Pervasive and Mobile Computing 15 (2014) 200–214

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

b-COELM: A fast, lightweight and accurate activity recognition model for mini-wearable devices

Lisha Hu [a,b,c,*], Yiqiang Chen [a,b], Shuangquan Wang [a,b], Zhenyu Chen [a,b,c,*]

[a] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, China

[b] Pervasive Computing Research Center, Institute of Computing Technology, Chinese Academy of Sciences, China

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

ARTICLE INFO

Article history:
Available online 28 June 2014

Keywords:
Wearable device
Activity recognition
Extreme learning machine
Proximal support vector machine

ABSTRACT

Various mini-wearable devices have emerged in the past few years to recognize activities of daily living for users. Wearable devices are normally designed to be miniature and portable. Models running on the devices inevitably face following challenges: low-computational-complexity, lightweight and high-accuracy. In order to meet these requirements, a novel powerful activity recognition model named b-COELM is proposed in this paper. b-COELM retains the superiorities (low-computational-complexity, lightweight) of Proximal Support Vector Machine, and extends the powerful generalization ability of Extreme Learning Machine in multi-class classification problems. Experimental results show the efficiency and effectiveness of b-COELM for recognizing activities of daily living.

1. Introduction

Wearable techniques focus on creating devices which can be worn, or embedded into clothes or accessories. Since wearable devices are much more promising and have potential in healthcare area, they occupy the main subfield of wearable devices. Wearable devices for healthcare improves users' health conditions by collecting information from their activities of daily living (ADL) and returning feedback to the users. Various wearable devices that can recognize ADL are emerging in the market recently. For example: wristbands, wristwatches and armbands.

Wearable devices are customarily designed to be miniature and portable so that they can be unobtrusive in users' daily lives (See Table 1). In order to recognize ADL for users, the models running inside mini-wearable devices should inevitably rise in several challenges. *First, the model should not require too much calculation*. This is because user's data are collected in real-time and need to be processed fast enough. *Second, the model should be lightweight as well.* Owing to the real-time requirements, in-memory computation needs to be performed in the model. However, the storage is limited due to hardware and space restrictions. So the model can recognize user's activities with less storage requirement. *At the end, the model should be accurate enough in activity recognition to guarantee the effectiveness of mini-wearable devices.*

User's sensor data are usually acquired constantly in real-time. The model should fulfill the activity recognition task in no time. So user can see his on-going states in real-time. For instance, a jogger should be able to see how many steps he

* Corresponding authors at: Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, China. Tel.: +86 18811324928.

E-mail addresses: hulisha@ict.ac.cn (L. Hu), chenzhenyu@ict.ac.cn (Z. Chen).

http://dx.doi.org/10.1016/j.pmcj.2014.06.002
1574-1192/

Table 1
Specifications of mini-wearable devices.

Type	Name	Weight (g)	Size (mm)
Wristband	Jawbone UP	19(S[a])	52 W[a](I[a]) × 35 H[a](I), P[a]: 141 (I) 66 W(O[a]) × 50 H(O), P:198 (O)
		22(M[a])	63 W(I) × 40 H(I), P: 167 (I) 76 W(O) × 54 H(O), P: 220 (O)
		23(L[a])	69 W(I) × 43 H(I), P: 180 (I) 81 W(O) × 56 H(O), P: 236 (O)
Wristband	Fitbit Flex	13.4(S) 14.6(L)	W:13.99, P: 140 (I)–176 (O) W:13.99, P: 161 (I)–209 (O)
Wristwatch	Pebble	150	33 W × 51 H × 10.2 T[a]
Wristwatch	Nike + SportWatch	66	38.1 W × 256.54 H × 15.24 T
Wristwatch	Basis B1	44	273 W × 273 H × 27 T
Wristwatch	Galaxy Gear	74	37 W × 57 H × 11.1 T
Armband	Fit Link	45.4	62 W × 55 H × 13 T

[a] 'S', 'M' and 'L' in Weight represent 'Small', 'Medium' and 'Large'. 'W', 'H', 'T' and 'P' in Size represent 'Width', 'Height', 'Thick' and 'Perimeter'. 'I' and 'O' in Size represent 'Inner' and 'Outer'.

has made when he is running, then to decide whether to continue or not. However, time spent in training process is often neglected since the model generation process is usually learned offline based on data of large amounts of users. Therefore, a pervasive activity recognition model is generated and utilized as the model of mini-wearable devices for all the users. For a specific user, the pervasive activity recognition model usually does not fit the user very well because of user's personalities. As is known to all, the best suitable model for user himself is the one which is retrained based on data of the user himself. However, such a model is hard to attain because of missing data with labels, time cost in training process etc. Even if we can get the user's personalized labeled data, the personalized model is still difficult to learn because the high time cost in training process is unbearable for the mini-wearable devices. Consequently, if we can cut down the time cost in training process to an acceptable level, we can learn the personalized model to better fit the user.

In order to deal with challenges above, in this paper, a new activity recognition model named b-COELM is proposed for mini-wearable devices. Analyses about computation complexity and memory requirement (Section 4.5) show that b-COELM owns superiorities in lightweight properties when compared with some existing approaches. Additionally, experimental results on four users' ADL datasets collected from real life scenarios validate the effectiveness and efficiency of the b-COELM model.

The remainder of this paper is structured as follows. Section 2 discusses related work. We review offline and online activity recognition systems from on-body sensors in Sections 2.1 and 2.2. We summarize machine learning techniques utilized in activity recognition literature in Section 2.3. Section 3 presents the b-COELM model generation procedures. Two related models of b-COELM are reviewed in Sections 3.1 and 3.2. After that, b-COELM is proposed in Section 3.3. Performance evaluation of the b-COELM model and comparison results with other related models are discussed in Section 4. Section 5 concludes the paper and discusses future extensions.

2. Related work

Activity recognition is a fast-growing area in ubiquitous computing domain, and a great deal of work has been proposed in the ubiquitous computing communities such as Ubicomp [1–13], IJCAI [14,15] and AAAI [16–18]. Recognizing ADL is a challenging but valuable task with many meaningful prospects such as healthcare, life pattern mining [19] etc. Since 2011, three international competitions named EvAAL (EValuating AAL systems through competitive benchmarking) have been organized with specific activity recognition aspects. Competitors from all over the world are gathered to compete with each other. A large number of high-quality activity recognition systems and models are generated in the competitions [20–28]. An overview of studies related to activity recognition is presented here from following three aspects.

2.1. Offline activity recognition from on-body sensor data

In recent years, extensive works have shown that knowledge of activity recognition can be mined from the on-body sensor data [29–33]. Bao et al. develop algorithms to detect physical activities from data acquired using five bi-axial accelerometers worn on different body parts. The recognition accuracy is over 80% on twenty daily activities by employing decision tree classifiers [29]. Berlin et al. propose an activity inference system to recognize leisure activities in continuous data for deployment in mood disorder research. A wrist-watch is utilized as wrist-worn data logger and the recorded data are processed relying on local signal features and motif discovery [30]. A two-stage recognition system for detecting arm gestures related to human meal intake is presented by Amft, in which data is derived from two wearable motion sensors on the subject's arm [31].

The activity recognition procedures described above are similar in nature. Data are all collected from on-body sensors and activity recognition is analyzed offline but not embedded in the wearable devices. Some representative online activity recognition systems in recent works are introduced in the following section.

2.2. Online activity recognition within wearable devices

With the development of mobile techniques, a wide range of high-precision sensors and extensive power processors have been incorporated into mobile phones. Existent researches show that humans' ADLs can be precisely mined relying on readings of the sensors deployed in mobile phones [34–39]. Several researches are able to monitor human activities in real-time. Physical activities are detected in [34–36,40] by a tri-axial accelerometer integrated in the phone. A system, named UbiFit Garden, is developed by utilizing on-body sensing, activity inference, novel personal and mobile display to encourage physical activity [37].

Except for mobile phones, mini-wearable devices also have been developed for real-time activity recognition [41,42]. Choudhury et al. utilize a wearable device named Mobile Sensing Platform (MSP) to recognize ADLs [41]. Another wearable sensor platform named eWatch, is employed by Maurer et al. for activity recognition and location recognition. Performances of Decision Tree (DT), k-Nearest Neighbor (k-NN), Naïve Bayes and the Bayes Net classifier are also evaluated and compared in [42].

In this paper, the b-COELM model is proposed for online activity recognition. Different with works introduced above, b-COELM model is constructed specially for the resource-constrained mini-wearable devices. The model should be extremely fast in both training and testing phases, be lightweight with as less storage requirement as possible, and be accurate enough to satisfy the activity recognition task.

2.3. Machine learning techniques in activity recognition

Human activity recognition is an important topic in pervasive computing community, and a great number of machine learning models have been utilized in this field. Zeng et al. propose a solution of learning Dynamic Bayesian Network (DBN) for human activity recognition, in which various domain knowledge are resorted to alleviate the problem of insufficient training data [43]. Experiments on single-subject's datasets show promising results in improving activity recognition. In particular, a DBN structure is learnt for each activity, which might be heavyweight and require a great deal of storage for multi-class activities. Gaikwad propose a Hidden Markov Models (HMMs) classifier to recognize human activities from video [44], in which threshold and voting based HMMs are utilized to recognize complex activities, whereas hybrids of HMM and Neural Network (NN) are applied for simple activity recognition. It is natural to suspect that this method might suffer in speed and accuracy during activity recognition due to the fuzziness of complexity. Oliver et al. present comparative analysis of HMMs and DBN for recognizing office activities. They conclude that each representation depends on different factors such as the available data resources, the nature of the data, the complexity of the domain etc. [45]. Other machine learning techniques such as Bayes classifiers [46], Boosting [47,48], DT [49,50], Support vector machine (SVM) [51–53] have been employed for classifying human's activity recognition in the literature.

SVM shows great potential in activity recognition compared with other machine learning methods [51–53]. SVM has been extensively used in small-scale binary classification problems due to its surprising classification capability [54,55]. Major limitations of SVM are its intensive computational complexities in training and testing processes. Though time limitation of SVM on the training stage can be avoided by learning the model offline, computations on the testing stage are still inevitable. In other words, it may take much time for predicting and requires extremely strong processing capability if SVM is employed as the activity recognition model in mini-wearable devices, as large numbers of inner-product computations need to be processed on the prediction phase. Proximal support vector machine (PSVM) [56] and least squares support vector machine (LS-SVM) [57] are proposed to compensate the shortage of SVM in time by solving a least squares optimization problem with equality constraints. PSVM retains comparable generalization ability of SVM with extremely faster learning speed, and can be considered as the ideal classifier for mini-wearable devices. It is natural to choose PSVM as the activity recognition model for mini-wearable devices. However, PSVM can only be utilized in binary classification problem whereas user's ADL are multi-class most of the time.

Several techniques [58–66] have been proposed to employ SVM or PSVM to deal with multiclass classification problems. For instance: methods of solving several small-scale binary classification problems utilizing One-Versus-One, One-Versus-Rest or voting techniques, methods of solving a single large-scale classification problem etc. How to extend PSVM to multi-class classification scenarios while retaining the superiorities of PSVM in binary classification situation is still a promising research field. Descriptions of SVM and PSVM are listed in Appendix A.

Extreme learning machine (ELM) [67–69] is proposed as a unified framework of solving binary classification, multi-class classification and regression problems. ELM is comparable to SVM in terms of accuracy with extremely fast learning speed. Authors in [70] proposed a constraint-optimization-based extreme learning machine (COELM) in order to extend ELM to kernel learning, as well as to provide a unified solution for ELM, PSVM and LS-SVM.

b-COELM is presented in this paper by adding bias in the optimization problem of COELM. Explicit descriptions of ELM, COELM and the proposed b-COELM are presented in Section 3.

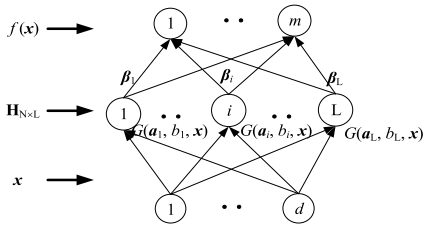

Fig. 1. SLFN.

3. Model design

3.1. ELM

Extreme learning machine (ELM) [67–69] is originally proposed for single hidden layer feedforward network (SLFN) with the purpose of solving both classification and regression problems. Only the classification problem is considered in this paper. We briefly review the structure of SLFN below.

The structure of SLFN (Fig. 1) consists of three layers from bottom to top: input layer, hidden layer and output layer. d input nodes in the input layer correspond to instance's d-dimensional features $\left(\boldsymbol{x} \in \mathbf{R}^{\mathrm{d}}, \boldsymbol{x} = (x_1, \ldots, x_{\mathrm{d}})^{\mathrm{T}}\right)$. Inputs are weighted and then processed by certain activation function g $(\boldsymbol{x})$ in the hidden layer. After that, the d-dimensional vector $\boldsymbol{x}$ in the input layer is mapped into the L-dimensional vector $(\mathrm{G}(\boldsymbol{a}_1, b_1, \boldsymbol{x}), \ldots, \mathrm{G}(\boldsymbol{a}_{\mathrm{L}}, b_{\mathrm{L}}, \boldsymbol{x}))^{\mathrm{T}}$ in the hidden layer. G $(\boldsymbol{a}_i, b_i, \boldsymbol{x})$ is the output of the ith additive hidden node and is given by Eq. (1). At the end, a m-dimensional vector $\boldsymbol{f}(\boldsymbol{x})$ is attained in the output layer by linear transformation on L-dimensional vector (Eq. (2)). m nodes in the output layer correspond to m classes.

$$G(\boldsymbol{a}_i, b_i, \boldsymbol{x}) = g(\boldsymbol{a}_i \cdot \boldsymbol{x} + b_i), \quad \boldsymbol{a}_i \in \mathbf{R}^d b_i \in \mathbf{R} \tag{1}$$

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\boldsymbol{a}_i, b_i, \boldsymbol{x}), \quad \boldsymbol{\beta}_i \in \mathbf{R}^m. \tag{2}$$

The learning algorithm of ELM utilizes a finite set of labeled instances for training: $\left\{\left(\boldsymbol{x}_j, \boldsymbol{t}_j\right) | \boldsymbol{x}_j \in \mathbf{R}^{\mathrm{d}}, \boldsymbol{t}_j \in \mathbf{R}^{\mathrm{m}}, j = 1, \ldots, N\right\}$, in which $\boldsymbol{x}_j$ is a vector of features and $\boldsymbol{t}_j$ is the label vector of $\boldsymbol{x}_j$ (see Fig. 1). In the training phase, each instance $\boldsymbol{x}_j$ is put into SLFN as the vector in the input layer, $\boldsymbol{t}_j$ is the expected output in the output layer. For each label vector $\boldsymbol{t}_j$ corresponding to the instance $\boldsymbol{x}_j$, one single element $\boldsymbol{t}_{jk}$ in the m-dimensional vector $\boldsymbol{t}_j$ is "1" representing the label of instance $\boldsymbol{x}_j$ being k, the other $m - 1$ elements are "−1". The values of all parameters $(\boldsymbol{a}_i, b_i, i = 1, \ldots, L)$ in hidden layer are randomly given. Variables $\left(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{\mathrm{L}}\right)$ between the hidden layer and the output layer can be calculated by:

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

where

$$\mathbf{H} = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_1) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ G(\boldsymbol{a}_1, b_1, \boldsymbol{x}_N) & \cdots & G(\boldsymbol{a}_L, b_L, \boldsymbol{x}_N) \end{bmatrix}_{N \times L} = [\boldsymbol{h}(\boldsymbol{x}_1), \ldots, \boldsymbol{h}(\boldsymbol{x}_N)]_{L \times N}^{T},$$

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} G(\boldsymbol{a}_1, b_1, \boldsymbol{x}) \\ \vdots \\ G(\boldsymbol{a}_L, b_L, \boldsymbol{x}) \end{bmatrix}_{L \times 1}, \quad \boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times m}, \quad \mathbf{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_N^T \end{bmatrix}_{N \times m}. \tag{4}$$

The least-squares solution $\boldsymbol{\beta}^*$ with minimal norm can be analytically solved using the Moore–Penrose "generalized" inverse:

$$\boldsymbol{\beta}^* = \mathbf{H}^{\dagger}\mathbf{T}. \tag{5}$$

The output function of ELM is: $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\beta}^{*\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})$.

3.2. COELM

Constrained-optimization-based extreme learning machine (COELM) [70] is proposed with the purpose of extending ELM with kernel learning. The classification problem of COELM is formulated as [70]:

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{\mathrm{C}}{2}\sum_{i}^{N}\left\|\boldsymbol{\xi}_i\right\|^2 \\ &\text{s.t. } \boldsymbol{h}(\boldsymbol{x}_i) \cdot \boldsymbol{\beta} = \boldsymbol{t}_i - \boldsymbol{\xi}_i, \quad i = 1, \ldots, N. \end{aligned} \tag{6}$$

Meanings of the notations are the same as in Section 3.1. The regularization factor C is the trade-off between two principles of training error minimization and margin maximization, and needs to be tuned properly. **I** represents the identity matrix. The solution $\boldsymbol{\beta}^*$ of COELM can also be analytically determined as [70]:

$$\boldsymbol{\beta}^* = \begin{cases} \mathbf{H}^T \left(\dfrac{\mathbf{I}}{\mathrm{C}} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}, & \mathrm{N} < L \\ \left(\dfrac{\mathbf{I}}{\mathrm{C}} + \mathbf{H}^T\mathbf{H} \right)^{-1} \mathbf{H}^T\mathbf{T}, & \mathrm{N} > L. \end{cases} \tag{7}$$

The output function of COELM is: $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\beta}^{*\mathrm{T}}\boldsymbol{h}(\boldsymbol{x})$.

3.3. b-COELM

For ELM in Section 3.1, the optimal output function $\boldsymbol{f}(\boldsymbol{x})$ can achieve minimal errors in training set based on the equation constraint in Eq. (3). For COELM in Section 3.2, with the help of $\boldsymbol{\xi}_i$ added into equation constraints in Eq. (6), the optimal output function $\boldsymbol{f}(\boldsymbol{x})$ can achieve minimal errors in training set even if fluctuations occurred in instances in the training set. A generalization form of COELM (named b-COELM) is proposed by adding the bias "$\boldsymbol{b}$" into output function ($\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{b}$) and equation constraints ($\boldsymbol{h}(\boldsymbol{x}_i)\cdot\boldsymbol{\beta} + \boldsymbol{b} = \boldsymbol{t}_i - \boldsymbol{\xi}_i i = 1, \ldots, N$). COELM can be recognized as a special case ($\boldsymbol{b} = \mathbf{0}$) of b-COELM. Moreover, other superiorities (strongly convexity and non-essential requirement of Mercer's positive definition condition) can be achieved if we add $\|\boldsymbol{b}\|^2$ into the objective function [56]. Therefore, it is natural to suppose that b-COELM can also inherit these superiorities by adding the bias "$\boldsymbol{b}$" into the constraints and the objective function.

A new model named b-COELM is presented in this paper that can be employed for both binary and multi-class classification problems in mini-wearable devices. b-COELM is formulated as:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{\xi}} \frac{1}{2}\left(\|\boldsymbol{\beta}\|^2 + \|\boldsymbol{b}\|^2\right) + \frac{\mathrm{C}}{2}\sum_{i}^{N}\left\|\boldsymbol{\xi}_i\right\|^2 \\ &\text{s.t. } \boldsymbol{h}(\boldsymbol{x}_i)\cdot\boldsymbol{\beta} + \boldsymbol{b} = \mathbf{t}_i - \boldsymbol{\xi}_i, \quad i = 1, \ldots, N. \end{aligned} \tag{8}$$

Notations are explained in Sections 3.1 and 3.2. Based on the KKT theorem, to train b-COELM is equivalent to solving its dual optimization problem:

$$D = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\|\boldsymbol{b}\|^2 + \frac{\mathrm{C}}{2}\sum_{i}^{N}\left\|\boldsymbol{\xi}_i\right\|^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\boldsymbol{\alpha}_{i,j}\left(\boldsymbol{\beta}_{:,j}^T\boldsymbol{h}(\boldsymbol{x}_i) + \boldsymbol{b}_j - \boldsymbol{t}_{i,j} + \boldsymbol{\xi}_{j,i}\right) \tag{9}$$

where each Lagrange multiplier $\boldsymbol{\alpha}_{i,:}$ (the ith row of $\boldsymbol{\alpha}$) is corresponding to the instance $\boldsymbol{x}_i$. The KKT conditions are attained as follows:

$$\begin{aligned} &\frac{\partial D}{\partial \boldsymbol{\beta}_{:,j}} = \boldsymbol{\beta}_{:,j} - \sum_{i=1}^{N}\boldsymbol{\alpha}_{i,j}\boldsymbol{h}(\boldsymbol{x}_i) = 0, \quad \forall j \to \boldsymbol{\beta} = \mathbf{H}^T\boldsymbol{\alpha} \\ &\frac{\partial D}{\partial \boldsymbol{b}_j} = \boldsymbol{b}_j - \sum_{i}^{N}\boldsymbol{\alpha}_{i,j} = 0, \quad \forall j \to \boldsymbol{b} = \boldsymbol{\alpha}^T\mathbf{1}_{N\times 1} \\ &\frac{\partial D}{\partial \boldsymbol{\xi}_{j,i}} = C\boldsymbol{\xi}_{j,i} - \boldsymbol{\alpha}_{i,j} = 0, \quad \forall i,j \to \boldsymbol{\xi}_i = \frac{1}{C}\boldsymbol{\alpha}_{i,:}^T, \ \forall i \\ &\frac{\partial D}{\partial \boldsymbol{\alpha}_{i,j}} = \boldsymbol{\beta}_{:,j}^T\boldsymbol{h}(\boldsymbol{x}_i) + \boldsymbol{b}_j - \boldsymbol{t}_{i,j} + \boldsymbol{\xi}_{j,i} = 0, \quad \forall i,j \to \boldsymbol{\beta}^T\boldsymbol{h}(\boldsymbol{x}_i) + \boldsymbol{b} - \boldsymbol{t}_i + \boldsymbol{\xi}_i = \mathbf{0}, \ \forall i \end{aligned} \tag{10}$$

where $\boldsymbol{\beta}_{:,j}$ represents the jth column of $\boldsymbol{\beta}$. Then we have:

$$\begin{aligned} &(\mathbf{H}^T\boldsymbol{\alpha})^T\boldsymbol{h}(\boldsymbol{x}_i) + \boldsymbol{\alpha}^T\mathbf{1}_{N\times 1} - \boldsymbol{t}_i + \frac{1}{C}\boldsymbol{\alpha}_{i,:}^T = 0, \quad \forall i = 1.\ldots, N \\ &\boldsymbol{\alpha}^T\mathbf{H}\mathbf{H}^T + \boldsymbol{\alpha}^T\mathbf{1}_{N\times N} - \mathbf{T}^T + \frac{1}{C}\boldsymbol{\alpha}^T = \mathbf{0} \\ &\boldsymbol{\alpha}^T\left(\mathbf{H}\mathbf{H}^T + \frac{1}{C}\mathbf{I}_{N\times N} + \mathbf{1}_{N\times N}\right) = \mathbf{T}^T \\ &\left(\mathbf{H}\mathbf{H}^T + \frac{1}{C}\mathbf{I}_{N\times N} + \mathbf{1}_{N\times N}\right)\boldsymbol{\alpha} = \mathbf{T} \end{aligned} \tag{11}$$

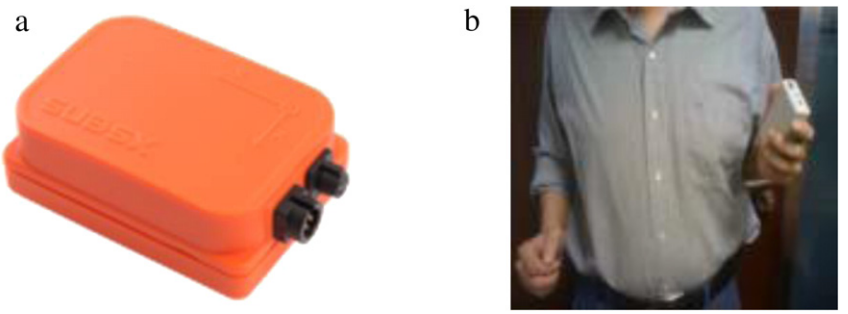

Fig. 2. Devices for data collection.

where $\mathbf{1}_{N\times 1}$ represents a $N \times 1$ vector of "1", $\mathbf{I}$ is the identity matrix. So, the optimal solution is:

$$\boldsymbol{\alpha}^* = \left(\mathbf{H}\mathbf{H}^T + \frac{1}{C}\mathbf{I}_{N\times N} + \mathbf{1}_{N\times N}\right)^{-1}\mathbf{T}. \tag{12}$$

According to results in Eq. (3), we have:

$$\boldsymbol{\beta}^* = \mathbf{H}^T\boldsymbol{\alpha}^*, \qquad \boldsymbol{b}^* = \boldsymbol{\alpha}^{*T}\mathbf{1}_{N\times 1}. \tag{13}$$

The output function of b-COELM is:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\beta}^{*T}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{b}^*. \tag{14}$$

The relations of b-COELM and PSVM are discussed in Appendix B. The algorithm of b-COELM is as follows.

Algorithm 1. b-COELM

Given N instances $\boldsymbol{x}_i$ in $\mathbf{R}^d$ with its label $\boldsymbol{t}_i$, choose a kernel function $k(\cdot,\cdot)$, Penalty parameter and the kernel parameter are assigned to appropriate values. We generate the b-COELM classifier as follows.

1. Define $\boldsymbol{\alpha}^*$ by (12) where $\mathbf{H}\mathbf{H}^T$ is equal to the $N \times N$ kernel matrix in which the value in the ith row and jth column of the kernel matrix is $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$, and $\mathbf{T}$ represents the $N \times m$ matrix in which each row of $\mathbf{T}$ is $\boldsymbol{t}_i$.
2. Define $\boldsymbol{\beta}^*$ and $\boldsymbol{b}^*$ by (13).

The output function $\boldsymbol{f}(\boldsymbol{x})$ is attained by (14).

4. Performance evaluation

4.1. Data collection & preprocessing

A human motion tracker named XSens MTx (Fig. 2(a)) is utilized to collect the data of a tri-axial accelerometer and a tri-axial gyroscope in the experiment. The device is embedded into a white box and held in the left hand by participants (Fig. 2(b)). All collected data is transmitted to a personal computer (PC) and all the data preprocessing and analysis are done in MATLAB R2013b on a PC with 32bit Windows 7 operating system, 4 GB Random Access Memory (RAM) and Intel(R) Core(TM) i5-3210M CPU.

The sampling rate of accelerometer is set to 100 Hz. Four participants (A–D) are recruited to perform six daily activities, which are stationary, walking, running, going upstairs, going downstairs and falling down.

For the data series of each axis, mean filtering is done to eliminate noise, using a sliding window of length 5. After that, sensor readings at each sample point are synthesized into one dimension according to Eq. (15).

$$sample_i = \sqrt{x_i^2 + y_i^2 + z_i^2}. \tag{15}$$

4.2. Feature extraction

The length of sliding window is set as 256 (i.e. 2.56 s) with 50% overlap between consecutive windows for activities: stationary, walking, running, going upstairs and going downstairs. The size of window and overlap percentage is chosen because of the following two reasons. First, at least one cycle of the above activities is statistically contained in the window and no cycles are lost and separated into two sub-parts; second, the value is set to be the nth power of 2 so that data in

Table 2
Feature definitions used for activity recognition.

Index	Feature	Description
1–3	1st, 2nd and 3rd Quartile	Three points dividing ordered signals into four equal parts
4	Mean	Average value of samples in window
5	Standard deviation (STD)	Square root of sample value with mean removed
6	Energy	Square of norm
7	Mean crossing rate	Rate of times signal crossing the mean value
8	Spectrum peak position	Position of spectrum peak value
9	Spectrum peak value	Maximum value of FFT series
10–13	Four PSD statistic features	Amplitude, STD, skewness, kurtosis

Table 3
Number of samples in different classes for different persons.

Dataset	Stationary	Walking	Running	Going upstairs	Going downstairs	Falling down
A[a]	393	399	390	344	342	36
B[a]	441	456	455	449	493	36
C[a]	447	343	413	421	397	36
D[a]	399	334	364	301	262	36

[a] A (B, C or D) represents the dataset of participant A (B, C or D).

Table 4
The optimal parameters for each method in different dataset.

Dataset	MPSVM		BR-MPSVM		COELM		b-COELM	
	C	g	C	g	C	g	C	g
A	256	0.0625	256	0.5	16	16	8	8
B	1024	0.0313	128	1	2	4	8	64
C	32	0.0625	128	1	64	16	64	16
D	1024	0.0625	64	1	16	16	32	32

a window can be easily employed in Fast Fourier Transform (FFT) algorithm. For the falling down activity, one window of same length is intercepted manually for each fall as it is not a continuous activity.

For each window, 13 features are extracted for both accelerometer and orientation readings. There are 26 features in all. All the feature types are listed in Table 2. In order to eliminate the scaling effects among different features, all the features are normalized using the z-score normalization algorithm [71]. In all, we get 7987 samples of six classes from the four people. The detailed information of the activity recognition dataset is described in Table 3. Four activity recognition tasks are generated corresponding to the four people.

4.3. Parameter selection

We evaluate the performance of b-COELM by employing it to recognize the human's ADL. b-COELM is compared with three multi-class classification models (MPSVM, BR-MPSVM and COELM).

Owning to the limited configuration of the personal computer, MPSVM and BR-MPSVM cannot be running on the whole dataset. For each person, 200 instances are randomly picked out from each of the five classes (Stationary, Walking, Running, Going upstairs, Going downstairs); Instances in the Falling down class are all picked out. All the instances picked out constitute a small-sized multi-class dataset for each person. In all four small-sized multi-class datasets corresponding to the four persons are generated for validation.

For each dataset, 60% of instances are randomly chosen from it to constitute the training set for model generation. Remaining instances compose the testing set. Gaussian kernel ($k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp(-g\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$) is utilized for each model. In order to equally compare the recognition abilities of the four models, two parameters (C and g) need to be tuned to find the optimal values for each problem.

Grid search technique is employed to fulfill the parameter tuning task. Ten different values $\left\{2^1, 2^2, \ldots, 2^9, 2^{10}\right\}$ are tried for the regularization factor C, and twenty-one values $\left\{2^{-10}, 2^{-9}, \ldots, 2^9, 2^{10}\right\}$ are tried for the kernel parameter g. Therefore, the optimal pair of (C, g) is chosen from a total of 210 pairs of (C, g). We employ a 2-fold cross-validation technique in the parameter tuning process. Fig. 3 shows the testing accuracy of b-COELM with different parameter pairs on dataset A. We can see that different values of regularization factor C have little impact on performance of b-COELM. The optimal parameter pair $\left(2^3, 2^6\right)$ is pointed out in Fig. 3. The optimal parameter values for each model are listed in Table 4.

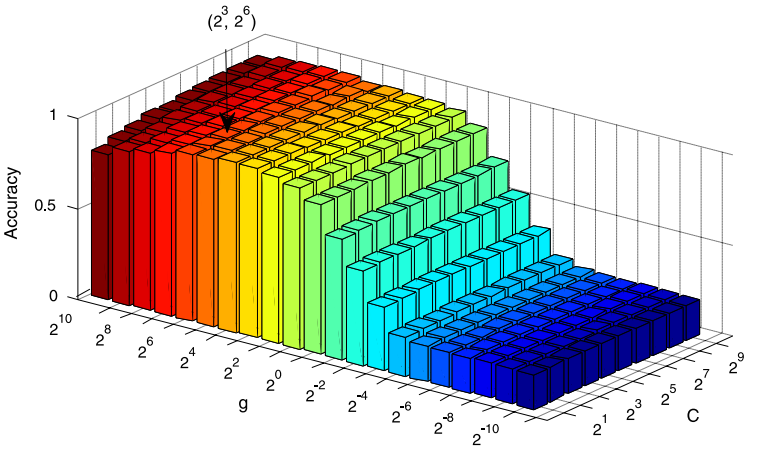

Fig. 3. Grid search for the optimal parameter pair (C, g) of b-COELM on dataset A.

Table 5
Performance comparisons of four models on training time.

Dataset	MPSVM	BR-MPSVM	COELM	b-COELM
A	3.3141	43.4449	0.0936	0.1029
B	3.0786	41.6839	0.0915	0.1073
C	3.0556	39.3322	0.0974	0.117
D	3.0298	43.1784	0.095	0.1041

Table 6
Performance comparisons of four models on testing time.

Dataset	MPSVM	BR-MPSVM	COELM	b-COELM
A	1.1816	5.7962	0.0235	0.0229
B	1.1862	5.9974	0.0231	0.0237
C	1.1754	5.6305	0.0362	0.022
D	1.7365	5.8912	0.0227	0.025

4.4. Experimental results comparisons

The optimization problems of four methods (MPSVM, BR-MPSVM, COELM, b-COELM) are employed on the whole training set with their corresponding optimal parameters. The whole procedure is conducted for ten times and results are averaged. At the end, the training and testing time are collected for comparison (Tables 5 and 6). The testing accuracies of the four models are also collected for comparison (Fig. 4).

Seen from the results above, MPSVM is weak both in speed and accuracy compared with the other three models. Although BR-MPSVM is effective in generalization ability, it takes too much time in training and testing procedures. b-COELM basically attains the best predictive ability. Besides, the time consumption in the training or testing procedures of b-COELM and COELM is roughly the same and negligible compared with MPSVM and BR-MPSVM (Tables 5 and 6).

According to the four groups of results in Fig. 4 we can see that b-COELM attains the highest prediction ability most of the time. Except in the first group, the testing accuracy of b-COELM is a little bit lower than that of BR-MPSVM. However, with the superiority of b-COELM in speed, b-COELM is still much more competitive than the BR-MPSVM model.

According to the comparison results of b-COELM and COELM we can conclude that: the generalization ability of COELM can be further improved by adding the bias term in the objective function and the constraints. In conclusion, b-COELM attains the highest predictive ability with extremely fast training and testing speed.

We also evaluated the predictive ability of each model on different data sources. All the samples are divided into four subsets corresponding to different persons. Among them, samples in three subsets are utilized for training and fourth subset for testing. The whole training–testing procedure executes four times in which each person's data is employed as the training set. Experimental results can be seen in Fig. 5. Comparing the results in Figs. 4 and 5 we can see that, the testing accuracy decreases at least 10% for each model in Fig. 5. As training and testing samples are from different sources (persons) and different people have different styles for each activity, which makes the data mining task extremely tough for each model. Inner combinations between sample features and the activity label are mined in different way for the four models. In Fig. 5 we can see that, compared with three models (MPSVM, BR-MPSVM, COELM) above, b-COELM can achieve the highest accuracy except "ABC–D" in which b-COELM is slightly lower than BR-MPSVM. In the same way, b-COELM has extremely

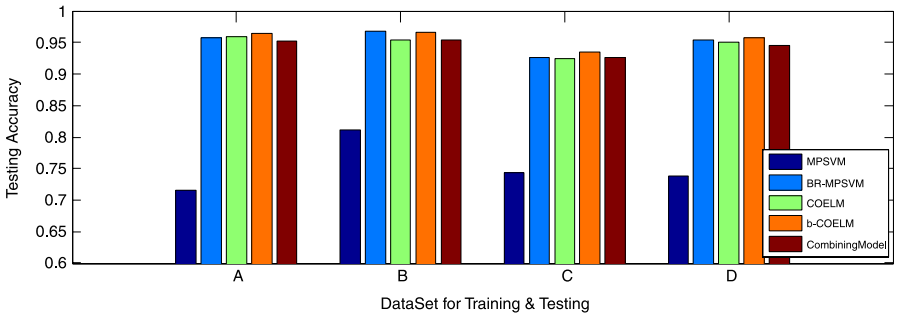

Fig. 4. Performance comparisons of four models on testing accuracy. A(B, C or D) represents each model is learned and tested on the data of person A (B, C or D).

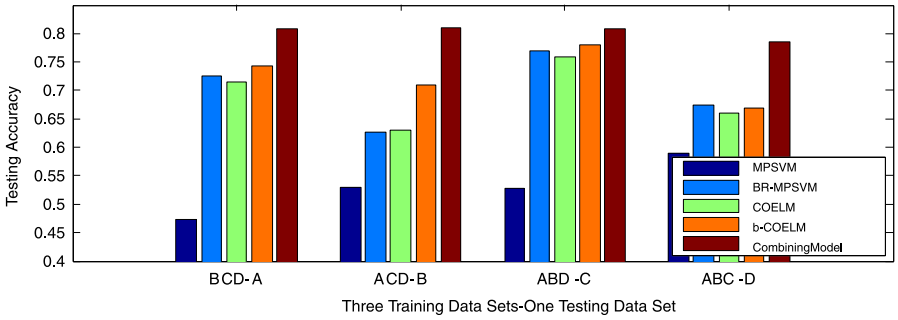

Fig. 5. Performance comparisons of four models on testing accuracy. ABC–D (ACD–B, ABD–C or BCD–A) represents each model is learned on the data of person A, B and C, tested on the data of person D (B, C or A).

faster training and testing speed than the BR-MPSVM model (see Tables 5 and 6). Compared with COELM, b-COELM can get higher accuracy in Fig. 5, which means that b-COELM gains higher generalization ability by adding the bias.

In Fig. 4, each model is trained on 60% of a single participant's data, and tested on the remaining 40%. Since data for training and testing are from the same data source (the single participant) with the same personality, results in Fig. 4 represent the learning ability of each model on the personalized data. In Fig. 5, however, data for training and testing are from different participants. For instance: "ABC–D" means data of participants A, B and C are for training the model, and data of participant D are for testing the model. Therefore, results in Fig. 5 represent the generalization ability of each model, how well each model mines the regular patterns from data of same activity but different participants.

In Figs. 4 and 5, we also list results of the "Combining Model" as a baseline for other four models. Actually, the "Combining Model" here is not a single model, but can be recognized as a mixture of eighteen models with the highest testing accuracy (last row in Table 7) of eighteen models' results on each dataset (underlined numbers in Table 7). Weka 3.6 [72] is utilized here to run all the eighteen models above. As can be seen in Table 5, accuracies of RBFNetwork, Dagging and LIBSVM constitute the results of "Combining Model".

Comparing b-COELM with the "Combining Model" in Figs. 4 and 5 we can see that b-COELM works slightly better than the "Combining Model" in Fig. 4 when 60% of a single participant's data are utilized for training and the remaining 40% for testing. Whereas b-COELM has lower accuracies than the "Combining Model" in Fig. 5 in circumstances that instances for training and testing are from different participants. b-COELM can achieve excellent generalization ability when instances for training are heterogeneous and representative. Nevertheless, b-COELM still can achieve higher or similar testing accuracies than the other three models (MPSVM, BR-MPSVM and COELM) in Fig. 5.

Besides, b-COELM is compared with RBFNetwork, Dagging and LIBSVM models on each dataset, and results are listed in Table 8. We can see that b-COELM can gain the highest accuracy when instances for training and testing are from the same persons. It means that b-COELM has stronger learning ability compared with the other three models (RBFNetwork, Dagging and LIBSVM). Meanwhile, RBFNetwork, Dagging and LIBSVM gains higher accuracy than b-COELM in the rest circumstances, none of them is the best in all cases.

4.5. Complexity analysis and comparisons

(1) Time complexity

Time costs of b-COELM, COELM, MPSVM and BR-MPSVM are analyzed and compared in Table 9. An inverse matrix of $N \times N$ (number of instances in the training set) is to be calculated in the training procedure of b-COELM, which is similar

Table 7
Eighteen models in Weka and the "Combining Model" on testing accuracy.

NO	Model	DataSet for training & testing				Training dataset–testing dataset			
		A	B	C	D	BCD–A	ACD–B	ABD–C	ABC–D
1	RBFNetwork	92.17%	94.69%	88.99%	90.72%	78.34%	78.52%	78.34%	**78.54%**
2	DMNBtext	77.97%	81.45%	78.16%	83.29%	68.38%	68.42%	68.53%	68.43%
3	NaiveBayes	83.38%	88.99%	86.67%	84.15%	75.85%	75.72%	75.67%	75.95%
4	NaiveBayesSimple	83.38%	85.96%	86.47%	84.06%	75.83%	75.71%	75.62%	75.76%
5	LWL	73.33%	81.45%	80.29%	80.77%	63.60%	63.57%	63.85%	63.32%
6	AdaBoostM1	38.55%	38.55%	38.55%	38.55%	41.27%	41.28%	41.28%	41.28%
7	ClassifyViaClustering	38.26%	38.55%	38.45%	38.55%	40.57%	40.77%	40.73%	40.75%
8	Dagging	87.83%	88.50%	84.25%	84.93%	***80.82%***	***80.95%***	***80.87%***	77.82%
9	MultiBoostAB	38.55%	38.55%	38.55%	38.55%	41.27%	41.28%	41.28%	41.28%
10	LIBSVM	***95.29%***	***95.39%***	***92.66%***	***94.49%***	72.48%	65.70%	76.74%	68.98%
11	Vote	19.23%	19.23%	19.23%	19.23%	21.04%	21.04%	21.04%	21.04%
12	HyperPipes	81.26%	82.32%	88.50%	77.87%	65.66%	65.68%	65.63%	65.77%
13	VFI	85.41%	89.37%	87.54%	84.54%	73.34%	73.22%	73.47%	73.35%
14	ConjunctiveRule	38.55%	38.55%	38.55%	38.55%	41.27%	41.28%	41.28%	41.27%
15	DecisionTable	81.55%	85.89%	82.42%	85.12%	80.22%	80.39%	80.35%	76.84%
16	OneR	61.06%	80.77%	70.92%	75.46%	68.66%	68.47%	68.47%	68.42%
17	DecisionStump	38.55%	38.55%	38.55%	38.55%	41.27%	41.28%	41.28%	41.28%
18	LADTree	90.24%	90.43%	87.54%	89.95%	79.93%	79.98%	79.74%	75.76%
MaxValue	**CombiningModel**	**95.29%**	**95.39%**	**92.66%**	**94.49%**	**80.82%**	**80.95%**	**80.87%**	**78.54%**

Table 8
Performance comparison of b-COELM and "Combining Model" members on testing accuracy.

NO	Model	Dataset for training & testing				Training dataset–testing dataset			
		A	B	C	D	BCD–A	ACD–B	ABD–C	ABC–D
1	RBFNetwork	92.17%	94.69%	88.99%	90.72%	78.34%	78.52%	78.34%	**78.54%**
2	Dagging	87.83%	88.50%	84.25%	84.93%	***80.82%***	***80.95%***	***80.87%***	77.82%
3	LIBSVM	95.29%	95.39%	92.66%	94.49%	72.48%	65.70%	76.74%	68.98%
4	b-COELM	***96.35%***	***96.55%***	***93.53%***	***95.72%***	74.28%	70.96%	78.10%	66.90%

Table 9
Time complexity analysis about four models of MPSVM, BR-MPSVM, COELM and b-COELM.

Model	Time complexity analysis
MPSVM	Solve m $N \times N$ inverse matrices
BR-MPSVM	Solve m $N \times N$ inverse matrices, m times iterative Newton Refinement procedures
COELM	Solve a single $N \times N$ inverse matrix
b-COELM	Solve a single $N \times N$ inverse matrix

to COELM. Therefore, the solutions of b-COELM can be attained analytically. Basically, MPSVM and BR-MPSVM are both 1-Versus-Rest PSVMs meaning that in all m PSVMs should be learned for either MPSVM or BR-MPSVM. It is natural to suspect that time cost in training MPSVM or BR-MPSVM might be at least m times slower than COELM and b-COELM. Except for the m-time training procedures issue, iterative Newton Refinement procedures also cost much time in the training phase of BR-MPSVM.

In Section 4.4, Tables 5 and 6 show experimental results of four models in training and testing time. We can see that COELM and b-COELM attain extremely faster speeds than the other two models in both training and testing procedures. Perhaps COELM is a little bit faster than b-COELM in training, since the bias in the formulation of b-COELM needs to be computed. Taking it by and large, b-COELM has comparable training speed with COELM. Because b-COELM is equal to PSVM for binary classification, PSVM owns the same training speed as b-COELM in the binary scenario. For multi-class classification scenarios, both MPSVM and BR-MPSVM are much slower than COELM and b-COELM.

(2) Space complexity

In order to measure memory usage and number of threads utilized for each method, we make an application named "Process Memory Monitor" by using C#. Interface is shown in Fig. 6. We can choose which process to monitor and the frequency of the monitoring process. Readings related to memories and thread numbers (Table 10 and Fig. 7) are dumped and saved in a text file. The frequency is set to be 1 s in our experiments.

Four classes of memory indicators are collected and shown in Table 10 for the above models when they are running in MATLAB. Memory usage when no function is running in MATLAB is recorded and excluded from the readings in Table 10. Descriptions about the four indicators are as follows.

- Memory (Private Working Set) is the amount of memory the process is using in the private working set, which is called *private RAM* in this paper.

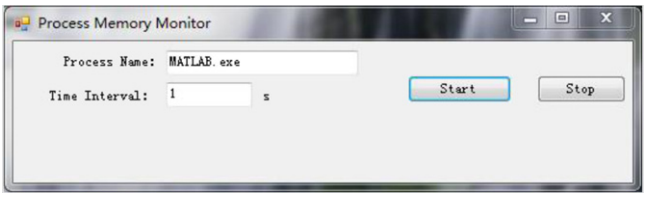

Fig. 6. Process & memory monitor.

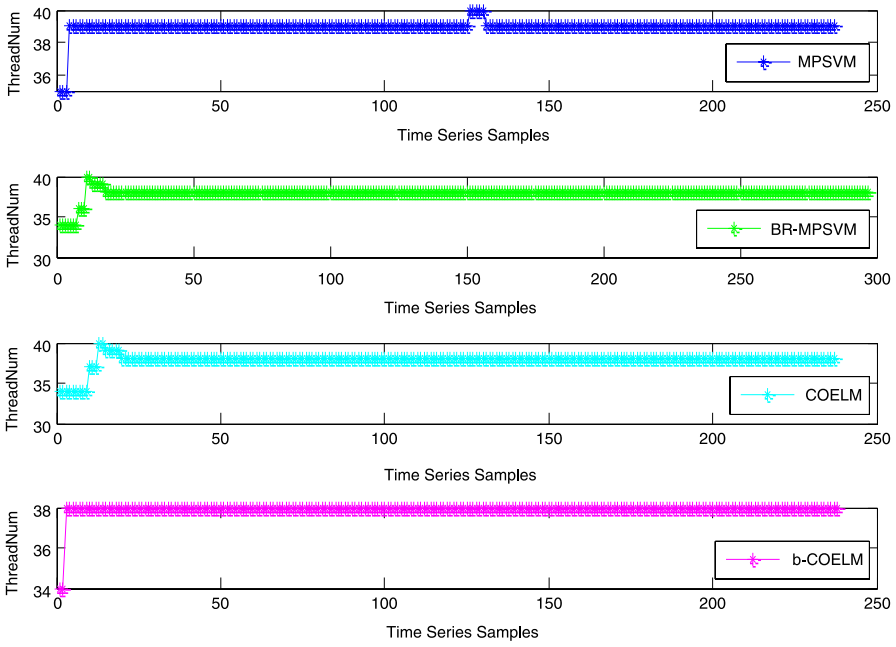

Fig. 7. Performance comparisons of four models on thread number.

Table 10
Performance comparisons of four models on memory usage.

Model	*RAM* (MB)		*Private RAM* (MB)		*Shared RAM* (MB)		*VM* (MB)	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
MPSVM	127.9621	69.9718	125.1432	70.023	2.8188	8.4182	144.2701	82.7893
BR-MPSVM	128.0534	72.9784	125.1322	72.512	2.9211	7.7565	134.2203	84.8734
COELM	45.2006	2.3572	42.6039	2.0851	2.5967	2.2705	50.2884	2.3273
b-COELM	40.8042	2.2306	38.2543	2.2148	2.5498	2.1541	33.6623	2.2668

- Memory (Shared Working Set) is the amount of memory the process is using that can be shared by other processes, which is called *shared RAM* in this paper.
- Memory (Working Set) is the sum of Memory (Private Working Set) and Memory (Shared Working Set), which is called *RAM* in this paper.
- Memory (Commit Size) is the amount of virtual memory that is reserved for use by the process, which is called *VM* in this paper.

We also compute the mean and Standard Deviation (STD) for above four indicators corresponding to the four models, results are shown in Table 10. From Table 10 we can see that: it cost about 40 MB *RAM* for COELM and b-COELM, about 128 MB *RAM* for MPSVM and BR-MPSVM models. Moreover, size of memory demanded for MPSVM and BR-MPSVM are rather unstable and pendulous (see *RAM* STD in Table 10). It means that additional 40 MB *RAM* is required stably to run COELM and b-COELM models, whereas the amount is extremely larger and turbulent for MPSVM and BR-MPSVM models. A similar phenomenon can be found in *VM* and *private RAM* usages, simply with different size and ranges. Moreover, COELM and b-COELM have relatively small amplitudes in *shared RAM* when compared with MPSVM and BR-MPSVM models (see *VM* STD in Table 10).

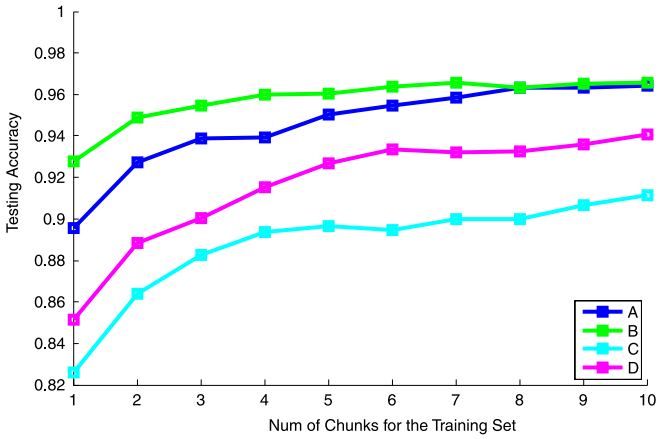

Fig. 8. Scalability of b-COELM. Numbers in *x* axis represent how many chunks in the training set is utilized for training a b-COELM model. Values in *y* axis represent the testing accuracy of the b-COELM model on the testing set.

(3) Threads

Since all models are running in the MATLAB software, a single process named MATLAB is generated to execute the program for each model. Moreover, number of threads created by the process is monitored for each model and shown in Fig. 7. When a function is started at first, several threads are created by the process to accomplish the computations. After that, the thread number becomes stable in the full running procedure. According to results in Fig. 7 we can see that: in all 38 threads are created for BR-MPSVM, COELM and b-COELM models, 39 for MPSVM model. Basically, number of threads corresponding to these four models does not make any difference.

4.6. b-COELM scalability analysis

In order to know how the generalization ability of b-COELM behaves when the size of instance for training is changing, the scalability of b-COELM is evaluated in this section. For each participant (A/B/C/D), instances collected from him/her are divided randomly into the training set and testing set as shown in Section 4.3. Instances in the training set are divided equally and randomly into 10 sequential chunks. Instances in the testing set are for evaluating the generalization ability. At the beginning, instances in the first chunk is utilized to train the b-COELM model, whose generalization ability (Testing Accuracy) is evaluated on the testing set. Then, instances in the second chunk is added into the training set and a new b-COELM model is learned and evaluated on the testing set as well. The whole procedures continue until all chunks of the participant have been added into the training set. Fig. 8 shows the experimental results of b-COELM for all the participants.

According to results in Fig. 8 we can see that: the testing accuracy of b-COELM increases with the number of chunks for training. In other words, the generalization ability of b-COELM can be improved if more instances are utilized in the training process. Moreover, the accuracy of the model grows fast at the beginning. After 5 or 6 chunks added into the training set, the incremental speed slows down and keeps static till the end. Besides, b-COELM models generated on different participant's instances have different generalization abilities.

5. Conclusion

Due to the portable and miniature properties of mini-wearable devices, several challenges (lightweight, low-computational-complexity and high-accuracy) have been faced in the model design phase for mini-wearable devices. In this paper, a new activity recognition model named b-COELM is proposed for mini-wearable devices. b-COELM model is constructed by adding the bias in optimization problem of COELM model. Superiorities of b-COELM are: (1) Theoretical analysis and experimental results have shown that b-COELM model is better than COELM model in both learning and generalization abilities, which explains that it is worthy of adding the bias. (2) As a novel multi-class proximal support vector machine (MPSVM), experimental results have shown that b-COELM is better than the existing MPSVM approaches in speed, storage requirement, learning and generalization abilities. (3) As a regular multi-class classification model, experimental results show that b-COELM can achieve higher learning ability than eighteen existing multi-class classification models, higher generalization ability than fifteen existing multi-class classification models. Therefore, b-COELM is both an effective and efficient activity recognition model for mini-wearable devices.

In the future, we plan to extend b-COELM model into the online mode when a single user's personal and informative data are collected one by one or chunk by chunk. User's data are employed to extend the pervasive b-COELM model into a personalized one which can preferably suit the user. Since we do not plan to retrain the model by both old data and new ones, the model should be updated adaptively to the personal data. During the updating process, new personalized data should

be recognized as more important than the various users' data. Once the data are collected and employed for updating the model, they will be discarded and only the model generated is reserved for next rounds of self-adaptation.

Acknowledgments

This work was supported by Natural Science Foundation of China under Grant No. 61173066, Beijing Natural Science Foundation under Grant No. 4144085, Strategic Emerging Industry Development Special Funds of Guangdong Province under Grant No. 2011912030, National Science and Technology Major Project under Grant No.2012ZX07205-005, and Open Project of Beijing Key Laboratory of Mobile Computing and Pervasive Device.

Appendix A

SVM

The goal of SVM is to look for a hyperplane which can correctly separate two classes of training instances (instances in the training set) with a maximal margin. The optimization problem of SVM is formulated as [54,55]:

$$\begin{aligned} &\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \mathrm{C}\sum_{i}^{N}\xi_{\mathrm{i}} \\ &\text{s.t. } t_i\left(\boldsymbol{\omega}\cdot\boldsymbol{\phi}\left(\boldsymbol{x}_{\mathrm{i}}\right)+b\right) \geq 1-\xi_i, \quad i=1,\ldots,N \\ &\xi_i \geq 0, \quad i=1,\ldots,N. \end{aligned} \tag{A.1}$$

Instance's label t_i is given to either "1" or "−1", corresponding to the positive and negative class respectively. $\boldsymbol{\phi}(\cdot)$ is a function which maps the instance $\boldsymbol{x}$ in $\mathbf{R}^{\mathrm{d}}$ onto $\boldsymbol{\phi}(\boldsymbol{x})$ in a higher dimensional feature space.

PSVM

Different from SVM in which instances are split up into two parts, PSVM classify instances by letting them cling to two parallel hyperplanes which are pushed apart as far as possible. The optimization problem of PSVM is formulated as [56]:

$$\begin{aligned} &\min_{\boldsymbol{\omega},b,\boldsymbol{\xi}} \frac{1}{2}\left(\|\boldsymbol{\omega}\|^2+b^2\right) + \frac{\mathrm{C}}{2}\sum_{i}^{N}\xi_{\mathrm{i}}^2 \\ &\text{s.t. } t_i\left(\boldsymbol{\omega}\cdot\boldsymbol{\phi}\left(\boldsymbol{x}_{\mathrm{i}}\right)+b\right) = 1-\xi_i, \quad i=1,\ldots,N. \end{aligned} \tag{A.2}$$

MPSVM

The MPSVM (1-Versus-Rest PSVM) and BR-MPSVM (Balanced and Refined 1-vs-rest PSVM) models are briefly reviewed here [66].

MPSVM is to learn a multi-class classifier by learning a set of binary PSVM classifiers utilizing 1-Versus-Rest techniques. Let the label of each instance in the ith ($i = 1, 2, \ldots, m,\ m > 2$) class be positive (+1), and the label of each instance in the other $m-1$ classes be negative (−1). Then all the instances constitute into the training dataset for binary classification problem. A PSVM classifier is learnt on this binary dataset. There are m PSVM classifiers are learnt in all. A new point is assigned to class s depending on which of the m half-spaces generated by the m classifiers it lies deepest in.

BR-MPSVM is a One-Versus-Rest PSVM with balancing and Newton refinement. Each instance is weighted by the reciprocal of the number of instances sharing the same label (positive or negative). The idea behind the Newton refinement is to move the plane learnt by the PSVM to minimize the misclassified instances.

Appendix B

Theorem. *Let the mapping functions $\boldsymbol{\phi}(\cdot)$ in PSVM and $\boldsymbol{h}(\cdot)$ in b-COELM be the same. The problem of b-COELM is equivalent to the problem of PSVM* [47] *in solving a binary classification problem.*

Proof. First of all, it is clear to see that two objective functions in Eqs. (B.2) and (1) are absolutely the same. Then we compare the constraints in (B.2) and (1). According to instances with positive or negative labels, constraints in (B.2) and (1) can be rewritten into constraints in (B.1) and (B.2). We compare them in cases individually.

$$\begin{aligned} &\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_i) = 1-\xi_i, \quad \forall i, t_i = 1 \\ &\boldsymbol{\omega}\cdot\boldsymbol{\phi}(\boldsymbol{x}_j) = -1+\xi_j, \quad \forall j, t_j = -1 \end{aligned} \tag{B.1}$$

$$\begin{aligned} &\boldsymbol{\beta}\cdot\boldsymbol{h}(\boldsymbol{x}_j) = -1-\xi_i, \quad \forall i, t_i = 1 \\ &\boldsymbol{\beta}\cdot\boldsymbol{h}(\boldsymbol{x}_j) = -1-\xi_j, \quad \forall j, t_j = -1 \end{aligned} \tag{B.2}$$

(1) For each positive instance $\boldsymbol{x}_i$ with its label $t_i = 1$, if the mapping functions $\boldsymbol{\phi}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are the same, then the corresponding constraints in (B.1) and (B.2) are exactly the same.
(2) For each negative instance $\boldsymbol{x}_j$ with its label $t_j = -1$, if the mapping functions $\boldsymbol{\phi}(\cdot)$ and $\boldsymbol{h}(\cdot)$ are the same, then the corresponding constraints in (B.1) and (B.2) are a bit different. Actually, the difference is: the value of ξ_j in (B.2) is the negative of the value of ξ_j in (B.1). Fortunately, only the square of ξ_j is considered in objective functions of (B.2) and (1), making this little difference irrelevant to the solution processes of two optimization problems.

In the end, we attain that if the mapping functions $\boldsymbol{\phi}(\cdot)$and $\boldsymbol{h}(\cdot)$ are the same, b-COELM (Eq. (1)) is equivalent to PSVM (Eq. (A.2)) in solving binary classification problems. This completes the proof.

According to theorem, we can conclude that: b-COELM can be considered as a new method of extending PSVM into multi-class classification problems.

References

[1] N.D. Lane, et al., Enabling large-scale human activity inference on smartphones using community similarity networks (CSN), in: Proceedings of the ACM Conference on Ubiquitous Computing, 2011, pp. 355–364.
[2] Y.Q. Chen, et al., Surrounding context and episode awareness using dynamic bluetooth data, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012,pp. 629–630.
[3] K. Ouchi, M. Doi, Indoor-outdoor activity recognition by a smartphone, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012, pp. 600–601.
[4] H. Lee, et al., Mobile posture monitoring system to prevent physical health risk of smartphone users, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012, pp. 592–593.
[5] J. Beltrń-Márquez, Activity recognition using a spectral entropy signature, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012, pp. 576–579.
[6] K. Yatani, K.N. Truong, Bodyscope: a wearable acoustic sensor for activity recognition, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012, pp. 341–350.
[7] J. Park, et al., Online pose classification and walking speed estimation using handheld devices, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012, pp. 113–122.
[8] Z.Y. Chen, et al., Inferring social contextual behavior from bluetooth traces, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2013, pp. 267–270.
[9] G. Spina, et al., COPDTrainer: a smartphone-based motion rehabilitation training system with real-time acoustic feedback, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2013, pp. 597–606.
[10] H. Hung, et al., Classifying social actions with a single accelerometer, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2013, pp.207–210.
[11] H. Gjoreski, et al., Ensembles of multiple sensors for human energy expenditure estimation, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2013, pp.359–362.
[12] Z.Y. Chen, et al., Contextsense: unobtrusive discovery of incremental social context using dynamic bluetooth data, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2014.
[13] R. Wang, et al., Studentlife: assessing mental health, academic performance and behavioral trends of college students using smartphones, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2014.
[14] V.W. Zheng, Qiang Yang, User-dependent aspect model for collaborative activity recognition, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 2011, pp. 2085–2090.
[15] Z. Zhao, et al., Cross-people mobile-phone based activity recognition, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 2011, pp. 2545–2550.
[16] X.Y. Gao, et al., SOML: sparse online metric learning with application to image retrieval, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
[17] Perera, J.F. Allen, Learning names for RFID-tagged objects in activity videos, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012, pp. 2449–2450.
[18] S. Abdullah, et al., Towards population scale activity recognition: a framework for handling data diversity, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012, 851–857.
[19] Z.Y. Chen, Mining individual behavior pattern based on significant locations and spatial trajectories, in: Proceedings of the International Conference on Pervasive Computing and Communications, 2012, pp. 540–541.
[20] S. Kozina, et al., Efficient activity recognition and fall detection using accelerometers, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 13–23.
[21] F. Palumbo, et al., Multisensor data fusion for activity recognition based on reservoir computing, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 24–35.
[22] M. Ángel Álvarez de la Concepción, et al., Activity recognition system using non-intrusive devices through a complementary technique based on discrete methods, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 36–47.
[23] J. Antonio Álvarez-García, Evaluating human activity recognition systems for AAL environments, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 131–136.
[24] L.M. Soria Morillo, et al., Activity recognition system using AMEVA method, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 137–147.
[25] J.H. Hong, et al., An activity recognition system for ambient assisted living environments, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 148–158.
[26] N. Li, et al., Visual experience for recognising human activities, in: Evaluating AAL Systems Through Competitive Benchmarking, Communications in Computer and Information Science, Vol. 386, 2013, pp. 173–185.
[27] P. Barsocchi, et al., RSSI localisation with sensors placed on the user, in: International Conference on Indoor Positioning and Indoor Navigation, 2010, pp.1–6.
[28] R. Guraliuc, et al., Limb movements classification using wearable wireless transceivers, IEEE Trans. Inform. Technol. Biomed. 15 (3) (2011) 474–480.
[29] L. Bao, S.S. Intille, Activity Recognition from User-Annotated Acceleration Data, in: Lecture Notes in Computer Science, vol. 3001, 2004, pp. 1–17.
[30] E. Berlin, K.V. Laerhoven, Detecting leisure activities with dense motif discovery, in: Proceedings of the ACM Conference on Ubiquitous Computing, 2012, pp. 250–259.
[31] Amft, et al., Detection of eating and drinking arm gestures using inertial body-worn sensors, in: Proceedings of the IEEE International Symposium on Wearable Computers, 2005, pp.160–163.

[32] K.V. Laerhoven, et al., Enabling efficient time series analysis for wearable activity data, in: Proceedings of the International Conference on Machine Learning and Applications, 2009, pp.392–397.
[33] D. Minnen, et al., Discovering characteristic actions from on-body sensor data, in: Proceedings of IEEE International Symposium on Wearable Computing, 2006, pp. 11–18.
[34] J.R. Kwapisz, et al., Activity recognition using cell phone accelerometers, ACM SIGKDD Explor. Newslett. 12 (2) (2010) 74–82.
[35] G. Bieber, et al., Activity Recognition for Everyday Life on Mobile Phones, in: Lecture Notes in Computer Science, vol. 5615, 2009, pp. 289–296.
[36] T. Brezmes, et al., Activity Recognition from Aaccelerometer Data on a Mobile Phone, in: Lecture Notes in Computer Science, vol. 5518, 2009, pp. 796–799.
[37] S. Consolvo, et al., Activity sensing in the wild: a field trial of UbiFit garden, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2008, pp. 1797–1806.
[38] Z.Y. Chen, et al., Unobtrusive sleep monitoring using smartphones, in: Proceedings of the 7th International ICST Conference on Pervasive Computing Technologies for Healthcare, 2013, pp. 145–152.
[39] Y. Xu, et al., Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns, in: Proceedings of the 17th ACM International Symposium on Wearable Computers, 2013, pp. 69–76.
[40] Z.Y. Chen, et al., Online sequential ELM based transfer learning for transportation mode recognition,in: Proceedings of the 6th IEEE International Conference on Cybernetics and Intelligent Systems, 2013, pp. 78–83.
[41] T. Choudhury, et al., The mobile sensing platform: an embedded activity recognition system, IEEE Trans. Pervasive Comput. 7 (2) (2008) 1268–1536.
[42] U. Maurer, et al., Location and Activity Recognition Using eWatch: A Wearable Sensor Platform, in: Lecture Notes in Computer Science, vol. 3864, 2006, pp. 86–102.
[43] Z. Zeng, Q. Ji, Knowledge Based Activity Recognition with Dynamic Bayesian Network, in: Lecture Notes in Computer Science, vol. 6316, 2010, pp. 532–546.
[44] MS. Kanchan Gaikwad, HMM classifier for human activity recognition, Internat. J. Comput. Sci. Eng. 2 (4) (2012) 27–36.
[45] N. Oliver, E. Horvitz, A Comparison of HMMs and Dynamic Bayesian Networks for Recognizing Office Activities, in: Lecture Notes in Computer Science, vol. 3538, 2005, pp. 199–209.
[46] Madabhushi, J.K. Aggarwal, A bayesian approach to human activity recognitio, in: Second IEEE Workshop on Visual Surveillance, 1999.
[47] Y.Q. Song, et al., Collaborative boosting for activity classification in microblogs, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 482–490.
[48] Reiss, et al., A competitive approach for human activity recognition on smartphones, in: Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013.
[49] J. Pärkkä, et al., Personalization algorithm for real-time activity recognition using PDA, wireless motion bands, and binary decision tree, IEEE Trans. Inform. Technol. Biomed. 14 (5) (2010) 1211–1215.
[50] J. Yang, Toward physical activity diary: motion recognition using simple acceleration features with mobile phones, in: Proceedings of the International Workshop on Interactive Multimedia for Consumer Electronics, 2009, pp. 1–10.
[51] H.Y. Zhao, Z.J. Liu, Human action recognition based on non-linear SVM decision tree, J. Comput. Inform. Syst. 7 (7) (2011) 2461–2468.
[52] D. Anguita, et al., Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine, in: Lecture Notes in Computer Science, vol. 7657, 2012, pp. 216–223.
[53] H.M. Qian, et al., Recognition of human activities using SVM multi-class classifier, Pattern Recognit. Lett. 31 (2010) 100–111.
[54] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[55] C. Cortes, V.N. Vapnik, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[56] G. Fung, O.L. Mangassarian, Proximal support vector machine classifiers, in: Proceedings of International Conference on Knowledge Discovery and Data Mining, 2001, pp. 77–86.
[57] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[58] J. Weston, C. Watkinis, Multi-class support vector machines, in: Proceedings of International Conference on European Symposium on Artificial Neural Networks, 1999.
[59] J.C. Plat, et al., Large margin DAG's for multiclass classification, Adv. Neural Inf. Process. Syst. (2000) 547–553.
[60] E.J. Bredensteiner, K.P. Bennett, Multicategory classification by support vector machines, Comput. Optim. Appl. 12 (1–3) (1999) 53–79.
[61] Y. Guermeur, Combining discriminant models with new multiclass SVMs, Pattern Anal. Appl. 5 (2) (2002) 168–179.
[62] C.W. Hsu, C.J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2) (2002) 415–425.
[63] Y.Z. Xing, et al., Multiclass least squares wavelet support vector machines, in: Proceedings of IEEE International Conference on Networking, Sensing and Control, April 6–8, 2008, pp. 498–502.
[64] T.V. Gestel, et al., Multiclass LS_ SVMs: moderated outputs and coding-decoding schemes, Neural Process. Lett. 15 (1) (2002) 45–58.
[65] M.R. Guarracino, A. Irpino, R. Verde, Multiclass generalized eigenvalue proximal support vector machines, in: Proceedings of International Conference on Complex, Intelligent and Software Intensive Systems, February 15–18, 2010, pp. 25–32.
[66] G.M. Fung, O.L Mangasarian, Multicategory proximal support vector machine classifiers, Mach. Learn. 59 (1–2) (2005) 77–97.
[67] G.B. Huang, et al., Extreme learning machine: theory and applications, Neurocomputing 7 (1–3) (2006) 489–501.
[68] G.B. Huang, et al., Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of IEEE International Joint Conference on Neural Networks, July 25–29, 2004, pp. 985–990.
[69] G.B. Huang, et al., Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Trans. Neural Netw. 17 (4) (2006) 879–892.
[70] G.B. Huang, et al., Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. 42 (2) (2012) 513–529.
[71] J.W. Han, et al., Data Mining: Concepts and Techniques, Morgan Kaufmann Publisher, 2000.
[72] M. Hall, et al., The WEKA data mining software: an update, ACM SIGKDD Explor. 11 (1) (2009) 10–18.