Cogn Comput (2014) 6:423–431
DOI 10.1007/s12559-014-9259-y

A Class Incremental Extreme Learning Machine for Activity Recognition

Zhongtang Zhao · Zhenyu Chen · Yiqiang Chen · Shuangquan Wang · Hongan Wang

Received: 12 August 2013 / Accepted: 17 March 2014 / Published online: 3 April 2014

Abstract Automatic activity recognition is an important problem in cognitive systems. Mobile phone-based activity recognition is an attractive research topic because it is unobtrusive. There are many activity recognition models that can infer a user's activity from sensor data. However, most of them lack class incremental learning abilities. That is, the trained models can only recognize activities that were included in the training phase, and new activities cannot be added in a follow-up phase. We propose a class incremental extreme learning machine (CIELM). It (1) builds an activity recognition model from labeled samples using an extreme learning machine algorithm without iterations; (2) adds new output nodes that correspond to new activities; and (3) only requires labeled samples of new activities and not previously used training data. We have tested the method using activity data. Our results demonstrated that the CIELM algorithm is stable and can achieve a similar recognition accuracy to the batch learning method.

Keywords Extreme learning machine · Incremental learning · Activity recognition · Mobile device

Z. Zhao
Zhengzhou Institute of Aeronautical Industry Management, Zhengzhou 450015, China

Z. Zhao (✉) · Z. Chen · Y. Chen · S. Wang
Pervasive Computing Center, Institute of Computing Technology, CAS, Beijing 100190, China
e-mail: zhaozhongtang@ict.ac.cn

Z. Chen
e-mail: chenzhenyu@ict.ac.cn

Y. Chen
e-mail: yqchen@ict.ac.cn

S. Wang
e-mail: wangshuangquan@ict.ac.cn

H. Wang
Institute of Software, CAS, Beijing 100190, China
e-mail: wha@iel.iscas.ac.cn

Introduction

Recently, the design and implementation of cognitive systems that can perceive, learn, memorize, decide, act, and communicate have attracted a lot of attention in both academia and industry [1, 2]. There are several important input modalities for natural interaction with intelligent systems. They include context-sensitive keyword [3], gaze [4], human speech and handwriting [5], and sensors [6]. Activity recognition in different kinds of cognitive systems [7, 8] plays a key role in diverse practical applications such as metabolic energy measurement and fall detection. Automatically recognizing various physical activities (e.g., standing still, walking, running, going upstairs or downstairs) has many applications in healthcare [9], monitoring of elderly people [10–12], energy expenditure [13], and so on [14, 15]. Because of their inherent properties, accelerometers are often used to measure movement for human activity recognition. In particular, smart phone-based activity recognition is a popular academic and industrial research topic that uses an embedded accelerometer sensor and has the obvious advantage of being unobtrusive to peoples' daily lives.

Most existing activity recognition methods collect and label samples of some predefined sort of activity to train a fixed activity recognition model. In [16, 17], the authors used multiple accelerometers to classify different activities. Chen [18] used a smart phone to detect six activities and

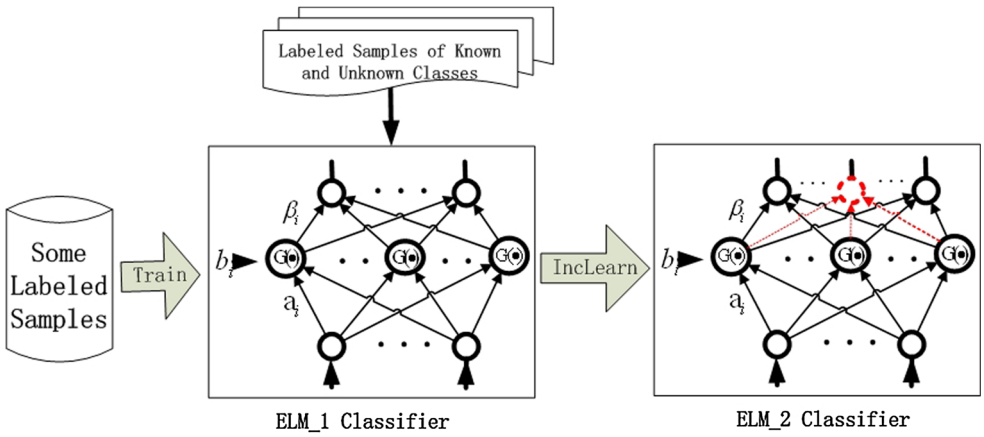

Fig. 1 CIELM framework

find the change point between different states. These models achieved high recognition accuracies, because their test and training samples were from the same data and followed the same distribution. These literatures did not consider the incremental activity recognition problem.

The fixed activity recognition model is not appropriate for meeting the diverse demands of different people. It is because that human beings learn motion activities as an incremental process. As a baby, we learn to sit, stand, and toddle. As a child, we learn to walk along the road, run back and forth, ride a bicycle, and so on. When we mature into adults, we may learn to how to drive a car. From the perspective of sports medicine, an individual's activities can reflect their physical condition. To address this problem, some existing methods retrain a new model by combining the original known activities with new activities [19, 20]. There are also some incremental learning methods. GAP-RBF [21] used feedforward networks with radial basis function (RBF) nodes. It pruned insignificant nodes from the networks. GAP-RBF attempted to simplify the sequential learning algorithms and increase learning speed. However, it requires information about the input sampling distribution or input sampling range, and the learning speed may still be slow for large applications. Fuzzy ARTMAP (FAM) [22] is a variant of an adaptive resonance theory map (ARTMAP) network, which is an incremental learning network based on adaptive resonance theory. It incorporates fuzzy set theory to govern the dynamics of ARTMAP. The major drawbacks of FAM are that it is sensitive to noise in the training dataset and often suffers from overfitting. This decreases its classification accuracy and generalization ability. The resource allocation network (RAN) [23] and its extensions are sequential learning algorithms that have also become popular feedforward networks. However, these sequential learning algorithms can only process one sample at a time, and not in larger sections. In [24], an online sequential extreme learning machine (OSELM) was introduced. OSELM can handle the training data in a sequential manner. At any time, only the newly arrived sample or a set of samples (instead of the entire past dataset) are used to train the incremental model. After the learning procedure, the current section of data is immediately discarded. However, RAN and OSELM cannot train using new classes of incremental data and integrate the results into an existing model.

It is not feasible to retrain a model on resource limited devices; they cannot store large amounts of training data and have low computing power. In this paper, we propose a class incremental extreme learning machine (CIELM). First, we train an ELM classifier using labeled samples of some predefined kinds of activities. Then, labeled samples of new activities are selected, and an incremental learning method is used to update the original ELM classifier.

The remainder of this paper is organized as follows. In "The Class Incremental Extreme Learning Machine for Activity Recognition" section, we present details of the CIELM. In "Experiments" section, and we give the results of our experiments. Section "Conclusion and Future Work" concludes the paper.

The Class Incremental Extreme Learning Machine for Activity Recognition

The proposed CIELM can be incrementally built in a similar manner to a human's cognitive processes. As illustrated in Fig. 1, CIELM has three main steps:

1. some activity samples are collected and labeled;
2. an ELM activity classifier is trained on these labeled samples (called ELM_1);
3. ELM_1 is increasingly adapted to a new classifier (ELM_2) using an incremental learning algorithm and labeled data of the known and new classes.

The ELM structure is changed after the adaption phase. Output neutrons that correspond to the new classes are added. They are represented by the dotted circles in Fig. 1.

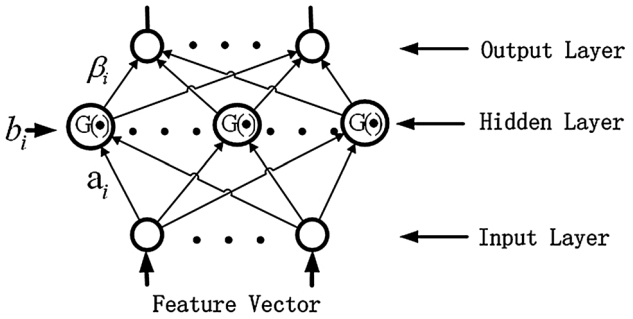

Fig. 2 ELM network structure

Brief Description of ELM

ELM is a recently developed neural network algorithm. It is known to perform well for complex problems and reduce computation time when compared with other machine learning algorithms. The ELM algorithm does not train the input weights or the biases of neurons. It acquires the output weights by using the least-squares solution and Moore–Penrose inverse of a general linear system. The final result is derived by selecting the maximum output value and its corresponding index.

Figure 2 shows the ELM network structure with a single hidden layer. The learning phase of the ELM algorithm can be summarized as in Algorithm 1.

Algorithm 1 Extreme Learning Machine

Input:

Given a training set, $\aleph = \{(x_i, t_i) | x_i \in \Re^n, t_i \in \Re^m, i = 1, 2, \cdots, N\}$, activation function, $G(a, b, x)$, and hidden node number, $\tilde{N}$.

Output:

The output weight β.

1: Assign random input weights a_i, and biases b_i, for $i = 1, 2, \cdots, \tilde{N}$;

2: Calculate the hidden layer output matrix:

$$H = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_N) \end{bmatrix}_{N \times \tilde{N}};$$

3: Calculate the output weight matrix, $\beta = H^{\dagger} T$, where $H^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $T = [t_1, t_2, \cdots, t_N]^T$.

In the offline phase, an ELM classifier can be trained using the labeled activity samples. After the classifier is deployed on a smart phone, it can be used to classify the users' activities in the online phase.

OSELM Algorithm

The ELM described in "Brief Description of ELM" section is a batch learning algorithm. It assumes that all the data are available for training. However, in real applications, the training data may arrive in sections or as individual samples. Therefore, the batch ELM algorithm must be modified so that it can deal with the online sequential problem [24].

Step 1 Given a chunk of the initial training set: $\aleph_0 = \{(x_i, t_i)\}_{i=1}^{N_0}$, where $x_i = \langle f_i^1, f_i^2, \ldots, f_i^n \rangle$, n is the feature number of the input data and also the input node number of the ELM. We select the activated function $G(a, b, x)$ and hidden node number $\tilde{N}$ and assign random input weights $a = \{a_i | a_i \in R^n\}_{i=1}^{\tilde{N}}$ and biases $b = \{b_i | b_i \in R\}_{i=1}^{\tilde{N}}$. The hidden layer output matrix, H_0, can be calculated using

$$H_0 = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_{N_0}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_{N_0}) \end{bmatrix}_{N_0 \times \tilde{N}}. \quad (1)$$

Then, the output weight β_0 can be calculated using

$$\beta_0 = K_0^{-1} H_0^{\mathrm{T}} T_0, \quad (2)$$

where $K_0 = H_0^{\mathrm{T}} H_0$, and $T_0 = [t_1, t_2, \ldots, t_{N_0}]^{\mathrm{T}}$.

Step 2 Suppose that we are given another chunk of data $\aleph_1 = \{(x_i, t_i)\}_{i=N_0+1}^{N_0+N_1}$. Then, using the parameters $a, b, G, \text{and} \tilde{N}$ in Step 1, H_1 can be calculated using

$$H_1 = \begin{bmatrix} G(a_1, b_1, x_{N_0+1}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_{N_0+1}) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_{N_0+N_1}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_{N_0+N_1}) \end{bmatrix}_{N_1 \times \tilde{N}}. \quad (3)$$

results:

$$K_1 = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} = \begin{bmatrix} H_0^{\mathrm{T}} & H_1^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} = K_0 + H_1^T H_1. \quad (4)$$

Therefore, if we combine the datasets $\aleph_0$ and $\aleph_1$ to train the ELM, β_1 can be calculated using

$$\beta_1 = K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} = \beta_0 + K_1^{-1} H_1^{\mathrm{T}} (T_1 - H_1 \beta_0). \quad (5)$$

As seen in Eq. (5), β_1 is a function of β_0, K_1, and H_1, and not a function of the dataset $\aleph_0$. Hence, we do not need to store the current data chunk after β has been calculated.

Class Incremental Extreme Learning Machine

As mentioned in "OSELM Algorithm" section, the OSELM model can be incrementally updated using individual samples or data chunks. The labels of the

incremental samples are the same as those in the training dataset. However, in reality, the user may form some new activities that the model needs to recognize. This case cannot be solved by OSELM. To address this problem, we extended OSELM so that it can be incrementally trained.

Suppose that we have a given labeled dataset $S = \{(x_s^{(i)}), (y_s^{(i)})|i = 1, 2, \ldots, N_0\}$ and a new dataset $D = \{(x_d^{(i)}), (y_d^{(i)})|i = 1, 2, \ldots, N_1\}$. Without loss of generality, we can simplify the description by assuming that there is only one new class label in D, and it is '$1 + \max\{(y_s^{(i)})|i = 1, 2, \ldots, N_0\}$'.

According to the ELM algorithm, after the training phase on the dataset of S is completed, the output weight can be calculated using

$$\beta_0 = K_0^{-1} H_0^{\mathrm{T}} T_0, \tag{6}$$

where

$$H_0 = \begin{bmatrix} G(a_1, b_1, x_s^{(1)}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_s^{(1)}) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_s^{(N_0)}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_s^{(N_0)}) \end{bmatrix}_{N_0 \times \tilde{N}}, \tag{7}$$

$$T_0 = \begin{bmatrix} t_1^{\mathrm{T}} \\ \vdots \\ t_{N_0}^{\mathrm{T}} \end{bmatrix}_{N_0 \times m}, \tag{8}$$

m is the number of sample classes in dataset S, and

$$K_0 = H_0^T H_0. \tag{9}$$

From the dataset of D and the function of the ELM algorithm, H_1 can be calculated using

$$H_1 = \begin{bmatrix} G(a_1, b_1, x_d^{(1)}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_d^{(1)}) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_d^{(N_1)}) & \cdots & G(a_{\tilde{N}}, b_{\tilde{N}}, x_d^{(N_1)}) \end{bmatrix}_{N_1 \times \tilde{N}}. \tag{10}$$

The labels of the samples in D can be represented by a multidimensional vector of dimension $m + 1$, where m is the column number of T_0.

$$T_1 = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 \end{bmatrix}_{N_1 \times (m+1)}. \tag{11}$$

If we combine datasets S and D to train ELM, β_1 can be calculated using

$$\beta_1 = K_1^{-1} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} T_0 \cdot M \\ T_1 \end{bmatrix}, \tag{12}$$

where

$$K_1 = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} = K_0 + H_1^{\mathrm{T}} H_1;$$

$$M = \begin{bmatrix} 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & 1 & 0 \end{bmatrix}_{m \times (m+1)}.$$

M is a transform matrix that adds one column of zeroes on the right of matrix T_0. Therefore, $T_0 \cdot M$ and T_1 have the same number of columns.

Using Eq. (12) we can derive another form of β_1:

$$\begin{aligned} \beta_1 &= \beta_0 M + K_1^{-1} H_1^{\mathrm{T}} (T_1 - H_1 \beta_0 M) \\ &= \beta_0 M + (K_0 + H_1^T H_1)^{-1} H_1^{\mathrm{T}} (T_1 - H_1 \beta_0 M). \end{aligned} \tag{13}$$

It can be seen from Eq. (13) that β_1 can be calculated without the dataset S and can be achieved through incremental training. By introducing matrix M, we have changed the ELM structure. This means that we have added one or more output neurons to the initial ELM structure.

Experiments

Class incremental learning has many applications in the activity recognition field. We incrementally learn to perform motion activities as we mature. Thus, we have used a class incremental extreme learning machine to enable the existing model to distinguish new activities using the small amount of storage and computing power available on a smart phone.

Sample Preparation

We have considered three benchmark problems for these classification experiments: activity data from our research group and two datasets (image segments and satellite images) from UCI.[1] We used the activity dataset to show that CIELM performs well for the activity recognition problem. The experiments on the UCI datasets demonstrated that CIELM can be used in other applications.

Activity Data Preparation

In our experiments, Nokia N95 8GB mobile phones were used to collect accelerometer data. An activity database was built from the collected data. There were 12 participants and 5 activities

[1] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/.

(standing still, walking, running, and going upstairs or downstairs). The data collection procedure was designed as follows:

1. Standing still for at least 1 min at the first floor of a 14-story building.
2. Climbing up the stairs for at least eight stories.
3. Standing still for at least 1 min.
4. Climbing downstairs to the first floor.
5. Standing still for at least 1 min.
6. Walking outside of the building to a playground.
7. Standing still for at least 1 min.
8. Running for at least 3 min.
9. Standing still for at least 1 min.

The N95 phone collected the data while in the subject's right trouser pocket. A sliding window with a 50 % overlap was used to extract the features. The sampling frequency of the N95 accelerometer sensor was set to approximately 32 Hz using the Nokia accelerometers plug-in API. Our chosen window size was two seconds, and the overlap time was one second. Thus, a complete activity could be included in the window. Previous work has demonstrated that feature extraction using windows with 50 % overlaps was successful [25].

An accelerometer detects changes in capacitance and transforms them into an analog output voltage that is proportional to acceleration. For a triaxial accelerometer, the output voltages can be mapped into acceleration along three axes, a_x, a_y, a_z. As a_x, a_y, a_z are the orthogonal decompositions of real acceleration, the magnitude of the synthesized acceleration is $a = \sqrt{a_x^2 + a_y^2 + a_z^2}$ [19], where a is the magnitude of the real acceleration with no directional information. Thus, the acceleration magnitude-based activity recognition model is orientation-independent.

Using a data series of acceleration magnitude, 16 statistical features [26] were extracted from a sliding window of 64 samples with a 50 % overlap between consecutive windows. These features were maximum, minimum, mean, standard deviation, mode, mean-crossing rate, range, signal magnitude area, four amplitude statistical features, and four shape statistical features of the power spectral density (PSD), as in [19, 27]. Additionally, using an FFT transformation of the series of acceleration magnitudes, all frequency components from 1Hz to 64Hz were extracted and added to the feature vector (a total of 80 features). To eliminate scaling effects, all the features were normalized using the z-score normalization algorithm.

The numbers of samples for each activity are listed in Table 1.

Table 1 Activity samples

Name	Label	Number of samples
Still	1	5,256
Walking	2	4,012
Running	3	3,756
Upstairs	4	1,656
Downstairs	5	1,611

Table 2 UCI datasets

Dataset	♯ Attributes	♯ Classes	♯ Data
Image segment	19	7	2,310
Satellite image	36	6	6,435

UCI Dataset Preparation

The image segmentation problem consists of a database of images drawn randomly from seven outdoor images. It consists of 2,310 regions of 3×3 pixels. The goal is classify each region as belonging to one of the seven classes using 19 extracted attributes.

The satellite image problem consists of a database of images generated from a landsat multispectral scanner. One frame of landsat multispectral scanner imagery consists of four digital images of the same scene in four different spectral bands. The database is a (tiny) subarea of a scene, consisting of 82×100 pixels. Each image in the database corresponds to a region of 3×3 pixels. The aim is to classify the central pixel in a region into one of six categories (red soil, cotton crop, gray soil, damp gray soil, soil with vegetation stubble, and very damp gray soil) using 36 spectral values for each region.

The datasets used in our experiments are described in Table 2. Before being used, they were normalized using the z-score method.

Experimental Results

We evaluated the CIELM's performance using the datasets described in Tables 1 and 2. We used four evaluations: model selection, CIELM's performance according to the incremental data chunk size, stability test for activity recognition, and a performance comparison between batch learning and the CIELM. All the simulations were implemented in MATLAB2009a running on an ordinary PC with 2.6 GHz CPU. The ELM source code was downloaded from Professor Huang's homepage.[2] We used the sigmoid activation function.

[2] Source codes and some references for ELM can be found at www.ntu.edu.sg/home/egbhuang.

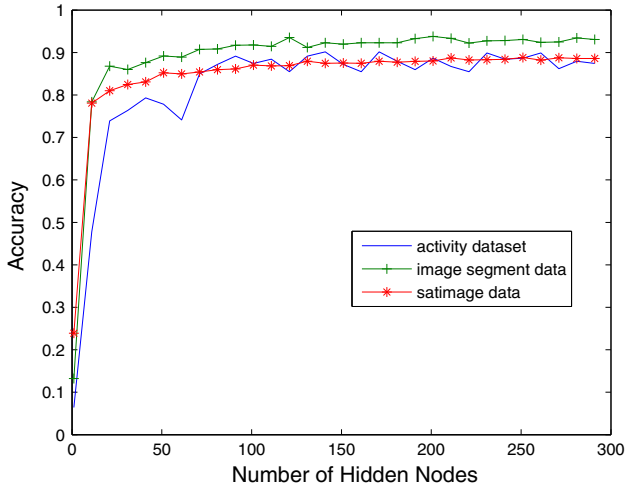

Fig. 3 Model selection

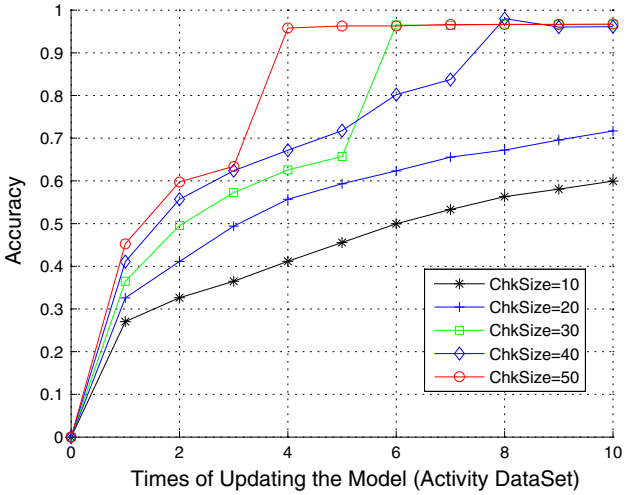

Fig. 4 CIELM's performance using the activity data

Model Selection

Model selection refers to the estimations of optimal architecture of the network and optimal parameters of the learning algorithm. It is problem-specific, and must be predetermined. For the CIELM, we only need to determine the optimal number of hidden nodes.

We used the training and validation methods to select the optimal number of hidden nodes. The total dataset was divided into two equal subsets such that the samples of each activity were divided into two equal subsets. We varied the number of hidden nodes between 1 and 300. The performance corresponding to each number is illustrated in Fig. 3.

As shown in Fig. 3, the recognition accuracy improved as the number of hidden nodes increased from 1 to 100. For the activity dataset, when the number of hidden nodes was larger than 100, the recognition accuracy remained stable with no significant changes. The same effect can be seen for the image segment and satellite image datasets when the number was larger than 50. Using more hidden nodes may achieve a high recognition accuracy, but there is an increased risk of overfitting. To achieve a trade-off between recognition accuracy and overfitting, we have used 100 hidden nodes for the activity recognition problem and 50 for the two UCI problems.

Change in CIELM's Performance According to Incremental Data Chunk Size

In this experiment, we evaluated CIELM's performance according to the incremental data chunk size. As our activity data were collected from 12 objects, we repeated our experiment for 12 times (one for each object). Each trial had two steps. First, the activity data for standing still, walking, running, and going upstairs were used to train the model M1. Then, the going downstairs data were divided into two sections (part_a and part_b). part_a was divided into equal size chunks and used to incrementally update M1. part_b was used to evaluate the new model after each dataset was added to the model. To observe the relationship between model performance and chunk size, we varied the size between 1 and 50 ("ChkSize" in Fig. 4). We took the average values of the 12 trials.

Three conclusions can be made from Fig. 4. First, the activity recognition accuracy increased with the number of model updates, and this situation did not change with varying chunk size. Second, a larger chunk size resulted in a better recognition result. Third, more updates resulted in a better recognition result.

We can conclude that when the chunk size was 50, the model accuracy gradually increased after four update procedures.

We repeated this experiment using the satellite image and image segmentation datasets. The experiment results are illustrated in Figs. 5 and 6.

For the satellite image data, when the chunk size was 60, 80 % accuracy was achieved after 5 update procedures. For the image segmentation data, when the chunk size was 6, 90 % accuracy was achieved after 4 update procedures.

Based on the results of these three experiments, we can conclude that the CIELM algorithm's performance is problem-related. To optimize the model, we should predetermine the chunk size and number of updates.

Stability Test

In this section, we evaluated the stability of the CIELM. We wished to determine whether it is new class-independent. We wish to make each of the five kinds of activities

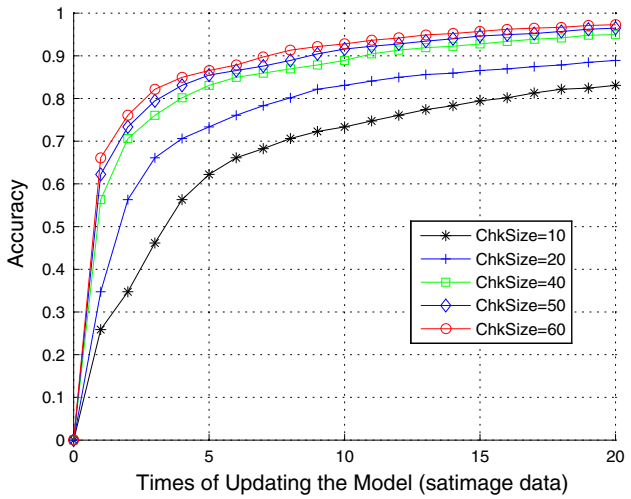

Fig. 5 CIELM's performance using the satellite image data

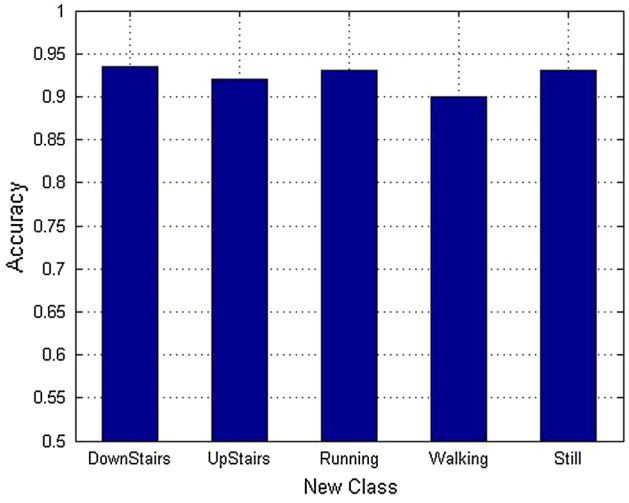

Fig. 7 Stability test

(A, B, C, D, and E) a new class. Without loss of generality, we can make A the new class. Samples of activities B, C, D, and E were used to train the model. Samples of activity A were divided into two parts: One part was used to incrementally train the new model, and the other was used to test it. For each subject, we used five trials. The process was repeated for twelve subjects. The average results are shown in Fig. 7.

It can be observed from Fig. 7 that the recognition accuracy for the new class samples was more than 90 %, regardless of the activity that was considered to be the new class. Thus, we have demonstrated that CIELM is a persuasive method for activity recognition and is not sensitive to the specific new class. When deployed on a smart mobile terminal that has limited memory, computation, and power resources, a personalized activity recognition model can be achieved through incremental learning.

Batch Learning Compared with CIELM

In the following experiment, we compared the performances of the batch learning and CIELM methods.

We can list the permutations of an activity set Act = {Still, Walking, Running, Upstairs, Downstairs}. For each subject, every activity permutation can be denoted Per = {A,B,C,D,E}. Samples of the jth(j $= 1, 2, \ldots, 5$) activity were divided into two equal parts. To simplify the description, we represent them using TR_j and TE_j.

1. Batch learning: For every permutation, the batch learning validation process consists of four steps. At the first step, an ELM classifier is trained using the $TR_1 \bigcup TR_2$ dataset and tested using $TE_1 \bigcup TE_2$. At the $k^{\text{th}}(k = 2, 3, 4)$ step, an ELM classifier is trained using $TR_1 \bigcup TR_2 \bigcup \cdots \bigcup TR_{k+1}$ and tested using $TE_1 \bigcup TE_2 \bigcup \cdots \bigcup TE_{k+1}$. The above process is repeated for all subjects and all permutations,

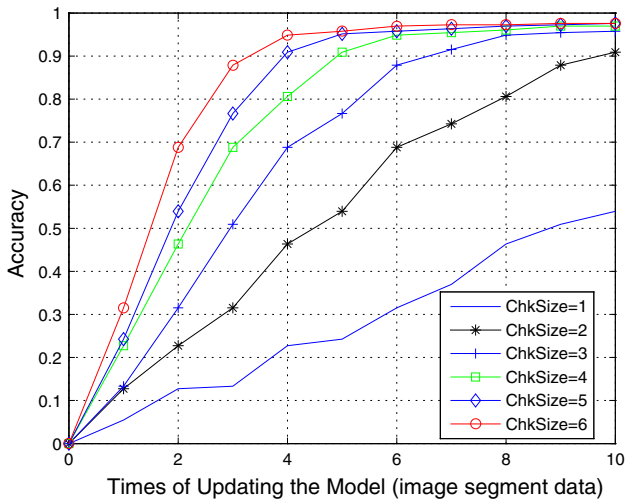

Fig. 6 CIELM's performance using the image segmentation data

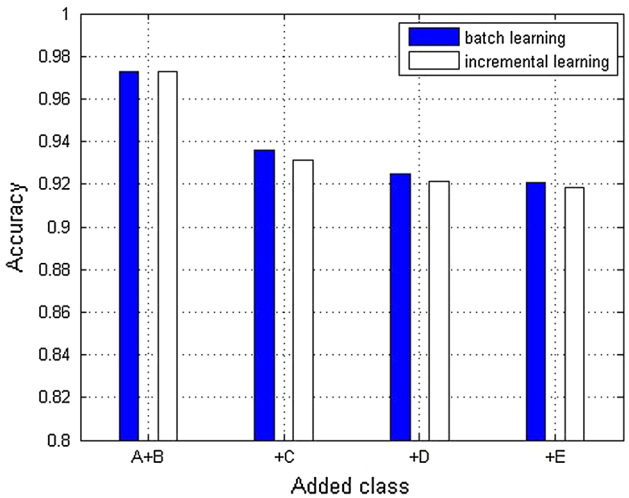

Fig. 8 Batch learning compared with CIELM

Table 3 Computational complexity comparisons

Method	Peak_Memory (KB)	Total_Time (s)
Batch learning	67,780.00	17.062
CIELM	37,108.00	3.016

and the average is taken as the result. The results using the batch learning method are shown in Fig. 8.

2. CIELM: As with batch learning, the incremental learning validation process consists of four steps. At the first step, an ELM classifier is trained using the $TR_1 \bigcup TR_2$ dataset and tested using $TE_1 \bigcup TE_2$. At the $k^{th}(k = 2, 3, 4)$ step, the model trained from the $(k-1)$th step is adapted by our proposed incremental learning method using samples from TR_{k+1} and tested using $TE_1 \bigcup TE_2 \bigcup \cdots \bigcup TE_{k+1}$. The above process is repeated for all subjects and all permutations, and the average is taken as the result. The results of the incremental learning method are shown in Fig. 8.

It can be seen from Fig. 8 that the batch learning and CI-ELM methods performed similarly when new classes were added to the model. When samples of the third activity class were used, the accuracy decreased from more than 96 % to less than 94 %. However, when the fourth and fifth activity classes were added, the recognition performance decreased very slowly.

We can also see that the CIELM performs slightly worse than the batch learning method. This is because batch learning can learn more because it has a larger training set than CIELM.

To evaluate the computational complexity of these two algorithms, we considered the peak memory and total execution time. We used a profile viewer to investigate the two methods. The related peak memory and total execution times are listed in Table 3.

As seen in Table 3, the CIELM algorithm uses less memory and consumes less CPU time than the batch learning algorithm.

Conclusion and Future Work

This paper proposed the CIELM algorithm to address the class-related incremental learning problem. Our empirical evaluations have shown the following. First, the CIELM model can be used to learn new classes, changing the structure of an existing ELM model by adding output nodes to the network. Second, the number of added samples can affect the performance of the classifier. Third, CIELM's performance is slightly worse than the batch learning method, which indicates that there is a trade-off between the optimal model and restricted resources.

Acknowledgments This work was supported in part by the Natural Science Foundation of China (61070110, 90820303), Beijing Natural Science Foundation (4112056, 4144085), Open Project of Beijing Key Laboratory of Mobile Computing and Pervasive Device, National Science and Technology Major Project (2012ZX07205-005), and Scientific and Technological Project of He'nan Province (No. 132102310258).

References

1. Taylor JG. Cognitive computation. Cogn Comput. 2009;1(1):4–16.
2. Cambria E, Hussain A. Sentic computing: techniques, tools, and applications. Springerbriefs in cognitive computation. Dordrecht: springer; 2012.
3. 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.
4. Mital P, Smith T, Hill R, Henderson J. Clustering of gaze during dynamic scene viewing is predicted by motion. Cogn Comput. 2011;3(1):5–24.
5. Wang QF, Cambria E, Liu CL, Hussain A. Common sense knowledge for handwritten Chinese recognition. Cogn Comput. 2013;5(2):234–42.
6. Roggen D, Magnenat S, Waibel M, Troster G. Designing and sharing activity-recognition systems across platforms. IEEE Robot Autom Mag. 2011;18:83–95.
7. Chen YQ, Chen ZY, Liu JF, Hu Derek H, Yang Q. Surrounding context and episode awareness using dynamic Bluetooth data. Pittsburgh, PA: UbiComp; 2012. p. 629–630.
8. Chen ZY, Chen YQ, Wang SQ, Liu JF, Gao XY, Campbell AT. Inferring social contextual behavior from Bluetooth traces. Zurich, Switzerland: UbiComp; 2013. p. 267–270.
9. Brajdic A, Harle R. Walk detection and step counting on unconstrained smartphones. In: UbiComp'1; 2013. p. 225–234.
10. Chen ZY, Lin M, Chen FL, Lane ND, Cardone G, Wang R, Li TX, Chen YQ, Choudhury T, Campbell AT. Unobtrusive sleep monitoring using smartphones. In: PervasiveHealth' 2013, p. 145–152.
11. Lane ND, Xu Y, Lu H, Hu SH, Choudhury T, Campbell AT. Enabling large-scale human activity inference on smartphones using community similarity. Networks (CSN). UbiComp; 2011, p. 355–364.
12. Lane ND, Xu Y, Lu H, Eisenmany SB, Choudhury T, Campbell AT. Cooperative communities (CoCo): exploiting social networks for large-scale modeling of human behavior. IEEE Pervasive Mag. 2011;10(4):45–53.
13. Chen S, Lach J, Amft O, Altini M, Penders J. Unsupervised activity clustering to estimate energy expenditure with a single body sensor. In: BSN'13; 2013.
14. Stanford V. Wearable computing goes live in industry. IEEE Pervasive Comput Mag. 2002;1(4):14–9.
15. Nachman L, Baxi A, Bhattacharya S, Darera V, Deshpande P, Kodalapura N, Mageshkumar V, Rath S, Shahabdeen J, Acharya R. Jog falls: a pervasive healthcare platform for diabetes management. In: Proceedings of the pervasive; 2010. p. 94–111.
16. Ravi N, Dandekar N, Mysore P, Littman ML. Activity recognition from accelerometer data. In: Proceedings of AAAI, 2005. pp. 1541–1546.
17. Ward Jamie A, Lukowicz Paul, Gellersen Hans W. Performance metrics for activity recognition. ACM Trans Intell Syst Technol 2011;2(1). doi:10.1145/1889681.1889687.
18. Chen YQ, Qi J, Sun Z, Ning Q. Mining user goals for indoor location based services with low energy and high qos. Comput Intell. 2010;26(3):318–36.

19. Chen YQ, Zhao ZT, Wang SQ, Chen ZY. Extreme learning machine based device displacement free activity recognition model. Soft Comput. 2012;16(9):1617–25.
20. Zhao ZT, Chen YQ, Liu JF, Shen ZQ, Liu MJ. Cross-people mobile-phone based activity recognition. In: Proceedings of the international joint conference on artificial intelligence(IJCAI2011); 2011. Barcelona, Spain, July 16–22.
21. Huang GB, Saratchandran P, Sundararajan N. An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern B Cybern. 2004;34(6):2284–92.
22. Carpenter GA, Grossberg S, Rosen D. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Netw. 1991;4:759–71.
23. Platt J. A resource-allocating network for function interpolation. Neural Comput. 1991;3:213–25.
24. Liang NY, Huang GB, Saratchandran P, Sundararajan N. A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw. 2006;17:1411–23.
25. Bao L, Intille S. Activity recognition from user annotated acceleration data. In: Proceedings of the 2nd international conference on pervasive computing; 2004. pp. 1–17.
26. Figo D, Diniz PC, Ferreira DR, Cardoso JMP. Preprocessing techniques for context recognition from accelerometer data. Pers Ubiquit Comput. 2010;14:645–62.
27. Chen ZY, Zhao ZT, Wang SQ, Shen ZQ, Chen YQ. Online sequential ELM based transfer learning for transportation mode recognition. In: The 6th IEEE international conference on cybernetics and intelligent systems (CIS 2013); 2013. Manila, Philippines.