Pattern Recognition 78 (2018) 277–290

Contents lists available at ScienceDirect

Pattern Recognition

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

A novel random forests based class incremental learning method for activity recognition

Chunyu Hu[a,b,c], Yiqiang Chen[a,b,c,*], Lisha Hu[a,b,c,d], Xiaohui Peng[a,c]

[a] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[b] University of Chinese Academy of Sciences, Beijing, China
[c] Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing, China
[d] Hebei University of Economics and Business, Shijiazhuang, China

ARTICLE INFO

Article history:
Received 22 May 2017
Revised 13 December 2017
Accepted 24 January 2018
Available online 31 January 2018

MSC:
00-01
99-00

Keywords:
Class incremental learning
Activity recognition
Random forests

ABSTRACT

Automatic activity recognition is an active research topic which aims to identify human activities automatically. A significant challenge is to recognize new activities effectively. In this paper, we propose an effective class incremental learning method, named Class Incremental Random Forests (CIRF), to enable existing activity recognition models to identify new activities. We design a separating axis theorem based splitting strategy to insert internal nodes and adopt Gini index or information gain to split leaves of the decision tree in the random forests (RF). With these two strategies, both inserting new nodes and splitting leaves are allowed in the incremental learning phase. We evaluate our method on three UCI public activity datasets and compare with other state-of-the-art methods. Experimental results show that the proposed incremental learning method converges to the performance of batch learning methods (RF and extremely randomized trees). Compared with other state-of-the-art methods, it is able to recognize new class data continuously with a better performance.

1. Introduction

In recent years, many studies [1,2] have shown that the ability to perform activities of daily living (ADLs) is an important indicator of the health condition of human beings. Thus, many researchers have paid great attention to intelligent ADLs monitoring. For instance, motion patterns, such as walking, sit-to-stand, are proved to be closely associated with Cerebral Small Vessel Disease in [3]. In [4], the ability of finger movements is shown to be very useful in stroke detection. A significant part of intelligent ADLs monitoring is automatic human activity recognition. With different types of data acquisition devices, activity recognition can be divided into video-based activity recognition and sensor-based activity recognition. Video-based activity recognition [5] tries to identify human activities from video or image sequence, while sensor-based activity recognition makes use of the sensor readings from accelerometer, gyroscope, magnetometer and so on. With the advance of pervasive computing and machine learning, various sensors and intelligent methods are developed for automatic human activity recognition. For hardware platform, accelerometer, gyroscope, and magnetometer are the most common sensors used in different smart devices. Many researchers adopt these three types of sensor in their work [6,7], which have achieved superior performance. For recognition model, a lot of machine learning algorithms, such as support vector machine (SVM) [8,9], decision tree [10,11], bagging [12], have been successfully applied in activity recognition.

However, traditional batch learning methods [13,14] recognize activities with fixed models, which are unable to adapt to the dynamic changes of human behavior. Generally, there are two types of dynamic changes in activity recognition. One is that different people may carry out the same activity in a different manner, which results in dynamic changes in data distribution. The other is that people may learn new motion activities over time, which is usually classified as dynamic changes in class. When a new kind of activity is performed or the behavioral pattern changes over time, devices with preinstalled activity recognition models may fail to recognize new activities or even known activities enacted in a different manner. To adapt to the changes of activities, traditional batch learning methods require retraining the whole model from scratch. This will result in a great waste of time and memory.

Some incremental learning methods [15–18] have been proposed to address these problems. Different from batch learning, incremental learning or on-line learning methods update existing

* Corresponding author at: Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.
E-mail addresses: huchunyu@ict.ac.cn (C. Hu), yqchen@ict.ac.cn (Y. Chen), hulisha@ict.ac.cn (L. Hu), pengxiaohui@ict.ac.cn (X. Peng).

https://doi.org/10.1016/j.patcog.2018.01.025
0031-3203/

models with new knowledge. Thus, it requires less training time and smaller memory. Among existing incremental learning or online learning methods, many variants of RF have been proposed for incremental activity learning and achieved remarkable performance [15,16]. However, most variants of online RF [15,16] focused on the dynamic changes of data distribution. Few of them address the class incremental learning problem.

In this paper, we propose a novel class incremental learning method—CIRF, which is specially designed for recognizing new activity classes. It is easier to implement CIRF with low computational cost while achieving comparable performance than other state-of-the-art algorithms [19]. The main contributions of this paper are listed as follows.

(1) We present a novel separating axis theorem (SAT) based splitting strategy with which changes of parent node will not lead to information loss. With this splitting strategy, insertion of new nodes is allowed without reconstructing a subtree.
(2) By combining SAT-based splitting strategy and traditional splitting strategy, we propose a novel class incremental learning method—CIRF. Two splitting strategies are adopted to make the tree grow reasonably in the case of class incremental learning.
(3) We applied the CIRF to an activity recognition problem. Experiments on three public datasets (Daily and Sports Activities Data Set (DSADS) [20], OPPORTUNITY [21] and Human Activity Recognition Using Smartphones Data Set (HARUSDS) [9]) from UCI are conducted to evaluate the performance of CIRF. Experimental results show that the proposed CIRF achieves comparable performance with offline learning methods in both accuracy and robustness. Moreover, the results indicate that CIRF performs better than other incremental learning algorithms on the test accuracy in most cases.

The rest of this paper is organized as follows. Section 2 mainly examines the related work on activity recognition and RF algorithm. Section 3 describes our proposed method. Section 4 presents the performance evaluation and comparisons with the other approaches on three public datasets. Section 5 concludes this paper with the future work.

2. Related work

Automatic activity recognition in a dynamic environment is a significant challenge. With dynamic changes in data distribution and activity class, traditional activity recognition methods may fail to respond with a correct recognition result. To address these challenges, many variants of online learning methods and incremental learning methods are proposed. In this section, we present an overview of traditional activity recognition methods as well as dynamic activity recognition methods.

2.1. Traditional activity recognition methods

Machine learning is known as an effective tool for activity recognition. A number of algorithms have been presented for applications such as health monitoring or elderly care.

Decision tree. Decision tree is a simple but very efficient tool for data mining. It constructs a hierarchical structure to map activity labels to corresponding leaves. In [10], CART and ID3 are used to detect ADLs, which can be further used for activity recognition based health monitoring. In [11], Zhao et al. propose a TransEMDT model to solve the cross-people mobile-based activity recognition problem.

Support Vector Machine (SVM). SVM [8,9] has also been successfully used in human activity recognition although it does not provide a set of human-understandable rules [22]. A basic implementation of SVM is limited to binary classification. However, some SVM-based techniques [23,24] have been developed to address this issue. In [8], SVM and decision tree are combined to form a multiclass classifier. It is successfully applied to recognize activities from video sequences. In [9], multi-class hardware-friendly SVM is implemented on a smartphone to monitor user's daily activities.

Artificial neural network (ANN). ANN [25], which has achieved good performance in many applications, is also an effective method for activity recognition. As a single hidden layer feed-forward neural network, extreme learning machine (ELM) [26] shows good performance in both classification accuracy and training time. In [27,28], different variants of ELM are proposed and tested on activity recognition datasets. Experimental results show that b-constrained-optimization-based extreme learning machine (b-COELM) is a model with low-computational-complexity and two-stage weighted ELM provides a good solution for fall detection of the elderly.

Ensemble classifier. Ensemble classifiers are commonly constructed with several homogeneous or heterogeneous individual learners to gain high accuracy. They have shown a great potential in activity recognition. In [12] and [29], bagging and boosting are successfully embedded in different activity recognition systems. RF is one of the most popular ensemble learning methods. It is easy to implement RF with low computational cost while achieving comparable performance with other state-of-the-art algorithms [19]. In [19], Feng et al. design an ensemble classifier system based on RF, which achieves high accuracy with less time cost.

2.2. Dynamic activity recognition methods

Online learning. Online learning is efficient in dealing with high-speed continuous data flows. Instances are processed sequentially and available in a few passes [30]. The classification models are trained iteratively with limited training time and limited memory.

In [31], Tudor et al. propose an Online Active Learning framework to improve the accuracy of personalized models. Instead of accumulating annotated training sets for building activity models, online active learning annotation decision heuristic is adopted to reduce the workload for annotation. In [32], Sun et al. put forward an online multitask learning method for personalized and continuous activity recognition. In addition to automatic activity recognition, this method is able to learn interesting task relationship from real-world data. In [33], a new online classifier, named Live Bayes, is presented to identify users based on their activities. With a small training set, Live Bayes is able to balance out the gap among imbalanced classes.

Particularly, variants of online RF have been proposed to handle massive amounts of data. ORF-Saffari [15] is an effective online RF algorithm, which has achieved a good performance in video tracking, image recognition, and other application fields. In [34] , ORF-Saffari is combined with active learning to construct a cross-subjects based activity recognition model. Experimental results show that it is efficient in building personalized activity recognition classifier. Similar to ORF-Saffari, ORF-Denil [35] constructs decision tree incrementally from an empty one. In the constructions of ORF-Saffari [15] and ORF-Denil [35], new data points are passed from the root to leaves. When the statistics information in a leaf node satisfy some criteria, the best split is chosen and two children nodes are added to this node. Mondrian forests [16] is another effective online RF algorithm, in which the split selection is

independent of labels. In addition to the update of existing leaves, the insertions of new nodes are also allowed in Mondrian forests.

Online learning methods are efficient in handling dynamic changes in data distribution. However, the classes are usually assumed to be known and fixed beforehand [36].

Incremental learning. Different from online learning, incremental learning methods process instances one by one or batch by batch [30]. There are usually fewer restrictive conditions for incremental learning. For example, the classes are not required to be known and fixed beforehand [36]. In [37], Zhou defines that online learning is a special case of incremental learning. Incremental learning updates the model when it is necessary, while online learning updates the model once a new instance is available.

In [38], hybrid user-assisted incremental model adaptation is proposed to recognize interleaved and concurrent activities in a dynamic smart-home scenario. Various data annotation strategies are adopted to reduce user intervention in a dynamic environment. In [17], incremental learning and active learning are adopted to learn with evolving data stream. Characteristics of clusters are extracted to enable the incremental and continuous learning. In [18], Wang et al. propose an incremental activity recognition classifier based on probabilistic neural networks (PNN) and adjustable fuzzy clustering (AFC). AFC is used to divide training instances into clusters and endow PNN with incremental learning ability. These incremental learning methods mainly focus on the changes of data distributions.

Some methods have considered class incremental learning cases. One-shot learning and zero-shot learning are two extreme forms of transfer learning. They are common used for learning new classes. In [39], Li et al. first propose the one-shot learning concept. It makes possible that learning knowledge about a new category from just one samples. In [40], Palatucci et al. introduce zero-shot learning to build a classifier that can recognize new classes, even without training examples for those particular classes. Both one-shot learning and zero-shot learning aims to share explicit knowledge among classes so as to reduce the amount of training data. Different from them, we aim at efficient integration of new classes. Ripple Down Rules (RDR) [41], which is designed for maintaining medium to large rule-based knowledge systems, is efficient in acquiring knowledge incrementally. It allows exception rules to be added to override previous knowledge, which results in the facts that the child rule's conclusion supersedes that of the parent rules. In [42], RDR is successfully applied in medical images segmentation. However, RDR may end up with knowledge repeated in many places throughout the tree [43]. In [44], You et al. propose a novel class incremental learning method for multi-label learning. It is able to model newly arrived labels with the help of the knowledge learned from past labels. In [45], Xu et al. present a new method to explore new views when collecting and preprocessing data from different view. Different from [44] and [45], we pay more attention to the scenario in which new class data is independent of the past labels. No obvious label relationship can be utilized in the recognition of new class data. In [46], Zhao et al. present a novel class incremental learning method, named class incremental extreme learning machine (CIELM), for human activity recognition. With CIELM, new class activities are able to be recognized dynamically. However, as the scale of class increases, the performance of CIELM fluctuates widely in accuracy (see Section 4.4). Thus, a robust and high accuracy class incremental learning method is needed urgently.

3. Our method

In this section, we describe the proposed CIRF algorithm. To recognize new class with an existing model, we introduce Axis-Aligned minimum Bounding Box (AABB) [47] into RF. With AABB, we present a separating axis theorem based splitting strategy, which enables us to insert new nodes into the decision tree without changing the structure of current decision tree. In CIRF, the insertion of new nodes and the split in leaves are both allowed. Before introducing our method, we firstly give a brief introduction to the RF.

3.1. Random forests

RF is an extension of the decision tree ensemble. Feature randomization and bootstrap sampling are combined to contribute to a good generalization performance. Information gain [48] (see Eq. (1)) and Gini index [49] (see Eq. (3)) are the most common used splitting criteria. The output of the RF is a combination of individual decision trees using different combining rules, such as simple averaging [50], weighted averaging [51], majority voting [52] and weighted voting [53]. Given a training set, $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, x_i is a K-dimensional feature vector of the ith sample and y_i is the label of x_i. The whole training set D is used to train a decision tree. A is the number of classes in the initial training set. The kth dimensional feature has M possible values. The proportion of the jth class data in D is represented by p_j. D^m is the set of samples with the mth possible value in feature k. The information gain and Gini index can be calculated by:

Information gain

$$Gain(D, k) = Entropy(D) - \sum_{m=1}^{M} \frac{|D^m|}{D} Entropy(D^m) \tag{1}$$

Information entropy

$$Entropy(D) = -\sum_{j=1}^{A} p_j \log p_j \tag{2}$$

Gini index

$$GiniIndex(D, k) = \sum_{m=1}^{M} \frac{|D^m|}{D} Gini(D^m) \tag{3}$$

Gini value

$$Gini(D) = -\sum_{j=1}^{A} p_j \left(1 - p_j\right) \tag{4}$$

3.2. Class incremental random forests

3.2.1. Motivation

Traditional decision trees usually grow following a top-down procedure. Once a split is determined, it is very difficult to modify the structure. The selection of attribute and cut-point for the split will further influence the construction of subtrees. For example, modification on internal node $S3$ leads to reconstruction of its subtree which consists of $L3$ and $L4$ (see in Fig. 1(a)). In the construction of existing RF variants (ORF-Saffari [15] and ORF-Denil [35]), new data points are passed from the tree root to leaves. When the statistics information in a leaf node satisfy some criteria, the best split is chosen and two children nodes are added to this node (see in Fig. 1(c)). However, with such an online tree growing mechanism, split occurs only in the leaf nodes. This will result in a redundant tree structure, especially in the class incremental learning case. To address this issue, we propose a novel tree growing strategy, in which both inserting new nodes and splitting leaves are allowed.

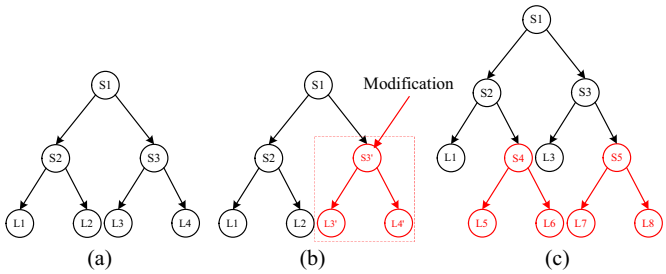

Fig. 1. Modification of an existing tree (a) structure of a decision tree; (b) modification on an internal node; (c) split on leaves.

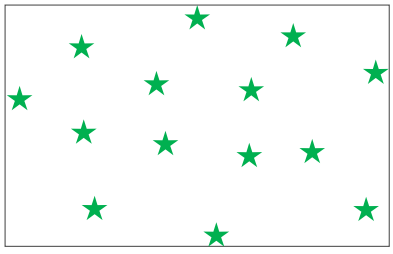

Fig. 2. The bounding box of a point set.

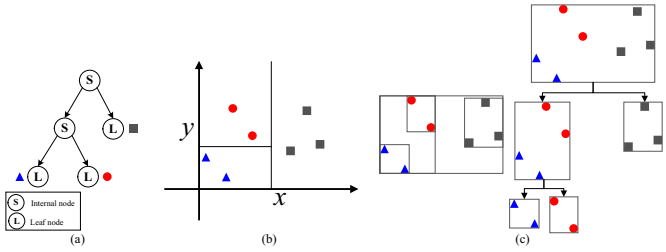

Fig. 3. A 2-dimensional decision tree. (a) a decision tree (b) the 2-dimensional space partition of a decision tree (c) the AABB corresponding to the tree in (a).

3.2.2. Axis-aligned minimum bounding box

In geometry, the minimum bounding box [54] for a set of points is the box with the smallest surrounding space in which all the discrete points lie. It is commonly used for detecting the intersection. If there is no intersection between two minimum bounding boxes, we can conclude that there is no overlap between corresponding point sets. Due to such a characteristic, minimum bounding box has been successfully applied in various areas, such as collision detection [55] and ray tracing [56]. To enable the insertion of a parent node, we introduce the minimum bounding box into the construction of an incremental learning decision tree. It is used for finding the intersection between an existing node and a new data batch. The most common minimum bounding box are AABB, arbitrarily oriented minimum bounding box, and object-oriented minimum bounding box [57]. In consideration of the axis-aligned splitting characteristics of RF, we adopt AABB in our method.

The AABB is the minimum bounding box with the edges of the box parallel to the coordinate axes. In a two-dimensional case, the AABB of a given point set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ is illustrated in Fig. 2. In this paper, AABB is formulated as: $B = \{\boldsymbol{R}_{\min}; \boldsymbol{R}_{\max}\}$, where $\boldsymbol{R}_{\min} = \min(\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_n)$, $\boldsymbol{R}_{\max} = \max(\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots; \boldsymbol{x}_n)$. It consists of the minimum value and the maximum value of each attribute, which is represented by $\boldsymbol{R}_{\min}$ and $\boldsymbol{R}_{\max}$ respectively. We can represent a decision tree in the form of AABB in which each node corresponding to an AABB. Figure 3 (b) shows the space partition of a decision tree and Fig. 3(c) shows the AABBs corresponding to the tree in Fig. 3(a).

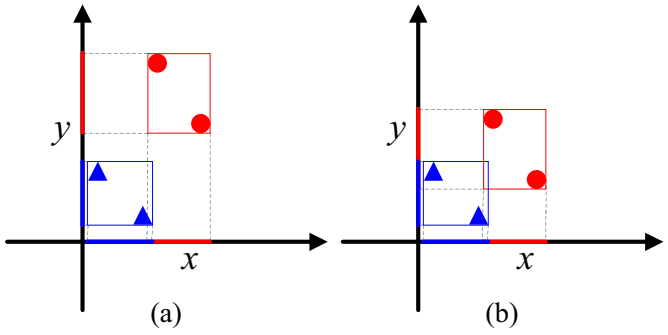

Fig. 4. Illustration of separating axis theorem. (a) two separated bounding boxes with their respective projections on x-axis and y-axis; (b) two intersected bounding boxes with their respective projections on x-axis and y-axis.

To facilitate the incremental learning ability of RF, we define two types of AABB: the AABB for the node in decision tree (NAABB) and the AABB for data batch (DAABB). The NAABB is used to record the boundary of the data points falling in this node (see Fig. 3(c)). While DAABB is used to describe the range of the data batch in each dimension (see Fig. 2).

3.2.3. Separating axis theorem based splitting strategy

Using the AABB, we are able to represent a decision tree in the form of hierarchical nested bounding boxes. When new class arrives, finding an appropriate attribute and position to split are significant issues for class incremental learning. In this paper, we introduce the Separating Axis Theorem (SAT) to solve this problem. The SAT is commonly used to determine whether two convex shapes are intersecting or not. In our work, it is used to find the splitting attribute and position.

Theorem 1. *Separating Axis Theorem [58]: If two convex objects are not penetrating, there exists an axis on which the projection of the objects will not overlap.*

Take two-dimensional AABB as an example. We illustrate the separating axis theorem in Fig. 4. Two AABBs are created according to the class and projected onto x-axis and y-axis respectively. The projections of two AABBs are drawn in corresponding colors. In Fig. 4(a), we can see that there is no overlap in their projections on the y-axis. According to SAT, the y-axis is the separating axis for these two AABBs. In Fig. 4(b), two AABBs are intersecting on both x-axis and y-axis, and we can't find a separating axis on which there is no overlap in their projections.

To find the splitting attribute and value, we first find the separating axis of the AABBs. The separating axis with the maximum margin is chosen as the splitting attribute (described in Eq. (5)). It is because that the separating axis with the maximum margin is more robust to the local disturbance of the training samples than the other axes. Take Fig. 5 as an example, with a larger margin, y-axis is selected as the splitting attribute. With the splitting attribute fixed, the splitting value is chosen as the midpoint of the interval on the splitting attribute (described in Eq. (6)). As shown in Fig. 5, $y = b$ is determined as the splitting value. The splitting strategy based on the SAT is illustrated in Fig. 5.

$$sAtt = \begin{cases} \arg\max\left(\boldsymbol{R}^1_{\min} - \boldsymbol{R}^2_{\max}\right), \\ \quad \max\left(\boldsymbol{R}^1_{\min} - \boldsymbol{R}^2_{\max}\right) > \max\left(\boldsymbol{R}^2_{\min} - \boldsymbol{R}^1_{\max}\right) \\ \arg\max\left(\boldsymbol{R}^2_{\min} - \boldsymbol{R}^1_{\max}\right), \qquad otherwise \end{cases} \tag{5}$$

Where $\boldsymbol{R}^i_{\min}$ and $\boldsymbol{R}^i_{\max}$ denote the lower boundary and upper boundary of the ith bounding box, respectively. $sAtt$ is the splitting

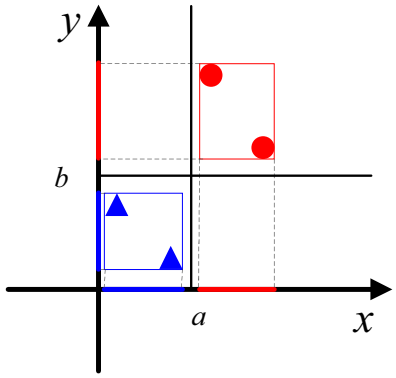

Fig. 5. Illustration of the splitting strategy based on the SAT.

attribute.

$$splitValue = \begin{cases} \frac{\boldsymbol{R}^1_{\min,sAtt}+\boldsymbol{R}^2_{\max,sAtt}}{2}, \max\left(\boldsymbol{R}^1_{\min}-\boldsymbol{R}^2_{\max}\right) \\ >0, \max\left(\boldsymbol{R}^1_{\min}-\boldsymbol{R}^2_{\max}\right) > \max\left(\boldsymbol{R}^2_{\min}-\boldsymbol{R}^1_{\max}\right) \\ \frac{\boldsymbol{R}^2_{\min,sAtt}+\boldsymbol{R}^1_{\max,sAtt}}{2}, \max\left(\boldsymbol{R}^2_{\min}-\boldsymbol{R}^1_{\max}\right) \\ >0, \max\left(\boldsymbol{R}^1_{\min}-\boldsymbol{R}^2_{\max}\right) \le \max\left(\boldsymbol{R}^2_{\min}-\boldsymbol{R}^1_{\max}\right) \end{cases} \quad (6)$$

Where *splitValue* is the splitting value on the splitting attribute.

Assume triangle and circle represent two different classes of data in the dataset $D1$. The number of triangle is $n1$, and the number of circle is $m1$. The splitting attribute and value chosen by SAT are y-axis and $y = b$, respectively. Then, the entropy gain of the splitting strategy based on the SAT is $Gain(D1, splitAtt1) = Entropy(D1)$.

As we know, $Gain(D,k) = Entropy(D) - \sum_{m=1}^{M} \frac{|D^m|}{D} Entropy(D^m) \le Entropy(D)$. Thus, we can conclude that the split determined by the SAT-based splitting strategy ($sAtt = y$, $splitValue = b$) is optimal and suitable for the partition of decision tree.

3.2.4. Class incremental tree growing mechanism

In order to accommodate to new class data, we propose a new incremental growing mechanism for the decision tree. By introducing this mechanism, we can insert a new node into an existing tree or split a leaf node without changing the original structure of the decision tree. The detailed construction process of tree growing mechanism is illustrated in Fig. 6.

Fig. 6(a) is an initial decision tree built with RF algorithm. As shown in Fig. 6, when new class data arrives, there will be four cases. In case 1, there is no intersection between the AABBs of the data batch and the tree node (Fig. 6(b)). Whether the node is a leaf or not, we insert a new node (S2 in Fig. 6(b)) as the parent of the current node (S1 in Fig. 6(b)) and a brother node (L3 in Fig. 6(b)) is created to be the branch for new class data. In this case, the parent node (S2 in Fig. 6(b)) is created to cover the range of new data batch and the current node (S1 in Fig. 6(b)). The split of the newly created parent node (S2 in Fig. 6(b)) is determined by the SAT-based splitting strategy. In case 2, a new batch data arrives in an internal node and the AABB of the data batch is within the range of the node's AABB (S1 in Fig. 6(c)). Then, no change is required for the current node. New data is partitioned by existing split (S1 in Fig. 6(c)) and passed to corresponding children nodes (L1 in Fig. 6(a)). In case 3, the new class data is passed into leaf nodes and still unable to be separated from known class data (Fig. 6(d)). In order to gain an optimal decision tree structure, we adopt information entropy or Gini index to split the leaf nodes. With such a strategy, we can also update the decision tree with

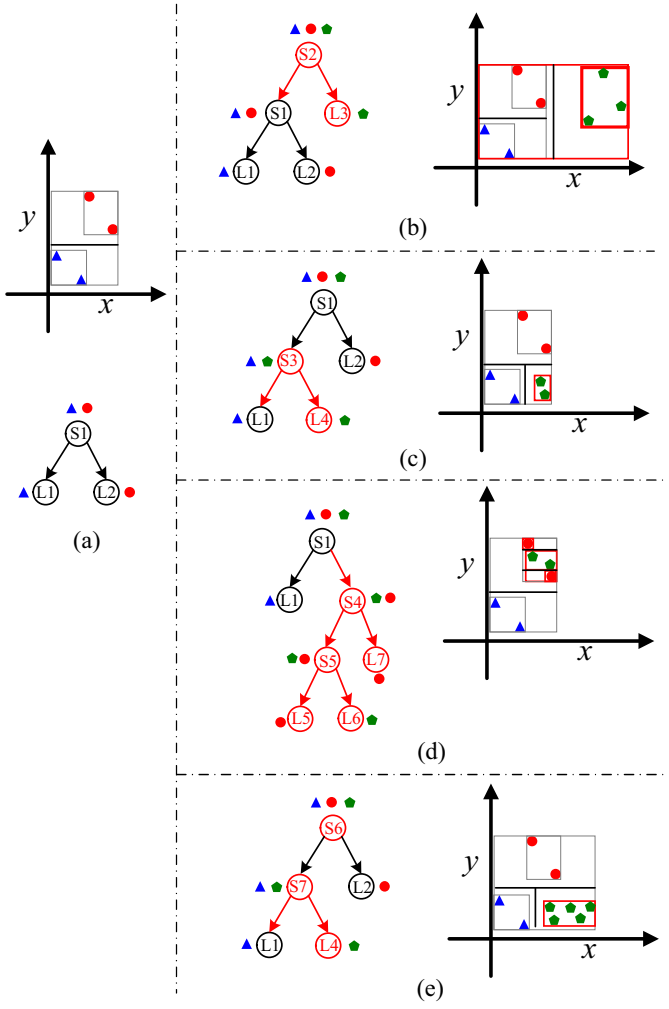

Fig. 6. Class incremental tree growing mechanism.

online examples of old classes. A more common case is illustrated in Fig. 6(e). New data partially falls into the range of the current node (S1 in Fig. 6(a)) and partially falls outside the range of the current node. In this case, we expand the AABB of S1 to cover all the new data points (S6 in Fig. 6(e)). No change is required for the splitting attribute and splitting value. New data is partitioned by the existing split and passed to corresponding subtrees (L1 and L2 in Fig. 6(a)). Then, the incremental learning phase repeats as an iterative process. The incremental growing procedure of a decision tree is depicted in Algorithm 1.

In the class incremental tree growing mechanism described above, there is no need to access original data in case 1, case 2 and case 4. The SAT based splitting strategy is able to make an optimal split with AABB recorded in each tree node. Old data is required only in the case 3. With all data accessible, we are able to update the RF model continuously with a little loss in performance.

3.2.5. Computational complexity of CIRF

In this section, we will discuss the computational complexity of the proposed CIRF. Assume there are M trees in the RF and K features in each instance. We randomly select k' ($k' = \sqrt{K}$) features as the candidate feature set. In the construction of the initial RF, we assume there are n samples which belong to two classes. The training computational complexity is analyzed as follows.

For the root node, all training samples are sorted first. For each dimension, the computational complexity of sorting operation is

Algorithm 1 Class incremental tree construction.

Input: incremental training set $D_1 = \{(\boldsymbol{x}_i^1, y_i^1)|i = 1, 2, !\text{`}, n_1\}$;decision tree $T_i, i = 1, 2, !\text{`}, M$.
Output: decision tree $T_i^{'}, i = 1, 2, !\text{`}, M$;

$T_i^{'} = ConstructClassIncrementalTree(D_1, T_i)\{$
$rangemin = T_i.min;$
$rangemax = T_i.max;$
$rangemin1 = \min(D_1);$
$rangemax1 = \max(D_1);$
$dist1 = rangemin1 - rangemax;$
$dist2 = rangemin - rangemax1;$
$[max1, index1] = \max(dist1);$
$[max2, index2] = \max(dist2);$
if $max1 > 0 \&\& max1 \geq max2$ **then**
  $T_i^{'}.leftchild = T_i;$
  $T_i^{'}.rightchild = CreateNewNode(D_1);$
  $T_i^{'}.splitAttribute = index1;$
  $T_i^{'}.splitValue = (rangemin1(index1) + rangemax(index1))/2;$
  $T_i^{'}.isLeaf = NaN;$
  $T_i^{'}.minrange = \min(rangemin1, rangemin);$
  $T_i^{'}.maxrange = \min(rangemax1, rangemax)$
else if $max2 > 0 \&\& max2 > max1$ **then**
  $T_i^{'}.leftchild = CreateNewNode(D_1);$
  $T_i^{'}.rightchild = T_i;$
  $T_i^{'}.splitAttribute = index2;$
  $T_i^{'}.splitValue = (rangemin(index2) + rangemax1(index2))/2;$
  $T_i^{'}.isLeaf = NaN;$
  $T_i^{'}.minrange = \min(rangemin1, rangemin);$
  $T_i^{'}.maxrange = \min(rangemax1, rangemax);$
else if $T_i.isLeaf == 1$ **then**
  if $\sim (isPure \| T_i.nodeDepth > maxDepth \| T_i.sampleCounts < counterThreshold)$ **then**
    $T_i^{'} = buildTree(T_i.samples, D_1);$
  else
    $UpdateStatistics(T_i, D_1);$
    $T_i^{'} = T_i$
  end if
else
  $UpdateStatistics(T_i, D_1);$
  $D_{11} = \{(x_j^1, y_j^1) \mid x_{j,splitAtt} < T_i.splitValue\};$
  $D_{12} = \{(x_j^1, y_j^1) \mid x_{j,splitAtt} \geq T_i.splitValue\};$
  $T_i.leftChild = ConstructClassIncremental Tree(D_{11}, T_i.leftChild);$
  $T_i.rightChild = ConstructClassIncremental Tree(D_{12}, T_i.rightChild);$
  $T_i^{'} = T_i;$
end if
}

about $O(n\log(n))$. The computational complexity for entropy calculation is $O(n) + (n-1)O(k^{'})$. Thus, for the tree root, the time complexity is:

$$O(k^{'} n\log(n)) + O(n) + (n-1)O(k^{'}) \approx O(k^{'} n\log(n))$$

On average, all training samples are equally divided into two parts and passed to the children nodes. For each node of the first layer, the computational complexity is $O(k^{'}\frac{n}{2}\log\frac{n}{2})$. Since there are two nodes in the first layer, the computational complexity for the first layer is $O(k^{'} n\log\frac{n}{2})$. In this way, we can deduce that the computational complexity of the second layer is $O(k^{'} n\log\frac{n}{4})$, the third layer is $O(k^{'} n\log\frac{n}{8})$, and so on. The depth of a decision tree is about $O(\log n)$. Therefore, the computational complexity for building a decision tree is $O(k^{'} n(\log(n))^2)$. As there are M trees in the RF, the computational complexity of RF is $O(Mk^{'} n(\log n)^2)$.

In the incremental learning phase, assume m new samples arrive. Firstly, we try to construct the AABB of the newly arrived data batch. To compute the AABB, we need to traverse the new data batch. Thus, the computational complexity is $O(m)$. To find a split with SAT-based splitting strategy, the time consumption is $O(K)$. As there are no more than n nodes in the existing tree, the complexity to insert new nodes in the tree is $O(Kn)$. On the other hand, the computational complexity to split leaves is $O\big(k^{'}(m+n)\log(m+n)\big)$. Thus, the computational complexity of incremental learning phase, recorded as CI, is $M(O(m) + O(Kn)) \leq CI \leq O(Mk^{'}(m+n)\log(m+n)(\log(m+n) - \log(n))$.

Without incremental learning mechanism, the computational complexity of RF is $O(Mk^{'}(m+n)(\log(m+n))^2)$. In the worst case, $CI = O(Mk^{'}(m+n)\log(m+n)(\log(m+n) - \log(n))$. Because $O(Mk^{'}(m+n)\log(m+n)(\log(m+n) - \log(n))$ is far less than $O(Mk^{'}(m+n)(\log(m+n))^2)$, we can conclude that the CIRF algorithm gains significant superiority in computational complexity compared with RF.

Table 1
Activities collected in the DSADS.

Label	Activity category
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 km/h
11	Walking in flat and 15° inclined positions
12	Running on a treadmill with a speed of 8 km/h
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

4. Experiments

To validate the performance of our method, we conduct experiments on three public activity datasets including UCI DSADS dataset [20], OPPORTUNITY dataset [21] and HARUSDS dataset [9]. All performance tests are conducted on a Dell OptiPlex 9020 (Intel Core i7-4790/24 GB DDR3) desktop computer with Matlab R2014a. In this section, we first give a brief description of these datasets. Then, we introduce the proposed feature extraction method in Section 4.2. Experimental settings are presented in Section 4.3. Finally, performance evaluations are further discussed in Section 4.4.

4.1. Dataset description

DSADS dataset The Daily and Sports Activities Data Set (DSADS) [20] from UCI Machine Learning Repository consists of 19 daily and sports activities performed by 8 subjects (4 males and 4 females). The activity categories and corresponding labels are listed in Table 1. Three types of sensors (3-axial accelerometer, 3-axial gyroscope, and 3-axial magnetometer) were placed on five different body parts including the torso, right arm, left arm, right leg and left leg (see in Fig. 7(a)). Thus, there are 45 sensor readings

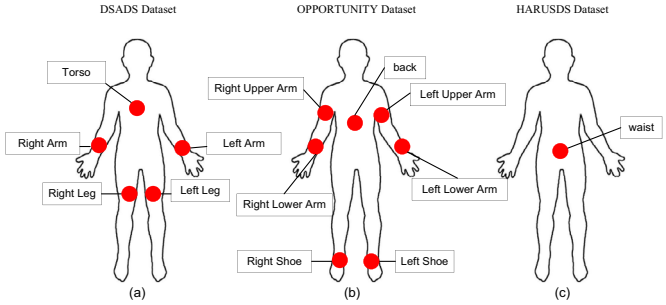

Fig. 7. Positions of sensors (a) DSADS dataset (b) OPPORTUNITY dataset (c) HARUSDS dataset.

Table 2
Summary of three public datasets .

Dataset name	DSADS	OPPORTUNITY	HARUSDS
Number of users	8	4	30
Position	5	7	1
Activity class	19	9	6
Data size	9120	2224	10299

in each frame. All subjects performed the activities in their own styles with no restriction.

OPPORTUNITY dataset The OPPORTUNITY dataset [21] is comprised of typical daily activities performed by 4 subjects. Six different runs (includes five runs of ADLs and a drill run) are recorded for each subject. Three types of sensors are employed to acquire data in a smart house. In our experiments, we pay more attention to the activity recognition problem. Thus, we use data acquired from on-body sensors. As there are plenty of missing data from Bluetooth sensors, we only use five XSense inertial measurement units mounted on a motion jackets and two commercial InertiaCube3 inertial sensors placed on feet. The positions of inertial measurement are depicted in Fig. 7(b). Each unit includes accelerometer, gyroscope and magnetometer sensors. In OPPORTUNITY dataset, activities are divided into 4 levels: modes of locomotion, low-level actions, middle-level gestures, and high-level activities. In this paper, we conduct experiments on the track which record the actions of the left hand to evaluate the effectiveness of our method.

HARUSDS dataset The Human Activity Recognition Using Smartphones Data Set (HARUSDS) [9] was collected with 30 subjects. A smartphone was fixed on the subject's waist (see in Fig. 7(c)) to acquire readings of the 3-axial accelerometer and 3-axial gyroscope for six activities (walking, walking upstairs, walking downstairs, sitting, standing, lying). This dataset has been preprocessed and divided into training and testing sets by the authors. In our experiments, we use the provided feature directly. However, the partition of the dataset is not consistent with our experimental goals. Thus, we merge the training and testing sets together and redistribute the dataset on demand.

To give an intuitive description of three datasets, we summarize the number of users, positions, activity classes and data size in Table 2.

4.2. Feature extraction

As features are provided in the HARUSDS dataset, we extract features only for the DSADS dataset and OPPORTUNITY dataset. In our experiments, we first synthesize the readings of 3 axes from one sensor into a synthetic value using the formula $a = \sqrt{(x^2 + y^2 + z^2)}$. This step will eliminate the influence caused by the orientation of sensors. Then, we adopt the sliding window technique to segment the sensor readings. We set the window size to 5 s [59] and 2 s for the DSADS dataset and OPPORTUNITY dataset respectively with 50% overlap between consecutive windows. For each window, we extract 27 features, which are listed in Table 3, from both time domain and frequency domain for a single sensor. After feature extraction, we learned 405 ($27 \times 3 \times 5$) features for the DSADS dataset and 459 ($27 \times 3 \times 5 + 27 \times 2$) features for the OPPORTUNITY dataset.

4.3. Experiment settings

In our experiments, RF [60], extremely randomized trees (ERT) [61], class incremental extreme learning machine (CIELM) [46], incremental RLSC with class recoding (IRLSC) [62] and SENCForest [63] are used as the comparison methods. Among these methods, RF and ERT are batch learning methods, which are used to compare the performance between batch learning and CIRF. CIELM, IRLSC and SENCForest are class incremental learning methods, which are used to validate the class incremental learning ability of CIRF.

As is common in the RF literature [60], we set the tree number as 100 and the minimum sample size for splitting a node as 2. The number of attributes randomly selected at each node is set as $\lfloor \sqrt{K-1} \rfloor$. For CIELM, the activation function is set as the sigmoid function. As stated in [64], ELM can achieve good generalization performance as long as the number of hidden nodes is large enough, and the performance of ELM with sigmoid additive hidden nodes is not sensitive to the number of hidden nodes. Therefore, we set the number of hidden nodes of CIELM as 1000. The parameters of SENCForest are set the same as [63], while the parameters of IRLSC are set the same as [62].

4.4. Experimental results and performance analysis

In our experiments, we first evaluate the performance of CIRF in recognizing new class data. Three class incremental learning methods are taken as comparisons. Then, we evaluate the ability of CIRF in continuous recognizing new classes. To evaluate the robustness of our method, we build generic models and personalized models, respectively. The comparisons on the number of leaves and influence of tree number are further explored in Section 4.4.3 and Section 4.4.4, respectively.

4.4.1. Performance in recognizing new class

In this section, we try to evaluate the ability of CIRF in recognizing new class incrementally. CIELM, IRLSC and SENCForest are taken as comparison methods. Three datasets are both divided into training set and testing set with equal size. In this experiment, one class is taken as new emerging class and the rest are taken as known classes. The initial model are constructed with data from known classes, while new emerging class data is used for incremental learning step. Experimental results are averaged over repeated trials (all classes are taken as new emerging class by turns). The experimental results are listed in Table 4.

From Table 4, we can see that CIRF gains best testing accuracy on both DSADS and OPPORTUNITY datasets. The testing accuracy of CIRF is superior to that of the other state-of-the-art methods by 6.69% on DSADS dataset and 2.13% on OPPORTUNITY dataset. On HARUSDS, the testing accuracy of CIRF is inferior to the state-of-the-art method by 0.27%. Overall, the performance of CIRF is better than the other state-of-the-art methods in recognizing new classes.

4.4.2. Performance in recognizing continuous new emerging classes

To eliminate the influence of data arriving order, we randomly sample five class sequences and take the mean performance as the final result. The mean performance is shown in the two-dimensional images in following sections.

Table 3
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 5 peaks after FFT
14–18	Frequency	Frequencies corresponding to the 5 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 4
Testing accuracy of CIRF, CIELM and IRLSC on three public activity datasets.

	CIRF		CIELM		IRLSC		SENCForest	
	Initial	Incremental	Initial	Incremental	Initial	Incremental	Initial	Incremental
DSADS	0.9340	**0.9851**	0.8872	0.8982	0.8657	0.9105	0.9078	0.9182
HARUSDS	0.8153	0.9742	0.8004	0.9257	0.8155	**0.9769**	0.7342	0.7570
OPPORTUNITY	0.7993	**0.8915**	0.5380	0.5868	0.7842	0.8702	0.7803	0.8452

For Figs. 8 and9, x-axis is the number of class. The values of y-axis represent predictive accuracy on test data, training time and test time, respectively.

1) *Performance comparisons with RRSS split*

We evaluate the performance of CIRF about generic model using repeated random sub-sampling (RRSS) technique [65]. Assume T represents the whole dataset, and T_{ij} represents the jth class instances of the ith user. T_{ij} is divided into two equal parts, which are represented by TR_{ij} and TE_{ij}. TR_{ij} is used for training and TE_{ij} is used for testing. In the first step, the model is trained with the data of first two classes (represented by $TR_{\cdot 1} + TR_{\cdot 2} = \bigcup_{i=1}^{u}(TR_{i1} + TR_{i2})$, where u is the number of subjects. The model is tested with $TE_{\cdot 1} + TE_{\cdot 2} = \bigcup_{i=1}^{u}(TE_{i1} + TE_{i2})$. At the kth step, the training set is $TR_{(\cdot(k+1)} = \bigcup_{i=1}^{u} TR_{i(k+1)}$ and the testing set is $TE_{(\cdot(k+1)} = \bigcup_{i=1}^{u} TE_{i(k+1)}$. Fig. 8 shows the test accuracy, training time and testing time comparisons of CIRF, CIELM, ERT and RF with RRSS split on three public datasets.

Test accuracy. Fig. 8(a) indicates that, in most cases, CIRF shows comparable performance with batch learning methods (ERT and RF) in test accuracy. The predictive accuracy of CIRF stays above 97% for DSADS dataset, 96% for HARUSDS dataset and 74% for OPPORTUNITY dataset. The performance of CIRF is significantly better than CIELM trained with the same fraction of training data. Besides, with data of unknown classes arrives, there are very small fluctuations on the accuracy curves, which indicate the performance of CIRF is relatively stable. For CIELM, we can see the performance of test accuracy fluctuates significantly when new class data arrives, especially on the DSADS and OPPORTUNITY datasets. Overall, the performance of CIRF is slightly worse than batch learning algorithms in predictive accuracy but is much better than CIELM in test accuracy and stability.

Time consumption. For two batch learning algorithms, the construction of RF requires more training time. This can be attributed to the splitting mechanism of RF. The naive strategy of enumerating all partitions and choosing the optimal split position enables the number of partitions grow exponentially, which is very time-consuming. On the contrary, ERT split nodes with fully randomized split-criterion. Thus, compared with RF, ERT shows a great advantage in training time. In the initial phase, CIRF constructs the RF using the same mechanism as RF. Therefore, the training time consumed by CIRF is the same with RF. Using SAT-based splitting strategy, CIRF is able to update recognition model to adapt to new class data. It shows great superiority in an incremental learning phase. With the number of training instance increases, the training time of CIRF is relatively stable, while the training time of RF and ERT grow exponentially and linearly respectively.

As shown in Fig. 8(c), the testing time of all four algorithms show linear upward trends with the number of testing samples increases. In the testing phase, the time consumption is directly related to the average depth of the leaves reached by the test samples [66]. With full randomization in choosing split-criterion, the structure of ERT is more complex than RF [61], which is mainly illustrated by the number of leaves and average tree depth. The optimal splitting strategy of RF contributes to an optimal model structure. Thus, we see a significant advantage of RF in testing time compared with ERT. Compared with ERT, CIRF spends less time on prediction. This result proves that CIRF constructs the model with better structure. On the other hand, CIRF performs competitively with RF on HARUSDS dataset and slightly worse than RF on the other two datasets in testing time. This is because that RF constructs a model with all training instances from scratch.

For incremental learning methods and online learning methods, the predictive time cost is more important than the training time cost. This is because of that, after model training, people expect an immediate response in practical application. Besides, less predictive time cost means the model has better structure and require less space to store the model. Thus, we adopt RF instead of ERF to construct our class incremental learning method in this paper.

For both training and testing time, CIELM shows a great superiority compared with RF, ERT, and CIRF. This is because CIELM is an individual learner while the other three algorithms are ensemble learners. To further evaluate the performance of CIRF and CIELM, another group of experiments is conducted in Section 4.4.4.

2) *Performance comparisons for personalized model*

To evaluate the performance of CIRF about personalized models, we build activity recognition model for individuals respectively. For personalized model, we build a model for each subject and evaluate the performance of the model by taking the mean of u users. In the first step, the model for the ith subject is trained with $TR_{i1} + TR_{i2}$ and tested with $TE_{i1} + TE_{i2}$. In the kth step, data of the

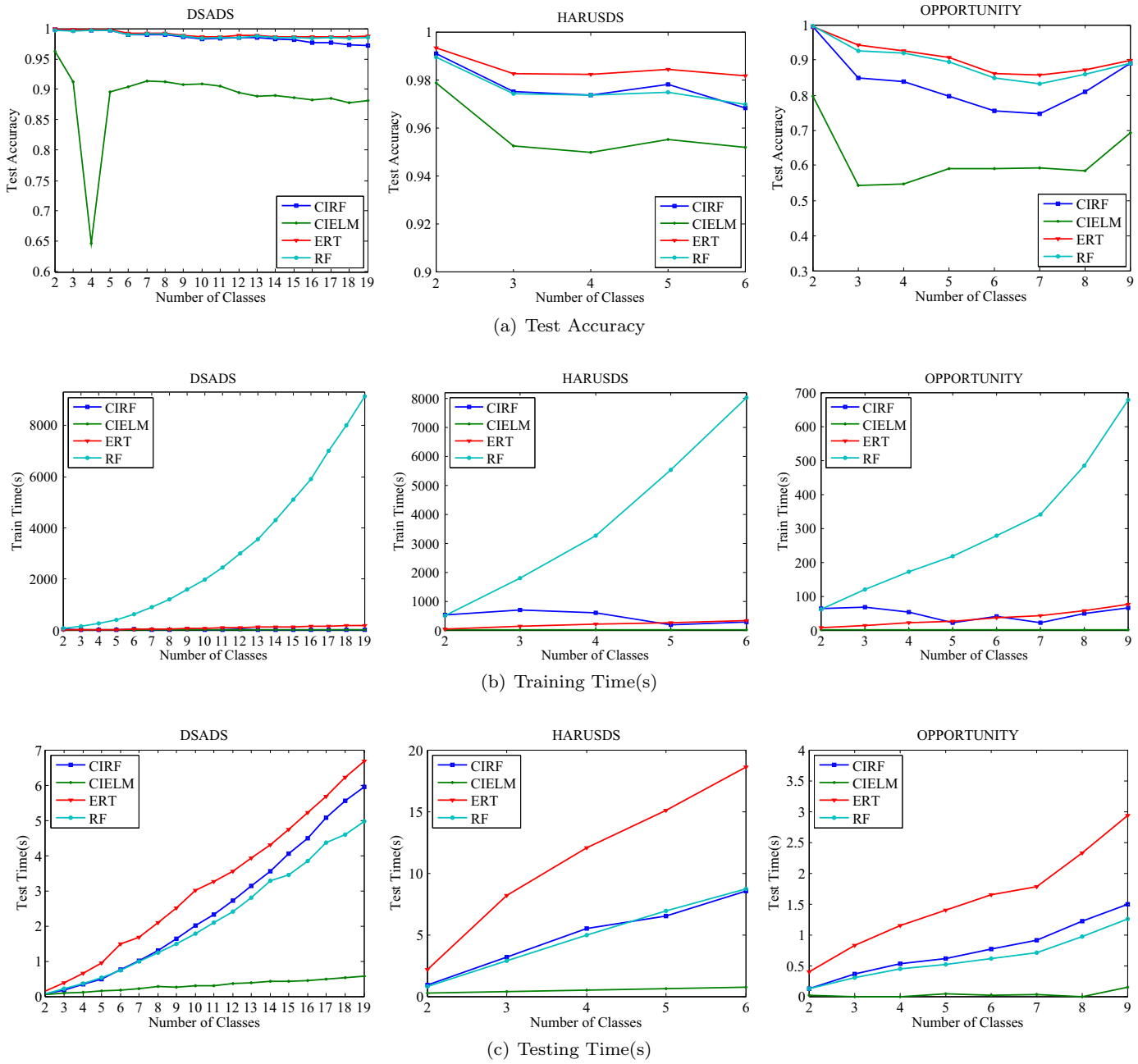

Fig. 8. Test accuracy, training time and testing time comparisons of CIRF, CIELM, ERT and RF with RRSS split on three public datasets. (a) Test accuracy; (b) Training time(s); (c) Testing time(s).

$(k+1)$th class $TR_{i(k+1)}$ is added to train the classifier and the testing set is $\bigcup_{j=1}^{k+1} TE_{ij}$. Fig. 9 shows the test accuracy, training time and testing time comparisons of CIRF, CIELM, ERT and RF for the personalized models on three public datasets.

Test Accuracy. In terms of test accuracy, CIRF is still close to the performance of batch learning algorithms (RF and ERT). The predictive accuracy of CIRF stays above 96% for DSADS dataset, 97.5% for HARUSDS dataset and 73% for OPPORTUNITY dataset. As the class number increases, its performance is relatively stable. On the contrary, the performance of CIELM algorithm is quite unstable. The classification accuracy fluctuates intensely, especially on the OPPORTUNITY data set. In addition, with the number of class increases, CIELM showed a significant downward trend on the DSADS dataset. To summarize, the performance of CIRF for the personalized model is similar to that for the generic model.

Time consumption. In this group of experiments, CIELM still shows the advantage of individual classifiers on time cost. Its performance is very good for both training and testing time. Compared with batch learning methods, CIRF shows a good time superiority. The time cost of CIRF is less than batch learning in most cases on all of the three data sets. It is worse than RF algorithm only on the DSADS dataset for the testing time. Nevertheless, in most cases, it is still better than the ERT algorithm, which can fully indicate the efficiency of the incremental learning algorithm.

4.4.3. Comparisons on number of leaves

To find out the differences in terms of structure between CIRF and RF, we record the number of leaf nodes obtained from exper-

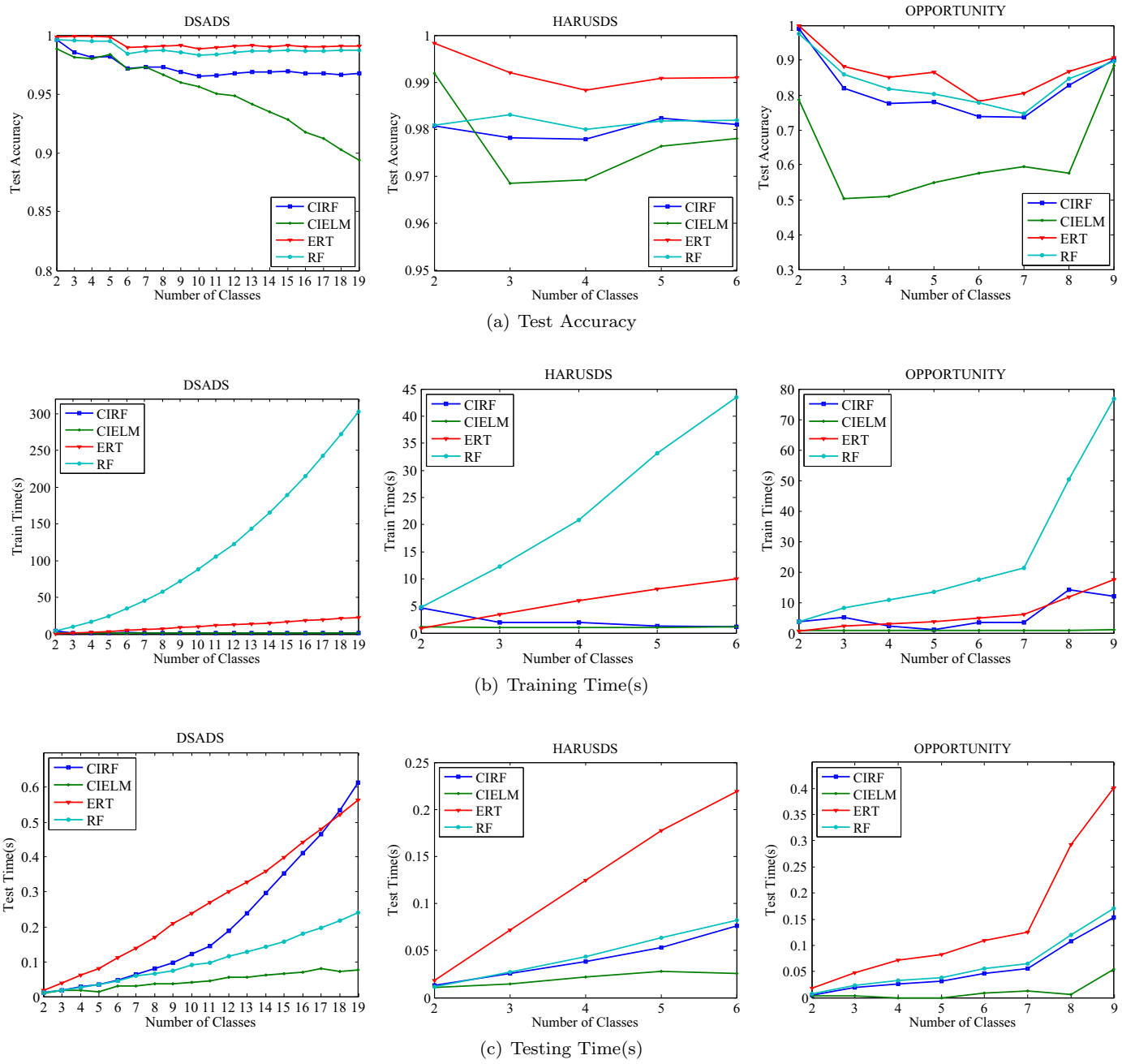

Fig. 9. Test accuracy, training time and testing time comparisons of CIRF, CIELM, ERT and RF for personalized model on three public datasets. (a) Test accuracy; (b) Training time(s); (c) Testing time(s).

Table 5
Comparisons on number of leaves for generic models.

Dataset	RF	CIRF
DSADS	15,205	14,447
HARUSDS	16,716	16,804
OPPORTUNITY	5861	6452

Table 6
Comparisons on number of leaves for individual models.

Dataset	RF	CIRF
DSADS	2682	2755
HARUSDS	1041	777
OPPORTUNITY	1623	1912

iments in Section 4.4.3. Table 5 shows the number of leaves for generic models and Table 6 shows the number of leaves for individual models.

From Tables 5 and 6, we can see that CIRF creates almost equal number of leaves as RF. For generic models, the average ratio is 0.998. While the average ratio is 1.018 for the individual models. Because the number of leaves grows exponentially with the tree depth [61], the tree depth of CIRF is inferred to be almost the same as RF. This is consistent with the experimental results on testing time. Thus, we can conclude that there is no obvious difference in terms of structure between the trees learned by the propose CIRF and RF.

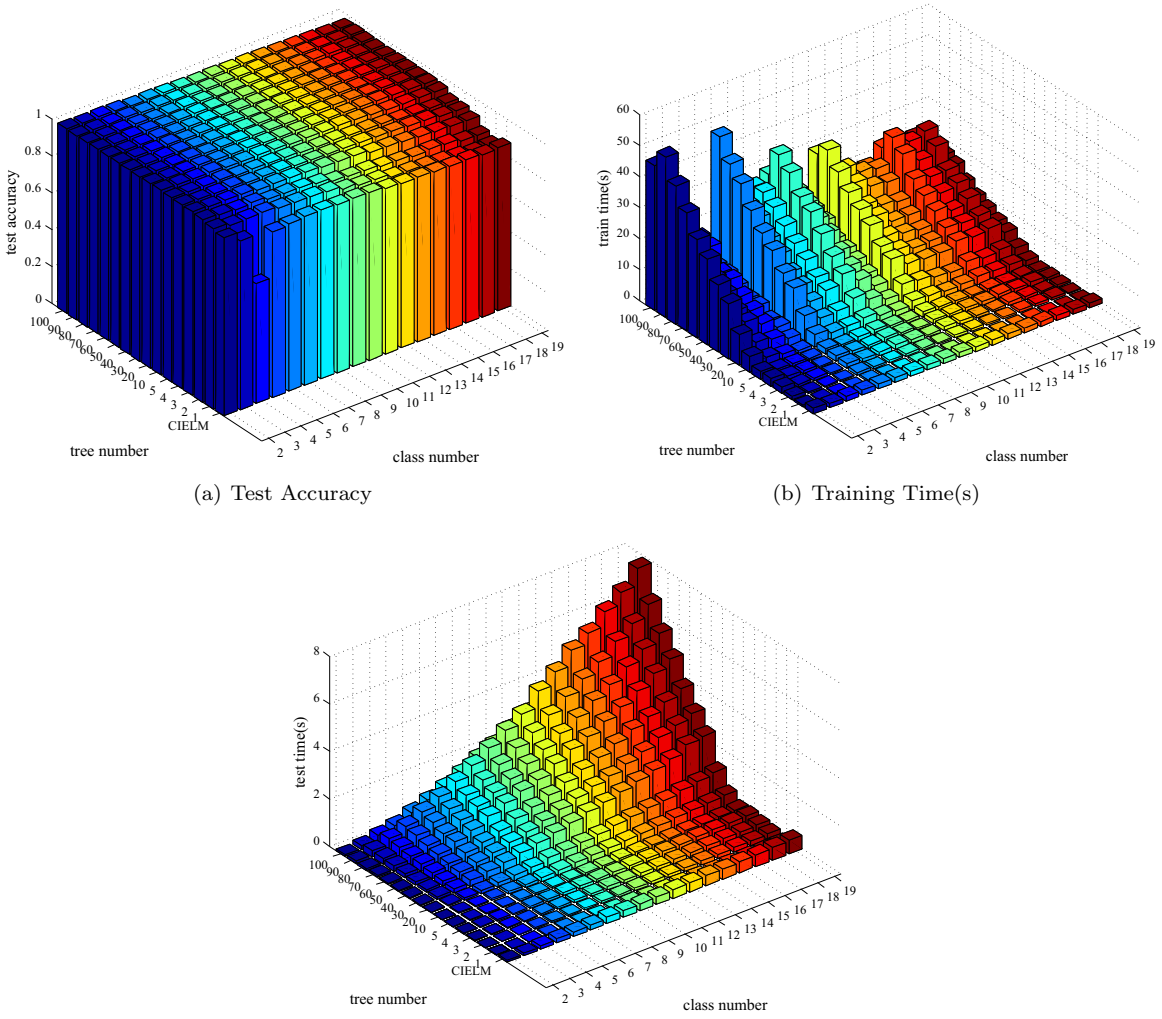

Fig. 10. The influence of tree number on the performance of CIRF and comparisons with CIELM. (a) Test accuracy; (b) Training time(s); (c) Testing time(s).

4.4.4. Influence of tree number

In Section 4.4.2, we have made comparisons among CIRF, RF, ERT, and CIELM on both test accuracy and time consumptions. However, it seems inadequate to compare the performance of CIRF and CIELM. As an individual learner, CIELM shows a significant advantage on time consumption while performs worse in test accuracy. In contrast, CIRF shows a good performance on testing accuracy with more time consumption. In this section, we increase the tree number of CIRF from 1 to 100 and compare the performance with CIELM. The influence of tree number and further comparisons between CIRF and CIELM are shown as follows.

With the number of trees increases from 1 to 100, the test accuracy of CIRF converges quickly. As we can see from Fig. 10(a), the predictive accuracy has reached above 91.61% with 3 trees in CIRF. After $M = 20$, the test accuracy exceeds 96% and the speed of convergence slows down. For both training time and testing time, we can see a monotonically increasing trend. This is because that CIRF consists of many individual trees. The more trees in CIRF, the more time consumes.

With five trees in CIRF, the training and testing time are both shorter than CIELM. At the same time, the test accuracy of CIRF is still above 93.78%, which is 5.68% higher than CIELM. Thus, we can conclude that the performance of CIRF is better than CIELM on both predictive accuracy and time consumption.

To summarize, with the number of trees increases, both test accuracy and time consumptions monotonically increase. The appropriate choice of tree number is able to compromise between predictive performance and computational requirements. Besides, with an appropriate choice of tree number, CIRF outperforms CIELM on both test accuracy and time consumption.

5. Conclusion and future work

Due to the incremental learning ability of humans, the manner of activity changes dynamically over time. The proposed CIRF is able to keep in line with the incremental learning ability of human while avoiding the failure caused by the emergence of new activity classes. CIRF is proposed with a novel splitting strategy, named SAT-based splitting strategy, which is firstly presented in this work. Experimental results on three public activity datasets show that our method is an effective solution to tackle the dynamic changes in activity recognition. For test accuracy, CIRF can achieve similar results as the batch learning algorithms. As the number of activity classes increases, the predictive performance of CIRF is relatively stable and robust. It also shows a significant advantage in time consumption compared with batch learning algorithms. With an appropriate choice of tree number, CIRF outperforms other related class incremental learning method on both test accuracy and time consumption. To compromise between testing accuracy and

time consumption, we can choose the number of trees which has the highest ratio on accuracy improvement to time consumption.

Similar to the other versions of online RF, the CIRF algorithm requires that all samples should be stored all the time. In future work, we plan to explore the possibility of constructing an incremental learning RF without maintaining old data, which will contribute to the reduction of storage space.

Acknowledgments

This work is partly supported by the National Key Research and Development Program of China [Grant no. 2017YFB1002801]; the Natural Science Foundation of China [Grant no. 61572471]; the Chinese Academy of Sciences Pioneer Hundred Talents Program [Grant no. Y704061000]; and the Science and Technology Planning Project of Guangdong Province, China [Grant no. 2015B010105001].

References

[1] B. Rong, F. Huiyong, Z. Jinfu, The effect of physical activity on cognitive function in older adults, Adv. Psychol. Sci. 19 (12) (2011) 1777–1787.
[2] M. Underwood, S.E. Lamb, S. Eldridge, B. Sheehan, A.-M. Slowther, A. Spencer, M. Thorogood, N. Atherton, S.A. Bremner, A. Devine, et al., Exercise for depression in elderly residents of care homes: a cluster-randomised controlled trial, The Lancet 382 (9886) (2013) 41–49.
[3] C. Yiqiang, H. Meiyu, H. Chunyu, Z. Yicheng, H. Fei, M. Chunyan, A coarse–to-fine feature selection method for accurate detection of cerebral small vessel disease, in: Proceedings of the International Joint Conference on Neural Networks, 2016, pp. 2609–2616.
[4] C. Yiqiang, Y. Hanchao, M. Chunyan, Y. Xiaodong, Using motor patterns for stroke detection, Sci. Adv. Comput. Psychophysiol. (2015) 12–14.
[5] Z. Feng, D. la Torre Fernando, H.J. K, Hierarchical aligned cluster analysis for temporal clustering of human motion, IEEE Trans. Pattern Anal. Mach. Intell. 35 (3) (2013) 582–596.
[6] M. Field, D. Stirling, P. Zengxi, M. Ros, F. Naghdy, Recognizing human motions through mixture modeling of inertial data, Pattern Recognit. 48 (8) (2015) 2394–2406.
[7] T.T. Ngo, Y. Makihara, H. Nagahara, Y. Mukaigawa, Y. Yagi, Similar gait action recognition using an inertial sensor, Pattern Recognit. 48 (4) (2015) 1289–1301.
[8] Z. Haiyong, L. Zhijing, Human action recognition based on non-linear svm decision tree, J. Comput. Inf. Syst. 7 (7) (2011) 2461–2468.
[9] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine, in: Proceedings of the International workshop on ambient assisted living, Springer, 2012, pp. 216–223.
[10] L.C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, W. Stork, Context-aware mobile health monitoring: evaluation of different pattern recognition methods for classification of physical activity, in: Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), IEEE, 2008, pp. 5250–5253.
[11] Z. Zhongtang, C. Yiqiang, L. Junfa, S. Zhiqi, L. Mingjie, Cross-people mobile-phone based activity recognition, in: Proceedings of the IJCAI, 11, 2011, pp. 2545–2550.
[12] O.D. Lara, A.J. Pérez, M.A. Labrador, J.D. Posada, Centinela: a human activity recognition system based on acceleration and vital sign data, Pervasive Mob. Comput. 8 (5) (2012) 717–729.
[13] L.A. Schwarz, D. Mateus, N. Navab, Recognizing multiple human activities and tracking full-body pose in unconstrained environments, Pattern Recognit. 45 (1) (2012) 11–23.
[14] H. Junker, O. Amft, P. Lukowicz, G. Tröster, Gesture spotting with body–worn inertial sensors to detect user activities, Pattern Recognit. 41 (6) (2008) 2010–2024.
[15] A. Saffari, C. Leistner, J. Santner, M. Godec, H. Bischof, On-line random forests, in: Proceedings of the IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), IEEE, 2009, pp. 1393–1400.
[16] B. Lakshminarayanan, D.M. Roy, Y.W. Teh, Mondrian forests: efficient online random forests, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 3140–3148.
[17] A.Z. Said, G.M. Medhat, S. Bala, K. Shonali, Adaptive mobile activity recognition system with evolving data streams, Neurocomputing 150 (PA) (2015) 304–317.
[18] W. Zhelong, J. Ming, H. Yaohua, L. Hongyi, An incremental learning method based on probabilistic neural networks and adjustable fuzzy clustering for human activity recognition by using wearable sensors, IEEE Trans. Inf. Technol. Biomed. Publ. IEEE Eng. Med. Biol. Soc. 16 (4) (2012) 691–699.
[19] F. Zengtao, M. Lingfei, L. Meng, A random forest-based ensemble method for activity recognition, in: Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2015, pp. 5074–5077.
[20] K. Altun, B. Barshan, O. Tunçel, Comparative study on classifying human activities with miniature inertial and magnetic sensors, Pattern Recognit. 43 (10) (2010) 3605–3620.
[21] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, et al., Collecting complex activity datasets in highly rich networked sensor environments, in: Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS), IEEE, 2010, pp. 233–240.
[22] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors., IEEE Commun. Surv. Tutor. 15 (3) (2013) 1192–1209.
[23] V. Franc, V. Hlavác, Multi-class support vector machine, in: Proceedings of the 16th International Conference on Pattern Recognition, 2, IEEE, 2002, pp. 236–239.
[24] H. Chihwei, L. Chihjen, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. 13 (2) (2002) 415–425.
[25] C. Randell, H. Muller, Context awareness by analysing accelerometer data, in: Proceedings of the Fourth International Symposium on Wearable Computers, IEEE, 2000, pp. 175–176.
[26] H. Guangbin, Z. Qinyu, S. CheeKheong, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2, IEEE, 2004, pp. 985–990.
[27] H. Lisha, C. Yiqiang, W. Shuangquan, C. Zhenyu, b-COELM: a fast, lightweight and accurate activity recognition model for mini-wearable devices, Pervasive Mob. Comput. 15 (2014) 200–214.
[28] C. Zhenyu, C. Yiqiang, H. Lisha, W. Shuangquan, J. Xinlong, Leveraging two-stage weighted ELM for multimodal wearables based fall detection, in: Proceedings of ELM, vol. 2, Springer, 2015, pp. 161–168.
[29] D. Minnen, T. Westeyn, D. Ashbrook, P. Presti, T. Starner, Recognizing soldier activities in the field, in: Proceedings of the 4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN), Springer, 2007, pp. 236–241.
[30] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. CSUR 46 (4) (2014) 1–44.
[31] T. Miu, P. Missier, T. Plötz, Bootstrapping personalised human activity recognition models using online active learning, in: Proceedings of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), IEEE, 2015, pp. 1138–1147.
[32] X. Sun, H. Kashima, N. Ueda, Large-scale personalized human activity recognition using online multitask learning, IEEE Trans. Knowl. Data Eng. 25 (11) (2013) 2551–2563.
[33] L. Haiguang, W. Xindong, L. Zhao, Online learning with mobile sensor data for user recognition, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 64–70.
[34] T. Sztyler, H. Stuckenschmidt, Online personalization of cross-subjects based activity recognition models on wearable devices, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE, 2017, pp. 180–189.
[35] M. Denil, D. Matheson, D. Nando, Consistency of online random forests, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1256–1264.
[36] M. Ristin, M. Guillaumin, J. Gall, L.V. Gool, Incremental learning of random forests for large-scale image classification, IEEE Trans. Pattern Anal. Mach. Intell. 38 (3) (2016) 490–503.
[37] Z. Zhihua, Machine Learning, Tsinghua University Press, 2016.
[38] L. Chinghu, H. Yuchen, C. Yihan, F. Lichen, Hybrid user-assisted incremental model adaptation for activity recognition in a dynamic smart-home environment, IEEE Trans. Hum. Mach. Syst. 43 (5) (2013) 421–436.
[39] L. Fei-Fei, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. 28 (4) (2006) 594–611.
[40] M. Palatucci, D. Pomerleau, G.E. Hinton, T.M. Mitchell, Zero-shot learning with semantic output codes, in: Proceedings of the Advances in Neural Information Processing Systems, 2009, pp. 1410–1418.
[41] B.R. Gaines, P. Compton, Induction of ripple-down rules applied to modeling large databases, J. Intell. Inf. Syst. 5 (3) (1995) 211–228.
[42] A. Misra, A. Sowmya, P. Compton, Incremental learning for segmentation in medical images, in: Proceedings of the 3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro,, IEEE, 2006, pp. 1360–1363.
[43] P. Compton, G. Edwards, B. Kang, L. Lazarus, R. Malor, T. Menzies, P. Preston, A. Srinivasan, C. Sammut, Ripple down rules: possibilities and limitations, in: Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, Calgary, Canada, University of Calgary, 1991, pp. 1–21.
[44] Y. Shan, X. Chang, W. Yunhe, X. Chao, D. Ta, Streaming label learning for modeling labels on the fly (2016), arXiv preprint arXiv:1604.05449.
[45] C. Xu, D. Tao, C. Xu, Streaming view learning (2016), arXiv preprint arXiv:1604.08291.
[46] Z. Zhongtang, C. Zhenyu, C. Yiqiang, W. Shuangquan, W. Hongan, A class incremental extreme learning machine for activity recognition, Cognit. Comput. 6 (3) (2014) 423–431.
[47] J. Arvo, Transforming axis-aligned bounding boxes, in: Graphics gems, Academic Press Professional, Inc., 1990, pp. 548–550.
[48] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[49] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.
[50] X. Lei, K. Adam, S.C. Y, Methods of combining multiple classifiers and their applications to handwriting recognition, IEEE Trans. Syst. Man Cybern. 22 (3) (1992) 418–435.

[51] M.P. Perrone, L.N. Cooper, When networks disagree: Ensemble methods for neural networks, in: How We Learn; How We Remember: Toward An Understanding Of Brain And Neural Systems: Selected Papers of Leon N Cooper, World Scientific, 1995, pp. 342–358.
[52] N. Jamali, C. Sammut, Majority voting: material classification by tactile sensing using surface texture, IEEE Trans. Rob. 27 (3) (2011) 508–521.
[53] N. Littlestone, M.K. Warmuth, The weighted majority algorithm, Inf. Comput. 108 (2) (1994) 212–261.
[54] G.T. Toussaint, Solving geometric problems with the rotating calipers, in: Proceedings of the IEEE Melecon, 83, 1983, p. A10.
[55] D. Eberly, Dynamic Collision Detection Using Oriented Bounding Boxes, Geometric Tools, Inc, 2002.
[56] C. Wächter, A. Keller, Instant ray tracing: The bounding interval hierarchy., Rendering Tech. 2006 (2006) 139–149.
[57] W. Liwen, B. Liu, J. Han, Survey of box-based algorithms for collision detection, J. Civil Aviation Univ. China 25 (4) (2007) 16–19.
[58] S. Gottschalk, Separating Axis Theorem, Technical Report, TR96-024, Department of Computer Science, UNC Chapel Hill, 1996.
[59] B. Barshan, M.C. Yüksek, Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units, Comput. J. 57 (11) (2013) 1649–1667.
[60] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[61] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[62] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, G. Metta, Incremental robot learning of new objects with fixed update time (2016), arXiv preprint arXiv:1605.05045.
[63] M. Xin, M. Kai, Z. Zhihua, Classification under streaming emerging new classes: a solution using completely-random trees, IEEE Trans. Knowl. Data Eng. PP (99) (2017) 1605–1618.
[64] H. Guangbin, Z. Hongming, D. Xiaojian, Z. Rui, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B Cybern. 42 (2) (2012) 513–529.
[65] K. Altun, B. Barshan, Human activity recognition using inertial/magnetic sensor units, in: Proceedings of the International Workshop on Human Behavior Understanding, Springer, 2010, pp. 38–51.
[66] G. Louppe, Understanding random forests: from theory to practice (2014), arXiv preprint arXiv:1407.7502.

Chunyu Hu received the B.S. and M.S. degrees in computer science and technology from Shandong Normal University, Jinan, China, in 2008 and 2015, respectively. Now she is a Ph.D. candidate in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. Her current research interests include machine learning, pervasive computing, and activity recognition.

Yiqiang Chen received the B.S. and M.S. degrees in computer science from Xiangtan University, Xiangtan, China, in 1996 and 1999, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2003.

Lisha Hu received the B.S. and M.S. degrees in applied mathematics from Hebei University, Baoding, China, in 2009 and 2012, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China,in 2017. She is currently a lecturer in Hebei University of Economics and Business. Her research interests include machine learning, activity recognition and wearable computing.

Xiaohui Peng received the B.S. degree from Northwestern Polytechnical University, Xi'an, China, in 2006, the M.S. degree from Peking University, Beijing, China, in 2009, and the Ph.D. degree from the University of Tokyo, Tokyo, Japan, in 2016. His research interests include edge computing and the Internet of Things.