J Supercomput (2016) 72:3927–3959
DOI 10.1007/s11227-016-1639-5

Improvised methods for tackling big data stream mining challenges: case study of human activity recognition

Simon Fong[1] · Kexing Liu[1] · Kyungeun Cho[2] · Raymond Wong[3] · Sabah Mohammed[4] · Jinan Fiaidhi[4]

Published online: 16 February 2016

Abstract Big data stream is a new hype but a practical computational challenge founded on data streams that are prevalent in applications nowadays. It is quite well known that data streams that are originated and collected from monitoring sensors accumulate continuously to a very huge amount making traditional batch-based model induction algorithms infeasible for real-time data mining or just-in-time data analytics. In this position paper, following a new data stream mining methodology, namely stream-based holistic analytics and reasoning in parallel (SHARP), a list of data analytic challenges as well as improvised methods are looked into. In particular, two types of decision tree algorithms, batch-mode and incremental-mode, are put under test at sensor data that represents a typical big data stream. We investigate whether and to what

✉ Simon Fong
ccfong@umac.mo

Kexing Liu
mb45462@umac.mo

Kyungeun Cho
cke@dongguk.edu

Raymond Wong
wong@cse.unsw.edu.au

Sabah Mohammed
sabah.mohammed@lakeheadu.ca

Jinan Fiaidhi
jfiaidhi@lakeheadu.ca

1 Department of Computer and Information Science, University of Macau, Macau, SAR, China

2 Department of Multimedia Engineering, Dongguk University, Seoul, Korea

3 School of Computer Science and Engineering, University of New South Wales, Sydney, Australia

4 Department of Computer Science, Lakehead University, Thunder Bay, Canada

extent of two improvised methods—outlier removal and balancing imbalanced class distributions—affect the prediction performance in big data stream mining. SHARP is founded on incremental learning which does not require all the training to be loaded into the memory. This important fundamental concept needs to be supported not only by the decision tree algorithms, but by the other improvised methods usually at the preprocessing stage as well. This paper sheds some light into this area which is often overlooked by data analysts when it comes to big data stream mining.

Keywords Data stream mining · Big data · Very fast decision tree · Resampling · Sensor data

1 Introduction

Recently the term "big data" was hyped up very largely by media in computing domain, as it is introducing three problematic issues infamously known as the 3V challenges: Velocity issue that leads to a huge amount of data to be processed at high speed; Variety issue that makes data processing and prediction model induction difficult because the data come from various sources in various formats; and Volume problem that makes archiving, processing, and analysis challenging, especially if a large bulk of data needs to be stored in the runtime memory for preprocessing.

In views of these 3V challenges, the traditional data mining approaches which are inherited from the full batch-mode learning may run short of satisfying the demand of analytic efficiency. That is due to the burden that the traditional data mining model construction techniques necessitate loading in the full set of data, and then the data set is partitioned by divide-and-conquer strategy. Two typical algorithms are classification and regression tree induction [1] and rough-set discrimination [2]. Whenever fresh data arrive, the size of the total training dataset increases, the traditional learning algorithm needs to re-run, and the current model needs to be re-built embracing the new data.

In contrast, a new breed of learning algorithms, called data stream mining methods [3], are capable of minimizing the impacts of the 3V big data problems. Data stream mining algorithm does not need to load up the full set of data. Rather it works by inducing a classification or prediction model incrementally by a bottom-up approach. Each segment of arriving data from the data streams prompts the model to update itself by only the current data segment. This type of read-and-forget design in data stream mining algorithms frees the system from reloading the whole data; hence it is able to handle big data streams that potentially accumulate to infinity without bound. This feature allows the data stream mining process run in limited runtime memory, learning and recognizing patterns from the data streams on the fly. Consequently, data stream mining methods are regarded as one of the big data analytics approaches to subside big data problems. Researchers agree lately that data stream mining algorithms are effective approaches to induce a prediction model from big data [4,5].

Recently a holistic data stream mining approach in gathering different auxiliary parts of data stream mining with a common aim of improving the ultimate prediction performance was proposed [6]. It is called stream-based holistic analytics and reasoning in parallel (or SHARP in short) which is based on principles of incremen-

tal learning and lightweight processing. SHARP is comprised of several components which cover a typical data-mining model construction process. They are lightweight feature selection, one-pass incremental decision tree induction, and incremental swarm optimization. Each one of these components in SHARP is supposed to complement each other towards the common objective of improving the classification/prediction performance as a whole. SHARP is scalable in computation; additional CPUs can be included in parallel for increasing the execution threads of independent performance enhancement. The benefits of SHARP include attaining the highest possible prediction accuracy while maintaining the computation as lightweight as possible; some components can be run independently, thereby allowing parallel processing and scalable solution. The operation of SHARP is in stochastic manner implying the longer it runs for, the better the performance it can achieve.

In this paper, we investigate the gaps between the proposed holistic methodology and the currently available so-called improvised methods in solving typical problems in data mining. As a pioneer attempt, we try subside the data outliers and imbalanced class distributions problems on the tradition decision tree algorithm and very fast decision tree algorithms. The effects of removing outliers and balancing imbalanced class distribution over the two types of algorithms are measured for comparison. The former algorithm is based on CART which represents traditional data mining, and the latter one is a classical algorithm in data stream mining. Do these improvised methods that worked for traditional mining algorithm still work for data stream mining? This is the research question in mind which would be answered, by conducting computer simulation experiment with a case study of human activity recognition over sensor data streams.

The remainder of the paper is structured as follows: Section 2 offers a background about the SHARP methodology and its components. Section 3 presents the challenges associated with human activity recognition and challenges in big data stream mining. The technological gaps between SHARP and improvised methods are identified too. The experimentation is described in detail in Sect. 4. Section 5 concludes the paper.

2 Background of SHARP methodology

A model of SHARP for data stream mining is shown in Fig. 1. It is comprised of several components that work cooperatively together during the data stream mining operation: (1) Cache receiver (CR); (2) Incremental classifier (IC); (3) Incremental feature selection module (IFS); (4) Factor analysis module (FA); and (5) Swarm optimizer (SO). The methodology offers a holistic approach which takes care of most if not all the possible aspects in data mining for improving performance. These five components aim at achieving the following objectives: CR-objective is to subside the problems of missing/incomplete data; IC-objective is to enable stream forecasting/prediction/classification by incremental learning manner; IFS and FA-objective is to understand the reasons and influences of the respective data attributes towards the predicted class; and SO- objective is to fine-tune the parameter values including selecting the optimal feature subset regularly. All these components contribute to the overall performance improvement, and they can function concurrently as the data stream in.

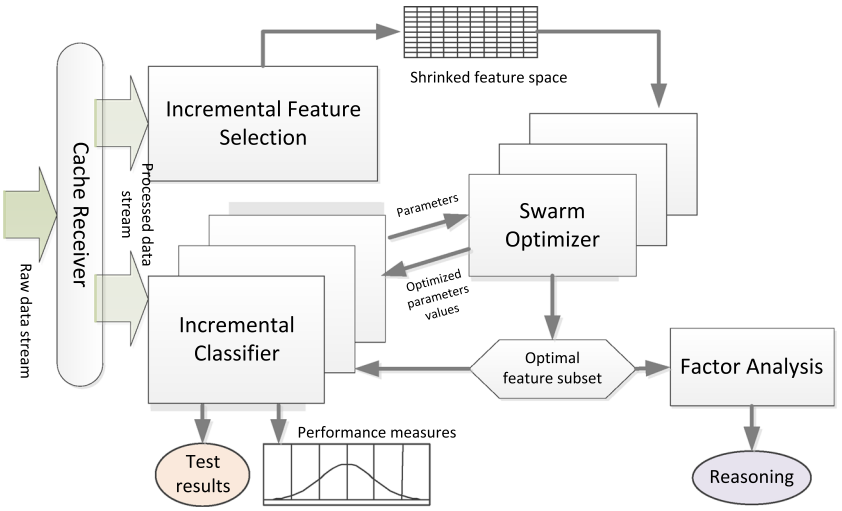


Fig. 1 SHARP processes

The multiple rectangles in Fig. 1 that represent IC and SO, respectively, depict the possibility that these processes can run concurrently over some parallel computing devices. The components are briefly described as follows: It is acknowledged that there exist many possible solutions or algorithms for implementing these components. Our discussion, however, highlights only some of the state-of-the-art components developed from our previous projects. The methodology serves as an abstract guideline on the possible integration of several data streaming components, discussing their functions (rather than implementation), interfaces, and advantages.

2.1 Cache receiver (CO)

The cache receiver is a front-end pre-processing mechanism that holds certain amount of data from the incoming data stream for a while. The main function is to minimize the latency of data arrivals as it is possible and likely that data streams that are being aggregated from various sources would be received at different speeds. CO acts as a delay regulator and buffer allowing opportunities for efficient data cleansing mechanism to operate upon, in real-time. It is not uncommon that data streams are stained with noise and incomplete information; techniques have been proposed and studied previously. The techniques mainly centered on delivering and synchronizing the cache-buckets, estimating missing data, and detecting and alleviating concept-drift problems, etc. CO also handles other basic data pre-processing tasks similar to those for traditional data mining in the KDD process. Data stream is partitioned into two portions: one for training that goes to the IFS and the other for testing at the IC. Some improvised methods mentioned in this paper such as outlier detection, missing-data estimations, class data rebalancing, and redundant data removal are implemented at CO, that is, where pre-processing by these improvised methods take place.

2.2 Incremental classifier (IC)

Many choices exist when it comes to data stream mining algorithms such as those which are available on massive online analysis (MOA) [7] developed by University of Waikato, New Zealand. Some popular algorithms include, but not limited to, decision stump, Hoeffding tree, Hoeffding option tree, Hoeffding adaptive tree, and ADWIN, etc. The algorithms have a common design basis that works by incremental learning approach. The model gets rebuilt partially by only seeing enough samples that are qualified (or biased) for growing an additional decision tree branch (or rule). In such way, the model in induced progressively from scratch to a full-grown mature decision tree, which has seen enough data stream samples, being able to recognize the mappings between the attributes and the target classes.

2.3 Incremental feature selection module (IFS)

The objective of this module is twofold; one is supposed to shrink down the total combination of feature subsets by simple selection algorithm, the other is to reason about the importance of the attributes with respect to the predicted classes, such as attribute scoring. There are plenty of available algorithms for incremental feature selection. Some popular ones include Grafting [8] which is based on the heuristic of gradient descent in function space, some are based on rough set theory on dynamic incomplete dataset [9], and the incremental feature ranking method over dynamic feature space [10], to just name a few. One of the latest state-of-art algorithms, called clustering coefficients, of variation (CCV) [11] is relatively simple and hence suitable for lightweight computation in SHARP. In the case of SHARP, CCV helps to shrink the feature space by eliminating the disqualified features and the combinations of such features. The feature space that has been reduced in size will then been used by the SO for finding the most optimal feature subset by metaheuristics search algorithms.

2.4 Factor analysis module (FA)

IFS and FA usually work together (or in parallel as in SHARP), IFS producing the selected features and FA offering insights into the significance of attributes to the predicted classes. In general, this method is to correlate a large number of features in a dynamic dataset with an outcome variable, such as the predicted class. Computationally this is done by scoring each feature by some statistical means (correlation is one of them). The other types of feature scoring exist such as gain ratio, information gain, Chi-square evaluation, etc. that have similar methods for scoring. As a result of FA, a list of features sorted by values in ascending or descending order would be produced; their rankings could be visualized too. It offers insights to users about the importance of each attribute for inquisitives.

2.5 Swarm optimizer (SO)

A swarm optimizer is essentially a search module that looks for optimal parameter values of the classifier in use and the optimal feature subset for the IC module. SO

takes the output of IFS which is a reduced search space of feasible combinations of feature subsets as input. Multiple search agents, which often are inspired by natural phenomena or behaviors of biological creatures, scout over the search space for the optimum solution. The search operation is in parallel as these multiple agents are working autonomously but collectively through some stochastic process. The search iterates through generations, thereby evolving the solution to an optimum one at the end. Some researchers call such methods meta-heuristics as it is meant to be a high-level strategy (therefore the name meta-) that guides the underlying heuristic search in achieving a goal. In the SHARP methodology, SO is an optimizer implemented by Swarm Search [12]. Swarm search is designed for finding the optimum feature subset using metaheuristics from large datasets.

Swarm Search is particularly useful for datasets that are characterized by a very large amount dimensionalities, so called features. Although the meta-heuristics is generic which is able to integrate any type of bio-inspired optimization algorithms into any type of classifier (at least theoretically), the work by [12] tested nine different combinations—three classifiers: neural network, decision tree, and Naïve Bayes, and three bio-inspired optimization algorithms: Wolf Search Algorithm, Particle Swarm Optimization, and Bat Algorithm.

For SHARP that demands for incremental learning and, therefore, progressive search for the best feature subset, Swarm Search should be configured to embed with only incremental algorithm. Depending on the setup at IC which may be an ensemble method where multiple classifiers are being tested, the most accurate model got selected, SO should operate in parallel too having each execution thread that corresponds to each candidate classifier as in IC. The composite optimization applies where the possible parameter values and the possible feature subset search space are blended together into a large search space, over which the Swarm Search attempts to find the best combination. This approach was pioneered by Iztok et al. in [13].

3 Computational challenges in mining sensor data stream

According to Google Trends, as shown in Fig. 2, in the past 9 years, the field of human activity recognition in the topic of gesture recognition has grown dramatically, reflecting its importance in many high-impact societal applications. They include smart surveillance, behavioral analysis, quality-of-life devices for elderly people, and human–computer interfaces.

At about the same time, data stream mining in the topic of stream processing emerged as a research trend that supports real-time incremental learning and fast data analysis. These two trends emerge and intertwine at almost the same period of time, indicating strong research interests over the years. By the forecast of the Google Trends, the futures of these two trends seem to meet again.

In the CVPR 2014 Tutorial on Emerging Topics in Human Activity Recognition, Michael Ryoo (NASA, ETRI, KAIST) gave an example of human activity recognition using CCTV video captures as sources of sensor data. Picture frames which are captured at the frame rate of dozens per second are analyzed and used to train a classifier to recognize a particular kind of human activity, e.g. whether the two people are pushing or not pushing by measuring their standing positions apart.

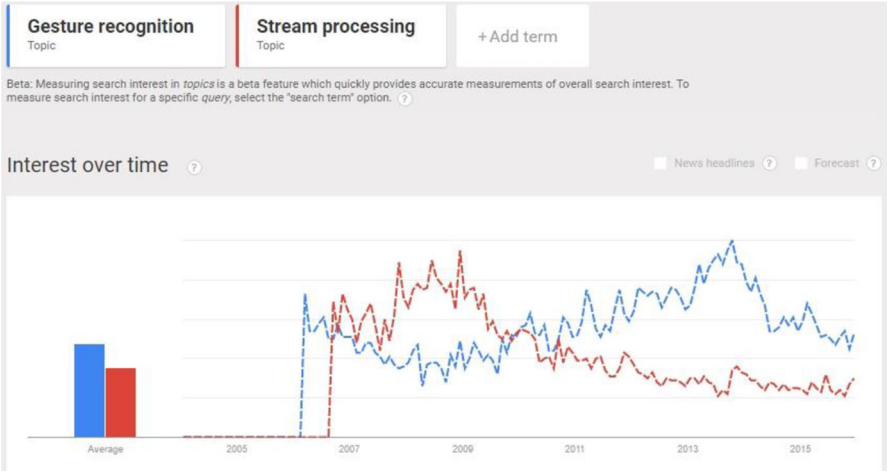


Fig. 2 Interest trends of human activity recognition and data stream mining

Ryoo advocated that human activity recognition, therefore, is comprised of the following computational tasks, if such complex activities are to be recognized by machine learning. The tasks are in four levels of abstractions and escalating complexities: (1) Tracking which keeps monitoring and estimating locations of human body parts of the subject; (2) Features—the bodily movements would have to be converted quantitatively into numerical status within a body part. e.g. lifting an elbow and swinging an elbow give rise to different numeric digests; (3) Action is a sequence of features that are narrated in the time domain; (4) Interaction is a composite of action sequences involving more than one party. Ryoo and his colleagues [14–16] summarized the various types of human activities to be recognized by computers: Gestures, Actions, Human-object interactions, Human–human interactions, and Group activities, as shown in Fig. 4. They are in different levels of complexities and, therefore, require different levels of information and computation methods.

We generalize these activities and their computational requirements, in the context of data mining, into the following: Increase of complexity in pattern recognition

Outlines → Features → Spatial information → Temporal sequence → Group contexts

The outlines and features of a gesture, e.g. hand gestures must be extracted into training/testing datasets for machine learning algorithms to understand and to learn. Gesture data are typically numeric data that are generated from some motion sensors as continuous data sequences in temporal domain. Usually the details of an image of a gesture is omitted, leaving only the outline of the subject and its motions. Gestures concern about merely the outline of the movement like hand sign language. Features are higher level information that usually comes with some semantics which are extracted from the raw outline data. Spatial information is important as it would be used in relation to the motion data of the subject. It adds meanings and references to the gesture data as well as bodily data and even group interaction data. Moreover, for better understanding the data, temporal sequence offers clues on the timing information and temporal relations among data segments when they are streaming in some ordered sequence.

Human-to-object recognition can be facilitated using RFID technology labeled on the object, tracking and analyzing both positions simultaneously. Note that computational complexity escalates as the level rises for recognizing human activities. At the end, given the context of groups, such as teams of players playing basketballs, high-level information like playing strategies, patterns of scoring or losing, can be derived from all these activity patterns made available from all the parties and environments involved.

Given the computational requirements of the human activity recognition, we relate them to the requirements in the context of data stream mining. Therefore, we formulate the following requirements for a future breed of data stream mining algorithms:

A new generation of data "stream" mining algorithms is needed:

1. Fast and accurate learner.
2. Learns details + sequences.
3. Learns changes dynamically.
4. Learns meta-information (the big picture).
5. Learns in parallel.

The five features are the 'must-have' elements to satisfy the data stream mining tasks for recognizing human activities by machines. Referring to Fig. 3, the rule induction process must consist of the five features so as to be able to handle the complex data stream mining tasks as indicated in Fig. 4 by Ryoo et al.

The algorithms in the rule induction process must be able to learn the activities from training data, fast and accurate. Speed and accuracy are usually in contradiction. Full learning requires processing all the data and this takes time, and vice-versa. Meta-learning is to be able to see the full picture. As shown in Fig. 5, the human activity recognition task can be viewed at different levels with different types of details. If a decision tree is to recognize the fine details like hand gesture, for example, the macro view may get missed out. In the example of a jigsaw puzzle in Fig. 5, it is analogous to build a decision tree to only recognize a single piece of jigsaw puzzle, e.g. when there is only a piece of green puzzle, it is inferred to just grass; when there is a cluster of puzzle pieces, it is inferred to a garden. Only when they are pieced together, they

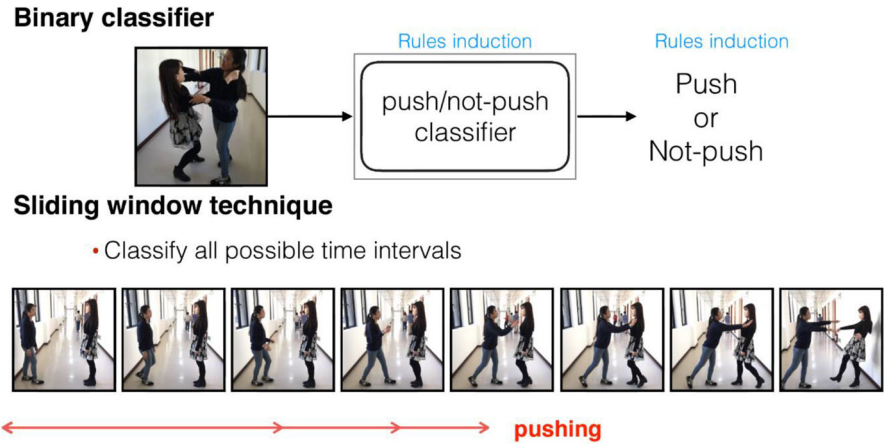


Fig. 3 Example of human activity recognition—push or not push, using CCTV captures

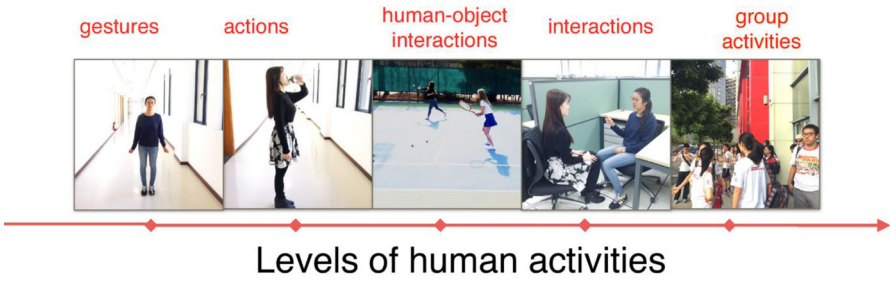


Fig. 4 Various types and levels of human activities

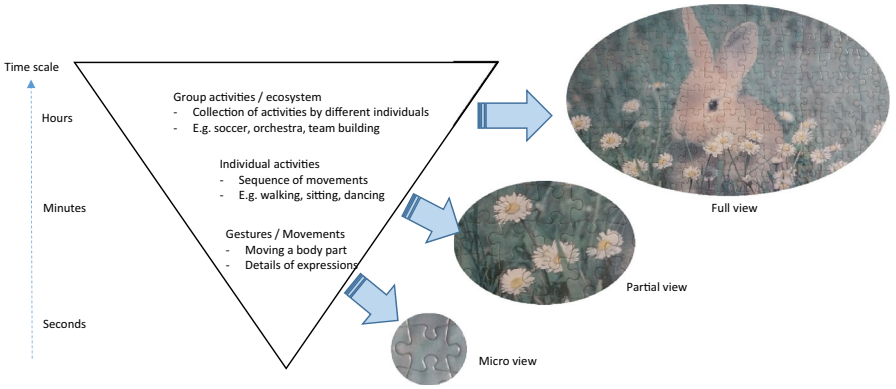


Fig. 5 Meta-learning is necessary to see the full picture

show a rabbit in a garden. It is hard to see the full picture, until when they are all pieced together. They show a face of smiley.

To sum up further from the desired list of new generation of data stream mining, existing improvised methods are proposed. By the definition from Oxford Dictionary, improvised means "produce or make (something) from whatever is available". Since early 70s over hundreds of methods (as evidenced by DM research publications) have been developed, yet there is no single perfect algorithm or a combination of standard algorithms that guarantees maximum performance. Ironically, speed and accuracy do contradict most of the times.

Some popular improvised methods include (but never limit to) the following:

- Outlier detection: For removing noise or finding salient features.
- Class balancing : For training a classifier that learns minority of samples.
- Feature selection: For finding a subset of significant features.
- Efficient rule induction: For generating a compact set of useful rules.

The contribution of this paper is to investigate the possibilities of applying the improvised methods for building a most appealing classifier which is fast and accurate. These improvised methods, outlier detection, class balancing and feature selection have been popularly used as preprocessing in traditional data mining. Now the same are applied on data stream mining models for inducing decision trees that recognize human activities.

4 Experiments

Two experiments are designed and conducted for verifying the efficacy of traditional decision tree (J48) and incremental decision tree, called very fast decision tree (VFDT), which classically represent the tree induction algorithms for batch data mining and data stream mining, respectively.

4.1 Experiment settings

Two sensor datasets which are generated from two types of motion detection are used for building decision trees for human activity recognition. The first data, which we term simply "3D" data, are collected from a smartphone with spatial information of X-Y-Z. The second data, more complicated, called "Kinect" data, are collected from Microsoft Kinect motion sensor. Both data have a target class called "Activity" with information labels that describe what the human activities are with respect to the instances of multi-attribute data. The 3D has 162,500 data instances. The Kinect data has 81,000 data instances. With such large data volumes and their streaming nature, these datasets are regarded as big data streams.

However, 3D data have only three attributes from the three spatial axes, X, Y, and Z that describe six human activities such as WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, and LAYING. The data are collected from volunteers who were wearing a smartphone (Samsung Galaxy SII) strapped on the waist. Using its embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity are captured at a constant rate of 50 Hz by the smartphone. The experiments have been video-recorded to label the data manually by a human annotator as a post-processing task. This type of data represents a scenario where data stream that is characterised by few spatial attributes is collected. The ratio of attributes to the categories of activities to be predicted is 3:6 (1:2). The data volume is double of that of Kinect data.

The Kinect data, on the other hand, are more complex, having 64 attributes that describe 30 different human activities. The attribute data are spatial X-Y-Z information of various parts of a human body such as head, shoulder, elbow, wrist, hand, spine, hip, knee, ankle, and foot, etc. The ratio of attributes to activities is 64:30 (2.13:1). They are downloaded from the Data Repository of D-lab of King Mongkut's University of Technology Thonburi,[1] Thailand. A volunteer performed in front of five Kinect cameras placed at different positions doing the 30 postures; the visual postures are coded according to 64 human-to-camera spatial information. The data of each position (x) was divided into test and train files ($P(x)$testingset.csv and $P(x)$trainingset.csv). The Kinect sensor was placed at five different positions facing the subject at five different angles. They are 0, 45, 90, 135, and 180 degrees. $P(x)$ stands for positive of camera at the xth angle of five different degrees used. An illustration that shows the five positions of the $P(x)$ Kinect sensors is shown in Fig. 6. This arrangement of sensors was chosen deliberately by testing the efficacy of human activity recognition

[1] https://dlab.sit.kmutt.ac.th/index.php/human-posture-datasets/.

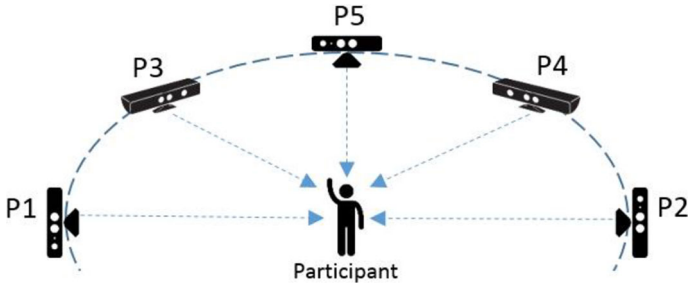


Fig. 6 Arrangement of Kinect sensors in the experiment

from sensors that may be placed randomly near the participant. The objective of the original application which built on the Kinect sensor data is to detect whether a user has been siting in the same posture for too long. The sensing data was used as a monitoring signal for preventing potential spine injury. From the group of Kinect sensor data, we adopted only a part called 'P(5) training' in the performance evaluation experiment. Classification model is built by using only P(5) which is the human activity data collected from one camera angle. Then we test how effective the classification model can recognize sensor data that were collected from the other angles. This arrangement is to simulate real-life situation where monitoring cameras may likely be placed at different positions during application deployment from the camera which was used to register the training dataset from the human participant.

These two data streams, 3D data and Kinect data, pose different computational challenges in the context of human activity recognition. 3D data have attribute-to-activity ratio of 1:2, little attributes for predicting more activities; Kinect data have attribute-to-activity ratio of approximately 2:1 which is the other way round, where more attributes are to predict less activities, relatively.

In particular, as the objective of this paper was to investigate the impacts of improvised methods in data mining, the two datasets offer insights about the effects of the improvised methods on big data streams that are generated from different types of sensors with different attribute-to-activity ratios. Two different types of mining algorithms are tested too, batch mode and incremental mode. In the context of human activity recognition, the research questions we want to ponder on are as follows: (1) How effective are the improvised methods on sensor data streams that have different data dimensions (scaled by the number of data attributes/data descriptors)? (2) How effective are the improvised methods on the data mining modes, batch learning mode, and incremental learning model, with respect to decision tree induction?

Four improvised methods are applied in the experimentation, mainly in testing with the two types of stream datasets by batch data mining and data stream mining. The common aim of applying the improvised methods is to develop an "accurate and fast" classifier. Accurate, in general means good classification performance, which is refined by a number of performance criteria, such as accuracy, kappa, coverage, precision, recall, and F-measure. Accuracy is simply the number of instance that has been correctly classified divided by the total number of instances. Kappa and Coverage are statistically measures of how generalized the classification model is when different

testing datasets are to be used. Precision and recall are often used if there is a large skew in the class distribution, meaning the class that we are interested in is of minority, e.g. in CCTV surveillance security, there are many normal human activities, but rare ones like those of alarming violence. Precision and recall focus on the performance of the models over the rare events, while the accurate predictions of those uninteresting class are ignored. Precision measures the ratios of correctly predicted events by the classification model that are really of interest. Recall measures the ratios of predicting events occurring in the class that are generally learnt by the models, which may include those false alarm signals from other uninteresting classes. A single weighted score is the F-measure which serves as a balance of the two measures.

In the hope of improving the performance, we wanted to apply FS on 3D data and Kinect Data. However, due to the fact that 3D data have only a few attributes this improvised method of FS is not applicable for 3D data but suitable for Kinect data that have a large number of 64 attributes. The two new improvised FS methods, namely Swarm-FS and Accelerated Swarm-FS, are applied only on Kinect data in the experimentation. For the 3D data, outlier removal method and class balancing method are applied, which are suitable to the nature of the 3D data that carries certain outliers and the class distribution is imbalanced. Different datasets are tested with different types of improvised methods where they are applicable for high- or low-dimensional datasets, and lengthy or relatively few data instances.

The experiments are run on an Intel Core i7 CPU at 2.2 GHz, x64-based processor; the benchmarking data mining software programs in use are WEKA for batch mode data mining, and MOA for data stream mining. WEKA[2] is collection of machine learning algorithms, implemented as a popular benchmarking platform for data mining tasks. MOA[3] which stands for Massive Online Analysis is also a similar benchmarking simulation framework for data stream mining. Both WEKA and MOA are written in JAVA programming language as open source, by the University of Waikato, New Zealand.

4.2 Mining 3D sensor data streams

The 3D dataset is characterized mainly by its sheer volume of data but by a relatively few number of attributes; there are only three attribute information, x, y and z. Therefore, improved methods like feature selection may not be applicable. Other methods such as outlier detection/removal and class balancing techniques would be applied. By simply visualizing the 3D sensor data, it can be seen that certain outliers do exist. There are data instances situated far away from the main data clusters, especially the two items of class Working_at_Computer that have exceptionally large y-axis values. Plotting the data distributions on statistical box-plots confirms that a large amount of outliers are present at the y-axis, as shown in Fig. 7. Removing them is the first pre-processing step prior to model construction from these data.

[2] http://www.cs.waikato.ac.nz/ml/weka/.

[3] http://moa.cms.waikato.ac.nz/.

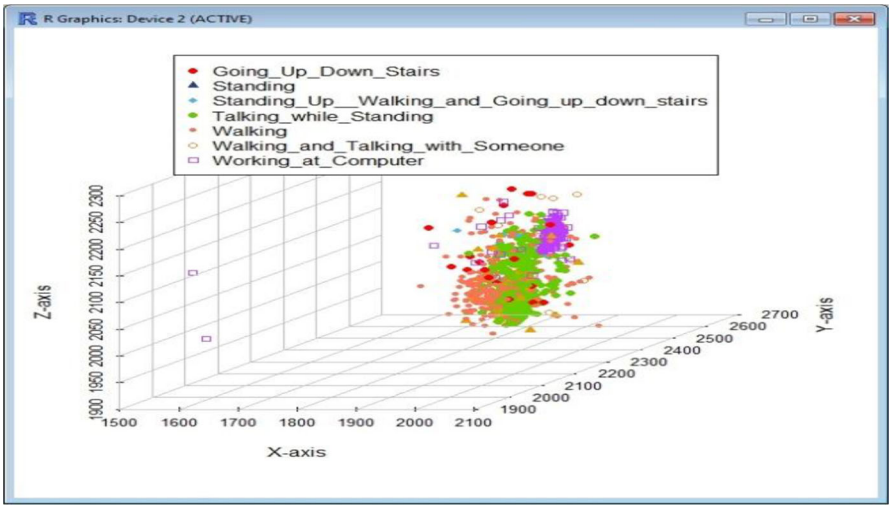


Fig. 7 Visualization of 3D sensor data

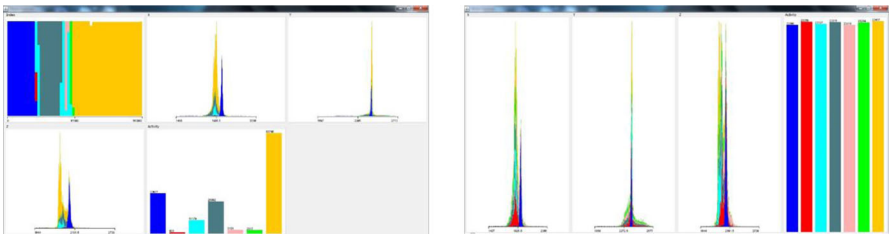


Fig. 8 Visualization of the class distributions; (*left*) imbalanced before SMOTE; (*right*) balanced after SMOTE

The second improvised method applied in the 3D sensor data is the class rebalancing filter, namely SMOTE [17] that stands for synthetic minority over-sampling technique. The filter helps to rebalance the uneven data distributions by artificially inflating the samples from the minority class. However, this function may not guarantee the best performance because the most suitable of parameter values and the exact proportion of the majority and minority data distributions for yielding the maximum performance are unknown in advance. One contemporary method is to use swarm search method to find the best combinations of SMOTE parameter values [18] in lieu of brute-force. Figure 8 below shows the snapshots of visualizations of the class distribution before and after the rebalancing.

By now the dataset has been pre-processed and improved from potential problems of noise and imbalanced class distributions. The dataset is now subject to two decision tree induction algorithms, J48 and VFDT. Three basic performance measures are observed: accuracy in percentage, time taken to construct the models, and the sizes of the resultant decision trees. The results are tabulated in Table 1.

In general, the J48 decision tree has always higher accuracy than the VFDT—(84.8 vs 83.9) before being balanced, and (85.9 vs 84.1) after being balanced. The marginal gain in accuracy by J48 has a high cost of training time spent on building up the

Table 1 Performance results of building decision trees by J48 and VFDT, before and applied SMOTE

	J48 decision tree	Very fast decision tree
Before balanced		
Accuracy (%)	84.7828	83.9286
Time (s)	29.76	2.66
Tree size	3663	47
After balanced		
Accuracy (%)	85.9237	84.0769
Time (s)	47.09	2.77
Tree size	11279	64

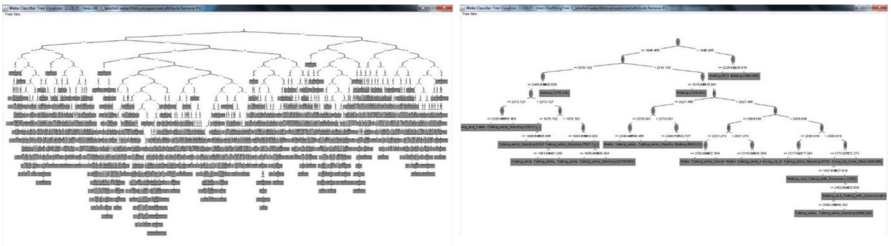


Fig. 9 Decision trees generated by J48 (*left*) and VFDT (*right*)

decision tree in batch training mode. VFDT, on the other hand, is fast in terms of incremental model refresh. It requires only less than 10 % of time in tree induction for VFDT compared to J48. This phenomenon is more apparent after the rebalancing filter is applied; the time cost by J48 soared from 58 % to 47 s. In comparison, VFDT has only a slight increase in time consumption from 2.66 to 2.77 s. This is due to the fact that J48 works by considering the full set of training data; training a tree with a collection of evenly distributed class samples requires a long processing time. Substantial amounts of decision rules are to be learnt from each distribution of class data, and there are now multiple of them in contrast to a singular large data distribution from the majority class beforehand.

The tree sizes have an implication of required memory storage size possibly in the operating system. Tree size is defined by summing up the number of testing nodes and predicted target classes in the decision tree. The larger the tree size the more memory space is required for storing them in real-time, usually in the random-access memory. Figure 9 shows the screen captures of the two generated decision trees by J48 and VFDT. Essentially the J48 is bushy having many tree branches whereby the tree should be able to cover most of the decision cases. Hence a slightly higher classification accuracy can be yielded. The decision tree is quite balanced as it can be seen by its overall shape. On the other hand, VFDT tree has a compact structure that is composed of conditional testing nodes that are significantly contributing to the predictive power of the decision tree. The compact tree size means that VFDT is able to produce decision rules that are compact and suitable for embedded system that has stringent memory size constraint.

4.3 Mining Kinect sensor data streams

Feature selection as an improved method is applied in this case, because Kinect dataset has a relatively large number of attributes or features. The high dimensionality of the dataset give rises to a very large number of feature subsets, $2^{64} = 1.8 \times 10^{19}$ which is almost computationally forbidden if all these combinations would have to be tried sequentially. For this reason, swarm search which does not require searching through the whole search space is employed in coupling with the classifier-wrapped type of feature selection (FS) method.

In this set of experiment, one part of the Kinect data, namely P5 (which is by the camera at the 5th capturing positon), is used for training a decision tree using J48 and VFDT. Then the decision trees are tested with the other testing data generated by the other camera positions from P1 to P4. In this scenario, the decision trees are trained by the postures of the person from one capturing position, and we attempt to put the recognition ability of the decision trees under test by trying to recognize the unseen postures data taken from the other camera positions. As seen in the Table 2 the training accuracy for inducing a decision tree is almost perfect (99.88 %) using J48. Testing the decision tree against the testing posture data collected from the same position, P5, yields an almost perfect accuracy and Kappa that represent the generality of the data, at 99.6 %. The accuracy and Kappa and other performance criteria decline dramatically when the decision tree that was built from posture data of one position is tested against those from the other positions. For instance, when it came to testing the classification models with the human activity data of P4, the classification accuracy rates of VFDT and J48 models dropped to 29.35 % and 29.9 % respectively. This indicates a situation where effective data pre-processing must be done as a remedy. Hence, optimization-based FS such as Swarm-FS and A-Swarm-FS that stand for swarm-search feature selection [12] and accelerated swarm-search feature selection [19], respectively, are applied, in an attempt to select only a subset of useful features for building a more accurate decision tree recognition model. The results of the Swarm-FS models are compared with CFS-FS model. CFS (stands for correlation-based feature selection) is a popular FS scheme which measures the worthiness of the attributes and attribute groups with respect to their correlation to the predicted target class. CFS is, therefore, used as a comparison benchmark here versus the Swarm type of FS [20]. CFS is known to be fast but it often finds a feature subset with minimum feature set size, though it yields good results in most cases.

The recognition results in terms of a spectrum of performance criteria are tabulated in Tables 2, 3, 4, 5, 6, 7, 8 and 9, and the means of the results are charted in bar-charts in Figs. 10 and 11 and in radar-charts in Figs. 12 and 13, respectively, over the two types of decision tree induction algorithms, J48 and VFDT.

It is generally observed that FS, regardless of which kind, has improvement over the performance of both J48 and VFDT indeed. One exception is A-Swarm-FS for J48 where its performance is worse than that of the original data without any FS applied. In contrast, A-Swarm-FS has great improvement for VFDT where the normal Swarm-FS has only slight increase in performance for VFDT. It can be seen that FS of all kinds have consistent but marginal performance improvement in J48, whereas they have large but varying performance gains in VFDT. These observations show

Table 2 Performance measures of J48 model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	99.879	0.121	0.9987	0.0001	0.0086	99.9198	0.999	0	0.999	0.999	0.999	0.999	1	18.99	533	1065
Test with P5testingset	99.6	0.4	0.9959	0.0003	0.0159	99.6543	0.996	0	0.996	0.996	0.996	0.996	0.999	1.73	n/a	n/a
Test with P4testingset	29.9494	70.0506	0.2753	0.0466	0.2143	31.1012	0.299	0.024	0.37	0.299	0.3	0.294	0.654	2	n/a	n/a
Test with P3testingset	34.2852	65.7148	0.3202	0.0438	0.2083	34.7494	0.343	0.023	0.369	0.343	0.337	0.324	0.681	2	n/a	n/a
Test with P2testingset	23.084	76.916	0.2043	0.0512	0.2253	23.9877	0.231	0.027	0.298	0.231	0.226	0.22	0.628	3	n/a	n/a
Test with P1testingset	20.8222	79.1778	0.1809	0.0527	0.2286	21.5914	0.208	0.027	0.235	0.208	0.193	0.181	0.606	3	n/a	n/a
Average	41.54816	58.45184	0.39532	0.03892	0.17848	42.2168	0.4154	0.0202	0.4536	0.4154	0.4104	0.403	0.7136	2.346	n/a	n/a

Table 3 Performance measures of J48 with CFS model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	99.8222	0.1778	0.9982	0.0002	0.0103	99.8901	0.998	0	0.998	0.998	0.998	0.998	1	3.89	636	1271
Test with P5testingset	99.4173	0.5827	0.994	0.0005	0.0192	99.5049	0.994	0	0.994	0.994	0.994	0.994	0.998	1	n/a	n/a
Test with P4testingset	31.1852	68.8148	0.2881	0.0459	0.213	31.2432	0.312	0.024	0.421	0.312	0.319	0.319	0.665	2	n/a	n/a
Test with P3testingset	36.1938	63.8062	0.3399	0.0426	0.2051	36.7284	0.362	0.022	0.428	0.362	0.369	0.361	0.698	2	n/a	n/a
Test with P2testingset	22.6765	77.3235	0.2001	0.0516	0.2255	23.1704	0.227	0.027	0.269	0.227	0.216	0.207	0.632	2	n/a	n/a
Test with P1testingset	19.9148	80.0852	0.1715	0.0534	0.2294	20.6444	0.199	0.028	0.218	0.199	0.185	0.17	0.606	2	n/a	n/a
Average	41.87752	58.12248	0.39872	0.0388	0.17844	42.25826	0.4188	0.0202	0.466	0.4188	0.4166	0.4102	0.7198	1.8	n/a	n/a

Table 4 Performance measures of J48 with Swarm FS model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	99.8815	0.1185	0.9988	0.0001	0.0085	99.9111	0.999	0	0.999	0.999	0.999	0.999	1	10.46	507	1013
Test with P5testingset	99.5753	0.4247	0.9956	0.0004	0.0167	99.5914	0.996	0	0.996	0.996	0.996	0.996	0.999	2	n/a	n/a
Test with P4testingset	33.4086	66.5914	0.3111	0.0444	0.2102	33.4333	0.334	0.023	0.413	0.334	0.337	0.333	0.67	2	n/a	n/a
Test with P3testingset	31.9247	68.0753	0.2958	0.0455	0.2126	32.0321	0.319	0.023	0.399	0.319	0.323	0.317	0.657	2	n/a	n/a
Test with P2testingset	24.1938	75.8062	0.2158	0.0505	0.2238	24.6938	0.242	0.026	0.278	0.242	0.229	0.22	0.628	3	n/a	n/a
Test with P1testingset	22.7395	77.2605	0.2008	0.0516	0.2264	22.9926	0.227	0.027	0.251	0.227	0.216	0.203	0.612	3	n/a	n/a
Average	42.36838	57.63162	0.40382	0.03848	0.17794	42.54864	0.4236	0.0198	0.4674	0.4236	0.4202	0.4138	0.7132	2.4	n/a	n/a

Table 5 Performance measures of J48 with accelerated Swarm FS model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	99.8605	0.1395	0.9986	0.0002	0.0092	99.9062	0.999	0	0.999	0.999	0.999	0.999	1	8.5	499	997
Test with P5testingset	99.6074	0.3926	0.9959	0.0003	0.0158	99.658	0.996	0	0.996	0.996	0.996	0.996	0.999	2	n/a	n/a
Test with P4testingset	29.2444	70.7556	0.268	0.0471	0.2165	29.7494	0.292	0.024	0.365	0.292	0.295	0.289	0.662	3	n/a	n/a
Test with P3testingset	34.6642	65.3358	0.3241	0.0435	0.2075	35.221	0.347	0.023	0.383	0.347	0.343	0.332	0.686	3	n/a	n/a
Test with P2testingset	22.9691	77.0309	0.2031	0.0513	0.2251	23.7654	0.23	0.027	0.273	0.23	0.219	0.21	0.63	2	n/a	n/a
Test with P1testingset	19.8173	80.1827	0.1705	0.0535	0.2299	20.2802	0.198	0.028	0.23	0.198	0.19	0.175	0.601	2	n/a	n/a
Average	41.26048	58.73952	0.39232	0.03914	0.17896	41.7348	0.4126	0.0204	0.4494	0.4126	0.4086	0.4004	0.7156	2.4	n/a	n/a

Table 6 Performance measures of VFDT model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	71.6875	28.3125	0.707	0.019	0.1354	73.425	0.717	0.01	0.779	0.717	0.705	0.71	0.904	0.17	n/a	n/a
Test with P5testingset	70.9125	29.0875	0.699	0.0195	0.1373	72.6125	0.709	0.01	0.771	0.709	0.697	0.702	0.9	13	n/a	n/a
Test with P4testingset	29.35	70.65	0.2689	0.0472	0.2158	29.975	0.294	0.025	0.334	0.294	0.25	0.253	0.712	13	n/a	n/a
Test with P3testingset	30.5875	69.4125	0.2817	0.0462	0.2139	31.475	0.306	0.024	0.345	0.306	0.268	0.27	0.725	13	n/a	n/a
Test with P2testingset	22.6375	77.3625	0.1994	0.0515	0.2261	23.2875	0.226	0.027	0.26	0.226	0.19	0.188	0.674	13	n/a	n/a
Test with P1testingset	19.025	80.975	0.162	0.054	0.2316	19.5875	0.19	0.028	0.2	0.19	0.161	0.151	0.651	13	n/a	n/a
Average	34.5025	65.4975	0.3222	0.04368	0.20494	35.3875	0.345	0.0228	0.382	0.345	0.3132	0.3128	0.7324	13	n/a	n/a

Table 7 Performance measures of VFDT with CFS model over Kinect sensor data

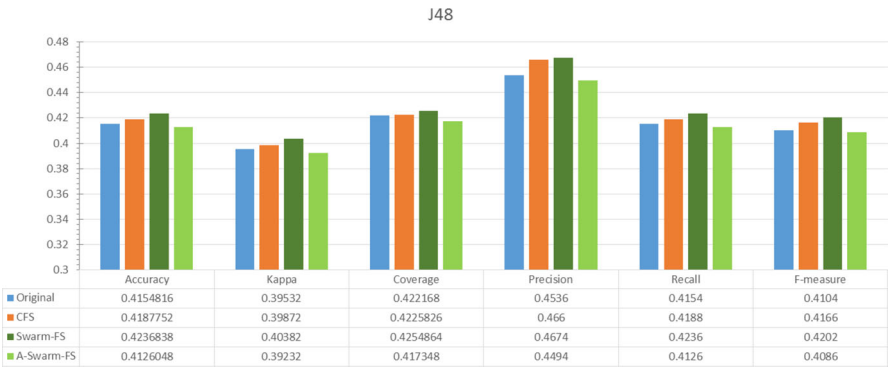
	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	71.575	28.425	0.7058	0.0201	0.131	77.775	0.716	0.01	0.732	0.716	0.699	0.701	0.935	0.03	n/a	n/a
Test with P5testingset	70.875	29.125	0.6986	0.0205	0.1326	76.9375	0.709	0.01	0.725	0.709	0.692	0.694	0.933	2	n/a	n/a
Test with P4testingset	30.9	69.1	0.2847	0.0465	0.2087	36.35	0.309	0.024	0.296	0.309	0.28	0.27	0.75	2	n/a	n/a
Test with P3testingset	38.1375	61.8625	0.3597	0.0417	0.1954	47.55	0.381	0.022	0.397	0.381	0.361	0.355	0.824	2	n/a	n/a
Test with P2testingset	30.1	69.9	0.2764	0.047	0.209	36.3625	0.301	0.025	0.306	0.301	0.268	0.263	0.77	2	n/a	n/a
Test with P1testingset	22.5	77.5	0.1976	0.0521	0.2217	26.2375	0.225	0.028	0.227	0.225	0.203	0.19	0.699	2	n/a	n/a
Average	38.5025	61.4975	0.3634	0.04156	0.19348	44.6875	0.385	0.0218	0.3902	0.385	0.3608	0.3544	0.7952	2	n/a	n/a

Table 8 Performance measures of VFDT with Swarm FS model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	69.6375	30.3625	0.6858	0.0204	0.1386	72.6375	0.696	0.01	0.741	0.696	0.687	0.689	0.92	0.26	n/a	n/a
Test with P5testingset	68.8625	31.1375	0.6778	0.021	0.1405	71.7375	0.689	0.011	0.735	0.689	0.68	0.681	0.917	9	n/a	n/a
Test with P4testingset	25.225	74.775	0.2264	0.0499	0.2217	26.3	0.252	0.026	0.251	0.252	0.221	0.214	0.694	9	n/a	n/a
Test with P3testingset	35.0125	64.9875	0.3277	0.0434	0.2061	37.0375	0.35	0.022	0.354	0.35	0.316	0.313	0.754	9	n/a	n/a
Test with P2testingset	25.4	74.6	0.2282	0.0497	0.221	27.2	0.254	0.026	0.272	0.254	0.222	0.22	0.718	9	n/a	n/a
Test with P1testingset	21.55	78.45	0.1885	0.0523	0.2262	23.7	0.216	0.027	0.209	0.216	0.185	0.174	0.673	9	n/a	n/a
Average	35.21	64.79	0.32972	0.04326	0.2031	37.195	0.3522	0.0224	0.3642	0.3522	0.3248	0.3204	0.7512	9	n/a	n/a

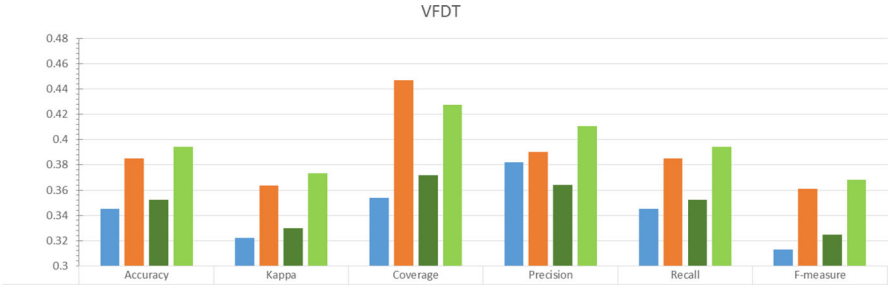
Table 9 Performance measures of VFDT with Accelerated Swarm FS model over Kinect sensor data

	Accuracy %	Error %	Kappa	MAE	RMSE	Coverage %	TP	FP	Precision	Recall	F-measure	MCC	ROC	Time (s)	#leaves	Tree size
Train with P5trainingset	74.6	25.4	0.7372	0.0171	0.1251	78.3625	0.746	0.009	0.766	0.746	0.736	0.733	0.954	0.08	n/a	n/a
Test with P5testingset	73.8	26.2	0.7289	0.0176	0.1271	77.5625	0.738	0.009	0.758	0.738	0.729	0.725	0.951	5	n/a	n/a
Test with P4testingset	33.3375	66.6625	0.3104	0.0444	0.207	36.8625	0.333	0.023	0.348	0.333	0.303	0.301	0.758	6	n/a	n/a
Test with P3testingset	39.75	60.25	0.3767	0.0401	0.1967	43.6625	0.398	0.021	0.408	0.398	0.368	0.366	0.815	6	n/a	n/a
Test with P2testingset	27.9875	72.0125	0.2549	0.0479	0.2161	31.025	0.28	0.025	0.287	0.28	0.243	0.241	0.752	6	n/a	n/a
Test with P1testingset	22.3125	77.6875	0.1962	0.0517	0.2246	24.55	0.223	0.027	0.253	0.223	0.199	0.192	0.696	6	n/a	n/a
Average	39.4375	60.5625	0.37342	0.04034	0.1943	42.7325	0.3944	0.021	0.4108	0.3944	0.3684	0.365	0.7944	5.8	n/a	n/a



	Accuracy	Kappa	Coverage	Precision	Recall	F-measure
Original	0.4154816	0.39532	0.422168	0.4536	0.4154	0.4104
CFS	0.4187752	0.39872	0.4225826	0.466	0.4188	0.4166
Swarm-FS	0.4236838	0.40382	0.4254864	0.4674	0.4236	0.4202
A-Swarm-FS	0.4126048	0.39232	0.417348	0.4494	0.4126	0.4086

Fig. 10 Bar chart of performance comparison of different J48 models over Kinect sensor data



	Accuracy	Kappa	Coverage	Precision	Recall	F-measure
Original	0.345025	0.3222	0.353875	0.382	0.345	0.3132
CFS	0.385025	0.3634	0.446875	0.3902	0.385	0.3608
Swarm-FS	0.3521	0.32972	0.37195	0.3642	0.3522	0.3248
A-Swarm-FS	0.394375	0.37342	0.427325	0.4108	0.3944	0.3684

Fig. 11 Bar chart of performance comparison of different VFDT models over Kinect sensor data

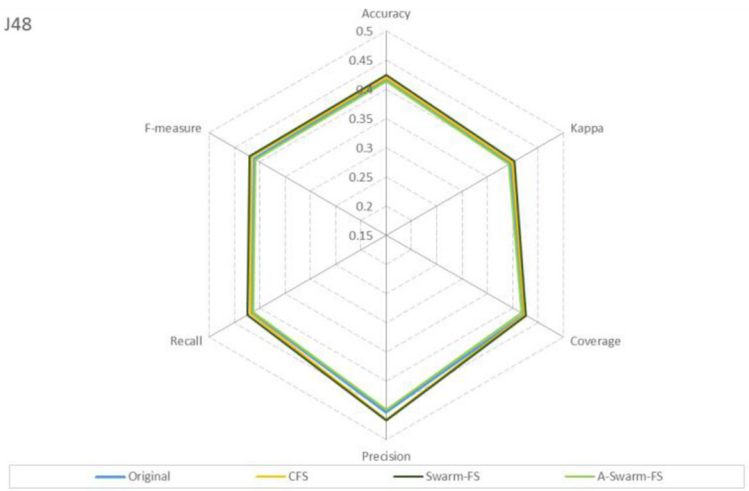


Fig. 12 Radar chart of performance comparison of different J48 models over Kinect sensor data

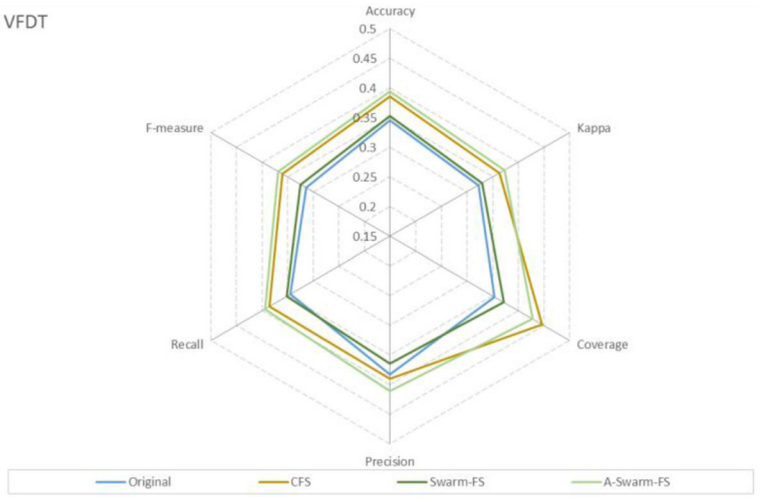


Fig. 13 Radar chart of performance comparison of different VFDT models over Kinect sensor data

an important implication—Swarm-FS works best with J48 and A-Swarm-PS works best with VFDT instead. This phenomenon is largely due to the natures of the mining algorithms; Swarm-FS in batch learning mode is able to find the best subset when all the data are available at the same time. This, however, is not true when it comes to data stream mining in incremental learning where the data are only partially available. In fact, the overall decision tree performance in VFDT is down-rated by approximately 20 % when compared to the full data training in J48. In data stream mining model, when the training/testing datasets possess a large number of attributes, full swarm optimization over finding the ideal subset has an edge in performance. The full swarm optimization loses its appeal when the training goes incremental. Even simple FS like CFS which is based on the correlation measures outperforms Swarm-FS. The search space for finding the best feature subset in VFDT is incomplete as only a segment of data is available during the training process, thereby limiting the efficacy of the Swarm-FS. Swarm-FS was designed to work with a complete search space assuming all the full set of training data is available.

However, A-Swarm-FS is capable in yielding the best overall performance in data stream mining mode for VFDT. The merit of A-Swarm-FS comes from the initialization process where the starting positions of the metaheuristic search are not taken in random. Rather, the starting positions of A-Swarm-FS are inherited by the heuristics from the best found positions from the last generation of optimization cycles. The retained starting positions contribute to speed-up in the subsequent swarm search of ideal feature subset (therefore, the name accelerated swarm), as well as making the FS adaptive to the incoming of new training samples by swarm search.

Overall, as revealed from the radar charts, FS is certainly effective for both J48 and VFDT. However, the effects of FS are more prominent in VFDT than in J48, because the incremental learning by VFDT is dynamic, finding the appropriate features is important to VFDT. In contrast J48 manages to build decision paths from the full

dataset according to the information gain principle; it is able to grow a good quality decision tree in all test cases of data even without any FS applied.

4.4 Pattern analysis over the sensor data

It is intellectually interesting to try find out the cause of the accuracy declines along the data stream. The accuracy curve of training with the P5 dataset using VFDT algorithm is plotted in Fig. 14. It was empowered by a data stream mining software benchmarking platform, called Massive Online Analysis. The accuracy curve is a real-time information curve produced by the VFDT model incrementally learnt from the data stream feed. As it can be seen clearly from the curve, there are many sudden drops in accuracy, indicating that the VFDT model failed many times to recognize the newly arrived data. Initially, the classifier is able to achieve a perfect 100 % accuracy; soon it dips to close to zero when new data comes. Then the model picks up the learning again, refreshes the decision tree embracing the new data, and the accuracy bounces back to high. These drastic fluctuations are largely due to the emerging new concepts introduced by the new arriving data. The classifier that learnt the existing concept from the old data is unable to cope with accurate prediction.

To further verify this situation ,the Kinect dataset, P5, is plotted as a time-series in Figs. 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24 for visual inspection. The time-series charts show a subset of patterns from the 3-axis information of some selected

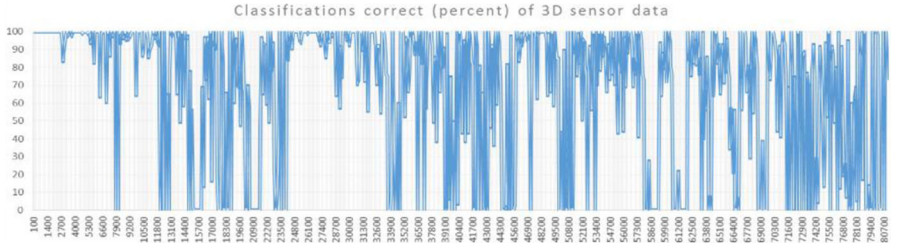


Fig. 14 Accuracy curve of Kinect data produced by MOA using VFDT algorithm

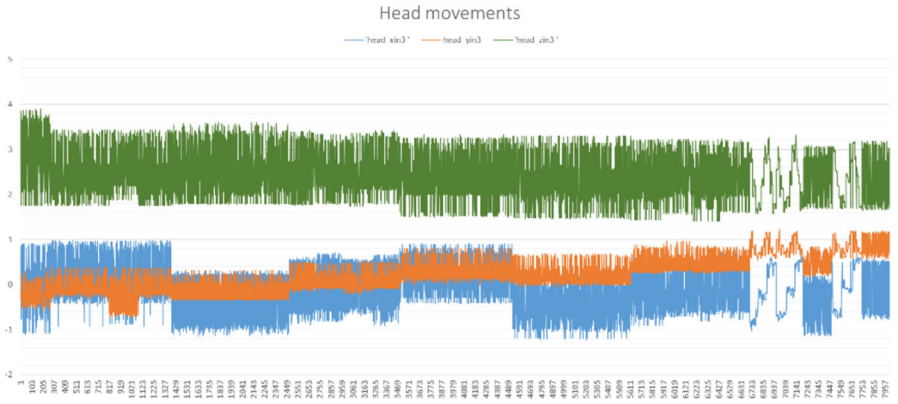


Fig. 15 Time series plot of the Kinect data, the head only

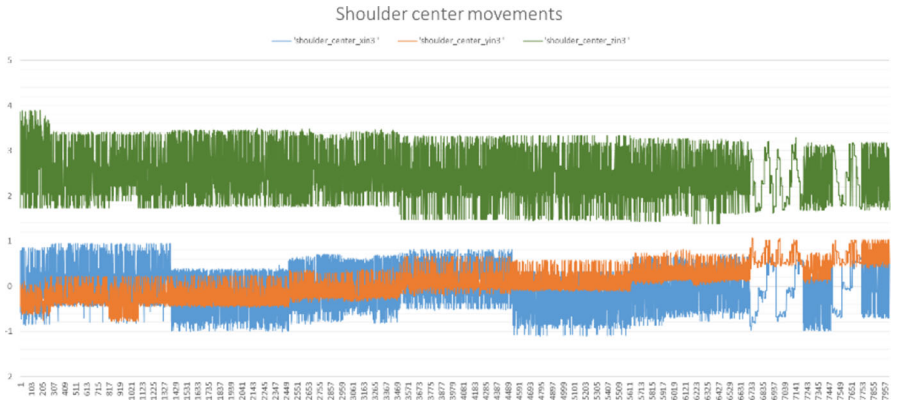


Fig. 16 Time series plot of the Kinect data, the center shoulder only

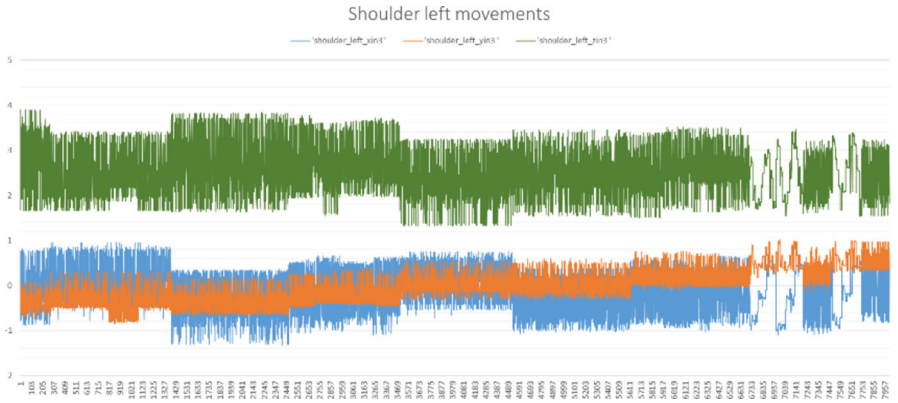


Fig. 17 Time series plot of the Kinect data, the left shoulder only

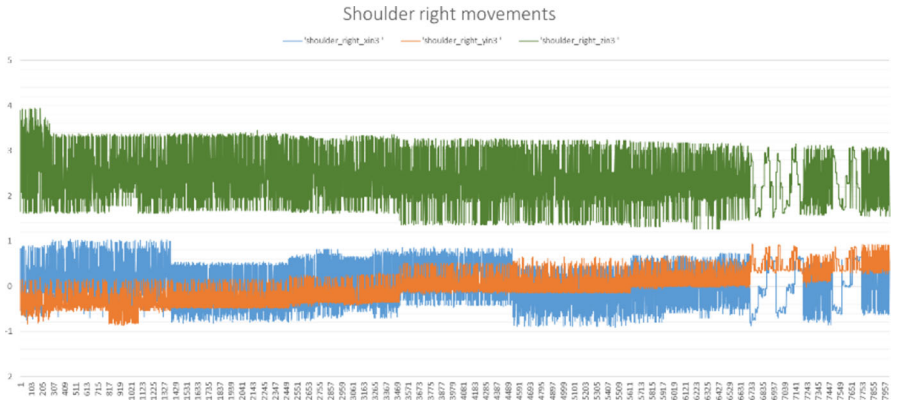


Fig. 18 Time series plot of the Kinect data, the shoulder right only

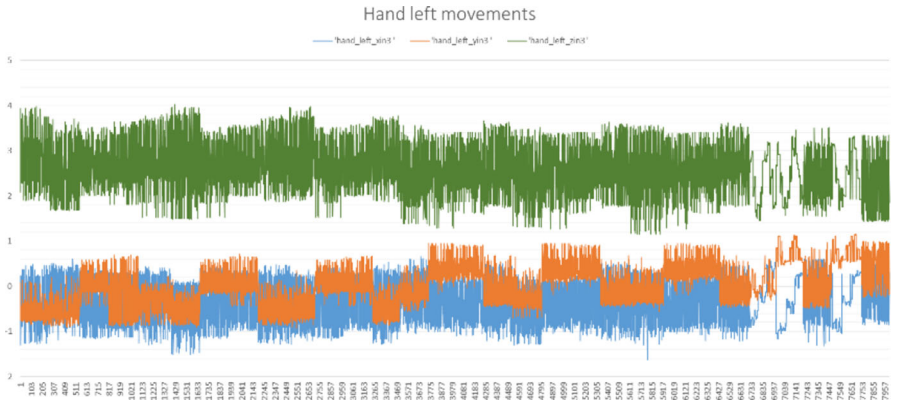


Fig. 19 Time series plot of the Kinect data, the left hand only

Fig. 20 Time series plot of the Kinect data, the right hand only

body parts, head, shoulder, hand, knee, and foot. In the time series, the patterns were kind of fragmented intermittently along the whole data sequence. That shows certain activities have been changed at different times as reflected by the fragmented patterns. Nevertheless, the purpose of showing the accuracy curve of the incremental classifier by data stream mining and just the data series is to compare and contrast vis-à-vis the two patterns. It is, however, observed that the accuracy dips and rises somehow correspond to the changes of concepts (target classes) which are the activity labels in the data sequence. In other words, when the subject is switching the activity from one to another, new concept needs to be learnt, and that is reflected by a sudden decline of the accuracy curve. In particular, the data values of the hand movements, as shown in Figs. 19 and 20, fluctuate very often when compared to those of other body parts. Relatively, the other body parts do not fluctuate so much, and their data form almost flat lines especially for the head foot. It is deduced that performing the activities do not require much movements from the head and feet but involve many hand movements.

The phenomenon of relation between activity changes and accuracy drops is further validated using the 3D sensor data which is simpler in attributes but longer in sequence.

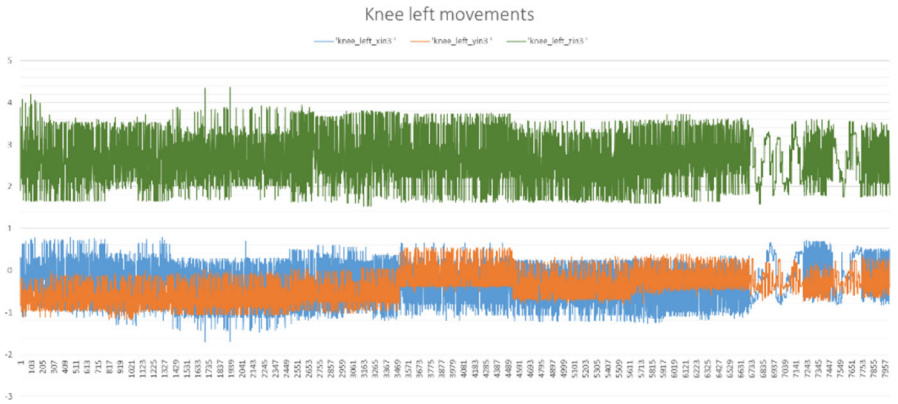


Fig. 21 Time series plot of the Kinect data, the left knee only

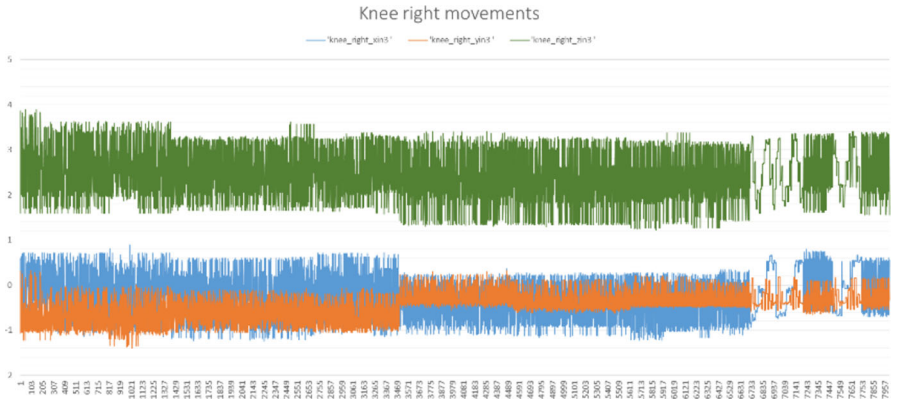


Fig. 22 Time series plot of the Kinect data, the right knee only

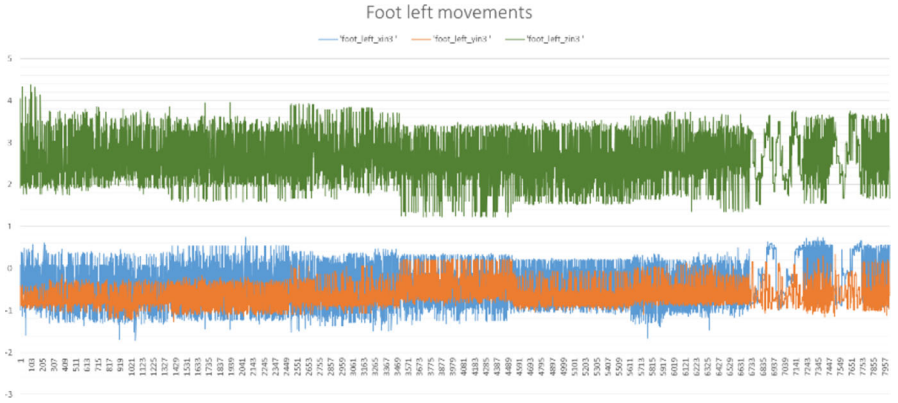


Fig. 23 Time series plot of the Kinect data, the left foot only

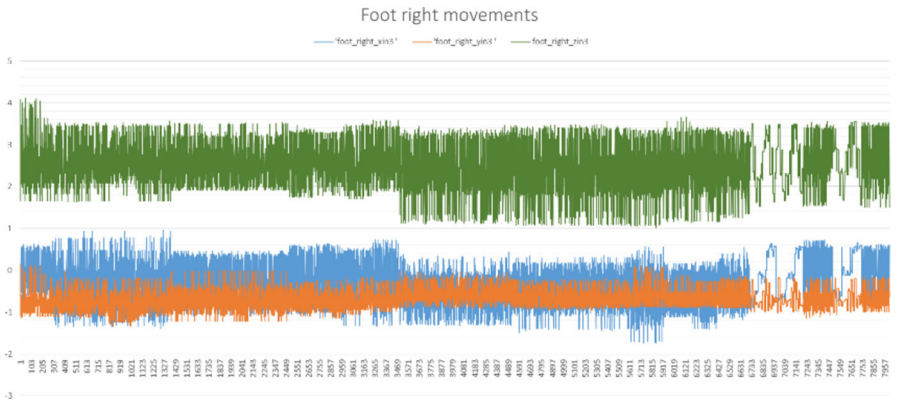


Fig. 24 Time series plot of the Kinect data, the right foot only

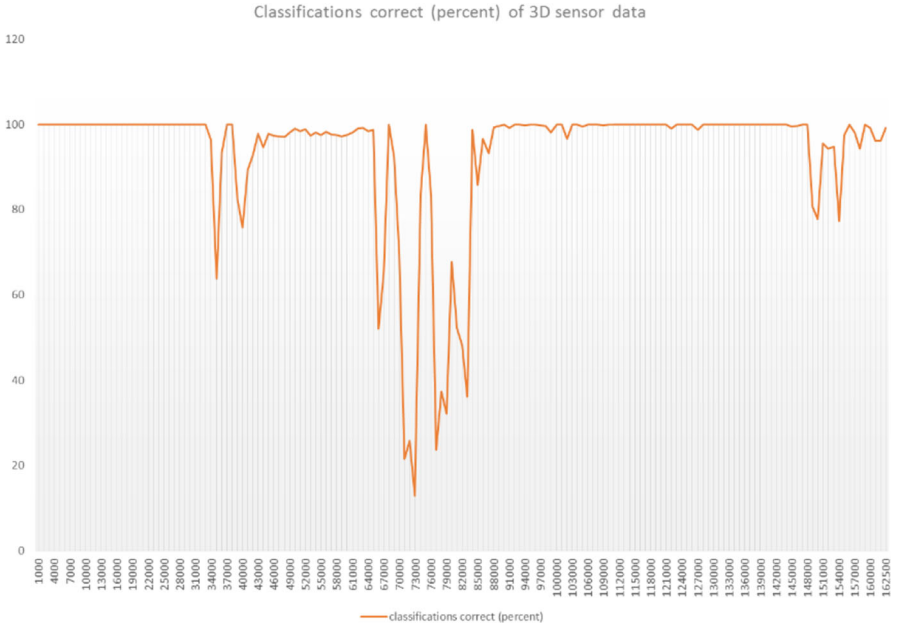


Fig. 25 Accuracy curve of 3D data produced by MOA using VFDT algorithm

The accuracy curve by data stream mining the 3D sensor data is shown in Fig. 25. The 3D sensor data series are plotted in Fig. 26. One can notice easily that the valleys of the accuracy curve correspond right to the changes of activities in the 3D sensor data sequence. During the data collection process, the volunteer performed a series of different actions at different stages of time. The different actions were captured and correspondingly reflected on the sequential pattern of the sensor data. The two cases of 3D sensor and Kinect sensor data confirm concordantly that the accuracy in data stream mining is largely related to the changes of activities in the context of human activity recognition.

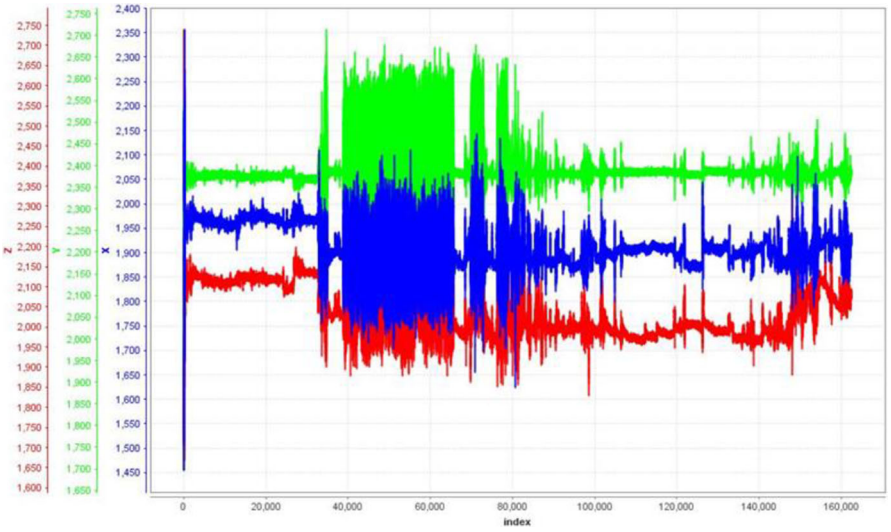


Fig. 26 Time series plot of the 3D sensor data

5 Conclusion

In this is paper a holistic methodology of data stream mining, called SHARP, is reviewed. Following the proposed methodology, data stream mining experiments are conducted with a case of human activity recognition. Two types of data streams with one type representing a long but low-dimensional data stream, and the other type representing a high-dimensional data streams are put under test using a collection of improvised methods discussed in this paper. The improvised methods are classical data mining methods; however, they might not have been tested extensively in data stream mining. Therefore, the objective of this paper was to shed some light on applying improvised methods on data stream mining; these methods range from outlier detection, class rebalancing to feature selection. Two contemporary types of feature selection methods based on swarm search have been applied in data stream mining in the context of human activity recognition. The swarm search feature selection methods are both shown to improve performance of data stream mining. This finding is important because a major challenge in big data stream mining is lack of suitable tools in reducing the dimensionality of the data streams in quick and effective ways, whereas brute-force is out of the question given the sheer volume of data stream. The experimentation results indicate that the improvised methods indeed have advantages in improving the VFDT decision tree performance. Comparing to traditional decision tree by batch learning, improvised methods, especially the swarm type of feature selections, have significant enhancement in performance for data stream mining. In other words, data stream mining may need such feature selection more desperately than traditional data stream mining. This work has proven the concept of SHARP methodology as a success example. It is a step towards the progress of implementing a scalable and efficient data stream mining framework. Further works include detecting

concept drifts in real-time, as well as migrating the codes from CPU to GPU that stands for graphics processing unit (or other distributed computing environment) for parallelizing the executions of several optimization components that involve stochastic operations. GPU computing helps to speed up the extra computational loads incurred by the improvised methods in pre-processing in real-time. As future work, the priority is on quickly finding suitable feature subset out of high-dimensional sensor data. The feature selection process is to be empowered by GPU programming which boosts parallel execution. Finding optimal feature subset by swarm intelligence is known to be a NP-hard combinatorial problem that needs to be solved before SHARP methodology can reach its full efficacy.

Acknowledgments The authors are thankful for the financial support from the research grant "Temporal Data Stream Mining by Using Incrementally Optimized Very Fast Decision Forest (iOVFDF)", Grant No. MYRG2015-00128-FST, offered by the University of Macau, FST, and RDAO.

References

1. Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann Publishers, San Francisco
2. Pai P-F, Chen T-C (2009) Rough set theory with discriminant analysis in analyzing electricity loads. Expert Syst Appl 36:8799–8806
3. Gaber MM, Zaslavsky A, Krishnaswamy S (2005) Mining data streams: a review. ACM SIGMOD Rec 34(2):18–26
4. Fan W, Bifet A (2005) Mining big data: current status, and forecast to the future. SIGKDD Explor 14(2):1–5
5. Murdopo A (2013) Distributed decision tree learning for mining big data streams. Master of Science Thesis. European Master in Distributed Computing
6. Fong S, Zhuang Y, Wong R, Mohammed S (2014) A Scalable data stream mining methodology: stream-based holistic analytics and reasoning in parallel. In: Proceedings of the 2nd International symposium on computational and business intelligence, New Delhi, 7–8 Dec 2014, pp 110–115
7. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 99:1601–1604
8. Perkins S, Lacker K, Theiler J (2003) Grafting: fast, incremental feature selection by gradient descent in function space. J Mach Learn Res 3:1333–1356
9. Shu W, Shen H (2014) Incremental feature selection based on rough set in dynamic incomplete data. Pattern Recognit 47(12):3890–3906
10. Katakis I, Tsoumakas G, Vlahavas I (2005) On the utility of incremental feature selection for the classification of textual data streams. In: PCI 2005, LNCS 3746. Springer, pp 338–348
11. Fong S, Liang J, Wong R, Ghanavati M (2014) A novel feature selection by clustering coefficients of variations. In: Proceedings of the 9th International conference on digital information management (ICDIM), Phitsanulok, 29 Sept–1 Oct 2014, pp 205–213
12. Fong S, Deb S, Yang X-S, Li J (2014) Feature selection in life science classification: metaheuristic swarm search. IT Prof 16(4):24–29
13. Brest J, Boskovic B, Zamuda A, Fister I, Mezura-Montes E (2013) Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Cancun, 20–23 June 2013, pp 377–383
14. Ryoo MS, Aggarwal JK (2011) Stochastic representation and recognition of high-level group activities. Int J Comput Vis (IJCV) 93(2):183–200
15. Fatima I, Fahim M, Lee YK, Lee S (2013) Analysis and effects of smart home dataset characteristics for daily life activity recognition. J Supercomput 66(2):760–780
16. Edwards Chris (2014) Decoding the language of human movement. Commun ACM 57(12):12–14
17. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res Arch 16(1):321–357

18. Li J, Fong S, Mohammed S, Fiaidhi J (2015) Improving the classification performance of biological imbalanced datasets by swarm optimization algorithms. J Supercomput, Springer, pp 1–21
19. Fong S, Wong R, Vasilakos A (2015) Accelerated PSO swarm search feature selection for data stream mining big data. IEEE Trans Serv Comput 99:1–12. doi:10.1109/TSC.2015.2439695
20. Fong S, Zhuang Y, Tang R, Yang X-S, Deb S (2013) Selecting optimal feature set in high-dimensional data by swarm search. J Appl Math 2013:18. doi:10.1155/2013/590614 (**Article ID** 590614)