Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Extreme learning machine for classification over uncertain data

Yongjiao Sun*, Ye Yuan, Guoren Wang

Northeastern University, Shenyang 110004, China

ARTICLE INFO

Article history:
Received 17 June 2013
Received in revised form
23 July 2013
Accepted 26 August 2013
Communicated by G.-B. Huang
Available online 18 October 2013

Keywords:
Extreme learning machine
Uncertain data
OS-ELM
SVM
Single hidden layer feedforward neural networks

ABSTRACT

Conventional classification algorithms assume that the input data is exact or precise. Due to various reasons, including imprecise measurement, network delay, outdated sources and sampling errors, data uncertainty is common and widespread in real-world applications, such as sensor database, location database, biometric information systems. Though there exist a lot of approaches for classification, few of them address the problem of classification over uncertain data in database. Therefore, in this paper, we propose classification algorithms based on conventional and optimized ELM to conduct classification over uncertain data. Firstly we view the instances of each uncertain data as the training data for learning. Then, the probabilities of uncertain data in any class are computed according to learning results of each instance. Finally, using a bound-based approach, we implement the final classification. We also extend the proposed algorithms to classification over uncertain data in a distributed environment based on OS-ELM and Monte Carlo theory. The experiments verify the performance of our proposed algorithms.

1. Introduction

Recently, classification over uncertain data has gained much attention, due to the inherent uncertainties of data in many real-world applications, such as sensor network monitoring [1], object identification [2], moving object search [3–5], and the like [6,7,36]. A number of factors induce the uncertainty, including data collection error, measurement, data sampling error, obsolete source, network latency and transmission error. For example, in the moving objects databases, due to the limited resources, it is impossible for the database server to know the exact positions of all objects all the time. In this condition, there are two kinds of uncertainty, measurement error and sampling error. The measurement errors are derived from the imprecision of GPS devices, while in the sampling errors, the uncertainty derives from the update frequency of moving objects. Therefore, it is very important to manage and analyze uncertain data effectively and efficiently.

However, many traditional data classification problems become particularly challenging in the uncertain case, since traditional classification algorithms cannot work for the uncertain data. An uncertain data object may have many instances, and the traditional classification algorithms view each instance as a data object. Thus an uncertain data object can be categorized into many classes, but an uncertain data object only belongs to one class actually. Moreover, an uncertain data object may be attached a probability density function (pdf) that describes the probability of each instance appearing in this uncertain object. The uncertain classification algorithm should consider this uncertain semantics and efficiently process the computation associated with pdf. Obviously, traditional classification algorithms cannot deal with such challenges. Therefore in this paper, based on *extreme learning machine* (ELM) [9–17], we propose a new classification algorithm to process uncertain data objects. Specifically, we use the conventional ELM [10] for uncertain data to obtain binary classifications and the optimized ELM [9] is used for binary and multiclass classifications over uncertain data. We also extend these algorithms to distributed environments based on OS-ELM [8]. Conventional ELM is a good learning method to class data due to good generalization performance as well as improving the learning speed of neural network, maximizing the separating margin, and minimizing the training errors. However, optimized ELM tends to have better scalability and achieve similar (for regression and binary class cases) or much better (for multiclass cases) generalization performance at much faster learning speed than conventional SVM and LS-SVM [18,19]. OS-ELM on the basis of ELM is an algorithm that can handle data arriving or chunk-by-chunk with varying chunk size.

To implement uncertain classifications, we model uncertain data as an object consisting of instances with arbitrary probability distribution. Based on ELM, firstly, we train each instance associated with the uncertain data object. Then, the class probabilities of each instance are computed according to the learning results. Finally, we can obtain the final classification results by using a probability bound-based approach. To obtain more accurate classification results,

* Corresponding author. Tel.: +86 1390 9838 790.
E-mail address: sunyongjiao@ise.neu.edu.cn (Y. Sun).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.08.011

we train the huge and non-huge samples using the optimized ELM. In both cases, we can obtain multiclass results at a time, while SVM needs a lot of iterations. In a distributed environment, based on OS-ELM, we train data one-by-one or chunk-by-chunk with varying or fixed chunk length, so that we can transfer data objects with minimal network overheads in the training and testing phases.

Our contributions in this paper are as follows:

- We develop efficient classification algorithms on uncertain data.
- We use the conventional ELM for uncertain data objects to obtain binary classifications.
- We use the optimized ELM for uncertain data objects to obtain binary and multiclass classifications.
- We also adapt these algorithms to distributed environments based on OS-ELM and a sampling method of Monte Carlo.

The remainder of the paper is organized as follows. The related works are presented in Section 2. We formally define uncertain classification and give a review of ELM in Section 3. We propose uncertain classification algorithms based on conventional and optimized ELMs in Section 5. We adapt the uncertain classification algorithms to distributed scenarios in Section 6. We discuss the results of performance tests on real datasets in Section 7 and the conclusions of our work in Section 8.

2. Related works

There are some works for centralized classification. For centralized massive data, base-level classifiers are generated by applying different learning algorithms with heterogeneous models [23,24], or a single learning algorithm to different versions of the given data. Lin et al. [25] theoretically analyzed the rationale behind plurality voting, and Demrekler et al. [26] investigated how to select an optimal set of classifiers. While [27,28] study the classification of uncertain data using the support vector model, [29] performs classification using decision trees. Artificial neural network has been used in model-based clustering with a probability gained from expectation maximization algorithm for classification-likelihood learning [30].

There are some classification approaches for distributed scenarios. Collaborative [31,32] are the mainly P2P classification approaches. Collaborative generates a single model for the classification, while ensemble combines multiple models (classifiers) for predictions. This is a much more efficient approach which propagates only the statistics of the peers local data, with the decision tree of each peer converging to the global solution over time. Luo et al. [33] proposed building local classifiers using Ivotes [34] and performed prediction using a communication-optimal distributed voting protocol that requires the propagation of unseen data to most.

3. Problem definition and preliminaries

3.1. Problem definition

Uncertainty data model: Consider a set of *uncertain data objects* $U = \{U_1, \dots, U_n\}$. Each uncertain object U_i consists of a set of *instances* $u_1, \dots, u_m$. Each instance u_j is associated with a probability p_{u_j} called *appearance probability* with the constraint that $\sum_{j=1}^{m} p_{u_j} = 1$. Without loss of generality, we assume that each object is independent of other objects.

Note that each instance in an uncertain object is a multidimensional vector. Thus each instance can be viewed a multidimensional training data for ELM.

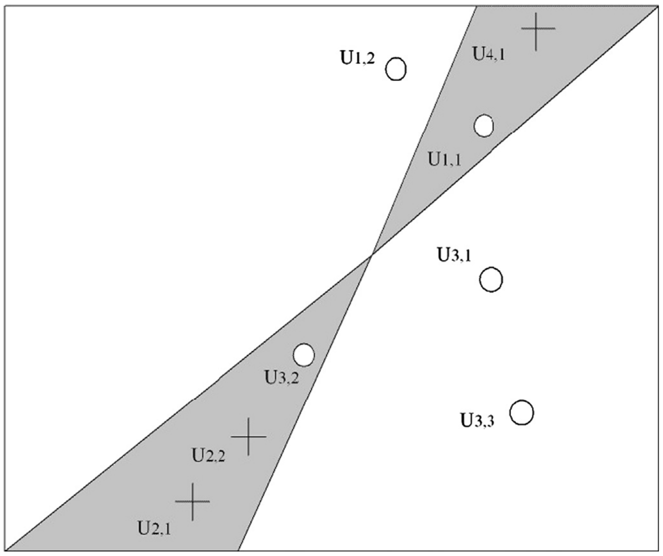

Fig. 1. An example of uncertain data model.

Table 1
Example of uncertain data model.

Uncertain object	# Instances	# Probability
U_1	$u_{1,1}, u_{1,2}$	$p_{1,1} = 0.2, p_{1,2} = 0.8$
U_2	$u_{2,1}, u_{2,2}$	$p_{2,1} = 0.3, p_{2,2} = 0.7$
U_3	$u_{3,1}, u_{3,2}, u_{3,3}$	$p_{3,1} = 0.1, p_{3,2} = 0.5, p_{3,3} = 0.4$
U_4	$u_{4,1}$	$p_{4,1} = 1$

For example, Fig. 1 shows uncertain objects U_1, U_2, U_3 and U_4 whose instances and corresponding appearance probabilities are given in Table 1.

In this example, we assume that the classification of uncertain data objects is a simple linear classification. In Fig. 1, the shadow area and the white area represent two classes, and each instance is classified into a corresponding category. However, many instances in the same uncertain data object are classified into different categories. Thus the problem of uncertain classification should consider the probability distributions of its all instances. The uncertain data classification is defined as follows.

Definition 1. Let Ω be the set of instances in all uncertain objects. Given the number of classes m, ELM can classify all instances in Ω into m categories. Then an uncertain object U belongs to a class C_i $(1 \le i \le m)$ if the summarized probability of instances in C_i is the largest.

4. Preliminaries

This subsection briefly gives an overview of ELM.

ELM was originally proposed for the single-hidden-layer feedforward neural networks and then extended to the generalized single-hidden layer feedforward networks where the hidden layer need not be neuron [5,6,37]. In ELM, all the hidden node parameters are randomly generated without tuning. The output function of ELM for generalized SLFNs is

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \boldsymbol{\beta} \cdot \mathbf{h}(\mathbf{x}) \tag{1}$$

where $\boldsymbol{\beta} = [\beta_1, \dots, \beta_L]^T$ is the vector of the output weights between the hidden layer of L nodes and the output node. $\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]^T$ is the output (row) vector of the hidden layer with respect to the input **x**. $\mathbf{h}(\mathbf{x})$ actually maps the data from the d-dimensional input space to the D-dimensional hidden-layer feature space (ELM feature space) H, and thus, $\mathbf{h}(\mathbf{x})$ is indeed a feature mapping.

Eq. (1) can be written compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{2}$$

where

$$\mathbf{H}(\mathbf{a}_1,\dots,\mathbf{a}_L,b_1,\dots,b_L,\mathbf{x}_1,\dots,\mathbf{x}_L) = \begin{bmatrix} G(\mathbf{a}_1,b_1,\mathbf{x}_1) & \cdots & G(\mathbf{a}_L,b_L,\mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1,b_1,\mathbf{x}_L) & \cdots & G(\mathbf{a}_L,b_L,\mathbf{x}_L) \end{bmatrix} \tag{3}$$

H is called the hidden layer output matrix of the network [26]; the ith column of **H** is the ith hidden node's output vector with respect to inputs $\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_L$ and the jth row of **H** is the output vector of the hidden layer with respect to input $\mathbf{x}_j$.

In the binary classification case, ELM only uses a single-output node, and the class label closer to the output value of ELM is chosen as the predicted class label of the input data.

To the problem of multiclass classifier, ELM has been optimized by unifying LS-SVM and PSVM [20]. Due to ELM can approximate any target continuous functions and the output of the ELM classifier $\boldsymbol{\beta}\mathbf{h}(\mathbf{x})$ can be as close to class labels in the corresponding regions as possible, the classification problem for a single-output node can be formulated as

$$\text{Minimize}: L_{P_{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2$$
$$\text{Subject to}: \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i - \xi_i, \quad i=1,\dots,N, \tag{4}$$

where C is a user-specified parameter. Based on the Karush–Kuhn–Tucher (KKT) [35] theorem, to train ELM is equivalent to solving the following dual optimization problem:

$$L_{D_{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - t_i + \xi_i) \tag{5}$$

where each Lagrange multiplier α_i corresponds to the ith training sample.

To the problem of multiclass classifier, ELM has multioutput nodes instead of single-output node, and m-class of classifiers have m output nodes. This problem can be formulated as

$$\text{Minimize}: L_{P_{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2$$
$$\text{Subject to}: \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i^T - \xi_i^T, \quad i=1,\dots,N. \tag{6}$$

The same as the problem of single-output node, to train ELM is equivalent to solving the following dual optimization problem:

$$L_{D_{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}(\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}_j - t_{i,j} + \xi_{i,j}) \tag{7}$$

where $\boldsymbol{\beta}_j$ is the vector of the weights linking hidden layer to the jth output node and $\boldsymbol{\beta}_j = [\boldsymbol{\beta}_1,\dots,\boldsymbol{\beta}_m]$.

From the above equation, we can see that the single-output node case considered a specific case of multioutput nodes when the number of output nodes is set to one. From aforementioned optimization problem, two kinds of solutions can be obtained in different size of training datasets. The first one is that the number of training samples is not huge. The output function of ELM classifier for this solution is

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{1}}{\mathbf{C}} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}. \tag{8}$$

The second solution is that the number of training samples is huge. In the case, the output function of ELM classifier is

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{1}}{\mathbf{C}} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \tag{9}$$

5. Classification over uncertain data based on ELM

5.1. Uncertain classification based on conventional ELM

The classification based on conventional ELM can be summarized as two steps. In the training, the instances of all uncertain objects are learned by ELM for obtaining binary classes. In the testing, we can determine the categories of an uncertain object based on the probability of its instances appearing which class.

We step up an example to show the detail processes of two steps. In Fig. 2, we assume that the classification of uncertain data objects is a linear classification, and the shadow and white areas denote binary classes obtained in the training for all instances. In the testing, consider uncertain objects A and B in Fig. 2. According to Definition 1, the highest sum value of instance probabilities in each class determines the final result of an uncertain object. In Fig. 2, the *separating boundary* reflects different classes in the training step. In the figure, an uncertain object A locates in both the shadow and white areas, hence it may be assigned to the shadow class based on ELM. However, if we consider the distribution of A, the probability of appearing in white area is much larger than that in the shadow area. Based on this intuition, object A should be classified into the class of white area. Obviously, this approach will perform correct uncertain classifications, since it takes account into the complete probability distribution of A across different classes.

A more efficient uncertain classification approach can be derived from the following theorem:

Theorem 1. *Let s_l be the summarized probability of instances, of uncertain object U_i, appearing in class l. If $s_l < 0.5$, U_i is not in class l, otherwise U_i is classified into category l.*

In the binary classification of conventional ELM, its decision function is given by

$$f_L(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{L}\beta_i G(\mathbf{a}_i,b_i,\mathbf{x})\right) = \text{sign}(\boldsymbol{\beta}\cdot\mathbf{h}(\mathbf{x})) \tag{10}$$

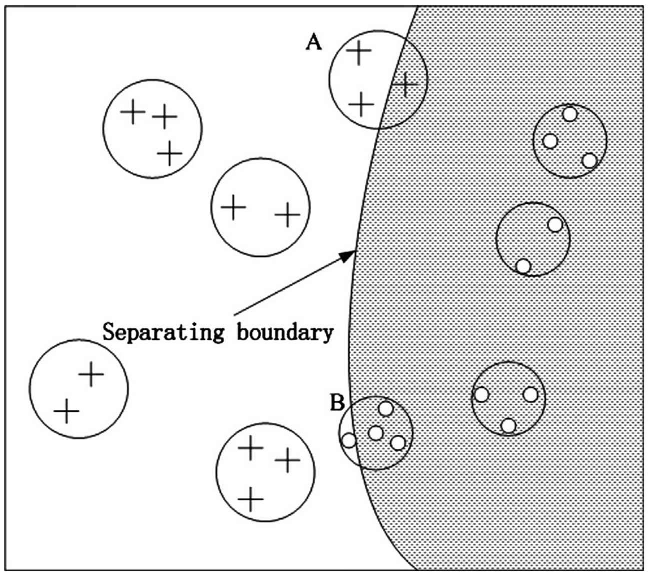

Fig. 2. Classification over uncertain data.

Recall that an uncertain object can be modeled as a region, in multidimensional space, associated with a pdf. This region is called an uncertain region.

For an uncertain object, it is possible that its uncertain region crosses the separating boundary in the multidimensional space, i.e., uncertain objects A and B in Fig. 2. In this case, A and B have positive probabilities in both classes, and the membership of class will be determined by which class has a higher probability. To simplify this computing, we construct an function from Eq. (10) as

$$f(\mathbf{S})=\begin{cases}1, & \mathbf{S}\cdot \mathrm{sign}(\boldsymbol{\beta}\cdot \mathbf{h}(\mathbf{x}))\geq 0.5\\ 0, & \mathbf{S}\cdot \mathrm{sign}(\boldsymbol{\beta}\cdot \mathbf{h}(\mathbf{x}))<0.5\end{cases} \tag{11}$$

where $\mathbf{S}$ is the sum probability of instances in one class. The testing procedure of binary classification is shown as follows:

Algorithm 1. Uncertain binary classification().

```
1: $U_1=\{u_{11},\dots,u_{1m}\},\dots,U_n=\{u_{n1},\dots,u_{nm}\}$;
2: Construct a container $\Omega=\{\{u_{11},\dots,u_{1m}\},\dots,\{u_{n1},\dots,u_{nm}\}\}$;
3: Input $\Omega$ into Eq. (12);
4: We compute $\mathbf{S}$ based on Definition 1 in the training results;
5: if $\mathbf{S}\cdot \mathrm{sign}(\boldsymbol{\beta}\cdot \mathbf{h}(\mathbf{x}))<0.5$ then
6:    Uncertain object $U_i$ is pruned from the current category;
7: else
8:    Uncertain object $U_i$ is classified into the current category;
9: end if
```

5.2. Uncertain classification based on optimized ELM

For *binary classification* case over non-huge training samples, the output node is one ($m=1$), and the decision function of *optimized ELM* classifier is

$$f(\mathbf{x})=\mathbf{sign}\left(\mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{1}}{\mathbf{C}}+\mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}\right). \tag{12}$$

In Eq. (12), class of classifier has one output node. The expected output vector is

$$T=\begin{bmatrix}t_1\\ \vdots\\ t_N\end{bmatrix}. \tag{13}$$

Similarly, for binary classification cases over huge training samples, the decision function of ELM classifier is

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

However, for multiclass classifier with multioutputs, it is obviously that Theorem 1 cannot be applied to uncertain multiclass classification. For example, assume we want to classify uncertain objects into three categories. If all summarized probabilities in three classes are smaller than 0.5, Theorem 1 and Eq. (11) for binary classification cannot be used for multiclass classification. Therefore, we revise Theorem 1 as follows:

Theorem 2. *Assume uncertain object U_i is classified into m categories, $c_1,c_2,\dots,c_m$, and the summarized values of instance probabilities in each class are $s_1,s_2,\dots,s_m$. If $s_l<1/m$, then U_i is not in class l.*

Proof. If uncertain object U_i is not classified into category l, then $s_l<\sum_1^m s_i/(m-1)$, where $i\neq l$. Thus, $(1-s_l)/(m-1)-s_l>0\Rightarrow s_l<1/m$. □

Based on Theorem 2, for *multiclass* cases over non-huge training samples, the output number of nodes is larger than one ($m>1$). The predicted class label of sample $\mathbf{x}$ is

$$label(\mathbf{x})=\arg\max_{i\in\{1,\dots,m\}} f_i(\mathbf{x}) \tag{15}$$

where $f_i(\mathbf{x})$ is the output function of the jth output node. The expected output vector is

$$\mathbf{T}=\begin{bmatrix}t_{11} & \cdots & t_{1m}\\ \vdots & \vdots & \vdots\\ t_{N1} & \dots & t_{Nm}\end{bmatrix} \tag{16}$$

For multioutput nodes over huge training samples, the predicted class label of a given testing sample is the output node that has the highest output. The testing procedure of multiclass classification using pruning based on ELM, called PE, is shown as follows:

Algorithm 2. Uncertain multiclass classifications().

```
1: $U_1=\{u_{11},\dots,u_{1m}\},\dots,U_n=\{u_{n1},\dots,u_{nm}\}$;
2: Construct a container $\Omega=\{\{u_{11},\dots,u_{1m}\},\dots,\{u_{n1},\dots,u_{nm}\}\}$;
3: Input $\Omega$ into Eq. (16);
4: We compute $\mathbf{S}$ based on Definition 1 in the training result;
5: if $s_l<1/m$; then
6:    $U_i$ is pruned from the class $l$;
7: else
8:    Compare with the values $s$;
9: end if
```

6. Uncertain classification in distributed network

To deal with distributed classification, we assume that all the uncertain objects are distributed over a set of sites, each of which contains parts of uncertain objects. Since the OS-ELM algorithm can handle data arriving chunk-by-chunk with varying chunk size, by viewing the data of a site as a chunk of the training dataset, we can easily design an approach to generate an uncertain distributed classifier based on OS-ELM, called (DOE).

In the learning of DOE, we propose a *distributed classifier* to train instances in all uncertain objects to obtain a binary classification. In the testing of DOE, any site can receive the results of binary classification and easily apply Algorithm 1 to classify its local uncertain data objects. Thus the learning process of DOE plays a critical role.

The learning of DOE mainly consists of two steps. First, a classifier is initialized in a site by learning the local all instances in uncertain objects. Second, the training results in that site are propagated to other sites to sequentially learn the remaining data in the network.

The learning process of distributed classifier is shown as follows: *Initialization phase*: Assume the network consists of sites $\{P_0,\dots,P_k,\dots,P_m\}$. From the given training set $\aleph$ a small chunk of training data $\aleph_0$, in a site P_0, is used to initialize the learning.

1. Assign random input weights $\mathbf{a}_i$ and bias b_i.
2. Calculate the initial hidden layer output matrix $\mathbf{H}_0$.
3. Estimate the initial output weight $\beta^{(0)}$:
4. Set $k=0$.

Sequential learning phase: Present the chunk of new observations $\aleph_{k+1}$ in the $(k+1)$th site P_{k+1}.

1. Calculate the partial hidden layer output matrix $\mathbf{H}_{k+1}$, where $\mathbf{H}_{k+1}$ is calculated by the $(k+1)$th chunk of data $\aleph_{k+1}$.
2. Set $\mathbf{T}_{k+1}$.
3. Calculate the output weight $\beta^{(k+1)}$.
4. Set $k=k+1$. Go to Sequential Learning Phase (1).

This learning process is finished until $k = m$.

The above algorithm is the result of adapting conventional ELM to the distributed scenario. Now we will present the distributed version of uncertain classification based on optimized ELM and a sampling method. This uncertain version mainly works as follows. First, we propose a *distributed sampling classifier* based on the optimized ELM to conduct classification for each instance associated uncertain objects across the network. Second, each uncertain object uses Algorithm 2 to conduct local classification based on the results of its instances in the first step.

Below we show the detail steps of the distributed sampling classification.

For an instance u of an uncertain object U, let X_u be a random variable as an indicator to the event that u is classified into some category. Then we define function $\phi(X_u)$,

$$\phi(X_u) = \begin{cases} 1 & \text{if } u \text{ is classified into the category } m \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

Algorithm 3 shows the concrete process of the distributed sampling classification based on ELM, called SE.

Algorithm 3. ApproxDistributeClassify.

```
1:  for $L = 0$ do
2:     Sample each instance $u$ across the network using
       probability $P_u$;
3:     Then we can obtain a snapshot of all uncertain objects.
       We apply optimized ELM to instances in the snapshot, and
       we can obtain the value of $\phi(X_u)$.
4:     if $((\phi(X_u) == 1)$ then
5:        $L = L + 1$;
6:     end if
7:     Until $N$ times;
8:  end for
9:  Attach class $m$ to $u$, and return $u$ to its corresponding
    uncertain object.
10: Return $L/N$.
```

When the sample size is large enough, the approximation quality can be guaranteed from the well known Chernoff–Hoeffding bound [21].

Theorem 3. *For arbitrarily* δ $(0 < \delta < 1)$ *and* τ $(\tau > 0)$, *if* $N \geq (3 \ln 2/\delta)/\tau^2$,

$$\Pr|L/N - E(\phi(X_u)) > \tau E(\phi(X_u))| \leq \delta \quad (18)$$

7. Performance evaluation

In this section, we implement all the proposed algorithms in this paper and evaluated the performance of our approach. We conduct extensive experiments by evaluating the efficiency of our techniques in different settings. All the evaluations are carried out in MATLAB 7.0 and C++ in a AMD Athlon Dual Core processor with 3.0 GHz and 2GB RAM.

7.1. Experiments setup

The experiments use real datasets taken from the UCI Machine Learning Repository [22] as original datasets. In order to extensively verify the performance of different algorithms, two types of datasets have been tested in our simulations, including binary and multiclass classification cases. For each uncertain object U, we first decide the center location C_U of its uncertainty region UR(U), where C_U is the real data. Then we randomly produce the instances in the UR(U) as the instances and the number of the instances are set $w \times m$, where $0 < w \leq 1$. Tables 2 and 3 show the training and testing instances number of binary and multiclass classifications respectively.

Our experiments are mainly conducted based on the following two aspects: (1)The comparisons with classification accuracy and training time among SVM, LS-SVM, ELM and PE (Algorithm 2) over varying parameters in centralized environment. (2)The comparisons with classification accuracy and training time among DOE (distributed classification based on OS-ELM), VE [38] and SE (Algorithm 3) over varying parameters in distributed environment. We set 1000 as the number of the Sigmoid and RBF nodes in ELM. The user-specified parameters are (C,L), which are chosen from the range $(2^{-7},\ldots,2^{25})$.

7.2. Performance evaluation of binary classification in centralized environment

Table 4 shows the classification accuracy (CA) among SVM, LS-SVM, ELM and PE in a centralized environment. Since traditional ELM cannot classify uncertain data, in order to test the performance of PE, we select ELM to classify the certain data while PE classifies the uncertain data. From the table, we can observe that the classification accuracy is slightly lower than ELM. Table 5 shows the training time among SVM, LS-SVM, ELM and PE in centralized environment. Due to we use pruning method during uncertain object classification, the training time of PE is slower than ELM, but faster than SVM and LS-SVM.

The multiclass classification performance is presented in Tables 6 and 7. Table 6 shows the classification accuracy of SVM, LS-SVM, ELM and PE in centralized environment. Since ELM, SVM and LS-SVM are traditional classifier, in this part, we use the real data to learn the classifier, while PE classifies the uncertain data.

Table 2
Specification of binary classification problems.

Datasets	# Train	# Test
Diabete	768	384
Liver	345	180
Credit	700	360
Adult	7645	39 847
Mushroom	2300	9845

Table 3
Specification of multiclass problems.

Datasets	# Train	# Test	# Classes
Vowel	778	670	11
Iris	167	82	3
Glass	201	108	6
Segment	2370	1120	7
DNA	2987	1750	3
Satimage	6148	3375	6

Table 4
Classification accuracy comparisons with SVM, LS-SVM, ELM and PE over binary class datasets in centralized environment.

Datasets	SVM	LS-SVM	ELM			PE		
			Gaus	Sig	RBF	Gaus	Sig	RBF
	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)
Diabete	74.34	75.23	75.96	76.25	77.14	75.89	76.21	77.25
Liver	70.54	69.96	70.11	71.05	69.83	69.97	70.56	69.75
Credit	83.17	84.02	85.79	85.31	85.99	85.15	84.98	85.49

Table 5
Training time comparisons with SVM, LS-SVM, ELM and PE over binary class datasets in centralized environment.

Datasets	SVM	LS-SVM	ELM			PE		
			Gaus	Sig	RBF	Gaus	Sig	RBF
	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
Diabete	0.915	0.294	0.061	0.235	0.312	0.101	0.297	0.413
Liver	0.846	0.107	0.0073	0.075	0.0065	0.0094	0.091	0.0085
Credit	1.235	0.243	0.077	0.202	0.180	0.098	0.284	0.210

Table 6
Classification accuracy comparisons with SVM, LS-SVM, ELM and PE over multiclass classification in centralized environment.

Datasets	SVM	LS-SVM	ELM			PE		
			Gaus	Sig	RBF	Gaus	Sig	RBF
	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)	CA (%)
Vowel	53.23	50.02	55.36	51.08	52.64	53.25	48.97	50.64
Iris	91.38	91.99	92.04	92.88	92.71	88.65	89.57	89.04
Glass	63.25	62.56	65.77	62.03	61.14	62.64	60.12	59.38

Table 7
Training time comparisons with SVM, LS-SVM, ELM and PE over multiclass classification in centralized environment.

Datasets	SVM	LS-SVM	ELM			PE		
			Gaus	Sig	RBF	Gaus	Sig	RBF
	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)	Time (s)
Vowel	3.128	0.847	0.087	0.324	0.295	0.187	0.469	0.502
Iris	0.175	0.0098	0.0057	0.0201	0.0219	0.010	0.0223	0.0235
Glass	0.687	0.0164	0.0054	0.0325	0.0399	0.0085	0.0632	0.0581

Form the table, we can observe that the difference of classification accuracy between PE and ELM is not much. Table 7 shows the training time of SVM, LS-SVM, ELM and PE in centralized environment. Similarly, PE is also shower than ELM due to pruning process.

7.3. Performance evaluation classification in distributed environment

Table 8 presents the comparisons of the performance among DOE, VE and SE with Sigmoid additive hidden node. In these experiments, the method about ensemble ELM with weighted majority voting is used for VE which is a regular voting ensemble solutions. Each site builds classifier over their local training data based on optimized ELM, then propagates the classifiers to other sites. For constructing a distributed network, we adopt larger datasets, Adult and Mushroom, to test the performance of binary classification in a distributed environment. The scale of network is set to 100 nodes. In Table 8, though the classification accuracy of DOE is the highest, the response time of DOE is the lowest than VE and SE. Parallel computing of ELM makes VE be the fastest and sampling makes SE keep a better classification accuracy.

Table 9 shows the comparisons of the performance among DOE, VE and SE with Sigmoid additive hidden node over multiclass classification in distributed network. Larger datasets, Segment, Satimage and DNA, are used for testing in this experiment. Similar to binary classification in distributed network, DOE, VE and SE both have benefit for multiclass classification. DOE obtains the highest classification accuracy, and the fastest one is VE, while SE is the middle one on classification accuracy and response time.

Table 8
Comparisons with DOE, VE and SE over binary class datasets in distributed environment.

Datasets	DOE		VE		SE	
	Testing (%)	Time (s)	Testing (%)	Time (s)	Testing (%)	Time (s)
Adult	81.68	294	75.64	1.745	77.59	25.7
Mushroom	85.48	162	77.93	0.923	81.17	16.5

Table 9
Comparisons with DOE, VE and SE over multiclass classification in distributed environment.

Datasets	DOE		VE		SE	
	Testing (%)	Time (s)	Testing (%)	Time (s)	Testing (%)	Time (s)
Segment	92.78	174	81.25	0.197	88.14	18.4
Satimage	89.36	267	79.31	0.845	84.87	21.7
DNA	92.22	189	78.64	0.687	86.55	23.5

8. Conclusion

Conventional classification algorithms cannot be applied to uncertain data objects. In this paper, we propose classification algorithms based on ELM to conduct classification over uncertain data. Using a bound-based approach, we implement the binary and multiclass classification over uncertain data objects. We also extend the proposed algorithms to classification over uncertain data in a distributed environment based on OS-ELM and Monte Carlo theory. A large number of experiments verify the performance of our proposed algorithms.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China under Grant nos. 61025007, 60933001, 61202087, and 61100024; the Open Foundation of WUHAN University no. SKLSE2012-09-40; the Fundamental Research Funds of the Central Universities no. N120404012; the National Basic Research Program of China under Grant no. 2011CB302200-G; the 863 Program under Grant no. 2012AA011004, and the Public Science and Technology Research Funds Projects of Ocean Grant no. 201105033.

References

[1] A. Faradjian, J. Gehrke, P. Bonnet, Gadt: a probability space ADT for representing and querying the physical world (ICDE), in: International Conference on Data Engineering, 2002, pp. 201–211.
[2] C. Böhm, A. Pryakhin, M. Schubert, The Gauss-tree: efficient object identification in databases of probabilistic feature vectors (ICDE), in: International Conference on Data Engineering, 2006, p. 9.
[3] R. Cheng, D. Kalashnikov, S. Prabhakar, Querying imprecise data in moving object environments (TKDE), IEEE Trans. Knowl. Data Eng. 16 (2004) 1112–1127.
[4] L. Chen, M.T. Ösu, V. Oria, Robust and fast similarity search for moving object trajectories, in: ACM Special Interest Group on Management of Data (SIGMOD), 2005, pp. 491–502.
[5] V. Ljosa, A.K. Singh, APLA: indexing arbitrary probability distributions, in: International Conference on Data Engineering (ICDE), 2007, pp. 247–258.

[6] A.D. Sarma, O. Benjelloun, A.Y. Halevy, J. Widom, Working models for uncertain data, in: International Conference on Data Engineering (ICDE), 2006, p. 7.
[7] M.A. Soliman, I.F. Ilyas, K.C. Chang, Top-k query processing in uncertain databases, in: International Conference on Data Engineering (ICDE), 2007, pp. 896–905.
[8] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Trans. Neural Networks 17 (6) (2006) 1411–1423.
[9] G.-B. Huang, H. Zhou, X.-J. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B (TSMC) 42 (2) (2012) 513–529.
[10] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501.
[11] G.-B. Huang, C.-K. Siew, Extreme learning machine: RBF network case, in: Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision, Kunming, China, 2004, pp. 1029–1036.
[12] G.-B. Huang, C.-K. Siew, Extreme learning machine with randomly assigned RBF kernels, Int. J. Inf. Technol. 11 (1) (2005) 16–24.
[13] A.A. Mohammed, R. Minhas, Q.M. JonathanWu, M.A. Sid-Ahmed, Human facerecognition based on multidimensional PCA and extreme learning machine, Pattern Recognition 44 (10–11) (2011) 2588–2597.
[14] B.P. Chacko, V.R.V. Krishnan, G. Raju, P.B. Anto, Handwritten character recognition using wavelet energy and extreme learning machine, Int. J. Mach. Learn. Cybern. 3 (2) (2012) 149–161.
[15] J.-H. Zhai, H.-Y. Xu, X.-Z. Wang, Dynamic ensemble extreme learning machine based on sample entropy, Soft Comput. 16 (9) (2012) 1493–1502.
[16] N. Liu, H. Wang, Ensemble based extreme learning machine, IEEE Signal Process. Lett. 17 (8) (2010) 754–757.
[17] W. Jun, W. Shitong, F.-L. Chung, Positive and negative fuzzy rule system, extreme learning machine and image classification, Int. J. Mach. Learn. Cybern. 2 (4) (2011) 261–271.
[18] C. Cortes, V. Vapnik, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[19] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[20] G. Fung, O.L. Mangasarian, Proximal support vector machine classifiers, in: Knowledge Discovery and Data Mining (KDD), 2001, pp. 77–86.
[21] D. Angluin, L.G. Valiant, Fast probabilistic algorithms for Hamiltonian circuits and matchings, in: ACM Symposium on the Theory of Computing (STOC), 1977, pp. 30–41.
[22] C.L. Blake, C.J. Merz, UCI Repository of Machine Learning Databases, Department of Information and Computer Science, University of California, Irvine, CA, 1998. [Online]. Available: ⟨http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.
[23] C.J. Merz, Using correspondence analysis to combine classifiers, Mach. Learn. 36 (1–2) (1999) 33–58.
[24] G. Tsoumakas, I. Katakis, I.P. Vlahavas, Effective voting of heterogeneous classifiers, in: European Conference on Machine Learning (ECML), 2004, pp. 465–476.
[25] X. Lin, S. Yacoub, J. Burns, S. Simske, Performance analysis of pattern classifier combination by plurality voting, Pattern Recognition Lett. 24 (2003) 1959–1969.
[26] M. Demrekler, H. Altincay, Plurality voting-based multiple classifier systems: statistically independent with respect to dependent classifier sets, Pattern Recognition 35 (11) (2002) 2365–2379.
[27] J. Bi, T. Zhang, Support vector classification with input data uncertainty, in: Neural Information Processing Systems (NIPS), 2004.
[28] C.C. Aggarwal, On density based transforms for uncertain data mining, in: International Conference on Data Engineering (ICDE), 2007, pp. 866–875.
[29] S. Tsang, B. Kao, K.Y. Yip, W.-S. Ho, S.D. Lee, Decision trees for uncertain data, in: International Conference on Data Engineering (ICDE), 2009, pp. 441–444.
[30] S.-S. Cheng, H.-C. Fu, H.-M. Wang, Model-based clustering by probabilistic self-organizing maps, IEEE Trans. Neural Networks 20 (5) (2009).
[31] K. Bhaduri, R. Wolff, C. Giannella, H. Kargupta, Distributed decision-tree induction in peer-to-peer systems, Stat. Anal. Data Mining (SADM) 1 (2) (2008) 85–103.
[32] H.-H. Ang, V. Gopalkrishnan, S.-C.-H. Hoi, Wee Keong Ng: cascade RSVM in peer-to-peer networks, in: European Conference on Machine Learning/European Conference on Principles of Data Mining and Knowledge Discovery, 2008, pp. 55–70.
[33] P. Luo, H. Xiong, K. Lü, Z.-Z. Shi, Distributed classification in peer-to-peer networks, in: Knowledge Discovery and Data Mining (KDD), 2007, pp. 968–976.
[34] L. Breiman, Pasting small votes for classification in large databases and on-line, Mach. Learn. 36 (1–2) (1999) 85–103.
[35] R. Fletcher, Practical methods of optimization, Constrained Optim. 2 (1981).
[36] X.-Z. Wang, Y. DS, T. ECC, A comparative study on heuristic algorithms for generating fuzzy decision trees, IEEE Trans. Syst. Man Cybern. Part B 31 (2) (2001) 215–226.
[37] G.-B. Huang, D.H. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[38] G. Tsoumakas, I. Katakis, I.P. Vlahavas, Effective voting of heterogeneous classifiers, in: European Conference on Machine Learning (ECML), 2004, pp. 465–476.

Yongjiao Sun received the B.S. degree, M.S. degree and Ph.D. degree in Computer Science from Northeastern University, China, in 2006, 2008 and 2012 respectively. Currently, he is an associate professor at Computer Science Department of the Northeastern University. His main research interests include distributed data management, data analysis and probabilistic database.

Ye Yuan received the B.S. degree, M.S. degree and Ph.D. degree in Computer Science from Northeastern University, China, in 2004 and 2007 respectively. Currently, he is an associate professor at the Computer Science Department of Northeastern University, and he is also a visiting scholar at Department of Computer Science and Engineering in Hong Kong University of Science and Technology. His research interests include graph database, probabilistic database, P2P data management, and data piracy.

Guoren Wang received the B.Sc., M.Sc., and Ph.D. degrees in Computer Science from the Northeastern University, Shenyang, China, in 1988, 1991, and 1996, respectively. Now, he is currently a full professor with the School of Information Science and Engineering, Northeastern University. His research interests include uncertain data management, data-intensive computing, visual media data management and analysis, distributed query processing and optimization techniques, and bioinformatics. He has published more than 100 research papers in international conference proceedings and journals.