Neurocomputing 128 (2014) 119–127

Contents lists available at ScienceDirect

Neurocomputing

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

TOSELM: Timeliness Online Sequential Extreme Learning Machine

Yang Gu [a,b,*], Junfa Liu [a], Yiqiang Chen [a], Xinlong Jiang [a,b], Hanchao Yu [a,b]

[a] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[b] University of Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 15 September 2012
Received in revised form
17 December 2012
Accepted 11 February 2013
Available online 8 November 2013

Keywords:
Timeliness
Online sequential learning
Adaptive weight
Adaptive iteration

ABSTRACT

For handling data and training model, existing machine learning methods do not take timeliness problem into consideration. Timeliness here means the data distribution or the data trend changes with time passing by. Based on timeliness management scheme, a novel machine learning algorithm Timeliness Online Sequential Extreme Learning Machine (TOSELM) is proposed, which improves Online Sequential Extreme Learning Machine (OSELM) with central tendency and dispersion characteristics of data to deal with timeliness problem. The performance of proposed algorithm has been validated on several simulated and realistic datasets, and experimental results show that TOSELM utilizing adaptive weight scheme and iteration scheme can achieve higher learning accuracy, faster convergence and better stability than other machine learning methods.

1. Introduction

Nowadays, machine learning methods have been applied to kinds of fields, such as text classification [1], recognition [2], medical field [3] and so on. Machine learning methods allow computers to deduce behaviors based on empirical data, such as data from sensors or databases. A major goal of machine learning is to intelligently recognize complex patterns and give accurate results. However, the difficulties lie in that all the possible inputs are too large that the observed examples (training data) can seldom cover all of them; besides the training time will be unbearable if the input dataset becomes too big.

In 2004, Huang et al. [4–6] proposed a new machine learning algorithm, Extreme Learning Machine (ELM) for single-hidden layer feedforward neural networks (SLFNs). Comparing with traditional neural networks, it has two obvious advantages: fast learning speed and high learning accuracy. ELM is a batch learning method; however, in some real world applications, sequential learning algorithms may be preferred over batch learning algorithms, as they neither need to retrain the model with all the training data nor require all the training data fully prepared in advance. Due to the advantages of sequential learning, ELM has been updated to Online Sequential Extreme Learning Machine (OSELM) [7], a sequential learning algorithm, which can learn training data one-by-one or chunk-by-chunk and discard training data as long as the training phase is over. Taking this advantage, OSELM algorithm has been widely used in machine learning field [8,9]. There are also other incremental learning algorithms, using incremental data to update the training model. [10] utilizes new labeled samples in each feedback loop to incrementally train the classifier, and collects training data to alleviate the small sample problem, which finally improves the performance of sketch retrieval with little time cost. [11] adopts incremental learning method to study face recognition. The system does not need all the training data fully prepared before calculation. Instead, it uses incremental data to change training database, then implements online learning, which outperforms the previous work.

As to the concept of timeliness, time series related work should be mentioned. [12] proposes an adaptive ensemble of extreme learning machines for time series (stationary and non-stationary time series) to solve one step ahead prediction problem. The ensemble model structure is optimized by the model selection method, and the weights of different ELMs are first averagely assigned, then updated by the calculation error with the scheme that big error makes the model weight come down, and vice versa. Finally, this algorithm achieves desirable prediction accuracy with little computation and time consumption. [13] adopts multiple Support Vector Regression (SVR) models to realize iterated time series prediction. The multiple SVR models are trained independently based on the same training data with different targets. The prediction outputs of models are considered as the next inputs to make further prediction. [14] proposes a robust recurrent kernel online learning (RRKOL) algorithm to predict time series, which achieves the weight convergence with regularized risk management through the recurrent hyper-parameters. And the sparsely selected meaningful data helps to reduce the calculation burden.

Although sequential learning methods and time series prediction methods are widely studied, seldom of them take the common

* Corresponding author at: Institute of Computing Technology, Chinese Academy of Sciences, Pervasive Computing Research Center, No.6 Kexueyuan South Road Zhonggu, Haidian District, Beijing 100190, China. Tel.: +86 15201279517.
E-mail address: guyang@ict.ac.cn (Y. Gu).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.02.047

existed timeliness problems into consideration. The timeliness here means that with time passing by, the distribution of data or the trend of data changes. Timeliness problems extensively exist in our life, such as price prediction in stock market and real-estate market, climate data prediction, gesture recognition and so on. Even though for common sequential learning algorithms, the newly incremental data are used to update the training model, the distribution of incremental data is usually considered to be similar to the original data's. However, in fact, changing environment, users and devices always lead to different data distribution and trend. Considering this, Timeless Online Sequential Extreme Learning Machine (TOSELM) is proposed, which brings an adaptive weight scheme and an adaptive iteration scheme to OSELM so that the incremental data can contribute reasonable weight to represent the current situation and ensure the stability and convergence of the model. The rest of this paper is organized as follows: Section 2 presents the proposed TOSELM, it first briefly overviews the ELM and OSELM then introduces TOSELM in detail; Section 3 gives the performance evaluation on simulated data and realistic data with the proposed method and other machine learning methods. Section 4 concludes the paper.

2. Timeless Online Sequential Extreme Learning Machine

In this section, we first briefly overview ELM and OSELM, and then introduce TOSELM with motivation, modeling, and algorithm steps.

2.1. Brief of ELM and OSELM

ELM is a single hidden layer feedforward neural network (SLFN). Given N arbitrary distinct samples (x_i, t_i) $i=1,2,\ldots,N$, where $x_i=[x_{i1},x_{i2},\ldots,x_{in}]$, and $t_i=[t_{i1},t_{i2},\ldots,t_{im}]$, the neural network with L hidden neurons can be illustrated in Fig. 1, and the output of this network is as follows:

$$f_L(x)=\sum_{i=1}^{L}\beta_i G(a_i,b_i,x),\ a_i\in\Re^n,\ b_i\in\Re,\ \beta_i\in\Re^m \tag{1}$$

$a_i=[a_{i1},a_{i2},\ldots,a_{in}]$ is the weight vector connecting the ith hidden neuron and input neurons, and b_i is the threshold of the ith hidden neuron, $G(a_i,b_i,x)$ is the output of the ith hidden neurons. $\beta_i=[\beta_{i1},\beta_{i2},\ldots,\beta_{im}]^T$ is the weight vector connecting the ith hidden neuron and the output neurons. If the activation function of hidden neurons is $g(x)$, then the output of the ith hidden neuron is

$$G(a_i,b_i,x)=g(a_i x+b_i),\ a_i\in\Re^n,\ b_i\in\Re \tag{2}$$

The above two equations can be written compactly as

$$H\beta=T \tag{3}$$

where

$$H=\begin{bmatrix} G(a_1,\ b_1,\ x_1) & \cdots & G(a_L,\ b_L,\ x_1)\\ \vdots & \ddots & \vdots\\ G(a_1,\ b_1,\ x_N) & \cdots & G(a_L,\ b_L,\ x_N)\end{bmatrix}_{N\times L},\quad \beta=\begin{bmatrix}\beta_1^T\\ \vdots\\ \beta_L^T\end{bmatrix}_{L\times m},\quad T=\begin{bmatrix}t_1^T\\ \vdots\\ t_N^T\end{bmatrix}_{N\times m} \tag{4}$$

As discussed in [15], the parameters of hidden neuron a_i and b_i do not need to be adjusted in model training phase, only need to be randomly assigned. Hence the hidden layer output matrix of the neural network H and the output matrix T are known, the solution of parameter β can be obtained as follows:

$$\beta=H^{\dagger}T=(H^TH)^{-1}H^TT \tag{5}$$

where $H^{\dagger}$ is the Moore–Penrose generalized inverse (pseudo inverse) of the hidden layer output matrix H. This equation can be seen as a linear system, the implementation of least squares solution of Eq. (5) is the solution of ELM.

OSELM, an incremental learning algorithm, is derived from ELM by putting newly incremental data into training. OSLEM algorithm does not need to retrain the model with all the data; instead it just updates the model with incremental data. The effect of incremental data is reflected by the correction, $\Delta\beta$, which modifies the old model $\beta_{(0)}$ to form a new model β^* according to the following equation:

$$\beta^*=\beta_{(0)}+\Delta\beta(X^*) \tag{6}$$

Apparently, the calculation burden is lighter, because it only utilizes incremental data to update the model instead of retraining the model with all the data.

In [16], the author gives a solution to this model. According to Eq. (5), assuming the current learning system $\beta_{(0)}=K_0^{-1}H_0^TT_0$, where $K_0=H_0^TH_0$, and the N_1 newly incremental samples are $\{(x_i,t_i)\}_{i=N_0+1}^{N_0+N_1}$, then

$$K_1=\begin{bmatrix}H_0\\ H_1\end{bmatrix}^T\begin{bmatrix}H_0\\ H_1\end{bmatrix}=\begin{bmatrix}H_0^TH_1^T\end{bmatrix}\begin{bmatrix}H_0\\ H_1\end{bmatrix}=K_0+H_1^TH_1 \tag{7}$$

Therefore, the new model parameter becomes

$$\begin{aligned}\beta_{(1)}&=K_1^{-1}\begin{bmatrix}H_0\\ H_1\end{bmatrix}^T\begin{bmatrix}T_0\\ T_1\end{bmatrix}=K_1^{-1}(K_1\beta_{(0)}-H_1^TH_1\beta_{(0)}+H_1^TT_1)\\ &=\beta_{(0)}+K_1^{-1}H_1^T(T_1-H_1\beta_{(0)})\end{aligned} \tag{8}$$

The solution corresponds to the idea of Eq. (6) that $\beta_{(1)}$ modifies $\beta_{(0)}$ only with the calculation of incremental data, which greatly improves the efficiency of model generation. With the increase of increment number, after $k+1$ times incremental learning, the model parameter can be written as follows:

$$\beta_{(k+1)}=\beta_{(k)}+K_{k+1}^{-1}H_{k+1}^T(T_{k+1}-H_{k+1}\beta_{(k)}) \tag{9}$$

The part $K_{k+1}^{-1}H_{k+1}^T(T_{k+1}-H_{k+1}\beta_{(k)})$ can be seen as the model correction made by new samples $\{X_{k+1},T_{k+1}\}=\{(x_i,t_i)\}_{i=\left(\sum_{j=0}^{k}N_j\right)+1}^{\sum_{j=0}^{k+1}N_j}$.

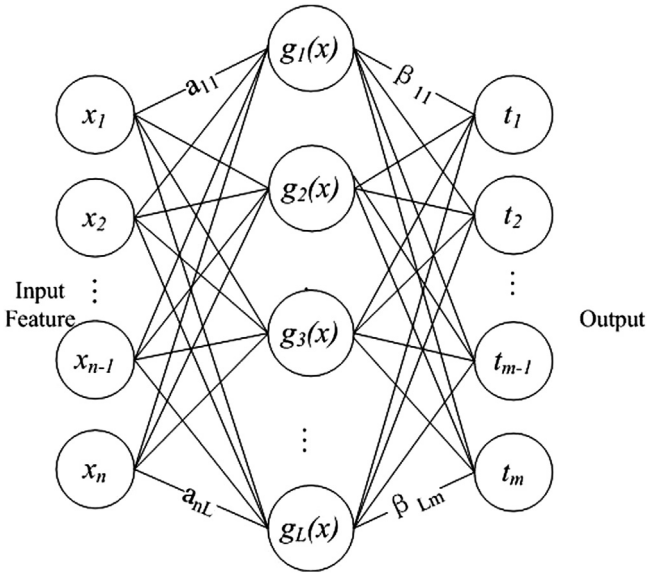

Fig. 1. SLFN with L hidden neurons.

2.2. TOSELM: Timeliness Online Sequential Extreme Learning Machine

We have reality driven motivation to extend OSELM to TOSELM. Since data for real world application are unstable, changing with time. To better predict or present the latest situation, the newly incremental data should contribute more

than the previous data. Therefore, base on the framework of OSELM, we put forward the adaptive timeliness weight scheme and iteration scheme to improve the performance of incremental learning for timeliness problem.

In Eq. (9), the component $(T_{k+1}-H_{k+1}\beta_{(k)})$ can be seen as the bias when using old model parameter $\beta_{(k)}$ to predict the output, which is also the effect brought by incremental data. For regression problem, the newly incremental data are always gathered in the latest situation which can better represent the current situation, therefore given a suitable weight, the prediction result can be more accurate, such as the stock price prediction problem; for classification problem, the distribution of collected data for the same class may be different due to the changing environment, devices, users and so on, therefore given a reasonable weight to the newly incremental data, the model can be more compatible for data in the same class but with different distribution, such as the gesture recognition problem. Hence, in order to reflect the timeliness of new samples, component $(T_{k+1}-H_{k+1}\beta_{(k)})$ should be given a suitable penalization, so that the correction of the old model cannot only represent the current situation but also decrease the error of the model. The revised form of Eq. (9) can be concisely expressed as follows:

$$\beta_{(k+1)}=\beta_{(k)}+\omega K_{k+1}^{-1}H_{k+1}^{T}(T_{k+1}-H_{k+1}\beta_{(k)}) \tag{10}$$

The penalization weight ω reflects the timeliness effect of newly incremental data which not only enlarges the coverage of model, but improves the compatibility of the model as well. There are two factors should be taken into consideration when calculating ω: (1) the mean difference between incremental data and adjacent history incremental data and (2) the variance difference between incremental data and adjacent history incremental data. From the statistical aspect, mean value and variance value are simple and effective elements to depict data characteristic. Mean value describes the central tendency of data, and variance value reflects the dispersion of data. Theoretically, the relationship between mean, variance and contribution is illustrated by the following Table 1.

The newly incremental data should give more contribution than history data is the main idea of timeliness weight scheme, that is ω must be larger than 1. According to Eq. (9), when $\omega=1$, incremental data and history data take 50% proportion respectively. However, not all the newly incremental data are meaningful; some incremental data may be meaningless even wrong. Therefore, the weight should not be obtained through single incremental learning; instead it should be calculated with continuous incremental learning. If mean value and variance value of the nth incremental data are greatly different from the $(n-1)th$ incremental data, the weight should be small to avoid overfitting, and if the following $(n+1)th$ data are similar to the nth data, that is the difference of mean value and variance value is small, which leads to the nth and the $(n+1)th$ data more meaningful than meaningless, then the $(n+1)th$ data can give larger weight to represent the new data distribution. Since the relationship between weight value and mean value difference satisfies a decreasing function, we adopt exponential function and also utilize the variance different to reflect the effect of data dispersion. As the analysis above, the weight is expressed in the form of exponential with the difference of mean and variance

$$\omega=1+2\exp(-|\mathrm{mean}(X1)-\mathrm{mean}(X2)|^{|\mathrm{var}(X1)-\mathrm{var}(X2)|}) \tag{11}$$

Here $X1$ is the newly incremental data, $X2$ is the history adjacent incremental data, **mean** is the function to obtain mean value and **var** is the function to obtain variance value (if either of the newly incremental data or adjacent history incremental data is a single sample, then the weight function becomes $\omega=1+2\exp(-|\mathrm{mean}(X1)-\mathrm{mean}(X2)|)$). Even though we get the adaptive penalization factor ω, only one time correction cannot guarantee the stability and convergence of the algorithm. Therefore, we adopt iteration scheme like Newton's method to gradually correct the model to a certain level of stability and convergence predefined by users. The proposed iteration scheme is indicated by the following equation:

$$\beta_{k(j+1)}=\beta_{k(j)}+\omega K_{k+1}^{-1}H_{k+1}^{T}(T_{k+1}-H_{k+1}\beta_{(k)}) \tag{12}$$

For the kth increment, the final iteration number is constraint by $|\beta_{k(j+1)}-\beta_{k(j)}|<\varepsilon$. Where $\beta_{k(j)}$ is the jth iteration result for the kth increment and ε is a small value defined by users. If this constraint is satisfied, then $\beta_{(k)}=\beta_{k(j+1)}$. Finally after $(k+1)$ times incremental learning, the timeliness model parameter becomes:

$$\beta_{(k+1)}=\beta_{(k)}+\omega K_{k+1}^{-1}H_{k+1}^{T}(T_{k+1}-H_{k+1}\beta_{(k)}) \tag{13}$$

As mentioned above, TOSELM algorithm can be summarized in the following steps:

1. Determine the model parameters by N initial samples, such as the number of hidden neurons L, the activation function $g(x)$.
2. Randomly assign the value of weight vector a_i and bias scalar $b_i, i=1,2,\ldots,L$.
3. Calculate the original hidden layer output matrix H_0 and the initialization model parameter $\beta_0=H_0^{\dagger}T_0$.
4. Calculate the weight ω according to the kth newly incremental data $\{X_k,T_k\}$ and adjacent history incremental data by Eq. (11).
5. Calculate H_k and β_k with the kth incremental data $\{X_k,T_k\}$.
6. Calculate the $(j+1)th$ iteration of model parameter $\beta_{k(j+1)}$ by Eq. (12).
7. If $|\beta_{k(j+1)}-\beta_{k(j)}|<\varepsilon$, the iteration stops, else go to step 6 for another iteration.
8. Output the final parameter $\beta_{(k)}=\beta_{k(j+1)}$.

Steps 4–7 are used for incremental learning, when there are new data, the algorithm will repeat these steps. The flowchart of the algorithm can be concisely given in Fig. 2.

Table 1
Relationship between mean, variance and weight.

Value	Difference between incremental data and history incremental data			
Mean	small	small	big	big
Variance	small	big	small	big
Weight	big	big	small	small

3. Experimental performance

In this section, we will give the performance evaluation on simulated data and practical data. The algorithms are run on the computer with the following configuration: Windows XP OS, 2.8 GHz Main Frequency, 4 GB RAM memory, and Intel Core i5-2003 Processor. Here we mainly compare algorithms ELM, OSELM, Weight OSELM (WOSELM), Back Propagation (BP), and Support Vector Regression (SVR) with TOSELM. BP is a traditional artificial neural network method, but it is a batch learning method which needs training data fully prepared before training a model, and if new data are arrived, the model should be retrained with the new training dataset, which is time consuming. SVR is the regression version of Support Vector Machine (SVM), only using support vector to generate model, therefore the calculation burden and time consumption are relatively lighter than BP. WOSELM is a transitional algorithm from OSELM to TOSELM, this algorithm adopts the same weight scheme as TOSELM, but it does not have the iteration step to guarantee the stability and convergence of the model.

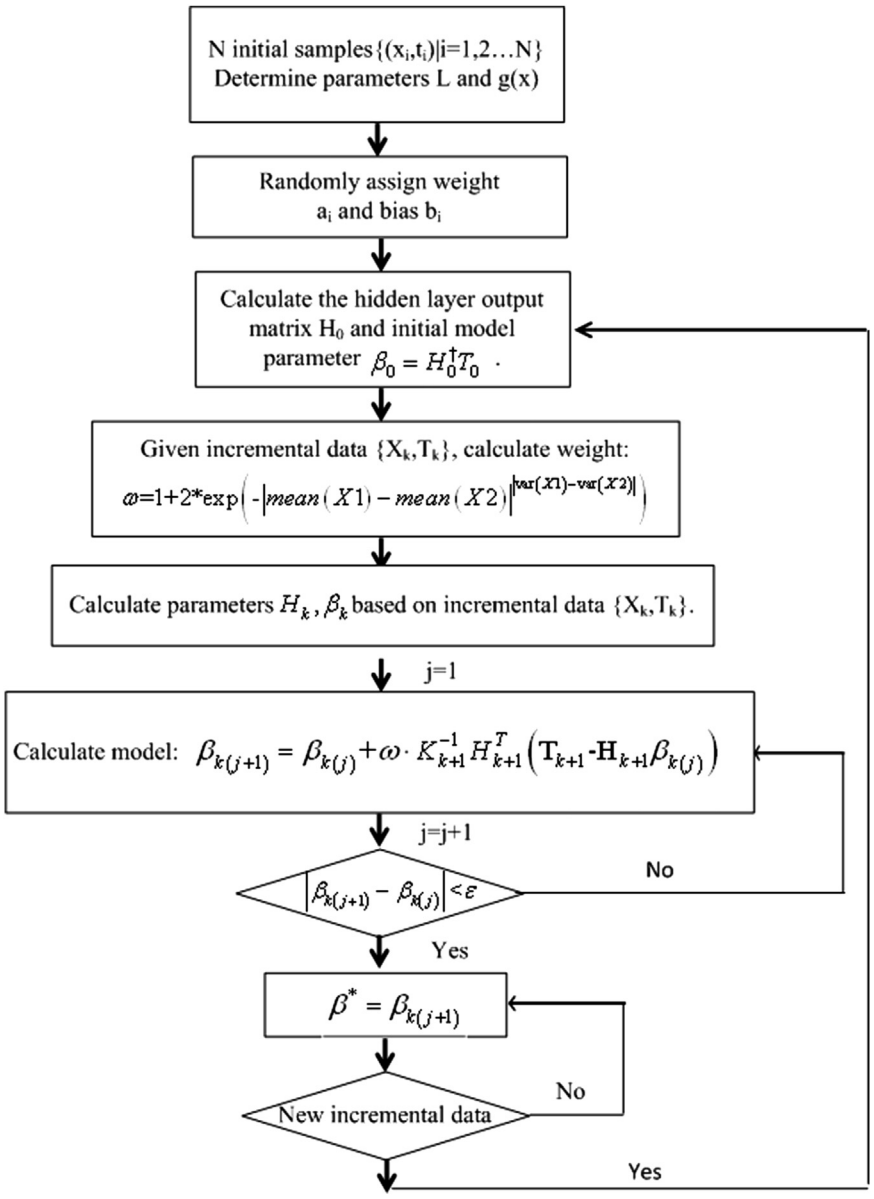

Fig. 2. The flowchart of algorithm.

3.1. Performance on simulated data

The data here used are similar to the data in [5], and we compare our proposed algorithm with other neural network algorithms: ELM, OSELM, WOSELM, BP and SVR.

3.1.1. Initial data preparing

The initial training are generated the same as [5]. The input X is randomly generated in the range $[-10, 10]$ by the following equation:

$$X = 20\ \mathrm{rand}(1, N) - 10 \tag{14}$$

N is the number of data. The output Y satisfies the 'SinC' relationship with X:

$$Y = \frac{\sin(X)}{X} \tag{15}$$

In order to make it more like a real world problem, we add noise in the range $[-0.4, 0.4]$, that is:

$$Y = Y + 0.8\ \mathrm{rand}(1, N) - 0.4 \tag{16}$$

3.1.2. Incremental data preparing

In order to reflect the timeliness effect, the distribution of incremental data is not the same as initial data's. Here we introduce a scalar and a bias to adjust the distribution of the data. The range of incremental data X is [10,30], which makes:

$$X = 20\mathrm{rand}(1, N) + 10 \tag{17}$$

And the output Y is adjusted by a scalar and bias:

$$Y = A\frac{\sin(X)}{X} + B \tag{18}$$

The parameter can be seen as the fluctuation of data which is caused by different users, devices and environments, here we set $A = 1.5$ and $B = 2$.

3.1.3. Testing data preparing

The testing data are generated the same as incremental data. The training parameters for different methods are as follows: For ELM and OSELM related algorithm (OSELM, WOSELM, and TOSELM), the number of initial data is 1000, the number of hidden neurons is 25,

the activate function is radial basis function ('rbf'), the number of increment is 10, and the number of incremental data is 500 each time. For BP, the two parameters: number of hidden neuron and looping time are both 50. For SVR, we set kernel function 'rbf', and other parameters are selected by cross validation. We first study the relationship between different weights and testing performance, which is illustrated in Fig. 3.

Theoretically, the larger the weight is, the faster convergence will be. While in fact, the larger weight makes the model more suitable for the current incremental data, if the distribution of testing data is different from incremental data, it will lead to large testing error, which is 'overfitting'. Therefore, in order to guarantee the testing accuracy, convergence and generalization, the penalization weight should be in a certain range, so that it cannot only ensure the fast speed of convergence, but also guarantee the desirable testing accuracy. From Fig. 3, we can see that with the increasing value of weight in range [1,10], the testing error first comes down then goes up. When the weight is in a certain range which is not too large, taking range [1,2] for example, the newly incremental data not only gives more contribution to the model, but also reduces the testing error. However, when the weight goes too large, the model is over fitted for the incremental data, which increases the testing error. In order to achieve dynamical adaptation and improve the performance of algorithm, we use adaptive weight calculated in Section 2 instead of fixed weight. The performance of adaptive weight is shown in Fig. 4. Compared with OSELM, WOSELM decreases the testing error, which shows the effectiveness of proposed weight scheme.

The other improvement of TOSELM is adopting iteration scheme to calculate model parameter β. Even though in Fig. 4, the performance of WOSELM is better than OSELM, with more experiments we find that, for the same incremental data, executing the calculation in Eq. (12) once can neither make the system stable nor modify the old model thoroughly. Hence, we study the relationship between iteration and testing performance, the results are illustrated in Fig. 5. Fig. 5(a) shows the difference between new model parameter $\beta_{k(j+1)}$ and $\beta_{k(j)}$; Fig. 5(b) shows the testing error when the iteration number increases. During the iteration step, the termination condition threshold ε is set to be an empirical value 0.001 (for different realistic system ε can be set as $\varepsilon = 0.1,\ 0.01$ or other small value).

From Fig. 5(a), we can see a great difference of model parameter, which indicates the parameter of one time calculation is not stable. Fig. 5(b) illustrates, with the increase of iteration number, the test error becomes stable, which also confirms that updating the model parameter with Newton's method like iteration scheme can guarantee the stability and convergence of the model.

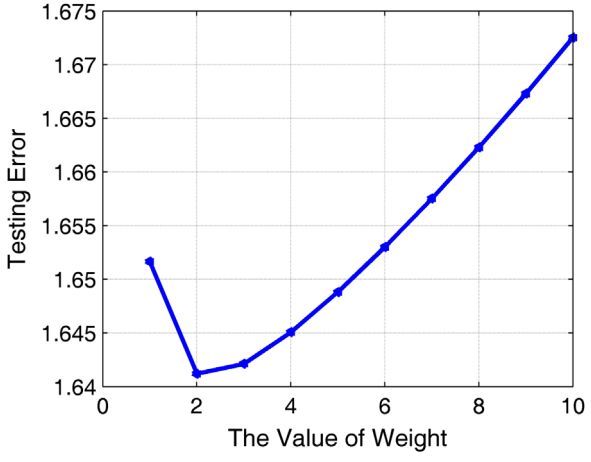

Fig. 3. Testing errors with different weights.

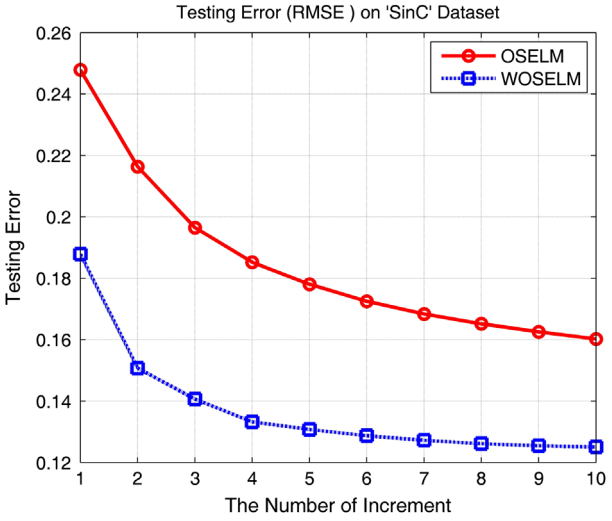

Fig. 4. Comparison between WOSELM and OSELM.

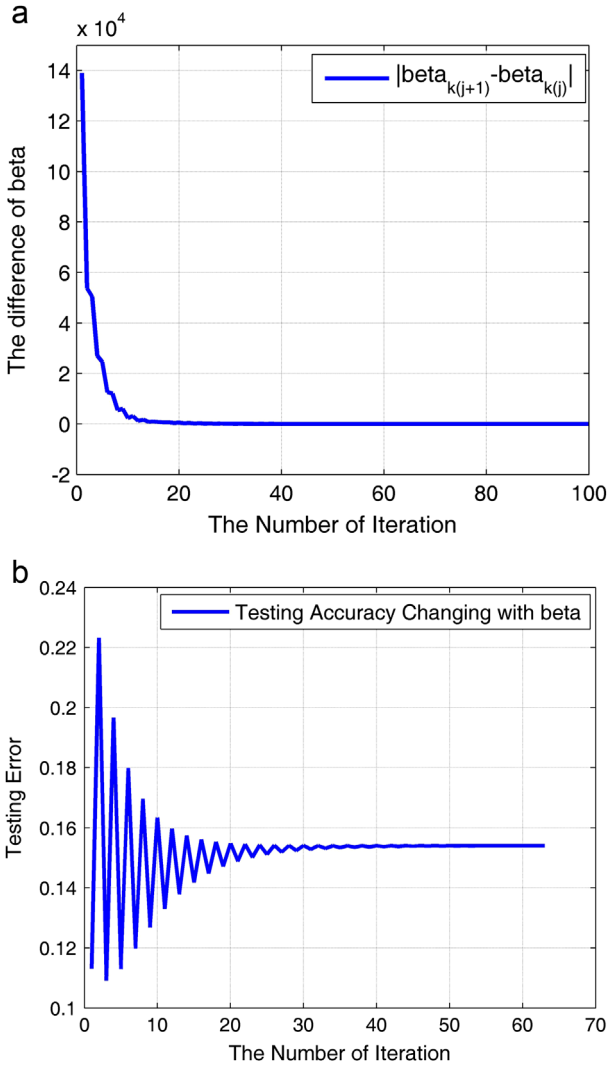

Fig. 5. Adaptive iteration performance. (a) Difference of model parameter and (b) difference of testing error.

After separately showing the performance made by the proposed weight scheme and iteration scheme, we compare the performance of TOSELM and other machine learning algorithms in Fig. 6.

Fig. 6(a) shows the testing error of the different algorithms. We adopt root-mean-square error (RMSE) to calculate the testing error. Obviously, with the increase of increment number, the training dataset becomes larger which covers more possible situations, therefore the testing errors for all these algorithms come down. Comparing with OSELM and WOSELM, TOSELM quickly achieves small testing error and stability due to the weight scheme and iteration scheme. Comparing with batch learning ELM, BP and SVR, the sequential learning TOSELM also outperforms the best, since batch learning methods do not take timeliness problem into consideration. Fig. 6 (b) shows the training time consumption. OSELM related algorithms (OSELM, WOSELM, and TOSELM) consume less time, even though TOSELM algorithm utilizes iteration to guarantee the convergence, it just costs a little more time than OSELM and WOSELM (since the feature dimension is small). BP and SVR are batch learning methods, therefore, with the increase of the training data, the training time shows a rising trend. Although ELM is also a batch learning method, its fast learning speed advantage makes it cost less time than BP and SVR.

3.2. Performance on real world problems

After examining the performance on the theoretical 'SinC' dataset, we also test the performances on real world datasets which have timeliness problem, such as stock data, climate data and gesture recognition data.

3.2.1. Stock price data

The stock data is downloaded from YAHOO Finance API (the download link is given in [17]), and we choose the stock whose code is $s=600005$. SS in Shanghai Index as the real-world stock dataset. Since the price of stock fluctuates every day, considering the timeliness effect, the long-ago data should give little contribution. Here 307 days' price data from 2011-03-03 to 2012-06-08 are chosen. The output is the opening price of a day and the input features are the closing price, the highest price and the lowest price of the day. The initial model is generated by 20 days' data, the testing data are the last 7 days' price and the rest data are incremental data for 15 times. The parameters for ELM and OSELM related algorithms are set as: the number of input neuron is 15, the activation function is 'rbf', and the number of increment is 15. For BP, the looping time is 50, and the number of hidden neuron is 25.

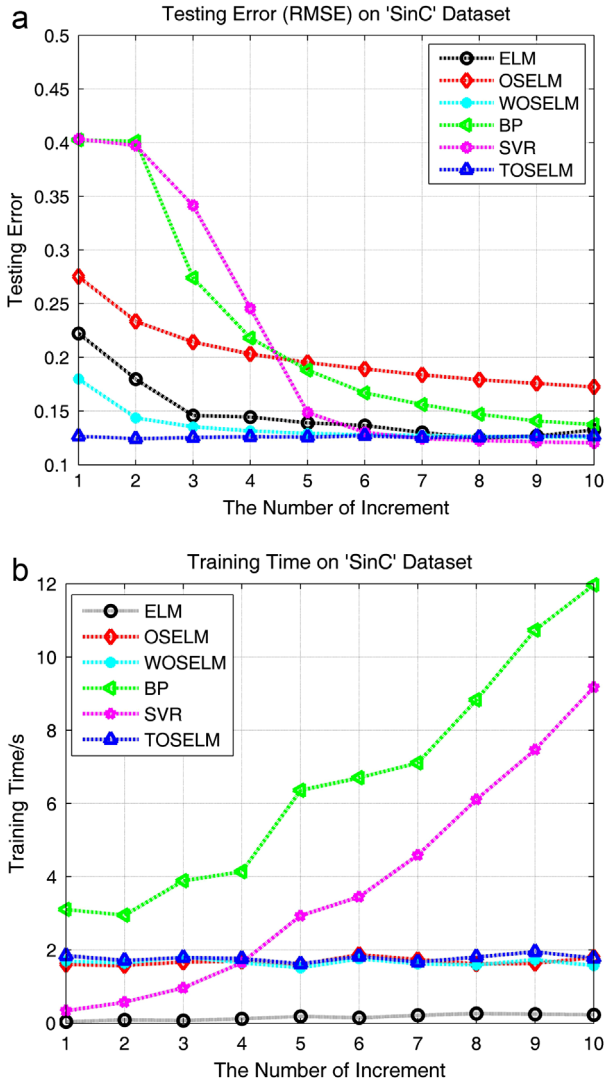

Fig. 6. Different algorithms' performance on 'SinC' data. (a) Testing error on 'SinC' data and (b) training time on 'SinC' data.

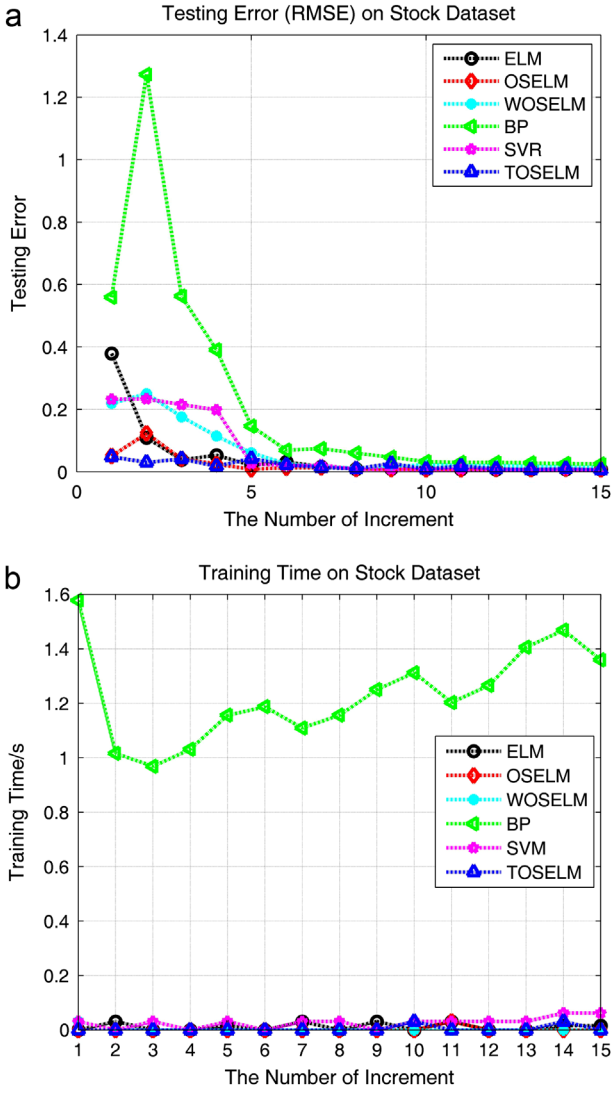

Fig. 7. Different algorithms' performance on stock price data. (a) Performance on stock price data and (b) training time on stock price data.

For SVR, 'rbf' is chosen as kernel function and other parameters are selected by cross validation. We adopt RMSE to calculate the testing error, and the final performance is illustrated in Fig. 7.

From Fig. 7(a) we can see that with the increase of increment number, the testing error shows a decreasing trend, and the proposed algorithm has the smallest testing error, which shows the effectiveness of adaptive weight scheme and iteration scheme. When the increment number reaches to 10, results of all these algorithms become stable. Fig. 7(b) shows the training time consumption of different algorithms. BP still has the highest time consumption, followed by SVR, and ELM and OSELM related algorithm have the lowest time consumption. Since the dataset is not so large (307 days' data with three input features and one output value), the trend of time consumption is not obvious when incremental data are put into the algorithms.

3.2.2. Climate data

For this part, the data are obtained from China meteorological Data Sharing Service System (the download link is given in [18]). We adopt the data collected from No. 54511 station in Beijing from 1989 to 2011. The output data is monthly average temperature, and the input data are monthly average air pressure, humidity, precipitation, wind speed, and sunlight perception. The initial data are the first 12 years' data (one year has 12 month data), the incremental data are the next following 10 years' data, and testing data are 12 months climate data in 2011. The parameters for ELM and OSELM related algorithms are set as: the number of input neuron is 25, the activation function is 'rbf', and the number of increment is 10, the number of each incremental data is 12. For BP, we set looping time 50 and number of hidden neuron 25. For SVR, we choose 'rbf' function and other parameters are selected by cross validation. RMSE is used to calculate the testing error and the result is shown in Fig. 8.

In Fig. 8(a), the proposed TOSELM has the smallest testing error. In Fig. 8(b), ELM and OSELM related algorithms still have the lowest time consumption, followed by SVR and BP. The analysis of performance is the same as stock price scenario. Therefore, for the climate timeliness problem, the proposed algorithm still achieves good performance both in accuracy and time consumption.

3.2.3. Gesture data

We also test the algorithm performance on our gesture recognition system. There are three different gestures: rock, paper, and scissors. The gesture data are captured by Kinect (Kinect is a motion sensing input device developed by Microsoft for the Xbox 360 video game console and Windows PCs). With time passing by, the Kinect captures gesture data from different users. There are 12 users (six male and six female; from age 20 to 30) giving three gestures in their own way, and finally we obtain 15,900 frames depth data: randomly choosing 10 users' 15,000 frame data (500

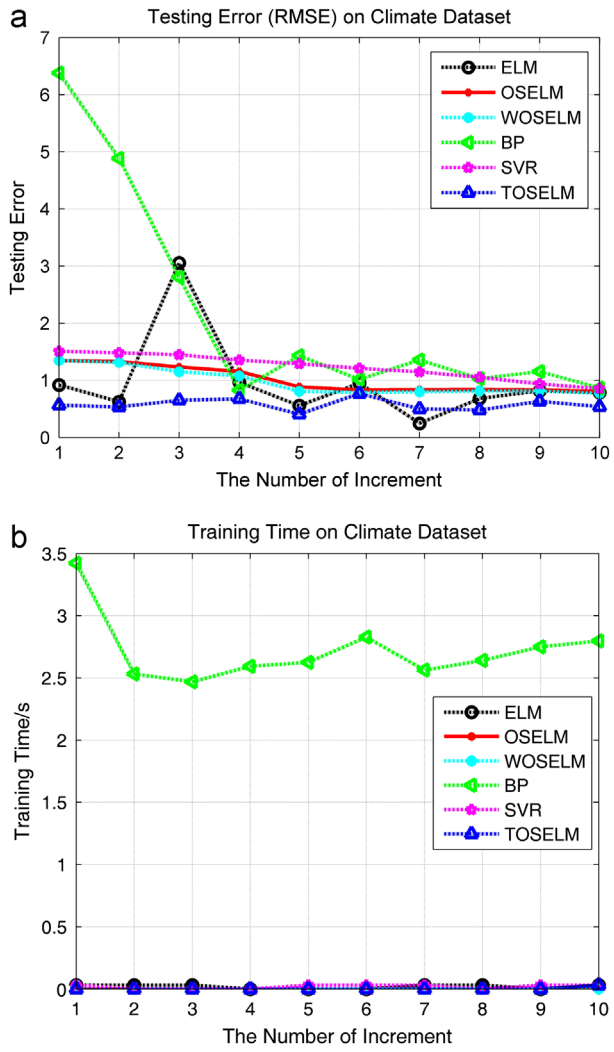

Fig. 8. Different algorithms' performance on climate data. (a) Testing error on climate data and (b) training time on climate data.

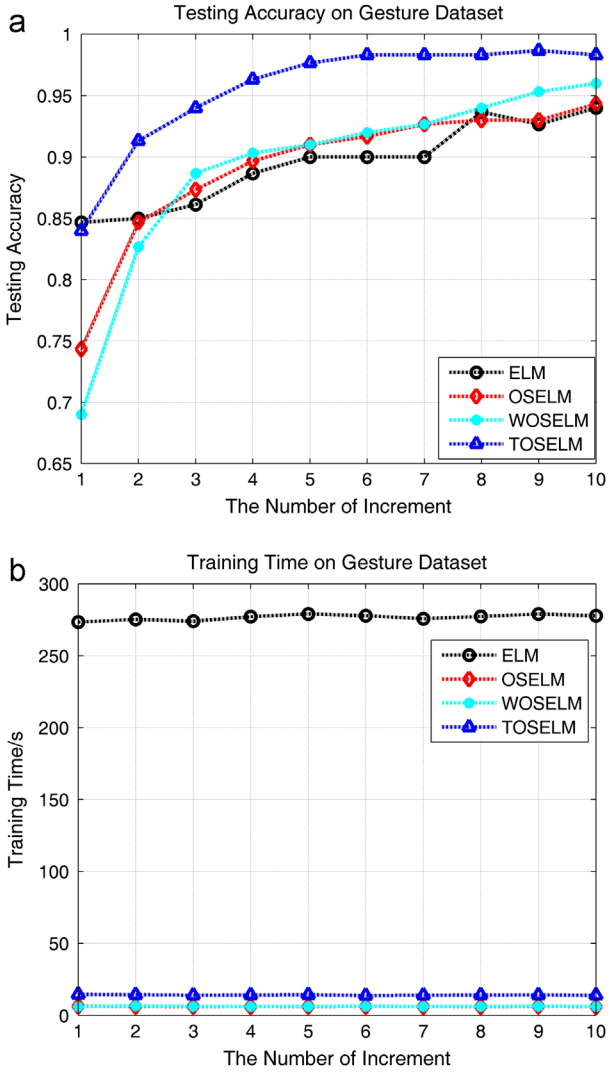

Fig. 9. Different algorithms' performance on gesture data (a) Testing Accuracy on gesture data and (b) training Time on gesture data.

frames for each gesture per user) as initial data and the rest 2 users' 900 data as incremental data and test data. Specifically, the number for 10 times' incremental data is 600; each time we get 60 frames incremental data, 20 for each gesture. The rest 300 frames data work as test data. In order to improve computing efficiency and recognition accuracy, the 3D point cloud data are converted into 3000 dimensions vectors by interpolation.

Due to the limitation of computer configuration, BP and SVM cannot execute successfully. Therefore, here we just compare the performance of ELM and OSELM derived algorithms. The parameters are set as follows: the number of hidden neurons is 3000; the activation function is sigmoid ('sig'). The recognition accuracy and training time are illustrated in Fig. 9.

From the figure we can see that for all these four algorithms, the proposed TOSELM achieves the best recognition accuracy, because when different users come into the system, the distribution of collected incremental data is always different from the history users', and the timeliness scheme of TOSELM can make the model adaptive to the new user. The accuracy of WOSELM is better than original OSELM as we expect due to the adaptive weight scheme. As to the performance of training time, since the dataset is large and the feature dimension is big, the difference of training time consumption is obvious. WOSELM has the same time consumption as OSELM, for the calculation of adaptive weight can be negligible. TOSELM costs more time than OSELM and WOSELM due to the iteration scheme. ELM is a batch learning method which needs to retrain the model with all the data, therefore its time consumption is obviously higher than the others'. Even though TOSELM costs more than OSELM and WOSELM, its accuracy is desirable; considering this tradeoff, TOSELM still outperforms others for gesture recognition problem.

4. Conclusion

In this paper we propose a sequential learning algorithm TOSELM under the framework of OSELM. Comparing with OSELM, TOSELM takes timeliness problem into consideration, and it has two improvements: one is the adaptive weight scheme; the other is the adaptive iteration scheme. The value of weight, reflecting the timeliness effect, is adaptively generated according to the differences between incremental data and history incremental data. The adaptive of iteration guarantees the stability and convergence of the system. The performances of TOSELM are tested both on simulated data as 'SinC' data and real world data like stock price data from Shanghai Index, climate data from China meteorological Data Sharing Service System and gesture data from interaction game system. All the results show that proposed algorithm has the least error and best accuracy. When the number of data and feature dimension are large, the proposed TOSELM consumes a little more time than the original OSELM for the iteration step, but it still spends less time than the batch learning method. Considering the tradeoff between time consumption and performance, TOSELM can settle the timeliness problem better than other methods.

Acknowledgment

This work is supported by NSFC General Program under Grant No. 61173066, Major Research Plan Project No. 90820303, and Guangdong Province Strategic Cooperation Project of the Chinese Academy of Sciences No. 2011A090100001.

References

[1] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. (CSUR) 34 (1) (2002) 1–47.
[2] F.M. Zanzotto, M. Pennacchiotti, A. Moschitti, A machine learning approach to textual entailment recognition, Nat. Lang. Eng. 15 (4) (2009) 551–582.
[3] Y. Wang, Y. Fan, P. Bhatt, C. Davatzikos, High-dimensional pattern regression using machine learning: From medical images to continuous clinical variables, NeuroImage 50 (4) (2010) 1519–1535.
[4] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the International Joint Conference on Neural Networks, 2004, pp. 985–990.
[5] G.B. Huang, C.K. Siew, Extreme learning machine: RBF network case, in: Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision (ICARCV), 2004, pp. 1029–1036.
[6] G.B. Huang, C.K. Siew, Extreme learning machine with randomly assigned RBF kernels, Int. J. Inf. Technol. 11 (1) (2005) 16–24.
[7] G.B. Huang, N. Ying Liang, H.J. Rong, P. Saratchandran, N. Sundararajan, On-line sequential extreme learning machine, in: Proceedings of the IASTED International Conference on Computational Intelligence, 2005.
[8] F.L. Chung, Z. Deng, S. Wang, From minimum enclosing ball to fast fuzzy inference system training on large datasets, IEEE Trans. Fuzzy Syst. 17 (1) (2009) 173–184.
[9] Y. Kim, K.A. Toh, Online AUC learning for biometric scores fusion, in: Proceedings of the 6th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2011, pp. 275–280.
[10] S. Liang, Z.X. Sun, Small sample incremental biased learning algorithm for sketch retrieval, J. Softw. 20 (5) (2009) 1301–1312. (in Chinese).
[11] C.X. Ren, D.Q. Dai, Incremental learning of bidirectional principal components for face recognition, Pattern Recognit. 43 (1) (2010) 318–330.
[12] M.V. Heeswijk, Y. Miche, T. Lindh-knuutila, P.A.J. Hilbers, T. Honkela, E. Oja, A. Lendasse, Adaptive ensemble models of extreme learning machines for time series prediction, Artif. Neural Netw. Lecture Notes Comput. Sci. 5769 (2009) 305–314.
[13] Li Zhang, W.D. Zhou, P.C. Chang, J.W. Yang, F.Z. Li, Iterated time series prediction with multiple support vector regression models, Neurocomputing 99 (2012) 411–422.
[14] Z. Xu, Q. Song, F. Haijin, D. Wang, Online prediction of time series data with recurrent kernels, in: Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN), 2012, pp. 1–7.
[15] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[16] N.Y. Liang, G.B. Huang, P. Saratchandran, N. Sundararajan, A fast and accurate on-line sequential learning algorithm for feedforward networks, IEEE Trans. Neural Netw. 17 (6) (2006) 1411–1423.
[17] ⟨http://ichart.yahoo.com/table.csv?s=600005.SS&a=00&b=01&c=2003&d=04&e=40&f=2012&g=d⟩.
[18] ⟨http://cdc.cma.gov.cn/⟩.

Yang Gu received the B.A.Sc. from Beijing University of Posts and Telecommunications, China in 2010. Currently, she is a Ph.D. student in the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). Her current research interests include machine learning, indoor localization and pervasive computing.

Junfa Liu received the Ph.D. degree in the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS) in 2009. Currently, he is a associate professor in ICT, CAS. His current research interests include machine learning, computational intelligence, human computer interaction and pervasive computing. He is a member of IEEE.

Yiqiang Chen received Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS), Beijing, China, in 2002. In 2004, he was a Visiting Scholar Researcher at the Department of Computer Science, Hong Kong University of Science and Technology (HKUST), Hong Kong. Currently, he is a professor and the director of the pervasive computing research center at ICT, CAS. His research interests include artificial intelligence, pervasive computing, and human computer interface. He is a member of IEEE.

Xinlong Jiang received the B.A.Sc. from Beijing University of Posts and Telecommunications, China in 2011. Currently, he is a master student in the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). Her current research interests include indoor localization and pervasive computing.

Hanchao Yu received M.A. degrees from Shandong Normal University, Jinan City, China, in 2011. Currently, he is a Ph.D. student in the Institute of Computing Technology, Chinese Academy of Sciences (ICT, CAS). His research interests include machine learning, human computer interaction and pervasive computing.