DOI: 10.3969/j. issn. 1673-5188. 2016. S0. 008
http://www.cnki.net/kcms/detail/34.1294.TN.20160603.1100.002.html, published online June 3, 2016

Human Motion Recognition Based on Incremental Learning and Smartphone Sensors

LIU Chengxuan[1], DONG Zhenjiang[2], XIE Siyuan[2], and PEI Ling[1]
(1. Shanghai Jiao Tong University, Shanghai 200240, China;
2. ZTE Corporation, Shenzhen 518057, China)

Abstract

Batch processing mode is widely used in the training process of human motion recognition. After training, the motion classifier usually remains invariable. However, if the classifier is to be expanded, all historical data must be gathered for retraining. This consumes a huge amount of storage space, and the new training process will be more complicated. In this paper, we use an incremental learning method to model the motion classifier. A weighted decision tree is proposed to help illustrate the process, and the probability sampling method is also used. The results show that with continuous learning, the motion classifier is more precise. The average classification precision for the weighted decision tree was 88.43% in a typical test. Incremental learning consumes much less time than the batch processing mode when the input training data comes continuously.

Keywords

human motion recognition; incremental learning; mapping function; weighted decision tree; probability sampling

1 Introduction

Human motion recognition involves recognizing what a person is doing and is an important aspect of context awareness [1]. It can be used for diverse purposes, such as ubiquitous computing, sports training, virtual reality, and health care. A promising application is remote monitoring of elderly people who live alone and need support. An emergency situation arising from a fall could be detected and responded to quickly [2]. Recently, sports bracelets that detect a person's motion have become very popular. Such bracelets can calculate how many calories a person consumes in a day and give reminders about healthy lifestyle. Human motion recognition has also been introduced into personal navigation to help increase the location accuracy [3]–[5].

Methods of motion recognition and classification include computer vision and inertial sensor data processing. Early research on motion recognition has focused on vision-based systems with one or more cameras [6], [7]. A camera system is practical when motion is confined to a limited area, such as an office or house, and the environment is well-lit. However, when a person is moving from place to place, a camera system is much less convenient because it cannot move in the same way a person does. In terms of privacy, a vision-based system puts a degree of psychological pressure on a person and causes them to act unnaturally. As well as vision-based solutions, sensor-based solutions are also extensively used to study human motion [8]–[11]. Most previous research on motion recognition has assumed that inertial sensors are fixed on the human body in a known orientation [12], [13]. In a well-cited work [14], multiple accelerometer sensors worn on different parts of the human body detect common activities, such as sitting, standing, walking or running. In [15], a small low-power sensor board is mounted at a single location on the body. Then, a hybrid approach is taken to recognize activities. This approach combines the boosting algorithm, which discriminatively selects useful features, and HMMs, which capture the temporal regularities and smoothness of activities. However, the assumption made in laboratory experiments usually cannot be made in a regular mobile environment. In some other research, a phone has been used as the sensor to collect motion data for offline analysis [16], [17]. In [18], a phone-centric sensing system is described. The position of the mobile phone on the human body is assumed to be fixed—e.g., in a pocket, clipped to a belt, or on a lanyard—and an inference model is trained according to the phone position. Compared to image-processing-based motion recognition, inertial-sensor-based motion recognition is cheaper, less limited by the environment, and involves smaller devices. Such sensors have already been integrated into smartphones, which are developing rapidly. Therefore, inertial-sensor-based motion recognition may be more popular in the future.

To recognize human motion, a classifier should be modeled. In the training process, an algorithm can be divided into batch-processing mode and incremental-learning mode. In batch-processing mode, all the history training data is used to model the motion classifier. When the new training data is available and the classifier needs to be updated, all the history data must be gathered again to retrain the classifier. Therefore, all the train-

This work is partly supported by the National Natural Science Foundation of China under Grant 61573242, the Projects from Science and Technology Commission of Shanghai Municipality under Grant No. 13511501302, No. 14511100300, and No. 15511105100, Shanghai Pujiang Program under Grant No. 14PJ1405000, and ZTE Industry-Academia-Research Cooperation Funds.

ing data must be preserved. This means that a huge amount of storage space is needed. As the amount of input training data increases, the training process becomes more complex and takes longer. Most current research on vision- and inertial-sensor-based human motion recognition focuses on this processing mode. Incremental learning only uses new incoming training data to update the classifier model; therefore, updating is more efficient, history data does not need to be stored, and much free space is spared. In [19], the authors propose pattern recognition based on neural networks and learning new chunks of patterns while keeping the previous ones intact. In [20], the authors suggest Learn ++, an approach inspired by Adaboost. This approach is based on neural-network-based ensemble classifiers working on a digital optics database. In [21], a Gaussian mixture model and resource allocation for learning were applied in the context of a dormitory to study the habits of students. In [22], the authors propose an approach to incrementally learning the relationship between motion primitives in order to form longer behaviors. In general, when the training data set is huge and data is continuously coming in, incremental learning mode is a better choice.

The main goal of our research is to provide a complete solution for human motion recognition based on incremental learning and smartphone inertial sensors. We illustrate the flow of motion pattern recognition and typical motion feature sets. We also show how to apply incremental learning to human motion recognition in detail. At the same time, we develop a weighted decision tree for the incremental learning framework.

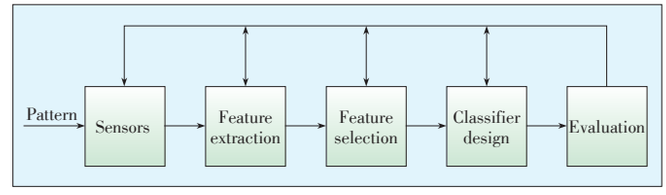
The remainder of this paper is organized as follows. In section 2, we describe the general pattern-recognition framework, date preprocessing, and feature extraction. In section 3, we discuss the incremental learning method used for human motion recognition. In section 4, we discuss the experimental results. In section 5, we make concluding remarks.

2 Motion Recognition

2.1 Pattern-Recognition Framework

Pattern recognition is a subject dealing with object classification and recognition. It encompasses face identification, character recognition, and voice recognition. It also encompasses human motion recognition. A general pattern-recognition system framework is shown in **Fig. 1**. The components of this framework are defined in **Table 1**.

Because raw sensor data is just a series of waves, it cannot be directly used for pattern recognition. Therefore, pattern features are extracted in order to describe the patterns. A pattern may have many features; however, more features do not mean better performance. If different patterns cannot discriminate when the chosen features are applied, the designed classifier performs badly. Feature selection (feature compression) is an important part of pattern recognition. With a well-selected feature subset, the generated classifier has a higher classification rate. Calculation complexity is also greatly reduced. The classifier is designed after the feature-selection process is completed. There are two methods for training a classifier: supervised learning (also called supervised clustering) and unsupervised learning. The main difference between these methods is that the labels of instances in supervised learning are known to the learner. This is not the case for unsupervised learning. The performance of the classifier is evaluated using test instances. The components shown in Table 1 are not independent. To improve overall performance, one component may feed back to the previous component, and the previous component is designed again.



▲Figure 1. Pattern-recognition system framework.

▼Table 1. Component definition of pattern recognition system

Component	Definition
Sensors	Obtain raw sensor data
Feature extraction	Extraction of features from raw sensor data
Feature selection	Selection of best feature subset
Classifier design	Modeling of classifier using training data
Evaluation	Evaluation of the performance of classifier model

In this paper, we use the general pattern-recognition framework to recognize human motion. We mainly focus on the classifier design component; the feature extraction and feature selection components are merged into one procedure.

2.2 Motion Definitions

Common human motions include keeping still, walking, running, climbing stairs, using an elevator, driving, and taking a bus. The possible motion set differs in different scenarios. In this paper, we limit the scenario to an office. The motions related to this scenario are shown in **Table 2**.

2.3 Data Collection and Preprocessing

In this paper, a three-axis linear smartphone accelerometer

▼Table 2. Motion state definition

Motion state	Definition
M1	Still
M2	Walking
M3	Climbing up stairs
M4	Climbing down stairs
M5	Running

is used to collect the raw motion data of a person. In fact, instead of the total acceleration, the linear acceleration infers the motions of a human. The collection process is controlled by an application we developed. Through a simple graphic user interface, we can start or stop collecting. The sensor sampling rate is 50 Hz, so we obtain 50 raw samples per second.

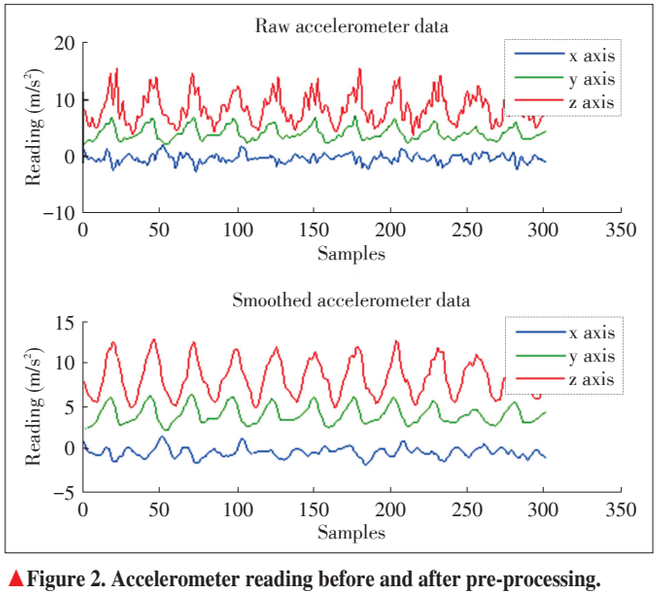
Because there are some noises in the raw sensor data, a five-stage moving window is used to eliminate them. **Fig. 2** shows the output of the moving window is smoother than the raw linear accelerometer data.

2.4 Feature Extraction

Features from the linear acceleration are used for motion recognition. In most of the previous work involving inertial sensors for motion recognition, the sensors are mounted on the human body in a known orientation and position. However, this is not practical in real life because of the flexible use of smartphones. To avoid the orientation problem, instead of directly using the features from x, y, z axes, the vertical and horizontal components of the linear acceleration are extracted by projecting the linear acceleration vector to the gravity vector. Some smartphones have the gravity sensor imbedded directly. With others, the gravity can be obtained using the following method. First, the smartphone is kept still for a few seconds. Then, the averages of the three axis readings are calculated as the approximation of the gravity vector [23].

Suppose $\vec{a}=(a_x,a_y,a_z)$ is the linear acceleration vector and $\vec{g}=(g_x,g_y,g_z)$ is the gravity vector. Then the vertical component of linear acceleration is $\vec{a}_v=\left(\frac{\vec{a}\times\vec{g}}{\vec{g}\times\vec{g}}\right)\vec{g}$ and the horizontal component of the linear acceleration is $\vec{a}_h=\vec{a}-\vec{a}_v$.

Both time and frequency domain features are extracted to construct the feature vector. In the time domain, mean, variance, skewness, kurtosis, and so on are the features used. The formulas for these features are shown in **Table 3**. In the frequency domain, an FFT algorithm is applied, and first dominant frequency, second dominant frequency, and the amplitude of the first and second dominant frequencies are the features used. All the features used are shown in **Table 4**. The instance window for extracting features is 2 s, and a 50% overlapping window is used to obtain feature vectors more efficiently. This has been demonstrated to be useful [24].



▲**Figure 2. Accelerometer reading before and after pre-processing.**

3 Incremental Learning

Recently, incremental learning has been popular in data mining and has attracted the attention of both academia and industry. Many of today's data-intensive computing applications require an algorithm that is capable of incrementally learning from large-scale dynamic data. An incremental learning algorithm learns new knowledge continuously over time and up-

▼**Table 3. Statistical feature definitions**

Feature	Definition
Mean	$\bar{m}=\frac{1}{N}\sum_{i=1}^{N}x_i$
Variance	$\sigma^2=\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{m})^2$
Skewness	$skew=\frac{1}{N\sigma^3}\sum_{i=1}^{N}(x_i-\bar{m})^3$
Kurtosis	$kur=\frac{1}{N\sigma^4}\sum_{i=1}^{N}(x_i-\bar{m})^4$

▼**Table 4. Human motion feature definitions**

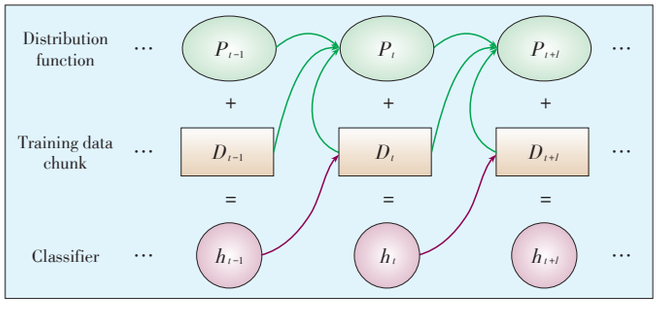
Feature	Definition
linaccM_mean	Mean of module of linear acceleration
linaccM_var	Variance of module of linear acceleration
linaccM_skew	Skewness of module of linear acceleration
linaccM_kur	Kurtosis of module of linear acceleration
linaccV_mean	Mean of vertical component of linear acceleration
linaccV_var	Variance of vertical component of linear acceleration
linaccV_skew	Skewness of vertical component of linear acceleration
linaccV_kur	Kurtosis of vertical component of linear acceleration
linaccH_mean	Mean of horizontal component of linear acceleration
linaccH_var	Variance of horizontal component of linear acceleration
linaccH_skew	Skewness of horizontal component of linear acceleration
linaccH_kur	Kurtosis of horizontal component of linear acceleration
firstfreq	The first dominant frequency
firstpeak	The amplitude of the first dominant frequency
secondfreq	The second dominant frequency
secondpeak	The amplitude of the second dominant frequency
energy	The mean energy of linear acceleration

dates the knowledge system to benefit future learning and decision-making.

In the human motion recognition domain, most researchers have used batch processing methods to recognize human motion. A classifier that uses batch processing keeps an invariable knowledge system after the learning. Usually, the classifier cannot learn new knowledge directly in order to expand itself unless all historical training data is gathered together with the new training data. Compared with incremental learning, batch mode learning, which updates itself, requires much storage space for historical training data, and the new training process is slower.

3.1 Framework for Incremental Learning

Inspired by the classification performance in [25], we use a similar incremental learning framework for human motion recognition. **Fig. 3** gives an overview of the algorithm. The learner is continuously presented with the training data flows over



▲**Figure 3. Incremental learning overview.**

time. The previous knowledge in this case includes the hypothesis, which was developed at time $t-1$, and the distribution function P_{t-1} is applied to the data set D_{t-1}. For the first block of the received training data, the initial distribution function P_1 is given a uniform distribution because nothing has been learned yet. As data blocks continually come in, a series of distribution functions is developed to represent the learning capability of the data in each block. Based on distribution function P_t, a hypothesis h_t, which is a human motion classifier, can be developed. In this classifier, the decision boundary is automatically forced to focus more on difficult-to-learn regions. When P_t and h_t have been obtained, the system uses its knowledge to facilitate learning from the next block of training data D_{t+1}.

Algorithm 1 can be divided into two parts. First, a mapping function estimates the initial distribution function of the current training data using the last input training data set and its corresponding distribution function. Second, a classifier using training data with probability distribution function is developed.

Algorithm 1. Incremental learning algorithm

Input: Sequence of data chunks $D_t, (t=1,2,3,...)$, each data chunk includes some instances. An instance is composed of a feature vector and a label.

Learning process:

1. For the first data set, suppose there are m samples, $D_1=\{x_i,y_i\}, (i=1,2,...,m_1)$, where x_i is the feature vector of the i-th sample and y_i is the label of the i*th* sample. The initial distribution function P_1 is a uniform distribution.

2. Train a classifier h_1 using the training data set D_1 and its corresponding probability distribution function P_1

3. For $t \geqslant 2$, use the last data set D_{t-1} and distribution function P_{t-1} to estimate the distribution function $\hat{P}_t$ of the new training data set D_t

4. Apply the last classifier h_{t-1} to the new data set D_t, calculate the pseudo error e_t:

$$e_t = \sum_{\substack{j:h_{t-1}(x_j)\neq y_j \\ (x_j,y_j)\in D_t}} \hat{P}_t(j) .$$

5. Set $\beta_t = e_t/((n-1)(1-e_t))$, n is the number of all different labels.

6. Update the distribution function of D_t, the misclassified sample will get a higher probability or weight:

$$P_t(j) = \frac{\hat{P}_t(j)}{Z_t}\begin{cases}\beta_t & if\ h_{t-1}(x_j)=y_j \\ 1 & otherwise\end{cases},$$

where Z_t is a normalization constant so that P_t is a distribution and $\sum_{j=1}^{m_t} P_t(j)=1$.

7. Model the classifier h_t using the training set D_{t+1} and its corresponding distribution function P_t

8. When new training data set comes, go back to 3 and repeat the procedure.

Output: After T classifiers have been trained, we obtain T basic classifiers and their corresponding weights. The final output is obtained by the combination of these. Here, $\log(1/\beta_t)$ is the weight of each basic classifier:

$$h_{final}(x) = \arg\max_{y\in Y} \sum_{t:h_T(x)=y} \log(1/\beta_t) .$$

3.2 Mapping Function Design

The mapping function is an important component in the incremental learning framework. It connects past experience to the newly received data and adapts such knowledge to the data sets received in future. Nonlinear regression models can be used as well as mapping functions. Here, we consider support vector regression (SVR) [26]. Suppose $y=f(x)$ is the estimated initial weight for instance x:

$$f(x)=<s,x>+b ,$$

where s and b are the slope and intercept of the linear estima-

tion function $f(x)$, respectively. To obtain an accurate estimation $f(x)$, a convex optimization problem can be extracted:

$$(s_t,b_t)=\arg\min_{s_t\in R^n,b_t\in R}\left(\sum_{i=1}^{m_{t-1}}\|y_i-<s_t,x_i>-b_t\|^2\right).$$

Alternative strategies can be used as well. For example, other types of regression model, such as multilayer perceptron or regression tree, can be integrated into the incremental learning framework according the application requirements.

3.3 Hypothesis Based on Probability-Distributed Training Data

In the conventional classifier design process, all instances in the training data set have the same weight. This means that instances that are difficult to classify are treated the same as those that are not. However, instances that are difficult to classify should be weighted more heavily so that they will be more easily recognized in the newly designed classifier.

Here, we introduce two methods for classifier modeling, both of which use the probability distributed training data. With the first method, the probability is used as the weight of the instance in the calculation of the decision boundary. We illustrate this method in the proposed weighted decision-tree algorithm. With the second method, the probability distribution function is used as the sampling probability. All training instances are given a sampling probability and sampled into the real training set. The higher the instance's probability, the easier it is added to the real training set.

3.3.1 Weighted Decision Tree

Decision tree is a classic algorithm used in pattern recognition and machine learning. It uses the information entropy gain to split training data, and it constructs a tree to represent the classifier. The information entropy of a tree node in the decision-tree algorithm is given by:

$$I(t)=-\sum_{i=1}^{M}P(\omega_i|t)\log_2 P(\omega_i|t),$$

where t is the current node, and $P(\omega_i|t)$ is the probability of the class ω_i in node t. In a conventional decision tree, it is N_{ω_i}/N_t, in which N_{ω_i} is the number of instances belonging to class ω_i, and N_t is the total number of instances in node t.

When node t is split, maximum entropy gain criterion is used. The formula is:

$$\Delta I(t)=I(t)-\frac{N_{t_l}}{N_t}I(t_l)-\frac{N_{t_r}}{N_t}I(t_r),$$

where t_l is the left child node, N_{t_l} is the number of the instances in the left child node, t_r is the right child node, and N_{t_r} is the number of instances in the right child node. In the conventional decision tree, all the instance have the same weight, i.e., $1/N_t$.

In this paper, we propose a weighted decision tree in which each training instance is given a different weight instead of $1/N_t$ (the original decision tree is a special case of the weighted decision tree). Thus, the heavier the weight of an instance, the bigger the information entropy of its class. This forces the decision boundary to focus on the more heavily weighted instances, which are difficult to learn. Thus, in the next incoming data block, the instance that is difficult to learn is weighted more heavily, and the final classifier will recognize it more easily.

In the information entropy formula, $P(\omega_i|t)$ is the sum of the weights of the instances in class ω_i. In the split formula, the factor $I(t_l)$ is the sum of the weights of instances in the left child node. The factor $I(t_r)$ is the sum of the weights of instances in the right child node. Thus, the instance that is difficult to learn will be weighted more heavily, and the instance that is easy to learn weighted more lightly.

3.3.2 Sampling Probability Function

With the sampling probability function method, all training data is sampled into the real training set according to their probability. The instance which is hard to learn will have a higher probability to be sampled into the real training set. However, low probability instance will have a lower probability to be chosen into the final training set. That means there may exist several duplicates of the hard instance in the real training set, and easy instance may not exist in the final training set. More basic classifier category can be contained in this framework.

4 Experiments and Evaluation

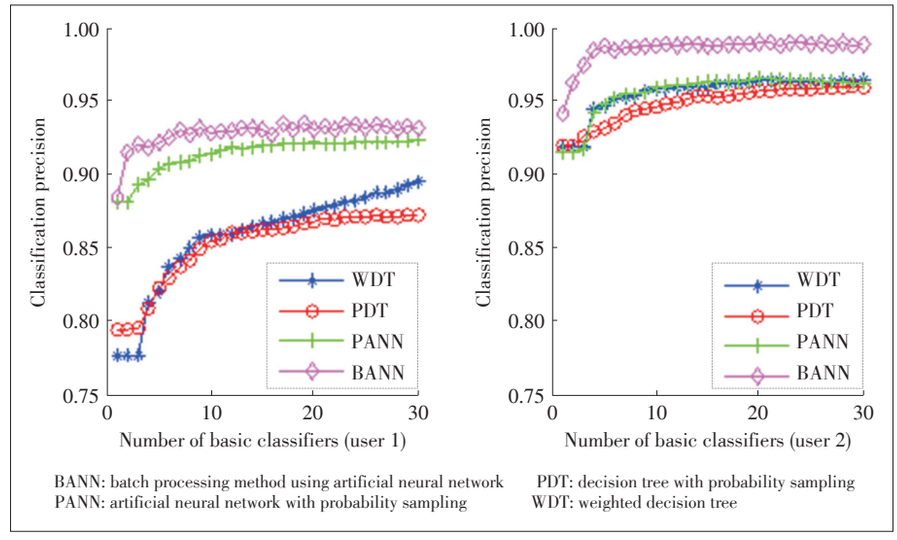
The device used in our experiments is a Google Nexus 5. This smartphone has a built-in tri-axial MPU 6515 accelerometer that records the user's raw motion data. The Android system uses a filtering algorithm to extract the linear accelerometer and gravity accelerometer. Experimental data was collected from two males. One was 1.85 m tall and weighed 70 kg. The other was 1.75 m tall and weighed 65 kg. Both males stood still, walked, ran, and climbed up and down stairs in and around our office building. The smartphone was held in the hand with the phone screen facing upwards. All sensor data was stored and processed offline.

4.1 Classification Precision Test

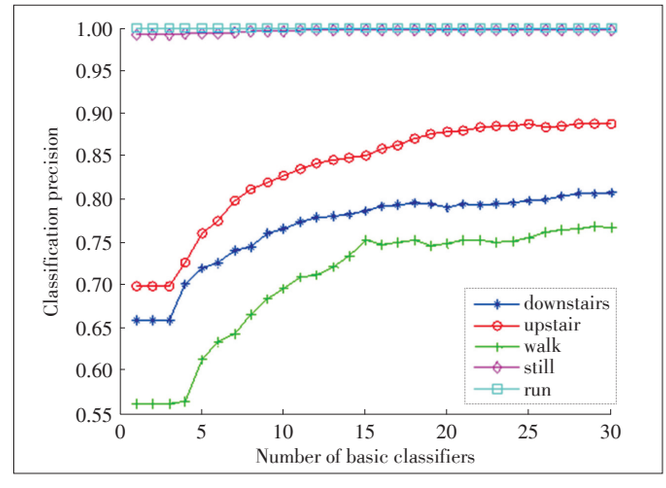
To determine the performance of incremental learning, the data was split into many blocks. In our test, there were 250 instances in a block; however, it is not essential to have the same number of instances in each block. The data blocks were input into the incremental-learning framework to simulate the continuous learning process. The two important parts of the incremental learning process are mapping function design and motion classifier modeling based on probability-distributed training

data. An artificial neural network was used for the mapping function because such a network is well integrated into the MATLAB toolbox. Weighted decision tree and sampling probability methods were used for classifier modeling. At the same time, batch processing method using an artificial neural network is used for the comparison.

The results of classification based on different learning methods are shown in **Fig. 4**. Incremental learning using three kinds of hypothesis strategy—weighted decision tree (WDT), decision tree with probability sampling (PDT), and artificial neural network with probability sampling (PANN) —results in a higher classification rate with continuous learning. For both user 1 and user 2 probability sampling, PANN performs better than PDT. When WDT is used for user 1, classification precision is only slightly higher than that of PDT when the number of basic classifiers is large enough. However, when WDT is applied to user 2, it performs better than both PDT and PANN, both of which use probability sampling. The increasing curves show that the incremental learning algorithm learns the new knowledge continuously and benefits future decision making. The classification rate of the batch processing method using artificial neural network (BANN) also increases when there is more training data. Both incremental learning and batch learning tend to be stable when there is enough training data. For example, PANN for user 2 needs be completed about ten times to be stable in this scenario. Because batch processing uses the training data sufficiently each time, it has a higher classification precision than incremental learning. Incremental learning using some strategy performs almost as well as batch processing. For user 1, the classification precision of PANN is slightly less than that of BANN (Fig. 4).



▲**Figure 4. Classification precision of incremental learning and batch learning.**



▲**Figure 5. Classification precision of each motion state.**

Fig. 5 shows how the classification of each motion changes for incremental learning. Take the incremental learning process of user 1 with WTD for example.

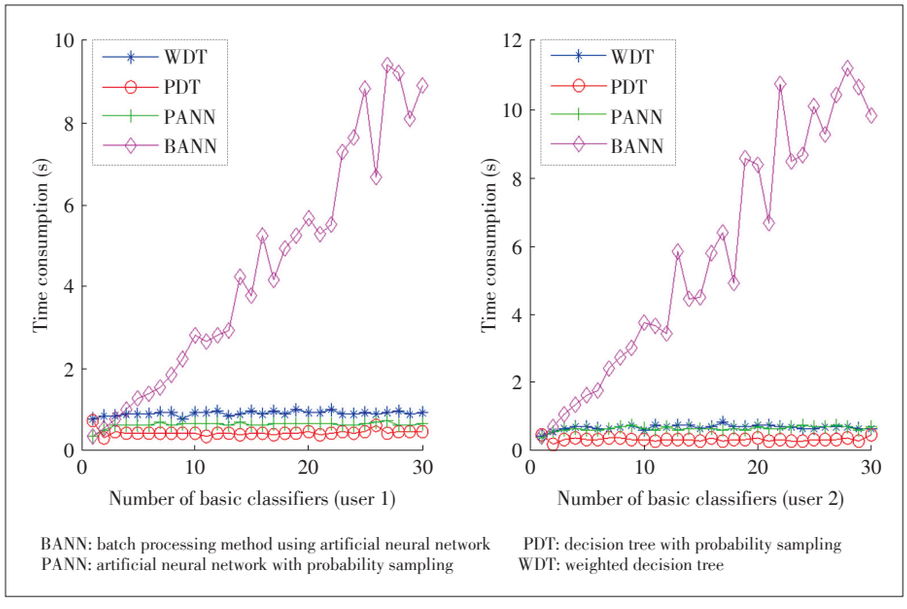
Because still and run have different feature spaces than other motions, they can be easily recognized. However, according to the smartphone inertial sensors, the person may appear to be doing similar motions when going downstairs, upstairs or walking. These three motions belong to classes of motions that are difficult to learn in this scenario. With continuous learning, the former basic classifier delivers its learned knowledge to the next classifier and forces the decision boundary to focus on the three hard-to-learn regions. Therefore, the hard-to-learn motions will have higher classification precision after learning. The confusion matrix after incremental learning is shown in **Table 5.** The average of the motion classification precision is 88.43%, which is high enough for many applications.

4.2 Computational Complexity Test

Although batch processing may be more precise than incremental learning, a huge amount of storage space and more complex computation are required. Incremental learning does not require historical training data to be stored. The computational complexity is shown in **Fig. 6**. The more complex the computa-

▼**Table 5. Motion classification confusion matrix**

Motion	Downstairs	Upstairs	Walking	Still	Running	Precision
Downstairs	115	13	17	2	0	80.60%
Upstairs	7	160	24	0	0	83.77%
Walking	21	19	140	0	0	77.78%
Still	0	0	0	226	0	100%
Running	0	0	0	0	202	100%



▲Figure 6. Time consumption of incremental learning and batch processing mode.

tion is, the more time the process requires. Therefore, the amount of time needed to complete the process reflects computational complexity. As training data is continuously input, the training process time is recorded for both incremental learning and batch processing modes (Fig. 6).

Regardless of which hypothesis which strategy used, the time required for incremental learning remains approximately constant, and training data is continually input. However, the amount of time needed for batch mode learning is linear to the amount of training data. With incremental learning only the new incoming data needs to be disposed. No historical training data needs to be stored or used in the new basic classifier training process, and the time to complete the process only needs to be approximately constant. Batch processing requires historical data to be stored and used to train the new classifier. Each time new training data arrives, the overall amount of training data increases linearly. Therefore, the time required in batch processing mode increases linearly at the same time. The average time consumption for both users is shown in **Table 6**. The time needed in batch processing mode BANN is several times that in incremental learning mode for both users. Because the time needed in batch processing mode increases linearly as data continually arrives, this ratio continues to increase.

▼Table 6. Average time consumption using different learning methods

	WDT	PDT	PANN	BANN
User 1	0.889 s	0.422 s	0.613 s	4.414 s
User 2	0.658 s	0.305 s	0.620 s	5.692 s

BANN: batch processing method using artificial neural network
PANN: artificial neural network with probability sampling
PDT: decision tree with probability sampling
WDT: weighted decision tree

5 Conclusion

In this paper, we have used the incremental learning method to recognize human motion. First, a mapping function was used to deliver learned knowledge to incoming data. Then, a basic classifier is modeled according to the training data with distribution. A weighted decision tree was proposed to illustrate hypothesis construction, and a sampling probability method was also employed. With continuous incoming data, the incremental learning method almost has the same classification precision as batch processing method. However, the incremental learning method has lower computational complexity and is much faster in the training phase. Considering the tradeoff between classification precision and training time, incremental learning is better than batch processing when there is huge amount of input data.

Acknowledgement

The authors would like to thank the editors and the reviewers for their very helpful comments and review. The authors are also grateful to the team members in the Shanghai Key Laboratory of Navigation and Location Based Services of Shanghai Jiao Tong University.

References

[1] L. Pei, R. Chen, J. Liu, et al., "Using motion-awareness for the 3D indoor personal navigation on a Smartphone," in *Proc. 24rd International Technical Meeting of the Satellite Division of the Institute of Navigation*, Portland, USA, Sept. 2011, pp. 2906–2912.

[2] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, Oct. 2010. doi: 10.1016/j.patcog.2010.04.019.

[3] C. Liu, L. Pei, J. Qian, et al., "Sequence-based motion recognition assisted pedestrian dead reckoning using a smartphone," in *6th China Satellite Navigation Conference (CSNC)*, Xi'an, China, 2015, pp. 741–751,. doi: 10.1007/978-3-662-46632-2_64.

[4] L. Pei, R. Chen, J. Liu, et al., "Motion recognition assisted indoor wireless navigation on a mobile phone," in *Proc. 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation*, Portland, USA, Sept. 2010, pp. 3366–3375.

[5] J. Qian, L. Pei, J. Ma, R. Ying, and P. Liu, "Vector graph assisted pedestrian dead reckoning using an unconstrained smartphone," *Sensors*, vol. 15, no. 3, pp. 5032–5057, Mar. 2015. doi: 10.3390/s150305032.

[6] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, Mar. 2001. doi: 10.1006/cviu.2000.0897.

[7] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, Mar. 2003. doi: 10.1016/S0031

-3203(02)00100-0.
[8] T. Y. Chung, Y. M. Chen, and C. H. Hsu, "Adaptive momentum-based motion detection approach and its application on handoff in wireless networks," *Sensors*, vol. 9, no. 7, pp. 5715–5739, Jul. 2009. doi: 10.3390/s90705715.
[9] D. T. P. Fong and Y. Y. Chan, "The use of wearable inertial motion sensors in human lower limb biomechanics studies: a systematic review," *Sensors*, vol. 10, no. 12, pp. 11556–11565, Dec. 2010. doi: 10.3390/s101211556.
[10] L. Pei, R. Guinness, R. Chen, et al., "Human behavior cognition using smartphone sensors," *Sensors*, vol. 12, no. 2, pp. 1402– 1424, Jan. 2013. doi: 10.3390/s130201402.
[11] R. Chen, T. Chu, K. Liu, J. Liu, and Y. Chen, "Inferring human activity in mobile devices by computing multiple contexts," *Sensors*, vol. 15, no. 9, pp. 21219–21238, Aug. 2015. doi: 10.3390/s150921219.
[12] B. Musleh, F. García, J Otamendi, et al., "Identifying and tracking pedestrians based on sensor fusion and motion stability predictions," *Sensors*, vol. 10, no. 9, pp. 8028–8053, Aug. 2010. doi: 10.3390/s100908028.
[13] W. Chen, Z. Fu, R. Chen, et al., "An integrated GPS and multi-sensor pedestrian positioning system for 3D urban navigation," *2009 Joint Urban Remote Sensing Event*, Shanghai, China, May 2009, pp. 1– 6. doi: 10.1109/URS.2009.5137690.
[14] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Computing*, vol. 3001, pp. 1–17, Apr.2004,. doi: 10.1007/978-3-540-24646-6_1.
[15] J. Lester, T. Choudhury, N. Kern, et al., "A hybrid discriminative/generative approach for modeling human activities," in *IJCAI*, Edinburgh, UK, 2005, pp. 766–772.
[16] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *Proc. 1st ACM International Workshop on Interactive Multimedia for Consumer Electronics*, Beijing, China, Oct. 2009, pp. 1–10. doi: 10.1145/1631040.1631042.
[17] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no.2, pp. 74–82, Dec. 2011. doi: 10.1145/1964897.1964918.
[18] E. Miluzzo, N. D. Lane, K. Fodor, et al., "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *Proc. 6th ACM Conference on Embedded Network Sensor Systems*, Raleigh, USA, Nov. 2008, pp. 337–350. doi: 10.1145/1460412.1460445.
[19] S. Ozawa, S. Pang, and N. Kasabov, "Incremental learning of chunk data for online pattern classification systems," *IEEE Transactions on Neural Networks*, vol. 19, no. 6, pp. 1061–1074, Mar. 2008. doi: 10.1109/TNN.2007.2000059.
[20] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, no. 4, pp. 497–509, Nov. 2001. doi: 10.1109/5326.983933.
[21] A. Bouchachia, M. Prossegger, and H. Duman, "Semi-supervised incremental learning," in *IEEE International Conference on Fuzzy Systems (FUZZ)*, Barcelona, Spain, Jul. 2010, pp. 1–6. doi: 10.1109/FUZZY.2010.5584328.
[22] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, Nov. 2011. doi: 10.1177/0278364911426178.
[23] L. Pei, J. Liu, R. Guinness, et al., "Using LS-SVM based motion recognition for smartphone indoor wireless positioning," *Sensors*, vol. 12, no. 5, pp. 6155–6175, May 2012. doi: 10.3390/s120506155.
[24] N. Ravi, N. Dandekar, P. Mysore, et al., "Activity recognition from accelerometer data," in *AAAI-05*, Pittsburgh, USA, Ju. 2005, pp. 1541–1546.
[25] H. He, S. Chen, K. Li, et al., "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, Oct. 2011. doi: 10.1109/TNN.2011.2171713.
[26] H. Drucker, C. J. C. Burges, L. Kaufman, et al., "Support vector regression machines," *Advances in Neural Information Processing Systems*, vol. 9, pp. 155–161, 1997.

Manuscript received: 2015-07-20

Biographies

LIU Chengxuan (lcxstorm@163.com) is studying for the ME degree in the Department of Information and Communication Engineering, Shanghai Jiao Tong University, China. He received his BE degree from Shanghai Jiao Tong University. His research interests include pattern recognition, multisource navigation, and positioning technology.

DONG Zhenjiang (dong.zhenjiang@zte.com.cn) is the vice president of the Cloud Computing & IT Research Institute of ZTE Corporation. His main research areas are cloud computing, big data, new media, and mobile internet technologies.

XIE Siyuan (xie.siyuan7@zte.com.cn) is a pre-research engineer at ZTE Corporation. His main research areas are indoor positioning, IoT, and mobile internet technologies.

PEI Ling (ling.pei@sjtu.edu.cn) is an associate professor in the Department of Electronic Engineering, Shanghai Jiao Tong University, China. He received his PhD degree from Southeast University, China. His research interests include indoor and outdoor seamless positioning, context-aware technology, and motion pattern recognition.