Int. J. Mach. Learn. & Cyber.
DOI 10.1007/s13042-017-0666-8

ORIGINAL ARTICLE

# OKRELM: online kernelized and regularized extreme learning machine for wearable-based activity recognition

**Lisha Hu**[1,2,3] · **Yiqiang Chen**[1,2,3] · **Jindong Wang**[1,2,3] · **Chunyu Hu**[1,2,3] · **Xinlong Jiang**[1,2,3]

Received: 30 September 2016 / Accepted: 27 March 2017


**Abstract** Miscellaneous mini-wearable devices (Jawbone Up, Apple Watch, Google Glass, et al.) have emerged in recent years to recognize the user's activities of daily living (ADLs) such as walking, running, climbing and bicycling. To better suits a target user, a generic activity recognition (AR) model inside the wearable devices requires to adapt itself according to the user's personality in terms of wearing styles and so on. In this paper, an online kernelized and regularized extreme learning machine (OKRELM) is proposed for wearable-based activity recognition. A small-scale but important subset of every incoming data chunk is chosen to go through the update stage during the online sequential learning. Therefore, OKRELM is a lightweight incremental learning model with less time consumption during the update and prediction phase, a robust and effective classifier compared with the batch learning scheme. The performance of OKRELM is evaluated and compared with several related approaches on a UCI online available AR dataset and experimental results show the efficiency and effectiveness of OKRELM.

**Keywords** Extreme learning machine · Kernel · Activity recognition · Online learning · Wearable computing

✉ Yiqiang Chen
yqchen@ict.ac.cn

Lisha Hu
hulisha@ict.ac.cn

Jindong Wang
wangjindong@ict.ac.cn

Chunyu Hu
huchunyu@ict.ac.cn

Xinlong Jiang
jiangxinlong@ict.ac.cn

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

[2] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing, China

[3] University of Chinese Academy of Sciences, Beijing, China

## 1 Introduction

Human activity recognition (AR) techniques promote the development of large amounts of meaningful applications such as context awareness [1, 2], energy expenditure [3], disease detection [4] and personal healthcare [5]. Moreover, with the development of wearable techniques in recent years, diverse of sensors (accelerometer, gyroscope, et al.) are embedded into the mini-wearable devices (e.g. smartwatch [6], wristband [7], armband [8], head-belt [9, 10]). Consequently, wearable AR techniques are widely employed to improve the users' health conditions by collecting and analyzing their data of activities of daily living (ADLs), and then giving them feedback.

Wearable AR technologies grow extremely fast and a great deal of work has been proposed [11, 12]. Owing to the limited computation and storage resources of the wearable devices, a wearable AR model ought to be lightweight with reduced computation complexity [13]. In some real applications, real-time feedback is greatly important and necessary. For instance, a jogger might see how many steps he has made when he is running, then decides whether to continue running or not. The AR model inside a wearable device should fulfill the recognition task in real time. To do this, a great number of machine learning algorithms (Decision Tree [14], Support Vector Machine [15, 16], Extreme Learning Machine [17], Dynamic Bayesian Network [18], Hidden Markov Models [19], Boosting [20], etc.) and a

Published online: 09 April 2017

couple of improved methods [21–26] have been proposed, then employed for a large number of application fields [27–32] and specially for wearable AR.

Among them, ELM is widely used owing to its good performance and extremely fast learning speed. ELM [33–36] is proposed as a single hidden layer feedforward neural network, in which the hidden layer doesn't need to be tuned and the output weight is computed by the Moore–Penrose generalized inverse. Due to its simple implementation and universal approximation capabilities, ELM has been widely used in multiple areas [37]. Recently the regularized extreme learning machine (RELM) [38–40] is proposed as a unified framework for binary, multiclass classification and regression problems. RELM is superior to ELM with more improved and stable generalization performance. Besides, RELM incorporated with kernels can be beneficial in contexts when an appropriate and explicit hidden layer mapping function is either unknown or difficult to find out. Recently, a new reduced extreme kernel sparse learning methodology is proposed in [41] for tactile object recognition to deal with the dictionary learning and the classifier learning simultaneously. Besides, authors in [41] developed a reduced kernel dictionary learning method to tackle the large storage requirement problems. Therefore, kernelized RELM (RELM with kernels) is attracting more and more researchers' attention.

The kernelized RELM deals with the batch learning problem in which the training data comes all together. Generally, such a generic and static AR model may not well fit for a specific user with distinctive personalities in terms of wearing styles and ADLs [42]. For example, an AR model learned based on the data from the dominant wrist (e.g. right wrist) may not work well for a target user wearing the device on his/her non-dominant wrist (e.g. left wrist), or having an opposite dominant wrist. Based on the incremental data of a target user, a generic model can adjust itself to be a personalized model according to the online learning mechanism.

Online learning is much relevant to the dynamic sequence recognition since both of them deal with the dynamic and time series signals. Besides, considering the fact that time series do not usually lie in the Euclidean space, conventional applications relevant to vector spaces (e.g. sparse coding [43]) may not work well in such applications. Fortunately, kernel trick is widely employed to address this problem. For instance, a joint kernel sparse coding method fuses the tactile sequences from separate fingers with a Gaussian DTW kernel [44] and a joint group kernel sparse coding method deals with the multivariate-time-series fusing both the tactile and visual data [45]. In the online learning scenario when training data comes one by one or chunk by chunk, online sequential extreme learning machine (OSELM) [46] is proposed based on ELM, and multiple online learning models have been proposed based on the kernelized RELM. For instance: OS-ELMK [47], FOKELM [48] and CF-FOKELM [49] have been proposed for time series prediction; KOS-ELM [50] is for binary classification and regression; KB-IELM [52] is for binary, multiclass classification as well as regression problems.

Since all the training data of KB-IELM [52] have to be saved after the training stage and used again during the prediction. It is greatly different from OSELM in that all the training data will be discarded once the training stage is finished. As a consequence, the memory requirement grows incrementally along with the online learning stages, which may not be affordable for the wearable devices. A decremental algorithm is developed for time series prediction to remove the oldest training data according to fixed memory schemes [47–49]. TransRKELM randomly chooses a subset of initial data to generate the initial model, and update with a subset of high confidential labeled incremental data during the multiclass classification problem [51]. For binary classification and regression problem, KOS-ELM [50] rejects to process an incremental data if its distance to the linear span of all the existing training data is less than a predefined threshold, and removes the least important data from the training dataset.

In this paper, we propose an online kernelized and regularized extreme learning machine (OKRELM) for AR with mini-wearable devices. Contributions of this paper are as follows:

- OKRELM is a lightweight and robust classification model compared with related methods since it uses a few and important data from each data chunk to update the model in the online training stage.
- OKRELM is able to deal with both the binary and multiclass classification problems.
- OKRELM shows its superiorities in dealing with wearable AR problems. We validate OKRELM on a UCI online activity recognition dataset. Experimental results show the efficiency and effectiveness of OKRELM in AR.

The rest of the paper is organized as follows. Section 2 gives the review of some related work. In Sect. 3, we elaborate the motivations and the OKRELM is proposed. The performance of OKRELM is validated in Sect. 4. At last, we conclude the paper and discuss some future extensions.

## 2 Related work

### 2.1 Regularized extreme learning machine (RELM)

Regularized extreme learning machine (RELM) is a single hidden layer feedforward neural network, which is

proposed as a unified framework for binary, multiclass classification and regression problems [38–40]. In this paper, we only focus on the classification problems using the kernelized RELM (RELM with kernels).

Let the training dataset be $\{(\boldsymbol{x}_i, \boldsymbol{t}_i) | \boldsymbol{x}_i \in \mathbf{R}^d, \boldsymbol{t}_i \in \boldsymbol{R}^m, i = 1, \ldots, N\}$, in which instance $\boldsymbol{x}_i$ is a $d$-dimensional feature vector and $\boldsymbol{t}_i$ is the label of $\boldsymbol{x}_i$. Within each label vector $\boldsymbol{t}_i = (t_{i1}, \ldots, t_{im})^T$, one single element $t_{ik}$ equals 1 representing the instance $\boldsymbol{x}_i$ belonging to class $k$, and the other elements $(t_{ij}, j \neq k)$ equal $-1$. $m$, $N$ represent the number of classes and instances, respectively.

The optimization problem of RELM is presented in Eq. (1). $\boldsymbol{\beta}$ is a $L \times m$ matrix to be solved. $\boldsymbol{\xi}$ is a $m \times N$ dimensional slack variable in which the $i$th column of $\boldsymbol{\xi}$, notated by $\boldsymbol{\xi}_{:,i}$, is the training error of the instance $\boldsymbol{x}_i$. $\boldsymbol{h}(\boldsymbol{x})$ maps $\boldsymbol{x}$ into some higher but unknown Hibert feature space using the kernel trick. C is the penalty parameter balancing the maximum generalization ability $1/2\left\|\boldsymbol{\beta}^2\right\|$ and minimum training error $1/2\sum_{i=1}^{N}\left\|\boldsymbol{\xi}_{:,i}\right\|^2$.

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{\xi}} \frac{1}{2}\boldsymbol{\beta}^2 + \frac{\mathrm{C}}{2}\sum_{i=1}^{N}{\boldsymbol{\xi}_{:,i}}^2 \\ &s.t. \boldsymbol{\beta}^{\mathrm{T}} \cdot \boldsymbol{h}\left(\boldsymbol{x}_i\right) = \boldsymbol{t}_i - \boldsymbol{\xi}_{:,i}, i = 1, \ldots, N. \end{aligned} \tag{1}$$

Based on the Karush–Kuhn–Tucker (KKT) Theorem, to solve the problem in Eq. (1) is equivalent to solve its dual problem. The objective function of its dual problem is in Eq. (2). $\boldsymbol{\alpha}$ is a $N \times m$ matrix in which the $i$th column of $\boldsymbol{\alpha}$ is the Lagrange multiplier of the instance $\boldsymbol{x}_i$. we call $\boldsymbol{\alpha}$ the Lagrange matrix.

$$D = \frac{1}{2}\boldsymbol{\beta}^2 + \frac{\mathrm{C}}{2}\sum_{i=1}^{N}{\boldsymbol{\xi}_{:,i}}^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\alpha_{i,j}\left(\boldsymbol{\beta}_{:,j}^T\boldsymbol{h}\left(\boldsymbol{x}_i\right) - \boldsymbol{t}_{i,j} + \boldsymbol{\xi}_{j,i}\right). \tag{2}$$

Let the partial derivative of $D$ with respect to all the variables be 0, we have the following equations. $\mathbf{H} = \left[\boldsymbol{h}\left(\boldsymbol{x}_1\right), \ldots, \boldsymbol{h}\left(\boldsymbol{x}_N\right)\right]^T$ is a matrix of $N$ rows.

$$\frac{\partial D}{\partial \boldsymbol{\beta}_{:,j}} = \boldsymbol{\beta}_{:,j} - \sum_{i=1}^{N}\boldsymbol{\alpha}_{i,j}\boldsymbol{h}\left(\boldsymbol{x}_i\right) = 0, \forall j \rightarrow \boldsymbol{\beta} = \mathbf{H}^T\boldsymbol{\alpha}, \tag{3a}$$

$$\begin{aligned} &\frac{\partial D}{\partial \boldsymbol{\xi}_{j,i}} = \mathrm{C}\boldsymbol{\xi}_{j,i} - \boldsymbol{\alpha}_{i,j} = 0, \forall i, j \rightarrow \boldsymbol{\xi}_i = \frac{1}{\mathrm{C}}\boldsymbol{\alpha}_{i,:}^T, \\ &\forall i \rightarrow \boldsymbol{\xi} = \frac{1}{\mathrm{C}}\boldsymbol{\alpha}^T \rightarrow \boldsymbol{\xi}^{\mathrm{T}} = \frac{1}{\mathrm{C}}\boldsymbol{\alpha}, \end{aligned} \tag{3b}$$

$$\frac{\partial D}{\partial \boldsymbol{\alpha}_{i,j}} = \boldsymbol{\beta}_{:,j}^T\boldsymbol{h}\left(\boldsymbol{x}_i\right) - \boldsymbol{t}_{i,j} + \boldsymbol{\xi}_{j,i} = 0, \forall i, j \rightarrow \boldsymbol{\beta}^T\boldsymbol{h}\left(\boldsymbol{x}_i\right) - \boldsymbol{t}_i + \boldsymbol{\xi}_i = 0, \forall i, \tag{3c}$$

Equation (3a) and (3b) are substituted in Eq. (3c), we can get Eq. (4). $\mathbf{T} = \left[\boldsymbol{t}_1, \ldots, \boldsymbol{t}_N\right]^T$ is a $N \times m$ matrix. $\mathbf{I}_{N\times N}$ represents an $N \times N$ identity matrix.

$$\boldsymbol{\alpha} = \left(\mathbf{H}\mathbf{H}^T + \frac{1}{\mathrm{C}}\mathbf{I}_{N\times N}\right)^{-1}\mathbf{T}. \tag{4}$$

Equation (4) is substituted in Eq. (3a), we attain:

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T + \frac{1}{\mathrm{C}}\mathbf{I}_{N\times N}\right)^{-1}\mathbf{T}. \tag{5}$$

At last, the output function of RELM is:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{T}^T\left(\mathbf{H}\mathbf{H}^T + \frac{1}{\mathrm{C}}\mathbf{I}_{N\times N}\right)^{-1}\mathbf{H}\boldsymbol{h}(\boldsymbol{x}). \tag{6}$$

For any testing instance $\boldsymbol{x}$, $\boldsymbol{f}(\boldsymbol{x})$ is a $m \times 1$ vector. The prediction of RELM on $\boldsymbol{x}$ is in Eq. (7). $f_k(\boldsymbol{x})$ is the $k$th element of $\boldsymbol{f}(\boldsymbol{x})$.

$$\text{Prediction}(\boldsymbol{x}) = \underset{k\in\{1,\ldots,m\}}{\operatorname{argmax}} f_k(\boldsymbol{x}). \tag{7}$$

In Eq. (6), replace $\mathbf{H}\mathbf{H}^T$ by the Gram matrix $\boldsymbol{\Omega}$ ($\boldsymbol{\Omega}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, \ldots, N$) of a kernel $k(\boldsymbol{u}, \boldsymbol{v})$ and $\mathbf{H}\boldsymbol{h}(\boldsymbol{x})$ by the kernel form, we have the output function of kernel based RELM:

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{T}^T\left(\boldsymbol{\Omega} + \frac{1}{\mathrm{C}}\mathbf{I}_{N\times N}\right)^{-1}\left[k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_N, \boldsymbol{x})\right]^T. \tag{8}$$

The kernelized RELM is a batch learning model. It can only deal with the problem that all the training data is available before training process begins. A generic RELM model could be learned offline based on the data from several people. The generic model requires update to better suit a single target user of the wearable device. Moreover, the update process needs to be done using online learning with the training data coming one-by-one or chunk-by-chunk. In the following section, we review KB-IELM, one of the online learning model proposed based on the kernelized RELM.

### 2.2 Kernel based incremental extreme learning machine (KB-IELM)

Based on the kernelized RELM, Kernel based incremental extreme learning machine (KB-IELM) is proposed to deal with the problem of training data coming one-by-one or chunk-by-chunk [52]. Two stages (which we call the initial and online stage) are contained in the online learning of KB-IELM. Here we introduce these two stages.

- In the initial stage, supposing that the initial data chunk $(\mathbf{X}_0, \mathbf{T}_0)$ contains $N_0$ training instances, $\mathbf{X}_0 = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_0}]^T, \mathbf{T}_0 = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_{N_0}]^T$. The Lagrange matrix $\boldsymbol{\alpha}^{(0)}$ and the output function $\boldsymbol{f}_0(\boldsymbol{x})$ are in

Eqs. (9) and (10). $\mathbf{\Omega}_0$ represents the $N_0 \times N_0$ Gram matrix with respect to the chunk $(\mathbf{X}_0, \mathbf{T}_0)$.

$$\boldsymbol{\alpha}^{(0)} = \left(\mathbf{\Omega}_0 + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0}\right)^{-1}\mathbf{T}_0, \tag{9}$$

$$\boldsymbol{f}_0(\boldsymbol{x}) = \left(\boldsymbol{\alpha}^{(0)}\right)^T \left[k(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}_{N_0}, \boldsymbol{x})\right]^T. \tag{10}$$

- In the online stage, supposing that the first data chunk $(\boldsymbol{X}_1, \mathbf{T}_1)$ contains $N_1$ training instances, $\mathbf{X}_1 = \left[\boldsymbol{x}_{N_0+1}, \ldots, \boldsymbol{x}_{N_0+N_1}\right]^T, \mathbf{T}_1 = \left[\boldsymbol{t}_{N_0+1}, \ldots, \boldsymbol{t}_{N_0+N_1}\right]^T$. The Lagrange matrix $\boldsymbol{\alpha}^{(1)}$ becomes:

$$\boldsymbol{\alpha}^{(1)} = \begin{bmatrix} \boldsymbol{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}, \tag{11}$$

where as

$$\begin{cases} \mathbf{A}_{11} = \mathbf{\Omega}_0 + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0} \\ \mathbf{A}_{12} = \mathbf{\Omega}_{01} \\ \mathbf{A}_{22} = \mathbf{\Omega}_1 + \frac{1}{\mathrm{C}}\mathbf{I}_{N_1 \times N_1} \end{cases}, \tag{12}$$

$\mathbf{\Omega}_1$ represents the $N_1 \times N_1$ Gram matrix with respect to the chunk $(\mathbf{X}_1, \mathbf{T}_1)$. $\mathbf{\Omega}_{01}$ represents a $N_0 \times N_1$ matrix in which $\left(\mathbf{\Omega}_{01}\right)_{ij} = k\left(\boldsymbol{x}_i, \boldsymbol{x}_{N_0+j}\right), i = 1, \ldots, N_0, j = 1, \ldots, N_1$. According to the block matrix inverse formula [53], The Lagrange matrix $\boldsymbol{\alpha}^{(1)}$ is calculated by:

$$\boldsymbol{\alpha}^{(1)} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{12}^T & \mathbf{B}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}, \tag{13}$$

where as

$$\mathbf{B}_{11} = \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}\mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}, \tag{14a}$$

$$\mathbf{B}_{12} = -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}, \tag{14b}$$

$$\mathbf{B}_{22} = \left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}. \tag{14c}$$

It is easy to see that $\boldsymbol{\alpha}^{(0)} = \mathbf{A}_{11}^{-1}\mathbf{T}_0$. In other words, $\mathbf{A}_{11}^{-1}$ is already computed in the initial stage. To compute $\boldsymbol{\alpha}^{(1)}$, we only need to invert a matrix of $N_1 \times N_1$ in Eq. (14c). Meanwhile, the output function $\boldsymbol{f}_1(\boldsymbol{x})$ is:

$$\boldsymbol{f}_1(\boldsymbol{x}) = \left(\boldsymbol{\alpha}^{(1)}\right)^T \left[k\left(\boldsymbol{x}_1, \boldsymbol{x}\right), \ldots, k\left(\boldsymbol{x}_{N_0+N_1}, \boldsymbol{x}\right)\right]^T. \tag{15}$$

After that, if another new data chunk $\left(\mathbf{X}_n, \mathbf{T}_n\right)$ comes, repeat steps (13)–(15) to update the output function $\boldsymbol{f}_n(\boldsymbol{x})$.

## 3 Online kernelized and regularized extreme learning machine (OKRELM)

### 3.1 Motivation

In this section, we will think of the update process from a different perspective. $\boldsymbol{\alpha}$ is a $N \times m$ matrix, whose rows increase in the online stage when the training instances come continuously. In this section, we change the notations a little bit different from those in Sect. 2.2.

- In the initial stage, supposing that the initial data chunk $(\mathbf{X}_0, \mathbf{T}_0)$ also contains $N_0$ training instances. Therefore, let the Lagrange matrix corresponding to $(\mathbf{X}_0, \mathbf{T}_0)$ in this stage be $\boldsymbol{\alpha}_0^{(0)}$, where the superscript represents the stage and the subscript represents the data chunk. After the initial training is finished, $\boldsymbol{\alpha}_0^{(0)}, \boldsymbol{\beta}_0$, and the output function $\boldsymbol{f}_0(\boldsymbol{x})$ can be computed by:

$$\boldsymbol{\alpha}_0^{(0)} = \left(\mathbf{H}_0\mathbf{H}_0^{\mathrm{T}} + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0}\right)^{-1}\mathbf{T}_0, \tag{16a}$$

$$\boldsymbol{\beta}_0 = \mathbf{H}_0^{\mathrm{T}}\boldsymbol{\alpha}_0^{(0)}, \tag{16b}$$

$$\begin{aligned} \boldsymbol{f}_0(\boldsymbol{x}) &= \boldsymbol{\beta}_0^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) \rightarrow \boldsymbol{f}_0(\boldsymbol{x}) \\ &= \mathbf{T}_0^{\mathrm{T}}\left(\mathbf{H}_0\mathbf{H}_0^{\mathrm{T}} + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0}\right)^{-1}\mathbf{H}_0\boldsymbol{h}(\boldsymbol{x}). \end{aligned} \tag{16c}$$

Replace $\mathbf{H}_0\mathbf{H}_0^{\mathrm{T}}$ by the Gram matrix $\mathbf{\Omega}_0$ and $\mathbf{H}_0\boldsymbol{h}(\boldsymbol{x})$ by the kernel form, we have:

$$\boldsymbol{\alpha}_0^{(0)} = \left(\mathbf{\Omega}_0 + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0}\right)^{-1}\mathbf{T}_0, \tag{17a}$$

$$\boldsymbol{\beta}_0 = \mathbf{H}_0^{\mathrm{T}}\boldsymbol{\alpha}_0^{(0)}, \tag{17b}$$

$$\boldsymbol{f}_0(\boldsymbol{x}) = \mathbf{T}_0^{\mathrm{T}}\left(\mathbf{\Omega}_0 + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0}\right)^{-1}\left[k\left(\boldsymbol{x}_1, \boldsymbol{x}\right), \ldots, k\left(\boldsymbol{x}_{N_0}, \boldsymbol{x}\right)\right]^T. \tag{17c}$$

- In the online stage, supposing that the first coming data chunk $(\mathbf{X}_1, \mathbf{T}_1)$ also contains $N_1$ training instances. At this time, the Lagrange matrix $\boldsymbol{\alpha}^{(1)} = \left[\boldsymbol{\alpha}_0^{(1)}, \boldsymbol{\alpha}_1^{(1)}\right]^T$, in which $\boldsymbol{\alpha}_0^{(1)}$ and $\boldsymbol{\alpha}_1^{(1)}$ correspond to $(\mathbf{X}_0, \mathbf{T}_0)$ and $(\mathbf{X}_1, \mathbf{T}_1)$, respectively. According to Eq. (4) we have:

$$\left(\begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}\begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T + \frac{1}{\mathrm{C}}\mathbf{I}_{(N_0+N_1)\times(N_0+N_1)}\right)\begin{bmatrix} \boldsymbol{\alpha}_0^{(1)} \\ \boldsymbol{\alpha}_1^{(1)} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}. \tag{18}$$

Equation (18) is equivalent to the following two equations:

$$\left(\mathbf{H}_0\mathbf{H}_0{}^T+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)\boldsymbol{\alpha}_0^{(1)}+\mathbf{H}_0\mathbf{H}_1{}^T\boldsymbol{\alpha}_1^{(1)}=\mathbf{T}_0, \tag{19a}$$

$$\mathbf{H}_1\mathbf{H}_0{}^T\boldsymbol{\alpha}_0^{(1)}+\left(\mathbf{H}_1\mathbf{H}_1{}^T+\frac{1}{\mathrm{C}}\mathbf{I}_{N_1\times N_1}\right)\boldsymbol{\alpha}_1^{(1)}=\mathbf{T}_1, \tag{19b}$$

$\boldsymbol{\alpha}_0^{(1)}$can be represented by $\boldsymbol{\alpha}_1^{(1)}$ in a manner of:

$$\boldsymbol{\alpha}_0^{(1)}=\boldsymbol{\alpha}_0^{(0)}-\left(\mathbf{H}_0\mathbf{H}_0{}^T+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)^{-1}\mathbf{H}_0\mathbf{H}_1{}^T\boldsymbol{\alpha}_1^{(1)}. \tag{20}$$

Equation (20) is substituted in Eq. (19b), we can solve $\boldsymbol{\alpha}_1^{(1)}$ in a way of:

$$\boldsymbol{\alpha}_1^{(1)}=\mathrm{C}\left(\boldsymbol{I}_{N_1\times N_1}+\mathbf{H}_1\boldsymbol{\gamma}_0\mathbf{H}_1{}^T\right)^{-1}\left(\mathbf{T}_1-\mathbf{H}_1\boldsymbol{\beta}_0\right), \tag{21}$$

$\boldsymbol{\gamma}_n$ is defined by:

$$\boldsymbol{\gamma}_n=\begin{cases}\left(\begin{bmatrix}\mathbf{H}_0\\ \vdots\\ \mathbf{H}_n\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{H}_0\\ \vdots\\ \mathbf{H}_n\end{bmatrix}+\frac{1}{\mathrm{C}}\mathbf{I}_{L\times L}\right)^{-1}, n=1,2,\ldots\\ \left(\mathbf{H}_0{}^{\mathrm{T}}\mathbf{H}_0+\frac{1}{\mathrm{C}}\mathbf{I}_{L\times L}\right)^{-1}, n=0\end{cases}, \tag{22}$$

$\boldsymbol{\gamma}_0$ is equal to Eq. (23) according to the Sherman–Morrison–Woodbury (SMW) formula [54].

$$\begin{aligned}\boldsymbol{\gamma}_0&=\left(\mathbf{H}_0{}^{\mathrm{T}}\mathbf{H}_0+\frac{1}{\mathrm{C}}\mathbf{I}_{L\times L}\right)^{-1}\\&=\mathrm{C}\left(\mathbf{I}_{L\times L}-\mathbf{H}_0{}^{\mathrm{T}}\left(\mathbf{H}_0\mathbf{H}_0{}^{\mathrm{T}}+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)^{-1}\mathbf{H}_0\right).\end{aligned} \tag{23}$$

According to Eq. (3a), $\boldsymbol{\beta}_1$ can be represented by:

$$\boldsymbol{\beta}_1=\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\boldsymbol{\alpha}_0^{(1)}\\ \boldsymbol{\alpha}_1^{(1)}\end{bmatrix}=\mathbf{H}_0{}^T\boldsymbol{\alpha}_0^{(1)}+\mathbf{H}_1{}^T\boldsymbol{\alpha}_1^{(1)}. \tag{24}$$

Equations (20)–(21) are substituted into Eq. (24). By making use of the SMW Formula again, $\boldsymbol{\beta}_1$ can also be represented by:

$$\boldsymbol{\beta}_1=\boldsymbol{\beta}_0+\boldsymbol{\gamma}_1\mathbf{H}_1{}^T\left(\mathbf{T}_1-\mathbf{H}_1\boldsymbol{\beta}_0\right). \tag{25}$$

According to Eq. (6), we can get $\boldsymbol{f}_1(\boldsymbol{x})$ by:

$$\boldsymbol{f}_1(\boldsymbol{x})=\boldsymbol{f}_0(\boldsymbol{x})+\left(\mathbf{T}_1-\boldsymbol{f}_0(\mathbf{X}_1)\right)^{\mathrm{T}}\mathbf{H}_1\boldsymbol{\gamma}_1\boldsymbol{h}(\boldsymbol{x}). \tag{26}$$

In fact, $\boldsymbol{\gamma}_n$ cannot be saved owing to the implicit mapping while using kernels. Therefore, we define a new function $\boldsymbol{g}_n(\boldsymbol{x},\boldsymbol{y})$ which can be saved during the online stage:

$$\boldsymbol{g}_n(\boldsymbol{x},\boldsymbol{y})=\boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{\gamma}_n\boldsymbol{h}(\boldsymbol{y}). \tag{27}$$

Therefore, the new output function $\boldsymbol{f}_1(\boldsymbol{x})$ can be updated based on the old one $\boldsymbol{f}_0(\boldsymbol{x})$ by Eq. (28). $\boldsymbol{g}_1\left(\mathbf{X}_1,\boldsymbol{x}\right)=\left[\boldsymbol{g}_1\left(\boldsymbol{x}_1,\boldsymbol{x}\right),\ldots,\boldsymbol{g}_1\left(\boldsymbol{x}_{N_1},\boldsymbol{x}\right)\right]^T$.

$$\boldsymbol{f}_1(\boldsymbol{x})=\boldsymbol{f}_0(\boldsymbol{x})+\left(\mathbf{T}_1-\boldsymbol{f}_0\left(\mathbf{X}_1\right)\right)^{\mathrm{T}}\boldsymbol{g}_1\left(\mathbf{X}_1,\boldsymbol{x}\right). \tag{28}$$

Generally, $\boldsymbol{f}_n(\boldsymbol{x})$ is updated by $\boldsymbol{f}_{n-1}(\boldsymbol{x})$ according to:

$$\boldsymbol{f}_n(\boldsymbol{x})=\boldsymbol{f}_{n-1}(\boldsymbol{x})+\left(\mathbf{T}_n-\boldsymbol{f}_{n-1}\left(\mathbf{X}_n\right)\right)^{\mathrm{T}}\boldsymbol{g}_n\left(\mathbf{X}_n,\boldsymbol{x}\right),\quad n=1,2,\ldots \tag{29}$$

To summarize,

$$\boldsymbol{f}_n(\boldsymbol{x})=\begin{cases}\boldsymbol{f}_{n-1}(\boldsymbol{x})+\left(\mathbf{T}_n-\boldsymbol{f}_{n-1}\left(\mathbf{X}_n\right)\right)^{\mathrm{T}}\boldsymbol{g}_n\left(\mathbf{X}_n,\boldsymbol{x}\right), n=1,2,\ldots\\ \mathbf{T}_0^{\mathrm{T}}\left(\mathbf{H}_0\mathbf{H}_0^{\mathrm{T}}+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)^{-1}\mathbf{H}_0h(\boldsymbol{x}),\quad n=0\end{cases}, \tag{30a}$$

$$\boldsymbol{g}_n(\boldsymbol{x},\boldsymbol{y})=\begin{cases}\boldsymbol{g}_{n-1}(\boldsymbol{x},\boldsymbol{y})-\boldsymbol{g}_{n-1}\left(\boldsymbol{x},\mathbf{X}_n\right)\left(\mathbf{I}_{N_n\times N_n}+\boldsymbol{g}_{n-1}\left(\mathbf{X}_n,\mathbf{X}_n\right)\right)^{-1}\boldsymbol{g}_{n-1}\left(\mathbf{X}_n,\boldsymbol{y}\right),\quad n=1,2,\ldots\\ C\left(\boldsymbol{h}(\boldsymbol{x})^T\boldsymbol{h}(\boldsymbol{y})-\boldsymbol{h}(\boldsymbol{x})^T\mathbf{H}_0{}^{\mathrm{T}}\left(\mathbf{H}_0\mathbf{H}_0{}^{\mathrm{T}}+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)^{-1}\mathbf{H}_0\boldsymbol{h}(\boldsymbol{y})\right),\quad n=0\end{cases}. \tag{30b}$$

Replace the Equations (30a) and (30b) by the kernel form, we have:

$$\boldsymbol{f}_n(\boldsymbol{x})=\begin{cases}\boldsymbol{f}_{n-1}(\boldsymbol{x})+\left(\mathbf{T}_n-\boldsymbol{f}_{n-1}\left(\mathbf{X}_n\right)\right)^{\mathrm{T}}\boldsymbol{g}_n\left(\mathbf{X}_n,\boldsymbol{x}\right), n=1,2,\ldots\\ \mathbf{T}_0^{\mathrm{T}}\left(\boldsymbol{\Omega}_0+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)^{-1}\left[k(\boldsymbol{x}_1,\boldsymbol{x}),\ldots,k\left(\boldsymbol{x}_{N_0},\boldsymbol{x}\right)\right]^T, n=0\end{cases}, \tag{31a}$$

$$\boldsymbol{g}_n(\boldsymbol{x},\boldsymbol{y})=\begin{cases}\boldsymbol{g}_{n-1}(\boldsymbol{x},\boldsymbol{y})-\boldsymbol{g}_{n-1}\left(\boldsymbol{x},\mathbf{X}_n\right)\left(\mathbf{I}_{N_n\times N_n}+\boldsymbol{g}_{n-1}\left(\mathbf{X}_n,\mathbf{X}_n\right)\right)^{-1}\boldsymbol{g}_{n-1}\left(\mathbf{X}_n,\boldsymbol{y}\right), n=1,2,\ldots\\ C\left(k(\boldsymbol{x},\boldsymbol{y})-\left[k\left(\boldsymbol{x},\boldsymbol{x}_1\right),\ldots,k\left(\boldsymbol{x},\boldsymbol{x}_{N_0}\right)\right]\left(\boldsymbol{\Omega}_0+\frac{1}{\mathrm{C}}\mathbf{I}_{N_0\times N_0}\right)^{-1}\left[k\left(\boldsymbol{x}_1,\boldsymbol{y}\right),\ldots,k\left(\boldsymbol{x}_{N_0},\boldsymbol{y}\right)\right]^T\right), n=0\end{cases}. \tag{31b}$$

If $\boldsymbol{f}_{n-1}\left(\mathbf{X}_n\right)=\mathbf{T}_n$, $\boldsymbol{f}_n(\boldsymbol{x})$ equals to $\boldsymbol{f}_{n-1}(\boldsymbol{x})$ according to Eq. (31a). Therefore, there is absolutely no need to update the output function. What if $\boldsymbol{f}_{n-1}\left(\mathbf{X}_n\right)\neq\mathbf{T}_n$ but all $\mathbf{X}_n$ (or a majority of them) can be correctly predicted using $\boldsymbol{f}_{n-1}(\boldsymbol{x})$? It seems we do not need to update the output function all the time. In the following section, we propose when to update and how to select a subset of the new data chunk to update.

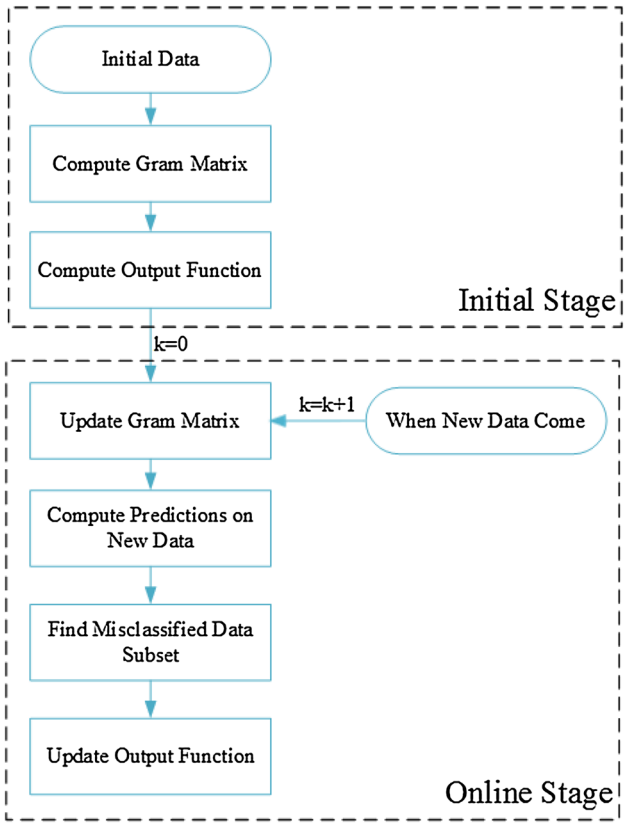


**Fig. 1** Main pipeline of OKRELM

### 3.2 The proposed OKRELM

Considering the large amounts of computation during the online stage and the high possibility of redundant data contained in the new data chunk. We propose the Online Kernelized and Regularized Extreme Learning Machine (OKRELM). The main contribution of OKRELM is the model updating. When new data chunk comes, we firstly predict their labels using the current model. If some of them are misclassified, we will update the output function based on only the misclassified ones. Figure 1 illustrates the main pipeline of OKRELM. The algorithm of OKRELM is shown in Table 1.

It consists of the initial and online stages during the training process of OKRELM, which is similar to the other online learning methods. During the initial stage, a Gram matrix is firstly computed according to the initial data chunk with $N_0$ instances, then we attain the output function by inverting a matrix of $N_0 \times N_0$ (Eqs. 32–33).The Gram matrix is updated first every time new data come in the online stage. Then we predict the labels of instances in the new data chunk using the current output function (Eqs. 34–35). After that, all the misclassified instances can be gathered into a subset for update by comparing the predictions with the ground truth (Eq. 36). In the end, we update the output function with those misclassified instances in the subset (Eq. 38). When another data chunk comes, repeat the steps in the online stage to update the output function.

Both the approaches in [41, 51] and OKRELM tackle the expensive storage requirement in the training phase and expensive time consumption in the testing process when experienced with the kernels. Their differences are listed as follows: (a) The approach in [41] is a batch learning model, whereas the approach in [51] and the OKRELM in this paper are especially proposed for incremental learning. (b) Authors in [41] selected a subset of training samples as prototypes by solving an optimization problem to decrease the large storage requirement; about 10% initial data are randomly selected to generate the initial model, and the data with high confidence levels are chosen from the data streams to update the model in [51]. For OKRELM, we use all the available data to initialize, and choose only the misclassified samples from each incremental data chunk to update the model in the online stage.

## 4 Performance evaluation

### 4.1 Data description and preprocess

We employ the Daily and Sports Activities Data Set (DSADS) [55] from UCI Machine Learning Repository to validate the performance of OKRELM. This dataset contains 19 daily and sports activities [(1) sitting,(2) standing, (3) lying on back, (4) lying on right side, (5) ascending stairs, (6) descending stairs, (7) standing in an elevator still, (8) moving around in an elevator, (9) walking in a parking lot, (10) walking on a treadmill with a speed of 4 kmh, (11) walking in flat and 15° inclined positions, (12) running on a treadmill with a speed of 8 kmh, (13) exercising on a stepper, (14) exercising on a cross trainer, (15) cycling on an exercise bike in horizontal positions, (16) cycling on an exercise bike in vertical positions, (17) rowing, (18) jumping, (19) playing basketball] performed by eight participants (four males and four females). Contributors explored the potential of recording activities on five different body parts (torso, right arm, left arm, right leg and left leg) with three-axial accelerometer, three-axial gyroscope and three-axial magnetometer on each part respectively. Each participant performed each activity in his own style. That produced 45 sensor readings per frame ($3 \times 5 \times 3$).

Features are extracted from every sensor. Firstly, we combine the three axes of one sensor together using $a = \sqrt{x^2 + y^2 + z^2}$. Then we exploit the sliding window technique to extract features (window length = 5 s [55]).

**Table 1** The OKRELM algorithm

- In the initial stage, supposing initial data chunk $(\mathbf{X}_0, \mathbf{T}_0)$ contains $N_0$ instances.
  1. Compute the Gram matrix $\boldsymbol{\Omega}_0, (\boldsymbol{\Omega}_0)_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), i, j = 1, \cdots, N_0$.
  2. Compute the matrix $\mathbf{A}$:

$$\mathbf{A} = \left(\boldsymbol{\Omega}_0 + \frac{1}{\mathrm{C}}\mathbf{I}_{N_0 \times N_0}\right)^{-1} \tag{32}$$

  3. Compute the output function $\boldsymbol{f}_0(\boldsymbol{x})$:

$$\boldsymbol{f}_0(\boldsymbol{x}) = \mathbf{T}_0^{\mathrm{T}}\mathbf{A}\left[k(\boldsymbol{x}_1, \boldsymbol{x}), \cdots, k(\boldsymbol{x}_{N_0}, \boldsymbol{x})\right]^T \tag{33}$$

- In the online stage, suppose the first coming data chunk $(\mathbf{X}_1, \mathbf{T}_1)$ also contains $N_1$ training instances.
  1. Compute the matrix $\boldsymbol{\Omega}_{01}$ , in which $(\boldsymbol{\Omega}_{01})_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), i = 1, \cdots, N_0, j = N_0 + 1, \cdots, N_0 + N_1$.
  2. Compute $\boldsymbol{f}_0(\mathbf{X}_1)$ by

$$\boldsymbol{f}_0(\mathbf{X}_1) = \mathbf{T}_0^{\mathrm{T}}\mathbf{A}\boldsymbol{\Omega}_{01} \tag{34}$$

  3. Compute the prediction results according to Equation (35) and generate the indicator set $\Delta$ of misclassification instances according to Equation (36).

$$\mathrm{Prediction}(\boldsymbol{x}_j) = \underset{k\in\{1,\cdots,m\}}{\mathrm{argmax}} (\boldsymbol{f}_0)_k(\boldsymbol{x}_j), j = N_0 + 1, \cdots, N_0 + N_1 \tag{35}$$

$$\Delta = \left\{ j \middle| \mathrm{Prediction}(\boldsymbol{x}_j) \neq \underset{i\in\{1,\cdots,m\}}{\mathrm{argmax}} t_{i,j}, j = N_0 + 1, \cdots, N_0 + N_1 \right\} \tag{36}$$

  4. Find the matrix $\boldsymbol{\Omega}'_{01}$:

$$\boldsymbol{\Omega}'_{01} = \left[\cdots, (\boldsymbol{\Omega}_{01})_{:,j}, \cdots\right], j \in \Delta \tag{37}$$

  5. Compute the Gram matrix $\boldsymbol{\Omega}_1, (\boldsymbol{\Omega}_1)_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j), i \in \Delta, j \in \Delta$.
  6. Compute the output function $\boldsymbol{f}_1(\boldsymbol{x})$:

$$\boldsymbol{f}_1(\boldsymbol{x}) = \begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix}^T \begin{bmatrix}\mathbf{B}_{11} & \mathbf{B}_{12}\\ \mathbf{B}_{12}^T & \mathbf{B}_{22}\end{bmatrix}^T \left[k(\boldsymbol{x}_1, \boldsymbol{x}), \cdots, k(\boldsymbol{x}_{N_0+N_1}, \boldsymbol{x})\right]^T \tag{38}$$

where as

$$\mathbf{B}_{11} = \mathbf{A} + \mathbf{A}\boldsymbol{\Omega}'_{01}\left(\boldsymbol{\Omega}_1 + \frac{1}{\mathrm{C}}\mathbf{I}_{|\Delta|\times|\Delta|} - (\boldsymbol{\Omega}'_{01})^T\mathbf{A}\boldsymbol{\Omega}'_{01}\right)^{-1}(\boldsymbol{\Omega}'_{01})^T\mathbf{A} \tag{39a}$$

$$\mathbf{B}_{12} = -\mathbf{A}\boldsymbol{\Omega}'_{01}\left(\boldsymbol{\Omega}_1 + \frac{1}{\mathrm{C}}\mathbf{I}_{|\Delta|\times|\Delta|} - (\boldsymbol{\Omega}'_{01})^T\mathbf{A}\boldsymbol{\Omega}'_{01}\right)^{-1} \tag{39b}$$

$$\mathbf{B}_{22} = \left(\boldsymbol{\Omega}_1 + \frac{1}{\mathrm{C}}\mathbf{I}_{|\Delta|\times|\Delta|} - (\boldsymbol{\Omega}'_{01})^T\mathbf{A}\boldsymbol{\Omega}'_{01}\right)^{-1} \tag{39c}$$

$|\Delta|$ represents the number of elements in the indicator set $\Delta$.

If another chunk of data $(\mathbf{X}_n, \mathbf{T}_n)$ comes, repeat steps (34) - (38) to update the output function $\boldsymbol{f}_n(\boldsymbol{x})$.

27 features from both time and frequency domain are extracted (see Table 2), leading to 405 dimensions in total $(27 \times 3 \times 5 = 405)$. After feature extraction, we perform Principal Component Analysis (PCA) to reduce the dimensions to 30 [55]. Specifications of the DSADS dataset is shown in Table 3.

### 4.2 Parameter selection

Gaussian kernel $\left(\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\boldsymbol{x}_i - \boldsymbol{x}_j{}^2/\mathrm{g})\right)$ is used for RELM, KB-IELM and OKRELM. Therefore, we need to find the optimal value combination for two parameters (penalty parameter C and kernel parameter g). Grid search

**Table 2** Features extracted per sensor on each body part

| ID | Feature | Description |
|---|---|---|
| 1 | Mean | Average value of samples in window |
| 2 | STD | Standard deviation |
| 3 | Minimum | Minimum |
| 4 | Maximum | Maximum |
| 5 | Mode | The value with the largest frequency |
| 6 | Range | Maximum minus minimum |
| 7 | Mean crossing rate | Rate of times signal crossing the mean value |
| 8 | dc | Direct component |
| 9–13 | Spectrum peak position | First five peaks after FFT |
| 14–18 | Frequency | Frequencies corresponding to the five peaks |
| 19 | Energy | Square of norm |
| 20–23 | Four shape features | Mean, STD, skewness, kurtosis |
| 24–27 | Four amplitude features | Mean, STD, skewness, kurtosis |

**Table 3** Specifications of the DSADS dataset

| Name | DSADS dataset |
|---|---|
| Class number | 1–19 |
| User number | 1–8 |
| Total data size | 9120 |

technique is employed and 50 different values $\{2^{-24}, 2^{-23}, \ldots, 2^{24}, 2^{25}\}$ are tried for both C and g. The optimal value combination of (C, g) is chosen from 2500 pairs.

We divide the DSADS dataset into the training and testing sets using repeated random sub-sampling (RRSS) technique [56]. That is, for all the instances of each class of each user, one-third is randomly selected at the testing set and the other two-thirds are used as the training set. Therefore, the training set and testing set consist of 6080 and 3040 instances, respectively.

For each pair of (C, g), the RELM, KB-IELM and OKRELM models are all learned. For KB-IELM, 1000 training instances are randomly selected as the initial data chunk, a series of data chunks with random size are randomly generated in the online stage. Figure 2 shows the testing accuracy of RELM, KB-IELM and OKRELM.

From the results in Fig. 2 we can see that, performance of RELM varies smoothly in the parameter space. KB-IELM and OKRELM are consistent with RELM in some places while inconsistent with RELM in the other places. Therefore, the optimal value of (C, g) is chosen to be $(2^2, 2^4)$ corresponding to the highest and consistent testing accuracy. In the following section, all experiments are run with the optimal values for the penalty parameter C and kernel parameter g.

## 4.3 Experimental results and analysis

### *4.3.1 Robustness comparison with different initial and online chunk size*

In order to evaluate the robustness of our proposed model, firstly we introduce two important concepts: initial trunk size and online trunk size. Initial trunk size refers to the number of training instances contained in the initial chunk while online trunk size means the number of training instances contained in each online chunk. In the experiments, we train KB-IELM and OKRELM with the initial chunk size taking values from {100, 200, 300,…, 2000} and the online chunk size taking values from {50, 100, 150,…, 1000}. Training and testing sets are generated using the RRSS technique as in Sect. 4.2. The whole process is repeated five times and results are averaged and shown in Fig. 3.

From the results in Fig. 3 we can see that: Given a random initial chunk size, the accuracies of both KB-IELM and OKRELM firstly grow with the increasing online chunk size, then converge to a high performance and remain stable at last. If we fix the online chunk size, the increase trend is not straightforward when the initial chunk size grows. Therefore, compared with the initial chunk size, the online chunk size has a much stronger influence on the final performance of both classifiers. To be more specific, KB-IELM needs an online chunk size of larger than 700 to achieve an accuracy of about 90%, while OKRELM only needs 250 (no matter whether misclassified or not). What's more, OKRELM may require less than 250 instances per chunk according to results in Sect. 4.3.3. In other words, OKRELM is less sensitive and more robust than KB-IELM when used to construct a generic AR model. Once the online chunk size is satisfied, both KB-IELM and OKRELM can achieve a satisfactory predictive

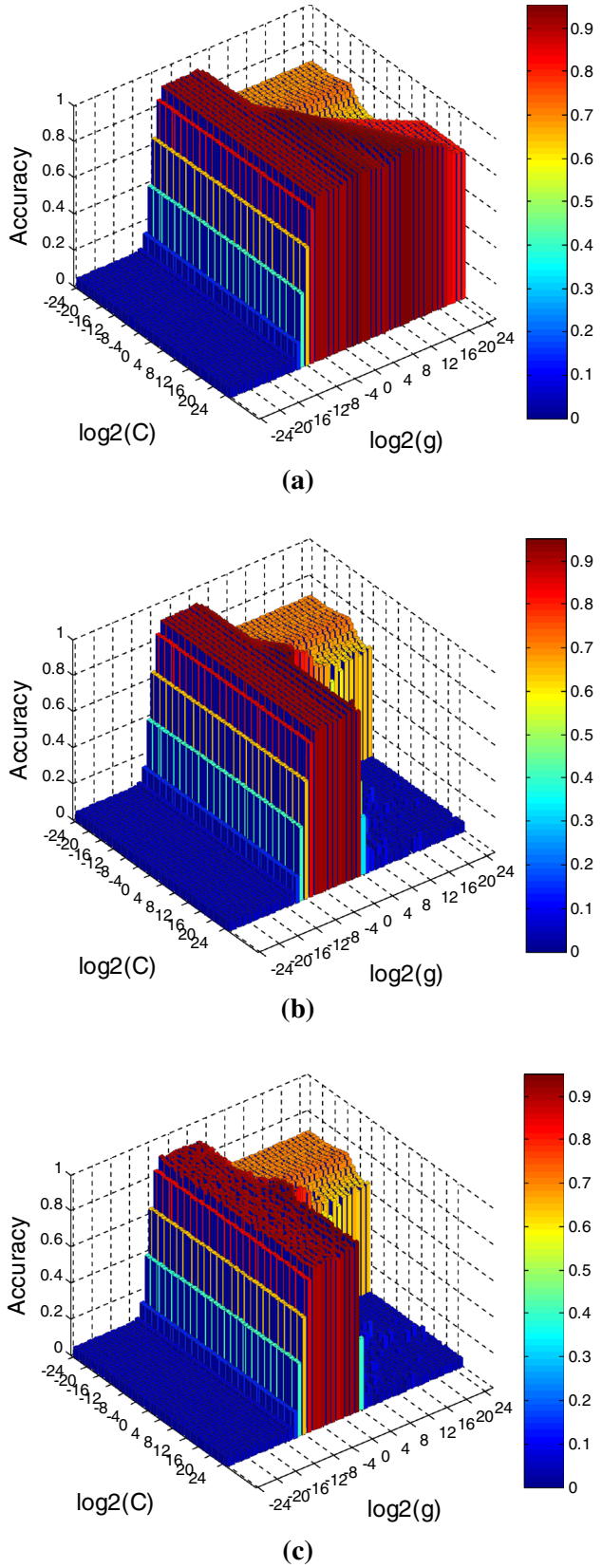


**Fig. 2** Testing accuracy of RELM, KB-IELM and OKRELM with different parameter pairs. **a** RELM; **b** KB-IELM; **c** OKRELM

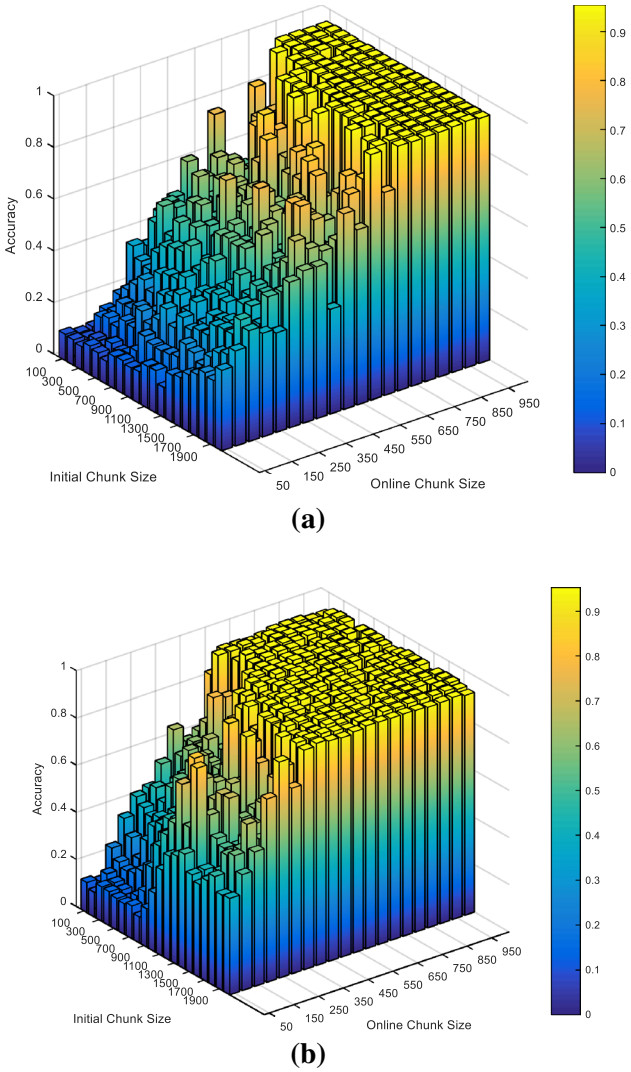


**Fig. 3** Testing accuracy of KB-IELM and OKRELM with different initial and online chunk sizes. **a** KB-IELM; **b** OKRELM

performance comparable to the RELM no matter how many instances are contained in the initial chunk.

#### *4.3.2 Accuracy and time consumption comparison with RRSS split*

To compare the effectiveness and efficiency of KB-IELM and OKRELM, we record their predictive accuracy on the testing set as the online chunk size increases. In the meantime, we also keep tracking of their time consumption during training and testing stages. Based on the results in Sect. 4.3.1, the initial chunk size is set to 100 and the online chunk size is set to 700. Training and testing sets are generated using the RRSS technique as in Sect. 4.2. The whole process is repeated five times and results are averaged and shown in Fig. 4.

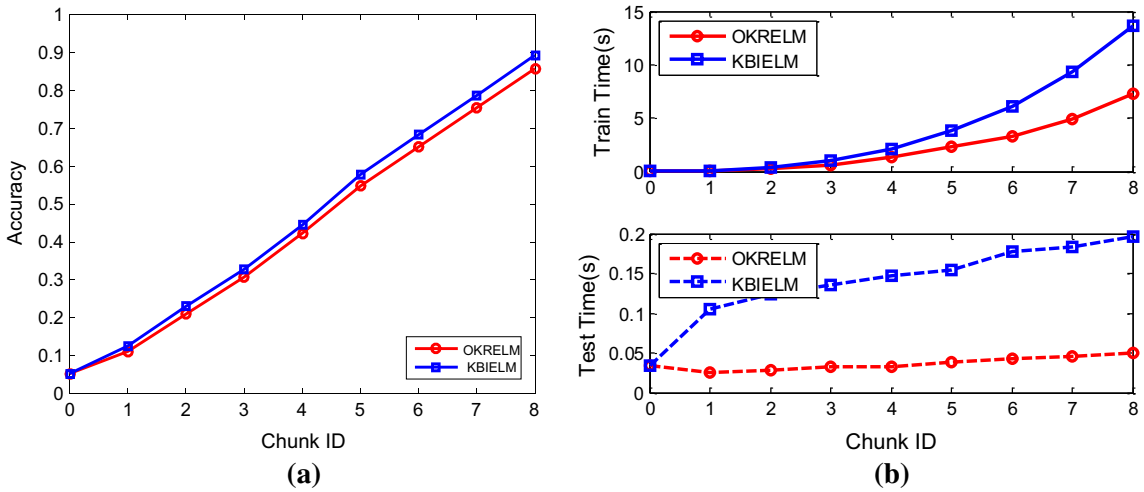


**Fig. 4** Accuracy, training time and testing time comparisons of KB-IELM and OKRELM with RRSS split during different phases in the online stage. **a** Predictive accuracy on the testing set; **b** training time and testing time consumptions

The predictive performance of KB-IELM and OKRELM is extremely low (5.25%, see Fig. 4a) at the beginning owing to the sparse initial training data (100 instances from 19 classes). As the incremental data chunks coming, the accuracy grows continuously and finally achieves a precision similar to RELM (94.53%). It verifies that the performance of online learning models KB-IELM and OKRELM can converge to the accuracy of the batch model RELM. The accuracy of OKRELM is a bit lower (3.97%) than that of KB-IELM, because large amounts of training data from the data chunks have been eliminated for OKRELM from updating the model compared with KB-IELM. Those data eliminated might not be unhelpful. However, such an elimination in OKRELM saves 46.49% training time and 74.61% testing time compared with KB-IELM (see Fig. 4b). It makes OKRELM a lightweight and effective AR model for mini-wearable devices.

### *4.3.3 Accuracy and time consumption comparison with LOO split*

To view the personalization learning probability, we compare the effectiveness and efficiency of KB-IELM and OKRELM which are trained initially based on seven users' data while updated and validated on the 8th user's data. The leave one subject out (LOO) cross validation technique [56] is used to generate the training and testing datasets. For each user's instances, one-half is used for updating the model (50 instances per chunk owing to the limitations instance number of one single user), and the other half is used as the testing set and 100 instances of the other seven users are randomly selected as the initial training set. The experiment is run 20 times for each user and the predictive performances are averaged and shown in Fig. 5.

All in all, the trends of accuracy in Fig. 5 are nearly the same for different users. Performances of KB-IELM and OKRELM improve at the beginning with KB-IELM a bit more accurate than OKRELM. During the last four chunks, the performance of KB-IELM goes down and becomes fluctuate while OKRELM becomes gradually convergent and stable in the predictive accuracy. To better observe and analyze, the results in Fig. 5 are averaged and shown in Fig. 6a. Besides, Fig. 6b shows the averaged time consumptions in the training and testing stages.

The predictive performance of OKRELM increases steadily along with the update process which is similar to the results in Fig. 4a. The results of OKRELM are very interesting since the chunk size in the online stage is only 50 (data of about 4.16 min, 50 instances$\times$5 s), which is less than needed (250, see the discussion in Sect. 4.3.1). Therefore, it reveals the robust performance of OKRELM in transforming a generic model into a personalized model. For KB-IELM, its accuracy increases faster in the first place, goes down abruptly and becomes fluctuant in the end. The reason of the superiority of KB-IELM at the beginning of the online stage has been discussed in Sect. 4.3.2. The fluctuation phenomenon of KB-IELM again confirms our conclusion in Sect. 4.3.1 that the performance of KB-IELM might be unstable when the chunk size is not large. To be more specific, KB-IELM requires about 700 instances (data of about 58.33 min, 700 instances$\times$5 s) each chunk to assure the update process going in a positive way (the accuracy will increase after the update). When chunk size is not enough (say 50 in our experiment), it is more likely to decrease for KB-IELM. Considering the jogger example introduced in the Introduction, KB-IELM updates every 58.33 min to attain a stable performance while OKRELM can update every 4.16 min. Therefore, OKRELM is more

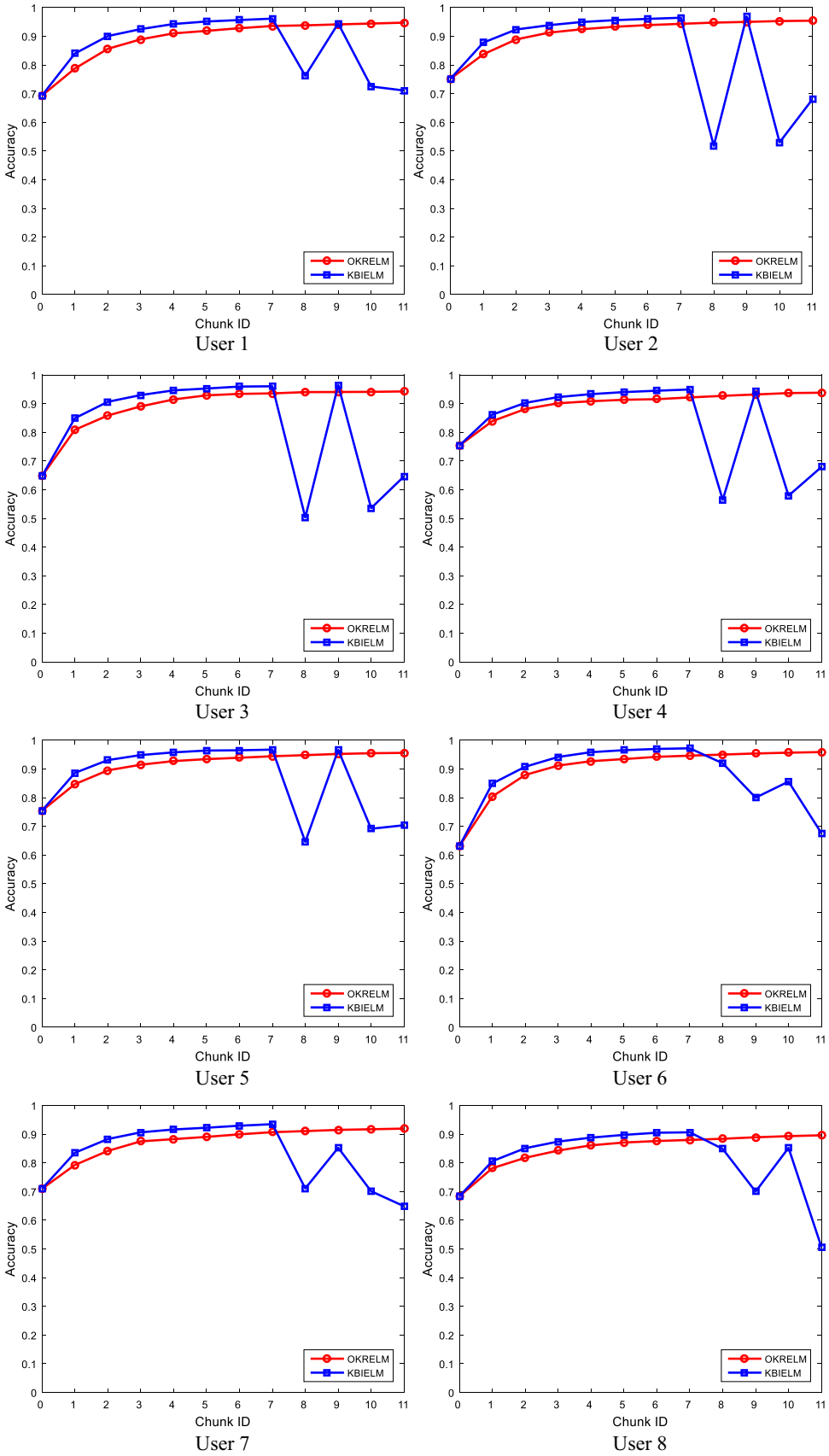


**Fig. 5** The predictive accuracy of KB-IELM and OKRELM on the datasets of different users with LOO split

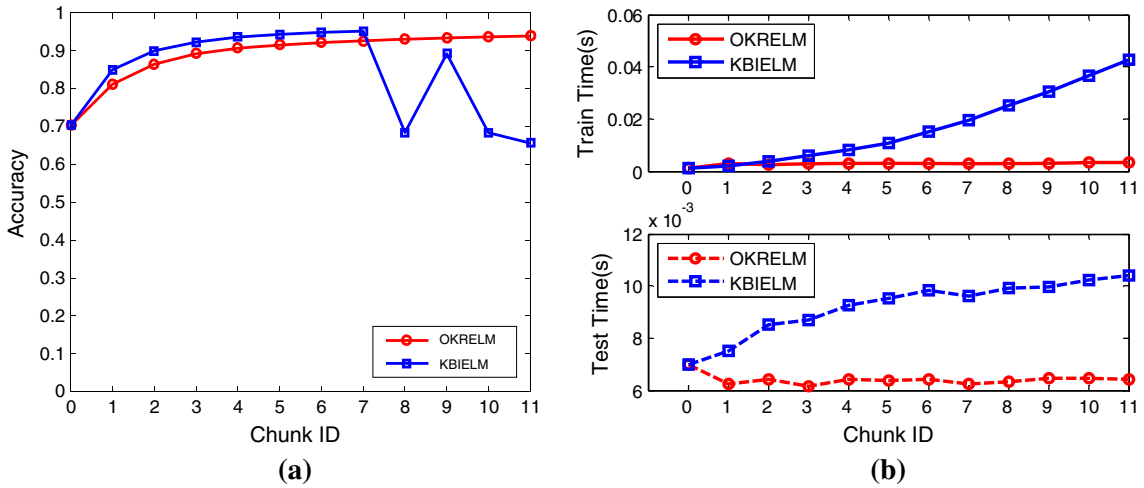


**Fig. 6** Average accuracy, training time and testing time comparisons of KB-IELM and OKRELM with LOO split during different phases in the online stage. **a** Predictive accuracy on the testing set; **b** training time and testing time consumptions

suitable for real-time update. Meanwhile, the time efficiency of OKRELM during the training process (Fig. 6b) is much more straightforward compared with the results in Fig. 4b. Specifically, the time consumption in the training stage of OKRELM is only 7.18% of the time consumption in the training stage of KB-IELM. OKRELM saves 38.17% time compared with KB-IELM during the prediction stage. That is, OKRELM is more suitable to predict user's ongoing activity in real-time.

To summarize, OKRELM is more stable and efficient than KB-IELM when used to construct a personalized AR model.

## 5 Conclusion

In this paper, we propose OKRELM, an online kernelized and regularized extreme learning machine for activity recognition using mini-wearable devices. OKRELM is more efficient than the related online learning models during both the training and prediction processes, and effective which is comparable to the batch mode model. Experimental results on an online UCI activity recognition dataset show the efficiency and effectiveness of the proposed OKRELM.

When a generic classifier is updated towards a personalized model by making use of online data of the target user, it may experience the class imbalance problem that large amounts of data in the upcoming chunk are from a few classes. In the future, we plan to modify the OKRELM to better suit the class imbalance problem in wearable activity recognition.

**Acknowledgements** The authors are much grateful to Prof. Xingyu Gao from Institute of Software Chinese Academy of Sciences for his constructive comments and suggestions that have helped to improve the quality of this paper. This work is supported by Natural Science Foundation of China under Grant Nos. 61572471 and 61210010, Chinese Academy of Sciences Research Equipment Development Project under Grant No. YZ201527 and Science and Technology Planning Project of Guangdong Province under Grant No. 2015B010105001.

## References

1. Chen Z, Chen Y, Hu L et al (2014) ContextSense: unobtrusive discovery of incremental social context using dynamic Bluetooth data[C]// International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. ACM, 23–26
2. Hu L, Chen Y, Wang S et al (2013) A nonintrusive and single-point infrastructure-mediated sensing approach for water-use activity recognition[C]//International Conference on embedded and ubiquitous computing (EUC). IEEE, 2120–2126
3. Wang S, Zhou G, Hu L et al (2015) CARE: chewing activity recognition using noninvasive single axis accelerometer[C]// International Joint Conference on pervasive and ubiquitous computing: Adjunct Publication. ACM, 109–112
4. Chen YQ, Yu HC, Miao CY et al (2015) Using motor patterns for stroke detection[J]. Science (Advances in Computational Psychophysiology) 350(6256):12–14
5. Wang R, Chen F, Chen Z et al (2014) StudentLife: assessing mental health, academic performance and behavioral trends of college students using smartphones[C]// International Joint Conference on pervasive and ubiquitous computing. ACM, 3–14
6. https://getpebble.com/steel. Accessed 2 June 2014
7. https://jawbone.com/up. Accessed 2 June 2014
8. http://www.bodymedia.com/Support-Help/BodyMedia-FIT-BW. Accessed 2 June 2014
9. Zhao Q, Hu B, Shi Y et al (2014) Automatic identification and removal of ocular artifacts in EEG—improved adaptive predictor filtering for portable applications[J]. IEEE Trans Nanobio-Science 13(2):109–117
10. Hu B, Majoe D, Ratcliffe M et al (2011) EEG-based cognitive interfaces for ubiquitous applications: developments and challenges[J]. IEEE Intell Syst 26(5):46–53

11. Bao L, Intille SS (2004) Activity recognition from user-annotated acceleration data[C]//International Conference on pervasive computing. Springer, Berlin Heidelberg, 1–17
12. Berlin E, Van Laerhoven K (2012) Detecting leisure activities with dense motif discovery[C]//International Conference on ubiquitous computing. ACM, 250–259
13. Hu L (2014) A lightweight and low-power activity recognition system for mini-wearable devices[C]//International Conference on pervasive computing and communications Workshops (PERCOM Workshops). IEEE, 166–167
14. Pärkkä J, Cluitmans L, Ermes M (2010) Personalization algorithm for real-time activity recognition using PDA, wireless motion bands, and binary decision tree[J]. IEEE Trans Inf Technol Biomed 14(5):1211–1215
15. Qian H, Mao Y, Xiang W et al (2010) Recognition of human activities using SVM multi-class classifier[J]. Pattern Recognit Lett 31(2):100–111
16. Hu L, Lu S, Wang X (2013) A new and informative active learning approach for support vector machine[J]. Inf Sci 244:142–160
17. Hu L, Chen Y, Wang S et al (2014) b-COELM: A fast, lightweight and accurate activity recognition model for mini-wearable devices[J]. Pervasive Mob Comput 15:200–214
18. Zeng Z, Ji Q (2010) Knowledge based activity recognition with dynamic Bayesian network[C]//European Conference on computer vision. Springer Berlin Heidelberg, 532–546
19. Gaikwad K (2012) HMM classifier for human activity recognition[J]. Comput Sci Eng 2(4):27
20. Song Y, Lu Z, Leung CW et al (2013) Collaborative boosting for activity classification in microblogs[C]// International Conference on knowledge discovery and data mining. ACM, 482–490
21. Gu B, Sheng VS (2016) A robust regularization path algorithm for $\nu$-support vector classification. IEEE Trans Neural Netw Learn Syst 99. doi:10.1109/TNNLS.2016.2527796
22. Gu B, Sun X, Sheng VS (2016) Structural minimax probability machine[J].IEEE Trans Neural Netw Learn Syst 99. doi:10.1109/TNNLS.2016.2544779
23. Gu B, Sheng VS, Wang Z et al (2015) Incremental learning for $\nu$-support vector regression[J]. Neural Netw 67:140–150
24. Gu B, Sheng VS, Tay KY et al (2015) Incremental support vector learning for ordinal regression[J]. IEEE Trans Neural Netw Learn Syst 26(7):1403–1416
25. Gu B, Sheng V S, Li S (2015) Bi-parameter space partition for cost-sensitive SVM[C]//IJCAI 3532–3539.
26. Zhang Y, Sun X, Wang B (2016) Efficient algorithm for k-barrier coverage based on integer linear programming[J]. China Commun 13(7):16–23
27. Wen X, Shao L, Xue Y et al (2015) A rapid learning algorithm for vehicle classification[J]. Inf Sci 295:395–406
28. Kong Y, Zhang M, Ye D (2017) A belief propagation-based method for task allocation in open and dynamic cloud environments[J]. Knowl Based Syst 115:123–132
29. Zheng Y, Jeon B, Xu D et al (2015) Image segmentation by generalized hierarchical fuzzy C-means algorithm[J]. J Intell Fuzzy Syst 28(2):961–973
30. Fu Z, Wu X, Guan C et al (2016) Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement[J]. IEEE Trans Inf Forensics Secur 11(12):2706–2716
31. Gao X, Hoi S C H, Zhang Y et al (2014) SOML: sparse online metric learning with application to image retrieval. In: Twenty-eighth AAAI conference on artificial intelligence. AAAI Publications
32. Chen Z, Chen Y, Gao X et al (2015) Unobtrusive sensing incremental social contexts using fuzzy class incremental learning[C]// International Conference on data mining (ICDM). IEEE, 71–80
33. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications[J]. Neurocomputing 70(1):489–501
34. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks[C]// International Joint Conference on Neural Networks. IEEE 2:985–990
35. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes[J]. IEEE Trans Neural Networks 17(4):879–892
36. Schmidt WF, Kraaijveld MA, Duin R P W (1992) Feedforward neural networks with random weights[C]// International Conference on Pattern Recognition. IEEE, 1–4
37. Huang G, Huang GB, Song S et al (2015) Trends in extreme learning machines: a review[J]. Neural Netw 61:32–48
38. Huang GB, Zhou H, Ding X et al (2012) Extreme learning machine for regression and multiclass classification[J]. IEEE Trans Syst Man Cybern Part B (Cybernetics) 42(2):513–529
39. An S, Liu W, Venkatesh S (2007) Face recognition using kernel ridge regression[C]//International Conference on Computer Vision and Pattern Recognition. IEEE, 1–7
40. Saunders C, Gammerman A, Vovk V (1998) Ridge regression learning algorithm in dual variables[C]//ICML 98:515–521
41. Liu H, Qin J, Sun F et al (2016) Extreme kernel sparse learning for tactile object recognition[J]. IEEE Trans Cybern 99. doi:10.1109/TCYB.2016.2614809
42. Hu L, Chen Y, Wang S et al (2016) Less Annotation on Personalized Activity Recognition Using Context Data[C]//International Conference on Ubiquitous Intelligence and Computing(UIC) 327–332.
43. Liu H, Liu Y, Sun F (2015) Robust exemplar extraction using structured sparse coding[J]. IEEE Trans Neural Netw Learn Syst 26(8):1816–1821
44. Liu H, Guo D, Sun F (2016) Object recognition using tactile measurements: kernel sparse coding methods[J]. IEEE Trans Instrum Meas 65(3):656–665
45. Liu H, Yu Y, Sun F et al (2016) Visual-tactile fusion for object recognition[J]. IEEE Trans Autom Sci Eng
46. Liang NY, Huang GB, Saratchandran P et al (2006) A fast and accurate online sequential learning algorithm for feedforward networks[J]. IEEE Trans Neural Netw 17(6):1411–1423
47. Wang X, Han M (2014) Online sequential extreme learning machine with kernels for nonstationary time series prediction[J]. Neurocomputing 145:90–97
48. Zhou X, Liu Z, Zhu C (2014) Online regularized and kernelized extreme learning machines with forgetting mechanism[J]. Math Probl Eng 2014:938548. doi:10.1155/2014/938548
49. Zhou XR, Wang CS (2016) Cholesky factorization based online regularized and kernelized extreme learning machines with forgetting mechanism[J]. Neurocomputing 174:1147–1155
50. Scardapane S, Comminiello D, Scarpiniti M et al (2015) Online sequential extreme learning machine with kernels[J]. IEEE Trans Neural Netw Learn Syst 26(9):2214–2220
51. Deng WY, Zheng QH, Wang ZM (2014) Cross-person activity recognition using reduced kernel extreme learning machine[J]. Neural Netw 53:1–7
52. Guo L, Hao JH, Liu M (2014) An incremental extreme learning machine for online sequential learning problems[J]. Neurocomputing 128:50–58

53. Hager WW (1989) Updating the inverse of a matrix[J]. SIAM Rev 31(2):221–239
54. Henderson HV, Searle SR (1981) On deriving the inverse of a sum of matrices[J]. SIAM Rev 23(1):53–60
55. Barshan B, Yüksek MC (2014) Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units[J]. Comput J 57(11):1649–1667
56. Altun K, Barshan B (2010) Human activity recognition using inertial/magnetic sensor units[C]//International Workshop on Human Behavior Understanding. Springer Berlin Heidelberg, 38–51