i \|x - a_i\|), \quad a_i \in R^d, \ b_i \in R^+ \tag{3}$$

The above equation can be written compactly as:

$$H\beta = T \tag{4}$$

where

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix} = \begin{bmatrix} G(a_1, b_1, x_1) \cdots G(a_L, b_L, x_1) \\ \vdots \\ G(a_1, b_1, x_N) \cdots G(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{5}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathrm{T} = \begin{bmatrix} \mathrm{t}_1^\mathrm{T} \\ \vdots \\ \mathrm{t}_\mathrm{N}^\mathrm{T} \end{bmatrix}_{\mathrm{N} \times \mathrm{m}} \tag{6}$$

H is called the hidden-layer output matrix of the *SLFN* [20, 32, 33]; the ith column of H is the ith hidden node output with respect to input $x_1, x_2, \ldots, x_N \ldots$, $h(x) = G(a_1, b_1, x), \ldots, g(a_L, b_L, x)$ is called the hidden-layer feature mapping. The ith row of H is the hidden-layer feature mapping with respect to the ith, input $x_i : h(x_i)$. It has been proved [31, 32] that from the interpolation capability point of view, if the activation function g is infinitely differentiable in any interval, the hidden-layer parameters can be randomly generated.

For the multiclass classifier with single output, ELM can approximate any target continuous function, and the output of the ELM classifier $h(x)\beta$ can be as close as possible to the class labels in the corresponding regions [34]. The classification problem for the proposed constrained-optimization-based ELM with a single-output node can be formulated as

$$\begin{aligned}&\text{Minimize} : \mathrm{L}_{\mathrm{PELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \\ &\text{Subject to} : \mathrm{h(x_i)}\beta = t_i - \xi_i, \quad i = 1,\ldots,N\end{aligned} \tag{7}$$

where C is user-specified parameter. Based on the KKT theorem [35], training ELM is equivalent to solving the following dual optimization problem:

$$L_{DELM} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i(h(x_i)\beta - t_i + \xi_i) \tag{8}$$

where each lagrange multiplier α_i corresponds to the ith training sample. The KKT optimality conditions given in [35] are as follows:

$$\frac{\partial L_{Delm}}{\alpha\beta} = 0 \longrightarrow \beta = \sum_{i=1}^{N}\alpha_i h(x_i)^T = H^T\alpha \tag{9}$$

$$\frac{\partial L_{Delm}}{\alpha\beta} = 0 \longrightarrow \alpha_i = C\xi_i, \ i = 1,\ldots,N \tag{10}$$

$$\frac{\partial L_{Delm}}{\alpha\beta} = 0 \longrightarrow h(x_i)\beta - t_i + \xi_i = 0, \ i = 1,\ldots,N \tag{11}$$

where $\alpha = [\alpha_1,\ldots,\alpha_N]$.

For a multiclass classifier, ELM requires multioutput nodes instead of a single-output node. Under these conditions, the classification problem can be formulated as

$$\begin{aligned}&\text{Minimize} : \mathrm{L}_{\mathrm{P_{ELM}}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \\ &\text{Subject to} : \mathrm{h(x_i)}\beta = t_i^T - \xi_i^T, \quad i = 1,\ldots,N\end{aligned} \tag{12}$$

where $\xi = [\xi_{i,1},\ldots,\xi_{i,m}]^T$ is the training error vector of the m output modes with respect to the training sample x_i. Similar as above, training ELM is equivalent to solving the following dual optimization problem:

$$\begin{aligned}L_{DELM} = {}& \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \\ & - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}(h(x_i)\beta_j - t_{i,j} + \xi_{i,j})\end{aligned} \tag{13}$$

where β_j is the vector of the weights linking hidden layer to the j-th output node and $\beta = [\beta_1,\ldots,\beta_m]$.

There are many variants of ELM, such as Circular-ELM [36], which augments the standard ELM paradigm by introducing a structural enhancement derived from the CBP network. Circular-ELM proves effective in addressing the visual quality assessment problem, in which the input vector of a conventional multilayer perceptron (MLP) is augmented by one additional dimension, the circular input. Circular-ELM handles the actual mapping of visual signals into quality scores, successfully reproducing perceptual mechanisms. [37] proposed a novel model for ELMs, which exploits random projection (RP) techniques. A hidden layer performs an explicit mapping of the input space to a feature space; the mapping is not subject to any optimization, since all the parameters in the hidden nodes are set randomly. The output layer includes the only degrees of freedom, i.e., the weights of the links that connect hidden neurons to output neurons. The variants of ELM mentioned above all work on determine data.

[38] proposed classification algorithms based on ELM to conduct classification over uncertain data. Using a bound-based approach, they implemented the binary and multiclass classification over uncertain data objects. They also extended the proposed algorithms to classification over uncertain data in a distributed environment based on OS-ELM and Monte Carlo theory. However, this algorithm is suitable for static data. This means that the algorithm don't have the ability to incremental updating. If there are changes in the underlying data objects, the algorithm must be executed from scratch, leading to performance degradation when operated on uncertain data streams.

Problem Definition

Uncertain Data Streams Model We assume that the uncertain data streams consist of a set of n d-dimensional uncertain tuples $x_1, x_2, \cdots, x_i, \cdots, x_n$, arriving with time stamp $T_1, T_2, \cdots, T_i, \cdots, T_n$, for any $i < n$, $T_i < T_n$. Each uncertain tuple has m possible instances and existence probabilities i.e. $x_i = \{(x_i^1, p_{x_i^1}), (x_i^2, p_{x_i^2}), \cdots, (x_i^j, p_{x_i^j}), \cdots, (x_i^m, p_{x_i^m})\}$, $\sum_{j=1}^{m} p_{x_i^j} = 1$, all instances are mutually independent. Each instance of uncertain tuple x_i is a

Table 1 Example of partial uncertain data tuple

Uncertain tuple	# Instance	# Probability
x_1	$x_1^1 = (6,6),\ x_1^2 = (7,6), x_1^3 = (6,7)$	$p_{x_1^1} = 0.6,\ p_{x_1^2} = 0.2,\ p_{x_1^3} = 0.2$
x_2	$x_2^1 = (6,2),\ x_2^2 = (8,1), x_2^3 = (9,3)$	$p_{x_2^1} = 0.5,\ p_{x_2^2} = 0.3,\ p_{x_2^3} = 0.2$
x_3	$x_3^1 = (2,3),\ x_3^2 = (2,2), x_3^2 = (3,2)$	$p_{x_3^1} = 0.2,\ p_{x_3^2} = 0.7,\ p_{x_3^3} = 0.1$
x_4	$x_4^1 = (3,8),\ x_4^2 = (6,8), x_4^3 = (4,7)$	$p_{x_4^1} = 0.6,\ p_{x_4^2} = 0.1,\ p_{x_4^3} = 0.3$

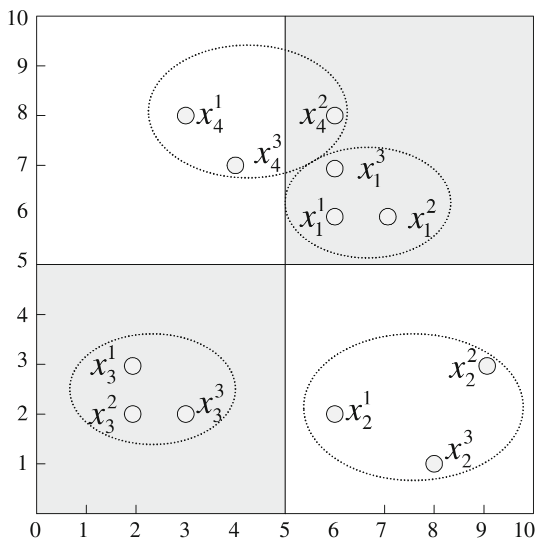

Fig. 1 Mapping uncertain tuples to a 2D-space

d-dimensional vector. Thus each instance can be be viewed as a multi-dimension data that is input into ELM.

As an example, Table 1 shows partial uncertain tuples in streams, x_1, x_2, x_3 and x_4, the instances of each uncertain tuple are shown in column instances, and the probabilities of instances are shown in column probability. Figure 1 maps these uncertain tuples into a 2D coordinate. For simplicity, we only consider two attributes in the example. Let the shaded area and the white area represent two classes, respectively, and each of the instances of uncertain tuple is to be classified into the corresponding category. When all instances of the same uncertain tuple are classified into the same class, then the uncertain tuple should belong to that particular class, as x_1, x_2, x_3. However, when instances in the same uncertain tuple are classified into different classes, then we will consider probability of each instance, as x_4. For an uncertain dataset, we need to consider all instances to determine which class the uncertain tuple belongs to.

Concept drift: Comparing with the old dataset, in the new dataset, the related features are changed, which will lead to the concept of the characteristics to be different. Specially, the concept drift discussed in this paper refers that the classification disappears, new classification emerges, or the old classification shifts too much.

Binary Classification of ELM

For example in Fig. 1, instances in all uncertain tuples are learned using ELM to obtain binary classes.

Theorem 1 *Let instance x_i^j denote the j-th instance of uncertain tuple x_i, $p_{x_i^j}$ is probability associated with x_i^j. Class set consists of two classes n and m. If $p_{x_i^j} \geq 0.5$ and x_i^j belongs class n, then tuple x_i is in class n, otherwise, x_i belongs class m.*

For binary classification case, ELM needs only one output node ($m = 1$), and the decision function of ELM classifier is:

$$f(x) = sign\left(\sum_{j=1}^{L} \beta_i G(a_i, b_i, x)\right) = sign(\beta \cdot h(x))$$

For an uncertain data tuple, its instances are classified into different classes, i.e., in tuple x_4. Let c_s and c_w be class labels of the shaded area and the white area, respectively. x_4^1 and x_4^3 are in class c_w, x_4^2 is in class c_s. The membership of class will be determined by the class, which has a high probability. To simplify this computing, we construct a function as follows:

$$f(x_i) = \begin{cases} 1, p_{x_i^j} \cdot sign(\beta \cdot h(x)) \geq 0.5 \\ 0, p_{x_i^j} \cdot sign(\beta \cdot h(x)) < 0.5 \end{cases}$$

Multiclass Classification of ELM

In this subsection, we introduce multiclass classification of ELM, the decision function is:

$$f(x) = sign\left(h(x)H^T\left(\frac{1}{C} + HH^T\right)^{-1} T\right) \tag{14}$$

The expected output vector is:

$$T = \begin{bmatrix} t_1 \\ \vdots \\ t_L \end{bmatrix}$$

For multiclass cases, m is the number of classes and output nodes of hidden layer. The predicted class label of a given testing sample is the index number of the output node, which has the highest output value for the given testing sample. Let $f_k(x_i^j)$ denote the output function of the k-th $(1 \leq k \leq m)$ output node, i.e., $f(x_i^j) = [f_1(x_i^j), \ldots, f_m(x_i^j)]$, then the predicted class label of sample x_i^j is:

$$label(x_i^j) = arg \max_{\tau \in (1,\ldots,m)} f_\tau(x_i^j)$$

For multiclass classifier with multioutputs, it is obviously that the Theorem 1 cannot be applied to uncertain multiclass classification. For example, we assume that uncertain tuples are classified into three categories. If all probabilities in three classes are smaller than 0.5, Theorem 1 binary classification cannot be used for multiclass classification. Therefore, we revise Theorem 1 as follows:

Definition 1 (*Probability of a classification instance*) Let x_i^j denote the j-th instance of uncertain tuple x_i, $p_{x_i^j}$ is probability associated with x_i^j. Given a number of classes L, ELM can classify the instance x_i^j into L classes. If the instance x_i^j belongs to the class c_l, $(1 \leq l \leq L)$, $p_{x_i^j}^{c_l}$ is probability of instance x_i^j belonging to the class c_l, then $p_{x_i^j}^{c_l} = p_{x_i^j}$.

Definition 2 (*Probability of classification uncertain tuple*) Let S_{x_i} be the set of instances of uncertain tuple x_i, each uncertain tuple has m possible instances and existence probabilities, i.e., $x_i = \{(x_i^1, p_{x_i^1}), (x_i^2, p_{x_i^2}), \ldots, (x_i^j, p_{x_i^j}), \ldots, (x_i^m, p_{x_i^m})\}$, $\sum_{j=1}^m p_{x_i^j} = 1$, all instances are mutually independent. Let $p_{x_i}^{c_l}$ be the probability of uncertain tuple x_i belonging to the class c_l, $p_{x_i}^{c_l}$ is sum of probabilities the instances which belong to class c_l, then

$$p_{x_i}^{c_l} = \sum_{x_i^j \in S_{x_i}} p_{x_i^j}^{c_l}$$

Definition 3 (*Classification of uncertain data*) Let S_c be the set of L classes, c_l be the class label, $(1 \leq l \leq L)$. x_i is uncertain tuple. If $p_{x_i}^{c_l} \geq \forall p_{x_i}^{c_j} (1 \leq j \leq L, j \neq l)$, then uncertain tuple x_i belongs class c_l.

For example, in Table 1, according to Definition 2 and Definition 3, let c_s and c_w be class labels of the shaded area and the white area, respectively. $p_{x_4}^{c_s} = p_{x_4^2}^{c_s} = p_{x_4^2} = 0.1$, and $p_{x_4}^{c_w} = p_{x_4^1}^{c_w} + p_{x_4^3}^{c_w} = p_{x_4^1} + p_{x_4^3} = 0.6 + 0.3 = 0.9$, then x_4 belongs to class c_w.

Theorem 2 *Let x_i be an uncertain tuple, which consist of m instances and existence probabilities, i.e., $x_i = \{(x_i^1, p_{x_i^1}), (x_i^2, p_{x_i^2}), \ldots, (x_i^j, p_{x_i^j}), \ldots, (x_i^m, p_{x_i^m})\}$, $\sum_{j=1}^m p_{x_i^j} = 1$, and all instances are mutually independent. Let $p_{x_i}^{c_l}$ be the probability of uncertain tuple x_i belongs to the class c_l. If $p_{x_i}^{c_l} \geq 0.5$, then uncertain tuple x_i belonging to the class c_l.*

According to the Theorem 2, when a calculated classification probability for a certain class exceeds 0.5, probabilities for the other classes do not need to be calculated. Thus, unnecessary calculations are then avoided, thereby enhancing efficiency. Based on the definition of classification of data stemming from uncertain data, we define the problem as follows:

Problem: Given uncertain data streams containing training data and testing data, output the class label for each uncertain data tuple while considering the risk of concept drift.

Classification Algorithm

In this section, we first introduce the weighted ensemble classifier based on ELM (WEC-ELM) algorithm, which is suitable for application in uncertain data and in the presence of concept drift. Secondly, we propose the uncertainty classifier based on ELM (UC-ELM) algorithm, which not only considers the value of the tuple, but also consider the uncertainty of the tuple.

Ensemble Classifier Based on ELM

In this section, a weighted ensemble classifier for uncertain data streams based on the ELM (WEC-ELM) algorithm is proposed. It consists of a number of component classifiers. The ordinary methods do not set the weights of training data; this section briefly describes the differences between existing methods. We use the ensemble classifier to distinguish the weighted training data, some tuples are classified correctly, and some are classified wrongly. The tuples that are classified incorrectly may represent the direction of the concept drift. These tuples need to be distinguished when the new classifier is created.

While concept drift occurs, a new component classifier needs to be trained and added to the ensemble classifier. If the number of component classifiers reached a preset maximum, the worst-performing component classifier is removed from the ensemble classifier, and a new component classifier is created by as trained by the new tuple. The standards to judge the classifier performance are accuracy of the previously classified data.

We first introduce the definition of the weight of a classifier and the weight of an instance which employs the idea of [39] as follows:

Definition 4 (*Error rate*) Let x_i^j be instance, C_n be the classifier label, $Err_{C_n}^{x_i^j}$ is error rate of C_n. If the instance x_i^j is classified incorrectly by classifier C_n then $Err_{C_n}^{x_i^j} = 1$, otherwise, $Err_{C_n}^{x_i^j} = 0$.

Definition 5 (*Error of a classifier*) Let $W_{x_i^j}$ be the weight of uncertain instance x_i^j, λ be the number of instances of current data chunk, E_{C_n} be the error of classifier C_n, then

$$E_{C_n} = \frac{\sum_{i=1}^{\lambda} W_{x_i^j} \times Err_{C_n}^{x_i^j}}{\sum_{i=1}^{\lambda} W_{x_i^j}}$$

Before the instance is classified, the weight of instance $W_{x_i^j}$ is 1, so

$$E_{C_n} = \frac{\sum_{i=1}^{\lambda} Err_{C_n}^{x_i^j}}{\lambda}$$

The number of instances which are classified incorrectly by classifier C_n is larger; the error of classifier C_n is larger. Otherwise, if the classifier C_n classify all instances correctly, then the error of classifier C_n is 0. If the classifier C_n classify all instances incorrectly, then the error of classifier C_n is 1, i.e., $E_{C_n} = 1$. The weight of each classifier is inversely proportional to the Error of classifier:

Definition 6 (*Weight of classifier*) Let W_{C_n} be the weight of classifier C_n, MSE_r be random prediction mean square error of classifier and E_{C_n} be the error of classifier C_n, then

$$W_{C_n} = MSE_r - E_{C_n}$$

where $MSE_r = \sum_{C_n} P(C_n)(1 - P(C_n))^2$, $P(C_n)$ is the proportion of each class in the data distribution. It follows that $P(C_n) \in (0, 1)$. The idea of fuzzy math is imported to calculate the fuzzy rate of instance classification:

Definition 7 (*Fuzzy instance classification*) Let W_{C_n} be the weight of classifier C_n, N be number of classifiers and $f_{x_i^j}$ be the fuzzy classification, then

$$f_{x_i^j} = 1 - \frac{\sum_{n=1}^{N} W_{C_n} \times Err_{C_n}^{x_i^j}}{\sum_{n=1}^{N} W_{C_n}}$$

We update the weight of instance based on the classified result.

Definition 8 (*Weight of an instance*) Let W_{C_n} be the weight of classifier and $C_n, f_{x_i^j}$ be the fuzzy classification, then

$$W_{x_i^j} = W_{x_i^j} \times (1 - f_{x_x^j}) = W_{x_i^j} \times \frac{\sum_{n=1}^{N} W_{C_n} \times Err_{C_n}^{x_i^j}}{\sum_{n=1}^{N} W_{C_n}}$$

For each instance, the weight will be updated according to the fuzzy rate for correctly classified. The weight of the instance which is classified incorrectly is small, and the weight of instance which is classified wrongly is larger. We should pay attention to the instances which have larger weight, they may represent the direction of concept drift. When we need to retrain the classifier, the instances which have larger weight should be preferred. Figure 2 shows the framework of WEC-ELM, and the update is as follows:

Initialize: We assume that the data streams arrive within chunks, the number of tuples in each chunk is λ. The first N chunks are selected to train N component classifiers based on ELM which form the ensemble classifier together.

Update: When the uncertain data chunk S_{N+1} arrives, the weight of each instance before being classified is 1. Each instance in data chunk S_{N+1} is classified using N existing component classifiers; the error rate E_{C_n} is computed according to the Definition 5. The maximum error rate is the maximum value of error rate of N Component Classifiers. If the maximum error rate is larger than a user-specified threshold E, it means that the component classifier whose error rate is the maximum error rate which is not suitable for the current data, in other words, the concept drift may occur. The component classifier with the largest error rate will be deleted, and a new component classifier is created to take its place. The new weights of N classifiers W'_{C_n} $(1 \leq n \leq N)$ are calculated according to the Definition 6. Next, the weight of each instance is updated according to the Definition 8. In order to get a better classifier, we use the instances in set S'_{N+1} to train the new classifier. The set S'_{N+1} consists of those instances whose weights are larger than a user-specified threshold W_{instance}. The new component classifier instead of being deleted is added to the integrated classifier, ready for classifying the next chunk uncertain data.

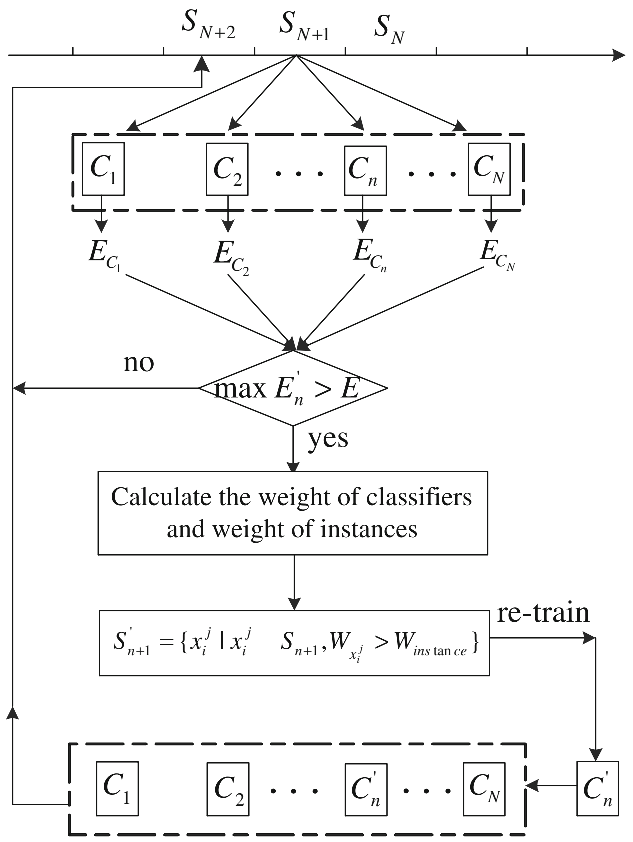
S_{N+2}
S_{N+1}
S_N
C_1
C_2
C_n
C_N
E_{C_1}
E_{C_2}
E_{C_n}
E_{C_N}
no
$\max E_n' > E$
yes
Calculate the weight of classifiers and weight of instances
$S'_{n+1} = \{x_i^j \mid x_i^j \quad S_{n+1}, W_{x_i^j} > W_{instance}\}$
re-train
C_n'

Fig. 2 Ensemble classifier framework

Uncertainty Classifier Based on ELM

In the field of uncertain data mining, tuple values are not the only issue to be considered. The presence of uncertainty can significantly affect the results of mining process [11]. The uncertainty classifier based on ELM (UC-ELM) takes both the value and the uncertainty of the tuple into account. Our method employs a two phase classification approach. First of all, according to the value of uncertain tuple, the K-classes set is composed, and next, it is decided which class absorbs the new tuple based on the uncertainty of the tuple. The definitions of instance uncertainty and tuple uncertainty [11] are as follows:

Definition 9 (*Instance uncertainty*) We assume that the uncertain tuple x_i consists of some possible instances; each possible instance consists of d-dimension value and probability of existence, i.e., $x_i = \{(x_i^1, p_{x_i^1}), (x_i^2, p_{x_i^2}), \ldots, (x_i^j, p_{x_i^j}), \ldots, (x_i^m, p_{x_i^m})\}$, where $\sum_{j=1}^m p_{x_i^j} = 1$, all instances are mutually independent. The instance uncertainty is defined are as follows:

$$U(x_i^j) = \log_2\left(\frac{1}{p_{x_i^j}}\right) = -\log_2(p_{x_i^j})$$

Definition 10 (*Tuple uncertainty*) We assume that the uncertain tuple x_i has m possible instances and existence probabilities, where $x_i = \{(x_i^1, p_{x_i^1}), (x_i^2, p_{x_i^2}), \ldots, (x_i^j, p_{x_i^j}), \ldots, (x_i^m, p_{x_i^m})\}$, and $\sum_{j=1}^m p_{x_i^j} = 1$, the uncertainty of tuple x_i is the expected value of its possible instance, denoted as follows:

$$U(x_i) = -\sum_{i=1}^{m} p_{x_i^j} \times \log_2(p_{x_i^j})$$

The uncertainty of a class is determined by the uncertainties of the tuples belonging to that the class:

Definition 11 (*Class uncertainty*) We assume that the class c_n consists of μ uncertain tuples, the uncertainty of class c_n is the average value of the uncertainty of each tuple in the class c_n, defined as follows:

$$U(c_n) = \frac{\sum_{i=1}^{\mu} U(x_i)}{\mu}$$

In order to avoid unnecessary computation, we will not compare all classes to decide which class will absorb the new tuple. Instead, we proposed a two phase method to handle this problem, choosing the optimal cluster.

Definition 12 (*K-classes*) We assume that there are L classes in the classes set, $P_{x_i}^{c_n}$ denotes the probability of the tuple x_i belongs to the class c_n $(1 \le n \le L)$. Based on $P_{x_i}^{c_n}$, the L classes in descending order select the top k classes as the K-classes set.

Figure 3 shows the framework for UC-ELM, which consists of two phases:

Initialize: The first N uncertain tuples to arrive are seen as the initialisation data set. The uncertain tuples in the initialization data set are used to train the ELM classifier.

Update: When a new uncertain tuple *xnew* arrives, the uncertainty of tuple *xnew* is calculated by Definition 10. Each instance of tuple *xnew* is classified by the ELM classifier. Using Definition 2, we obtain the probability of the new tuple *xnew* belonging to each class. According to the Definition 12, the classes in descending order according to the probability of the new tuple *xnew* belongs to each class select the top k classes as K-classes set; then, we retrieve the K-classes set for tuple *xnew*. According to the Definition 11, we select a class, which will reduce the uncertainty the most. In other words, the goal is to select one class with maximum value of $\triangle U(c) = U(c) - U(c \bigcup \{x_i\})$, where c is any class in the K-classes set, then the tuple *xnew* is absorbed by that class. The uncertainty of the class is updated after absorbing the new tuple

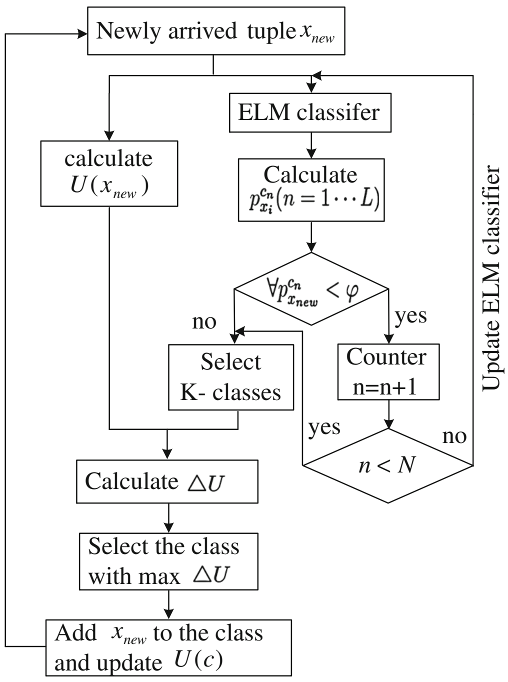
Newly arrived tuple x_{new}
ELM classifer
calculate $U(x_{new})$
Calculate $p^{c_n}_{x_i}(n=1\cdots L)$
$\forall p^{c_n}_{x_{new}} < \varphi$
no
yes
Select K- classes
Counter n=n+1
yes
no
$n<N$
Calculate $\triangle U$
Select the class with max $\triangle U$
Add x_{new} to the class and update $U(c)$
Update ELM classifier

Fig. 3 Uncertainty classifier framework

xnew. In order to solve the concept drift, a counter n is initialized to 0. Furthermore, let p^{c}_{xnew} denote the probability of *xnew* belonging the class which absorbs the new tuple *xnew*, if $p^{c}_{xnew} < \varphi$, then $n = n + 1$, where φ is user-specified threshold. Until $n = N$, where N is another user-specified threshold; we believe concept drift will occur, and the existing classifier is not suitable for current data; then, the classifier is deleted. Based on the recent training data set, a new classifier is trained.

Performance Verification

This section compares the performance of several algorithms (Support Vector Machine (SVM) [40], Dynamic Classifier Ensemble (DCE) [41], WEC-ELM, and UC-ELM,) in the real-world benchmark regression and multi-class classification data sets. We assume that all data arrive as chunks from a stream. All the evaluations are carried out in Windows 7, MATLAB R2012B, running on a Intel Core(TM) i5-2450M running at 2.50GHz and 4GB RAM.

Data Set and Experimental Setup

Because there is no real uncertain data set available, the data sets we used in the experiments are synthesized from real data sets. The values of each tuple of certain data set are fitted to four instances with four probabilities, respectively, and the expected value of the four values of instances is equal to the original value in the certain data set. All source data sets were taken from the UCI machine learning repository. Table 2 describe the data sets in detail.

- *Magic04:* 3,000 uncertain tuples are synthesized, each of which consists of 4 instances; the probability of each instance is randomly generated where the sum of probabilities of the instances in the same tuple is 1. The first 500 tuples are used as training data and are divided into 5 chunks; each of which contains 100 tuples, to train 5 classifiers, respectively. The others are used as testing data and arrive in a stream pattern, where each chunk contains 100 tuples.
- *Waveform:* 1,200 uncertain tuples are synthesized, each of which consists of 4 instances; the probability of each instance is randomly generated where the sum of probabilities of the instances in the same tuple is 1. The first 750 tuples are used as training data and are divided into 5 chunks; each of which contains 150 tuples, to train 5 classifiers, respectively. The others are used as testing data and arrive in a stream pattern, where each chunk contains 150 tuples.
- *Pendigits:* 2,000 uncertain tuples are synthesized, each of which consists of 4 instances; the probability of each instance is randomly generated where the sum of probabilities of the instances in the same tuple is 1. The first 1,000 tuples are used as training data and are divided into 5 chunks; each of which contains 200 tuples, to train 5 classifiers, respectively. The others are used as testing data and arrive in a stream pattern, where each chunk contains 200 tuples.
- *Letter:* 5,000 uncertain tuples are synthesized, each of which consists of 4 instances; the probability of each instance is randomly generated where the sum of probabilities of the instances in the same tuple is 1. The first 1,000 tuples are used as training data and are divided into 5 chunks; each of which contains 200 tuples, to train 5 classifiers, respectively. The others are used as testing data and arrive in a stream pattern, where each chunk contains 200 tuples.
- *Pageblocks:* 1,300 uncertain tuples are synthesized, each of which consists of 4 instances; the probability of

Table 2 Uncertain data set

Dataset	Number of tuples	Number of instances	Number of classes
Magic 04	3,000	4	2
Waveform	1,200	4	3
Pendigits	2,000	4	10
Letter	5,000	4	26
Pageblocks	1,300	4	5

Table 3 Training time comparison of SVM, DEC, WEC-ELM and UC-ELM

Datasets	SVM	DEC	WEC-ELM			UC-ELM		
			Sig Time(s)	Hardlim Time(s)	Sin Time(s)	Sig Time(s)	Hardlim Time(s)	Sin Time(s)
Magic 04	0.988	1.230	0.1310	0.1716	0.7662	–	–	–
Waveform	0.848	1.045	0.0624	0.0624	0.1256	0.0312	0.0312	0.0312
Pendigits	1.268	0.983	0.0942	0.0942	0.1584	0.0468	0.7509	0.0312
Letter	1.289	0.105	0.1032	0.1005	0.1382	0.0453	0.5528	0.0322
Pageblocks	1.118	0.979	0.6942	0.5942	0.892	0.0468	0.7509	0.0312

Table 4 Testing time comparison of SVM, DEC, WEC-ELM and UC-ELM

Datasets	SVM	DEC	WEC-ELM			UC-ELM		
			Sig Time(s)	Hardlim Time(s)	Sin Time(s)	Sig Time(s)	Hardlim Time(s)	Sin Time(s)
Magic 04	1.5231	1.9561	1.3709	1.7129	1.4135	–	–	–
Waveform	0.7256	0.9685	0.2808	0.4680	0.4251	0.1408	0.2028	0.2184
Pendigits	1.6312	2.3262	0.6597	0.7843	0.7786	0.5408	0.7509	0.4160
Letter	1.389	1.1232	0.7032	0.6986	0.6856	0.6805	0.5123	0.5987
Pageblocks	1.002	1.989	0.5645	0.687	0.596	0.4689	0.5961	0.426

Table 5 Accuracy comparison for SVM, DEC, WEC-ELM and UC-ELM

Datasets	SVM	DEC	WEC-ELM			UC-ELM		
			Sig	Hardlim	Sin	Sig	Hardlim	Sin
	TR (%)	TR (%)	TR (%)	TR (%)	TR (%)	TR (%)	TR (%)	TR (%)
Magic 04	75.23	74.65	76.17	73.62	64.37	–	–	–
Waveform	89.63	86.69	92.49	88.93	62.15	56.37	56.33	52.96
Pendigits	80.58	81.65	88.32	85.96	53.24	48.33	46.67	50.67
Letter	81.65	80.73	83.59	82.61	82.63	67.59	68.28	69.53
Pageblocks	76.32	79.59	80.57	82.62	82.12	60.21	62.17	62.36

each instance is randomly generated where the sum of probabilities of the instances in the same tuple is 1. The first 500 tuples are used as training data and are divided into 5 chunks; each of which contains 100 tuples, to train 5 classifiers, respectively. The others are used as testing data and arrive in a stream pattern, where each chunk contains 100 tuples.

In the experiments, we employ SVM [40] and DEC [41] as a competitive method. DEC is a recently proposed classifier ensemble for uncertain data streams classification. In order to show the performance of the ELM classifier, we use three different activate the functions (sig, hardlim and sin) to execution algorithm in the experiments. The number of classes of Magic04 is too small to run UC-ELM, so the UC-ELM algorithm run on Waveform data set and Pendigits. The default parameter settings: $E = 70\,\%, K = 2$.

Efficiency Evaluation

Firstly, we evaluate the efficiency of our methods. Table 3 shows the training time among SVM, DEC, WEC-ELM and UC-ELM. We can see that training time of WEC-ELM and UC-ELM is shorter than SVM and DEC. Table 4 shows the testing time for SVM, DEC, WEC-ELM and UC-ELM. As we expect, WEC-ELM and UC-ELM are faster than DEC.

Accuracy Evaluation

Table 5 shows the accuracy rate for SVM, DEC, WEC-ELM and UC-ELM with five data sets. We adopt the traditional learning principle in the training phase. We compare the accuracy among SVM, DEC, WEC-ELM and UC-

Table 6 Number of updates comparison of SVM, DEC, WEC-ELM and UC-ELM

Datasets	SVM	DEC	WEC-ELM			UC-ELM		
			Sig	Hardlim	Sin	Sig	Hardlim	Sin
	Times	Times	Times	Times	Times	Times	Times	Times
Magic 04	6	8	10	6	25	–	–	–
Waveform	1	1	1	1	3	0	1	1
Pendigits	2	3	1	3	5	1	2	4
Letter	3	3	5	5	4	4	3	4
Pageblocks	3	2	4	3	2	1	3	2

Table 7 Uncertainty comparison of SVM, DEC, WEC-ELM and UC-ELM

Datasets	SVM	DEC	WEC-ELM			UC-ELM		
			Sig	Hardlim	Sin	Sig	Hardlim	Sin
Magic 04	0.96	0.95	0.92	0.91	0.95	–	–	–
Waveform	0.93	0.91	0.91	0.91	0.93	0.70	0.68	0.72
Pendigits	0.80	0.86	0.80	0.82	0.83	0.65	0.66	0.75
Letter	0.83	0.80	0.82	0.81	0.82	0.59	0.68	0.61
Pageblocks	0.79	0.83	0.81	0.82	0.82	0.52	0.59	0.62

ELM as shown in Table 5. It can be seen that ELM can always achieve comparable performance as SVM and DEC. Seen from Table 5, different functions of ELM can be used in different data set in order to have similar accuracy in different size of data sets, although any output function can be used in all types of data sets. We can see that WEC-ELM and DEC performs as good as SVM.

Table 6 shows the number of updates among SVM, DEC, WEC-ELM and UC-ELM with five data sets. In WEC-ELM algorithm, since concept drift occurs, at least one component classifier is not suitable for the current data because of the concept drift. The worst-performing component classifier is deleted; the new component should be trained based on the current data and be added to ensemble classifier. In UC-ELM algorithm, when the counter is maximum value, the classifier should be deleted; we should train classifier according to current data. Due to the number of updates of WEC-ELM is more than UC-ELM algorithm, so UC-ELM is faster than WEC-ELM.

Uncertainty Evaluation

To the best of our knowledge, there is no existing standard for the evaluation of uncertain data streams classification. Due the nature of uncertain data, we propose the evaluation criteria of uncertain data classification based on [11]. Given a set of classes $c_l\ (1 \leq l \leq L)$ and corresponding uncertainty is $U(c_l)$, UM denote the uncertainty of the result of classification, $UM = \frac{1}{L}\sum_{1}^{L} U(c_l)$. UM is a positive number that allows comparison of the variation of classes that have significantly different tuples. In general, the smaller the UM value is, the greater quality of the classification result, as the Table 7 shows that the UC-ELM performs better than the other methods. The UC-ELM takes into account the uncertainty of the data tuples in order to decide the assignment of uncertain tuples to classes.

Conclusions

In this paper, we studied the problem of classification based on ELM over uncertain data streams. We first proposed a weighted ensemble classifier based on ELM (WEC-ELM), which can dynamic adjust the classifier and the weight of training uncertain data to solve the problem of concept drift. To get better classification results for uncertain data, an uncertainty classification based on ELM (UC-ELM) is proposed, which can balance the value and uncertainty of tuples. Finally, experiments were conducted on real data, which showed that the algorithms proposed in this paper are efficient and are able to deal with data streams in a real-time fashion.

An interesting direction for future work is classification on high-dimensional uncertain data. Another direction for future work is the classification for uncertain data streams in distributive environment.

Acknowledgments This research are supported by the NSFC (Grant No. 61173029, 61025007, 60933001, 75105487 and 61100024), National Basic Research Program of China (973, Grant No. 2011CB302200-G), National High Technology Research and Development 863 Program of China (Grant No. 2012AA011004) and the Fundamental Research Funds for the Central Universities (Grant No. N110404011).

References

1. Faradjian A, Gehrke J, Bonnett P. Gadt: a probability space adt for representing and querying the physical world. In: Proceedings of 18th international conference on data engineering, 2002.
2. Aggarwal CC. On density based transforms for uncertain data mining. ICDE 2007. IEEE 23rd international conference on data engineering, 2007.
3. Cheng R, Kalashnikov DV, Prabhakar S. Querying imprecise data in moving object environments. IEEE Trans Knowl Data Eng. 2004;16(9):1112–27.
4. Chen L, Özsu MT, Oria V. Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD international conference on management of data, 2005.
5. Ljosa V, Singh AK. Apla: indexing arbitrary probability distributions. In: ICDE 2007. IEEE 23rd international conference on data engineering, 2007.
6. Taylor JG. Cognitive computation. Cogn Comput. 2009;1(1): 4–16.
7. Wöllmer M, Eyben F, Graves A, Schuller B, Rigoll G. Bidirectional lstm networks for context-sensitive keyword detection in a cognitive virtual agent framework. Cogn Comput. 2010;2(3): 180–90.
8. Mital PK, Smith TJ, Hill RL, Henderson JM. Clustering of gaze during dynamic scene viewing is predicted by motion. Cogn Comput. 2011;3(1):5–24.
9. Cambria E, Hussain A. Sentic computing: techniques, tools, and applications. Dordrecht, Netherlands: Springer; 2012.
10. Wang Q-F, Cambria E, Liu C-L, Hussain A. Common sense knowledge for handwritten chinese text recognition. Common sense knowledge for handwritten Chinese text recognition, 2013.
11. Zhang C, Gao M, Zhou A. Tracking high quality clusters over uncertain data streams. ICDE. In: IEEE 25th international conference on data engineering, 2009.
12. Wang H, Fan W, Yu PS, Han J. Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining. 2003;39(4):226–35.
13. Zhu X, Wu X, Yang Y. Dynamic classifier selection for effective mining from noisy data streams. ICDM. In: Fourth IEEE international conference on data mining, 2004.
14. Zhang Y, Jin X. An automatic construction and organization strategy for ensemble learning on data streams. ACM SIGMOD, 2006.
15. Tsymbal A, Pechenizkiy M, Cunningham P, Puuronen S. Dynamic integration of classifiers for handling concept drift. Inf Fusion. 2008;9(1):56–68.
16. Kolter JZ, Maloof MA. Dynamic weighted majority: a new ensemble method for tracking concept drift. ICDM 2003. In: Third IEEE international conference on data mining, 2003.
17. Tsymbal A. The problem of concept drift: definitions and related work. Dublin: Computer Science Department, Trinity College; 2004.
18. Huang G-B, Chen L, Siew C-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw. 2006;17(4):879–92.
19. Huang G-B, Saratchandran P, Sundararajan N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans Neural Netw. 2005;16(1):57–67.
20. Huang G-B. Learning capability and storage capacity of two-hidden-layer feedforward networks. IEEE Trans Neural Netw. 2003;14(2):274–81.
21. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine. In: Technical report ICIS/03/2004 (also in http://www.ntu.edu.sg/eee/icis/cv/egbhuang.htm), (School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore), Jan 2004.
22. Huang G-B, Siew C-K. Extreme learning machine with randomly assigned RBF kernels. Int J Inf Technol. 2005;11(1):16–24.
23. Huang G-B, Chen L. Enhanced random search based incremental extreme learning machine. Neurocomputing. 2008;71:3460–8.
24. Huang G-B, Chen Y-Q, Babri HA. Classification ability of single hidden layer feedforward neural networks. IEEE Trans Neural Netw. 2000;11(3):799–801.
25. Huang G-B, Chen L. Convex incremental extreme learning machine. Neurocomputing. 2007;70:3056–62.
26. Huang G-B, Saratchandran P, Sundararajan N. An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern B. 2004;34(6):2284–92.
27. Huang G-B, Chen L, Siew C-K. Universal approximation using incremental feedforward networks with arbitrary input weights. In: Technical report ICIS/46/2003. School of Electrical and Electronic Engineering: Nanyang Technological University, Singapore; Oct 2003.
28. Huang G-B, Zhu Q-Y, Mao KZ, Siew C-K, Saratchandran P, Sundararajan N. Can threshold networks be trained directly? IEEE Trans Circuits Syst II. 2006;53(3):187–91.
29. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), vol. 2, (Budapest, Hungary), p. 985–990, 25–9 July, 2004.
30. Huang G-B, Siew C-K. Extreme learning machine: RBF network case. In: Proceedings of the eighth international conference on control, automation, robotics and vision (ICARCV 2004), vol. 2, (Kunming, China), p. 1029–36, 6–9 Dec 2004.
31. Huang G-B, Zhu Q-Y, Siew C-K. Extreme learning machine: theory and applications. Neurocomputing. 2006;70:489–501.
32. Huang G-B, Wang DH, Lan Y. Extreme learning machines: a survey. Int J Mach Learn Cybern. 2011;2(2):107–22.
33. Huang G-B, Babri HA. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw. 1998;9(1): 224–9.
34. Huang G-B, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern B Cybern. 2012;42(2):513–29.
35. Fletcher R. Practical methods of optimization. Chichester: Wiley; 1987.
36. Decherchi Sergio, Gastaldo Paolo, Zunino Rodolfo, Cambria Erik, Redi Judith. Circular-ELM for the reduced-reference assessment of perceived image quality. Neurocomputing. 2013;102(1):78–89.
37. Gastaldo Paolo, Zunino Rodolfo, Cambria Erik, Decherchi Sergio. Combining ELM with random projections. IEEE Intell Syst. 2013;28(5):18–20.
38. Sun Yongjiao, Yuan Ye, Wang Guoren. Extreme learning machine for classification over uncertain data. Neurocomputing. 2013;55:500–6.
39. Freund Y, Schapire RE. Adecision-the oretic generalization of online learning and an application to boosting. J Comput Syst Sci. 1997;55(1):119–39.
40. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;10(3):273–97.
41. Pan S, Wu K, Zhang Y, Li X. Classifier ensemble for uncertain data stream classification. In: Advances in knowledge discovery and data mining, vol. 6118. Berlin, Heidelberg: Springer; 2010. p. 488–495.