Hindawi Publishing Corporation
Journal of Sensors
Volume 2015, Article ID 205707, 16 pages
http://dx.doi.org/10.1155/2015/205707

Research Article

Gesture Recognition from Data Streams of Human Motion Sensor Using Accelerated PSO Swarm Search Feature Selection Algorithm

Simon Fong,[1] Justin Liang,[1] Iztok Fister Jr.,[2] Iztok Fister,[2] and Sabah Mohammed[3]

[1]*Department of Computer and Information Science, University of Macau, Macau*
[2]*Faculty of Electrical Engineering and Computer Science, University of Maribor, 2000 Maribor, Slovenia*
[3]*Department of Computer Science, Lakehead University, 955 Oliver Road, Thunder Bay, ON, Canada P7B 5E1*

Correspondence should be addressed to Iztok Fister; iztok.fister1@um.si

Received 17 November 2014; Accepted 7 March 2015

Academic Editor: Jian-Nong Cao

Copyright © 2015 Simon Fong et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Human motion sensing technology gains tremendous popularity nowadays with practical applications such as video surveillance for security, hand signing, and smart-home and gaming. These applications capture human motions in real-time from video sensors, the data patterns are nonstationary and ever changing. While the hardware technology of such motion sensing devices as well as their data collection process become relatively mature, the computational challenge lies in the real-time analysis of these live feeds. In this paper we argue that traditional data mining methods run short of accurately analyzing the human activity patterns from the sensor data stream. The shortcoming is due to the algorithmic design which is not adaptive to the dynamic changes in the dynamic gesture motions. The successor of these algorithms which is known as data stream mining is evaluated versus traditional data mining, through a case of gesture recognition over motion data by using Microsoft Kinect sensors. Three different subjects were asked to read three comic strips and to tell the stories in front of the sensor. The data stream contains coordinates of articulation points and various positions of the parts of the human body corresponding to the actions that the user performs. In particular, a novel technique of feature selection using swarm search and accelerated PSO is proposed for enabling fast preprocessing for inducing an improved classification model in real-time. Superior result is shown in the experiment that runs on this empirical data stream. The contribution of this paper is on a comparative study between using traditional and data stream mining algorithms and incorporation of the novel improved feature selection technique with a scenario where different gesture patterns are to be recognized from streaming sensor data.

1. Introduction

With the advance in the sensing technology [1] that is relatively easy-to-deploy and cost-effective in operation, video motion sensor finds its applications popularly across different domains, just to name a few successful cases in environmental sensing, useful applications such as hand motion gesture detection by smartphones for remote interaction [2], monitoring human movements for medical rehabilitation [3], daily activities in elderly homes [4], entertainments, and sports [5], as well as advanced computer-human interaction research [6, 7]. While the communication mechanisms and the general operation of the sensor applications have been intensively studied, the decision making of such sensor system which is often known as the analytical "brain" has not yet been explored in details. Kinect sensor, for example, [8], is a peripheral I/O device designed to provide a natural interface to gaming consoles without the need of conventional controllers. This sensing device features a simple color camera, depth sensor, and multiarray microphone, capable of delivering swift audio and visual data streams for supporting facial recognition, partial or full-body motion recognition, and acoustical detection recognition.

In general, human motion recognition is the process of first detecting and recording changes in position of a human posture or gesture (that depends on the context of a full

body or hands only), relative to its surroundings or backdrops that are corresponding to the previous positions in the video sequence. The collected data from sensors include motions of infrared signals, optical visions, radio frequency energy, or ultrasounds, depending on the types of motion sensors being used. Assuming that the deployed sensors are the eyes and ears that continuously and reliably collect perceived data from a moving subject, the data quickly stream to a preliminary image-processing that extracts the features from the images and then to a central processor that functions as a decision-support for the applications by interpreting the changing information of the features.

As a generalized process applicable to many applications, the remaining tasks after the data are collected and delivered to the decision-support center consist of the following: image processing, data preprocessing, model induction, rule extraction, and intelligence dissemination. In modern sensor hardware, the image processing is usually embedded as some low-level middleware in the device which processes and extracts information from raw images into characteristic features in abstract level.

Data preprocessing involves data transformation, data cleaning, and often feature selection for reducing the feature space for enhanced recognition accuracy; our previous papers have addressed in length this task for distributed wireless sensor networks [9, 10]. We focus in this paper on the tasks of model induction and rule extraction, specifically by using data streaming algorithms and lightweight feature selection scheme suitable for high-speed incremental machine-learning, for gesture recognition.

This paper investigates the remaining tasks after the data are collected and delivered to the decision-support center with the aim of finding the right combination of classification algorithms and feature selection algorithms for accurately recognizing human gestures on the fly. The sensing data concerned in this paper are the data that are collected from Microsoft Kinect sensor, which are used to capture the gesture, in terms of the body positions in 3D and their corresponding velocities and acceleration while the user poses in various postures. Our focus is the rigorous and comparative performance analysis over two groups of classification algorithms, namely, batch-learning and incremental or data-stream learning, in the light of achieving top accuracy at the shortest possible preprocessing times. We illustrate the efficacy of a newly proposed feature selection method around an illustrative example, that is, how to recognize human gesture pattern in an application of video sensors.

The main research challenge here is about finding the most appropriate model induction algorithm for gesture pattern recognition. The challenge rests on several stringent requirements in video motion sensor: first of all, the amount of data feed is potentially infinite, and the data delivery is continuous like a high-speed train of information. The processing hence is expected to be real-time and instantly responsive. This implies that the classification induction algorithm being deployed must be lightweight, incremental, and accurate for sure. The model update needs to be done quickly and on the fly upon each arrival of the new instance of data. As an additional feature, pertaining to the suitability for video sensor where the decision support process/operation may have to be embedded in a small mobile device, the memory requirement is opt to be as little as possible for obvious reasons of energy saving and fitting into a tiny device size. In other words, the learned model, probably in form of generalized nonlinear mappings between the values of the features to the predicted target classes, must be compact enough to be executed in a small run-time memory. No room is wasted for storing the features and their relations that neither are significant nor contribute little to the model accuracy. To this end, without using feature selection is out of the question. That is because the number of original features extracted from the video sequences could be very high. Feature selection is a heuristic process which retains only the significant features as an optimal subset of the full features, representative enough to induce an accurate classification model for pattern recognition.

Another complication on top of quantitatively computing the nonlinear relations between the feature values and the target classes is the temporal nature of such sensor data stream. One must crunch on the data stream long enough for modeling seasonal cycles or regular patterns if they ever exist. There are no straightforward relations that can easily map the attribute data into a specific class without a long-term observation. This impacts considerately on the data mining algorithm design that should be capable of just reading and forgetting the data stream (so is called "one-pass" algorithm), retaining nothing but just the required statistics for reasoning the long-term relations among the attributes values and the target classes.

Taken into account the aforementioned unique computational challenges associated with the motion data from video sensor, a rigorous analytical evaluation is both crucial and necessary in comparing some popular data mining algorithms, for gesture pattern recognition. This evaluation as reported in this paper offers insights to developers who want to design a video sensor network for the purpose of recognizing human activities or gestures through data stream mining.

The remaining of the paper is organized as follows. Section 2 introduces the background of our research via a discussion in both aspects of the experimental layout and the types of data mining algorithms to be tested with. In particular, the video sensor for measuring motion data as a result of human activities is described, and traditional and incremental decision trees are compared and contrasted. Section 3 covers the technical details of the data stream mining algorithms. Specifically the shortcoming of the traditional algorithms is discussed, as well as how the new functions of data stream mining algorithms that help overcome the limitation are narrated. An empirical video sensor dataset is applied in the experiment, in Section 4, with the aim of comparing several data mining algorithms vis-à-vis with respect to gesture recognition. Lastly Section 5 concludes the paper.

2. Background

2.1. Gesture Recognition and Dataset. In 2013, researchers Madeo et al. from University of Sao Paulo studied a gesture segmentation problem using support vector machine [11] and

reviewed temporal aspect of hand gesture analysis [12] and how gestures which are captured by video sensor can be recognized by incorporating the temporal aspects with references to the bodily positions [13]. Although the focus of their research is on gesture phase segmentation the research team pointed out that the gesticulation behavior may influence the performance of a classifier. It was known that different human users who were videoed doing the same gesture may yield different gesticulation behavior. Therefore an effective machine learning approach is needed in accurately classifying the motion pattern data into their corresponding gestures.

Their experimental dataset which is available for download is composed by 7 video sequences captured by using Microsoft Kinect sensor. In front of the Microsoft Kinect sensor, 3 human subjects were instructed to read 3 comic strips and to tell the stories from the comic strips using hand gestures and bodily postures, while the sensor is recording the motions. At the end of the image processing, the video contains a sequence of images, one for each frame, indexed by a timestamp. The video sequence is then formatted into a matrix of text file, with rows of data representing the temporal data instances and columns that characterize the spatial positions or X-Y-Z 3D coordinates of 6 articulation points. The articulation points are identified by the current positions of the limbs and body parts, such as the head, spine, left hand, left wrist, right hand, and right wrist. The measures of the positions were normalized into numeric values. Each data instance is the gesture information extracted per video frame, with each of which uniquely identified by a timestamp. The data instances are postprocessed by the help of a human specialist who manually segmented the file and associated it with a label of gesture. This manual postprocessing was needed for generating a ground truth for classification for the sake of evaluating the performance of the classification algorithms under test.

There are 32 attributes or features in the dataset, which are combined from the static positions of the body parts as in the video frame and the motion information. The target classes are the five phases of the gestures which are individually abstracted as Hold, Preparation, Rest, Retraction, and Stroke. A total of 50 attributes are used to characterize each instance that amounts to thousands depending on the length of the stories to be played. Out of the 50 attributes, 18 are the positions of the body parts, and 32 are the velocity and acceleration, in both vectorial and scalar forms of hands and wrists. Table 1 shows the 50 attributes. There are a total of 9,900 data instances which are extracted from the 7 videos available for training/testing the classification algorithms.

As an illustration, the three prime positions of a left hand are visualized for showing the fluctuation in values over time in Figure 1. Furthermore the normalized values of the three coordinates are visualized as parallel coordinate graph in Figure 2 that displays the wide ranges of the attribute values though they have been normalized between 0 and 1. The mapping relations of the three coordinates to the six phases (target classes) are visualized in 2D and 3D in Figures 3 and 4, respectively. They both point to the fact that the mapping relations which constitute the construction of a classification model are indeed very nonlinear. Highly nonlinear models

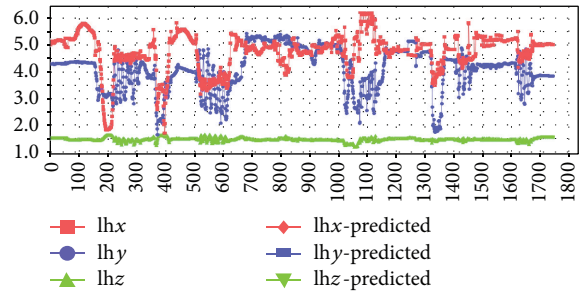


Figure 1: Fluctuation of coordinate values on the time-domain of the motion patterns from video sensor.

are known to be computationally difficult to induce especially if they need to achieve a good balance between generality and accuracy. With the additional requirement for data stream mining, processing must be fast which poses certain challenge in algorithm design.

2.2. Traditional and Incremental Model Learning Methods. Sensor data analysis is emerging and it demands an efficient classification model that is capable of mining data streams and making a prediction for unseen samples. Traditional classification approach is referred to a method of top-down supervised learning [14], where a full set of data is used to construct a classification model, by recursively partitioning the data into forming mapping relations for modelling a concept. Since these models are built based on a stationary dataset, model update needs to repeat the whole training process whenever new samples arrive, adding them to incorporate the changing underlying patterns. The traditional models might have a good performance on a full set of historical data, and the data are relatively stationary without anticipating much new changes. In dynamic stream processing environment, like gesture recognition using a video sensor, however, data streams are ever evolving and the classification model would have to be frequently updated accordingly. Therefore a new generation of algorithms, generally known as incremental classification algorithms or simply, data stream mining algorithms has been proposed to solve this problem [15].

Take decision tree construction for example, heuristic function is an important evaluation method that determines the split attributes for converting leaves into nodes, for instance, information gain used in C4.5 [16] and Hoeffding tree [17]. Traditional methods require the full dataset (newly arrival data and historical data) to update decision model while incremental methods implement a single-pass approach which is unnecessary to reload full dataset. Figure 5 shows the flow-charts of classification model induction by using these two families of learning methods.

As a technical drawback in the traditional methods, holding the whole execution process of model-induction in runtime memory is not favorable especially when the input training data are too large. Hence, incremental methods load only a small fragment of the input data stream at a time rather than filling all in one go, for refreshing the classification

Table 1: The gesture attributes of the sensor data.

Positions	Motions
(1) lh$x$: position of left hand (x coordinate)	(1) Vectorial velocity of left hand (x coordinate)
(2) lh$y$: position of left hand (y coordinate)	(2) Vectorial velocity of left hand (y coordinate)
(3) lh$z$: position of left hand (z coordinate)	(3) Vectorial velocity of left hand (z coordinate)
(4) rh$x$: position of right hand (x coordinate)	(4) Vectorial velocity of right hand (x coordinate)
(5) rh$y$: position of right hand (y coordinate)	(5) Vectorial velocity of right hand (y coordinate)
(6) rh$z$: position of right hand (z coordinate)	(6) Vectorial velocity of right hand (z coordinate)
(7) hx: position of head (x coordinate)	(7) Vectorial velocity of left wrist (x coordinate)
(8) hy: position of head (y coordinate)	(8) Vectorial velocity of left wrist (y coordinate)
(9) hz: position of head (z coordinate)	(9) Vectorial velocity of left wrist (z coordinate)
(10) sx: position of spine (x coordinate)	(10) Vectorial velocity of right wrist (x coordinate)
(11) sy: position of spine (y coordinate)	(11) Vectorial velocity of right wrist (y coordinate)
(12) sz: position of spine (z coordinate)	(12) Vectorial velocity of right wrist (z coordinate)
(13) lw$x$: position of left wrist (x coordinate)	(13) Vectorial acceleration of left hand (x coordinate)
(14) lw$y$: position of left wrist (y coordinate)	(14) Vectorial acceleration of left hand (y coordinate)
(15) lw$z$: position of left wrist (z coordinate)	(15) Vectorial acceleration of left hand (z coordinate)
(16) rw$x$: position of right wrist (x coordinate)	(16) Vectorial acceleration of right hand (x coordinate)
(17) rw$y$: position of right wrist (y coordinate)	(17) Vectorial acceleration of right hand (y coordinate)
(18) rw$z$: position of right wrist (z coordinate)	(18) Vectorial acceleration of right hand (z coordinate)
	(19) Vectorial acceleration of left wrist (x coordinate)
	(20) Vectorial acceleration of left wrist (y coordinate)
	(21) Vectorial acceleration of left wrist (z coordinate)
	(22) Vectorial acceleration of right wrist (x coordinate)
	(23) Vectorial acceleration of right wrist (y coordinate)
	(24) Vectorial acceleration of right wrist (z coordinate)
	(25) Scalar velocity of left hand
	(26) Scalar velocity of right hand
	(27) Scalar velocity of left wrist
	(28) Scalar velocity of right wrist
	(29) Scalar velocity of left hand
	(30) Scalar velocity of right hand
	(31) Scalar velocity of left wrist
	(32) Scalar velocity of right wrist

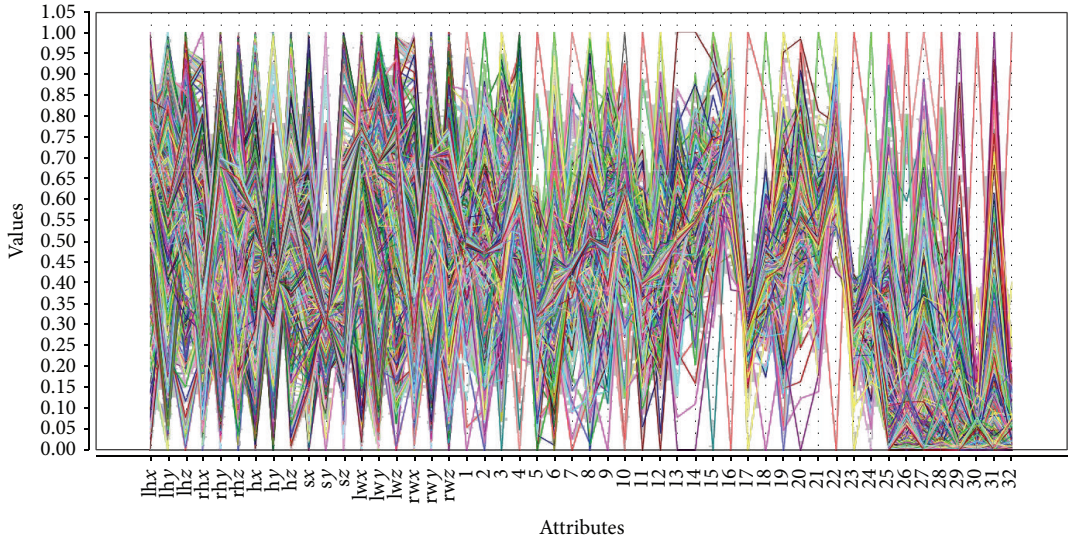


Figure 2: Normalized coordinate values of the motion patterns from video sensor.

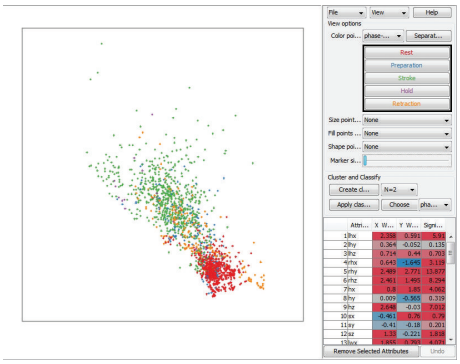


Figure 3: Visualization of three coordinate values with respect to their target phases in 2D.

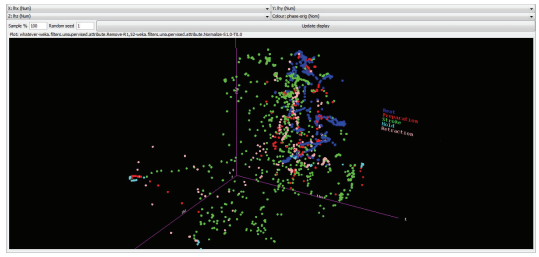


Figure 4: Visualization of three coordinate values with respect to their target phases in 3D.

model incrementally as shown in Figure 5. In incremental learning, Hoeffding bound (HB) is used to decide whether an attribute should be split to establish new nodes provided that sufficient samples for that attribute have appeared in the data stream. The new approach is designed for incremental decision trees, the pioneer of which is very fast decision tree (VFDT) and sometimes it is more generally called Hoeffding tree (HT) [17]. HT is a classical work using HB in the node-splitting test. This is attributed to the statistical property of HB that controls the node-splitting error rate on the fly.

3. Incremental Learning Model for Data Stream Mining

3.1. Batch-Learning Classification Problems. Here we review, via mathematics, why traditional model induction process may not function well for mining data stream from video sensor. Assume an instance of data arrives for model induction from the stream at timestamp t, D_t; it carries a vector of data of multiple attributes and a corresponding class value y_t as defined in

$$D_t = \left[X^t, y^t\right]. \tag{1}$$

During a time slot ($t = 1, 2, \ldots, T$) over the number of timestamp T, the data are collected into a data block D_T, where $T > 0$. D_T is defined in

$$D_T = \sum_{t=1}^{T} D_t = \begin{bmatrix} X^1 & y^1 \\ \vdots & \vdots \\ X^T & y^T \end{bmatrix}. \tag{2}$$

With the data block D_T which was collected so far on hand, a heuristic function is used for inducing a classification model. Let $H(\cdot)$ be such heuristic function; greedy search approach that works in the manner of divide-and-conquer is usually employed by traditional decision tree model that attempts to induce a globally optimal decision tree, $\mathrm{TR}_{\mathrm{GLOBAL}}$. This tree is ensured as global, because of the availability of the full collected dataset D_T. The role of $H(\cdot)$ is to one-by-one rank and select the attributes in the order of the highest information gain [18], as splitting tree nodes in the case of decision tree. There are other incremental learning methods though incremental decision tree is used for illustration here. So for each attribute X_i, of indices i and j where $i \leq M$ and $j \leq N$, for M, is the maximum number of attributes and N is the maximum number of instances received so far, x_{ij} is the splitting value. The function tries to pick the attribute X_i that has the maximum splitting value, by $x_{ij} = \arg\max H(x_{ij})$ from the splitting values ranging from x_{i1} to x_{ij}, which we have already known from D_T. This process ensures that the resultant model is globally optimal as far as the full data is collected in D_T, and it is defined in

$$\text{Maximize} \sum_{i=1}^{M} \sum_{j=1}^{N} H\left(x_{ij}\right). \tag{3}$$

For any given new instance that arrives in the future time t, X_t, the induced model will map it to a predicted class $\widehat{y_k^t}$ where k is the index to the possible set of classes, K. With reference to the data that have been collected and used for training so far, the induced model is being built with the aim of minimizing the classification error, as defined in (4). The Train$(\cdot)$ and Test$(\cdot)$ functions are generic, depending on the implementation and the choice of the classification algorithms. In general the Train$(D_T, H(\cdot))$ function takes two parameters, one is the data which would be used for supervised learning and $H(\cdot)$ is the heuristic function to be used in learning from the data D_T. The Test$(\mathrm{TR}_{\mathrm{GLOBAL}}, X^t)$ function produces a prediction result by testing the classification model which is supposed to be globally optimal over a testing sample X^t received at timestamp t:

$$\left.\begin{aligned} \mathrm{TR}_{\mathrm{GLOBAL}} &= \mathrm{Train}\left(D_T, H(\cdot)\right) \\ \widehat{y_k^t} &= \mathrm{Test}\left(\mathrm{TR}_{\mathrm{GLOBAL}}, X^t\right) \\ \mathrm{Error}_k^t &= \begin{cases} 1, & \text{if } \widehat{y_k^t} \neq y_k^t \\ 0, & \text{otherwise} \end{cases} \end{aligned}\right\} \tag{4}$$

$$\Longrightarrow \text{subject to Minimize} \sum_{t=1}^{T} \sum_{k=1}^{K} \mathrm{Error}_k^t.$$

Now consider a situation at timestamp t, the data is accumulated up to D_t, and a classification model $\mathrm{TR}_{\mathrm{GLOBAL}}$ has been induced, so far so good. When new data D_{t+1} arrives

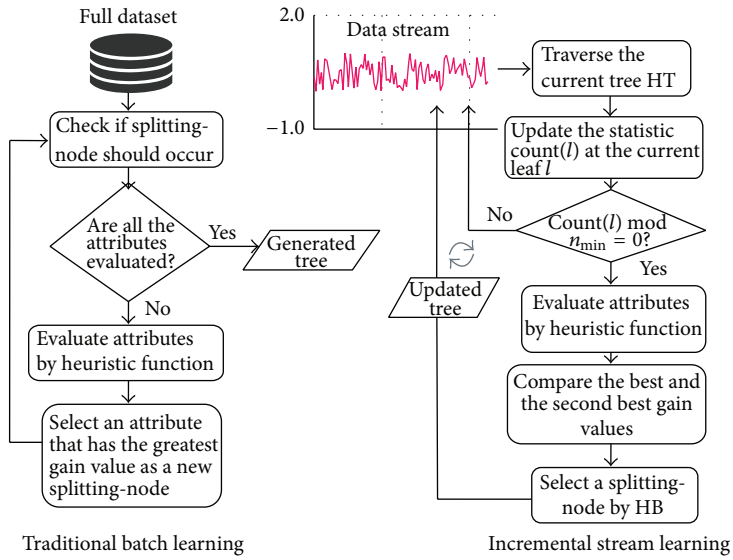


FIGURE 5: Comparison of approaches for traditional and incremental tree-building.

at $t + 1$, the classification model $\mathrm{TR}_{\mathrm{GLOBAL}}$ now needs to be updated by repeating the induction process defined by (3) and (4) with the inclusion of the new data, $D_t + D_t + 1$. The time taken for model rebuilding will only get longer as t and D_t increase. Each time it requires loading in all historical D_T repeatedly.

In mining sensor data, the collected data instances are massive in volume and ever new data are being generated frequently without end (in some cases like 24/7 video surveillance). How to keep up with the latest model efficiently is an open problem. For frequently updating model, recomputing historical data is not applicable when the data repository contains large millions of records. Some sort of incremental approach is required.

To solve this incremental problem, the authors in [17] proposed an alternative method for incrementally inducing a classification model, $\mathrm{TR}_{\mathrm{INCR}}$. This method is also known as any-time algorithm where the training data is read once only without storing or reloading it anymore. The induction method builds a tree by selecting an attribute for node-splitting by estimating the sufficient statistics that records the counts of each attribute value. This is done by computing the Hoeffding bound (HB) as defined in (5) that checks how often the attribute value x_{ij} of attribute X_i would have corresponded to class y_k:

$$\mathrm{HB} = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}, \tag{5}$$

where the class distribution is measured by R and the amount of instances that have been seen belonged to a class is n. Unlike the traditional approach, for attribute X_i the method checks on the splitting-value by nominating two best values. At any time, we have the best value of $H(\cdot)$ called x_{ia} such that $x_{ia} = \arg\max H(x_{ij})$. Likewise, the second best value is x_{ib} so $x_{ib} = \arg\max H(x_{ij})$, $\forall j \neq a$. These two best values are chosen incrementally as the induction goes and new data arrives. The difference between these two best values is calculated as in $\Delta H(X_i) = \Delta H(x_{ia}) - \Delta H(x_{ib})$ for each attribute X_i where $i \in I$. For n number of instances that have been observed so far, a confidence interval is computed by HB as in (5), called r_{true} by which we can be sure of relating the attribute value x_{ij} to class y_k. Incrementally, just by observing the confidence intervals as the only retained statistics for each attribute X_i, $r - \mathrm{HB} \leq r_{\mathrm{true}} < r + \mathrm{HB}$ where $r = (1/n)\sum_i^n r_i$. For assuring an attribute is to be nominated for node-splitting, a minimum amount of observed samples, n_{min} is required. Over the observed samples, if the inequality holds true for $r + \mathrm{HB} > 1$, and $r_{\mathrm{true}} < 1$, then the attribute x_{ia} being tested is the best candidate by the statistics based on only a part of the data stream over the entire data stream with good confidence.

In this way, we estimate the splitting-value x_{ij} of attribute X_i, without the need of knowing all attribute values from x_{i1} to x_{iN}. It hence frees us from reloading the full data for training the classification model as it learns incrementally when additional data come. The induced model can be useful in prediction at any time as well as being trained at any time by adjusting the statistics of the splitting values. While being able to embrace unlimited incoming samples from the data stream, the incremental learning is designed with the optimization goal of keeping the error minimum as follows:

$$\left.\begin{aligned} \mathrm{TR}_{\mathrm{INCR}} &= \mathrm{Train}\left(D_t, H(\cdot), \delta, n_{\mathrm{min}}\right) \\ \widehat{y_k^t} &= \mathrm{Test}\left(\mathrm{TR}_{\mathrm{INCR}}, X^t\right) \\ \mathrm{Error}_k^t &= \begin{cases} 1, & \text{if } \widehat{y_k^t} \neq y_k^t \\ 0, & \text{otherwise} \end{cases} \end{aligned}\right\} \tag{6}$$

$$\text{Subject to Minimize} \sum_{t=1}^{T}\sum_{k=1}^{K} \mathrm{Error}_k^t.$$

3.2. Incremental Learning Algorithms as Solutions. Two main schools of algorithms were designed for incremental learning: functional-based and decision tree-based. The former group of algorithms constructs a black-box model which is represented by numeric weights and coefficients for mapping the relations between the inputs and the predicted outputs. Two of the most popular functional-based incremental learning algorithms are KStar and Updatable Naïve Bayes.

The full name of KStar is "Instance-Based Learner Using an Entropic Distance Measure." As the name suggests, it learns incrementally per instance by some similarity function that measures the entropic distance between the test instance and the other instances. Motivated by information theory, the underlying similarity function solves the smoothness problem by summing the probabilities over all possible decision paths for attaining good overall performance. Due to the large amount of summation over all the possible paths, KStar usually required longer processing time than its counterparts. The details of the algorithm and its entropy-based distance function are described in full in [19]. In the same article, KStar was shown to outperform other rule based and instance based learning algorithms using some empirical datasets.

Updatable Naïve Bayes is extended from the famous Naïve Bayes classifiers which embrace a family of simple probabilistic classifiers founded on the principle of Bayes' theorem. The algorithm is designed with assumptions of possessing strong independence between the features. An advantage of this assumption is that it only requires a small amount of training data to estimate the means and variances of the features (variables) for computing the probabilities of all the possible outcomes for performing classification. Updatable Naïve Bayes is the online version of Naïve Bayes where the same algorithm continually updates its variables for tuning the hypothesis as it runs; it continually receives a new data instance and predicts its target class based on the current hypothesis; the new instance is used to further update its hypothesis accordingly too.

The other major group of algorithms is decision tree based. By the any-time tree induction principle as discussed in Section 3.2, several research papers have proposed different approaches to improve the accuracy of VFDT in the past decade. Some selected algorithms, together with KStar and Updatable Naïve Bayes will be put into experimental test in this paper. Such incremental decision tree algorithms using HB in node splitting test are so called Hoeffding tree (HT).

HOT [18] proposed an algorithm producing some optional tree branches at the same time, replacing those rules with lower accuracy by optional ones. The classification accuracy has been improved significantly while learning speed is slowed because of the construction of optional tree branches. Some of the options are inactive branches consuming computer resources are to be removed; some are random choices of trees for speed-up called random Hoeffding tree (RHT), and so forth. ADWIN [20] that stands for adaptive sliding window algorithm proposes a solution to detect changes by observing the recently seen data instances within a variable-sized sliding window. The node splitting value is judged on the variation in the average value of the instances as seen inside the window. Another adaptive algorithm is called "Concept Drift Active Learning Strategies" or just Active [21]. Active learning aims at learning an accurate model with as little tree branches as possible. It is known that during data streaming in, the data distribution in the data stream is prone to differ over time resulting in concept drift, hence the learnt model needs to adapt by relearning. Usually data stream learning focuses on checking through the uncertain instances which can be found near the decision boundary. So if the concept drift happens in other areas other than the boundary, the learning may fail to adapt. Active learning strategies make use of randomization of search space for evenly learning from the data stream. Another challenge associated with incremental learning in data streams is the huge volume of search space (as hyperplane) from which a representative feature subset needs to be derived, for efficient model induction without incurring a large latency for high-speed data stream mining. Usually the larger the amount of features in the data, the higher the cardinality of the dimensions, the huger the search space, and the extremely longer time it requires for processing. The next subsection deals with some techniques called feature selection to tackle this problem.

3.3. Feature Selection by Swarm Search and APSO. A contemporary type of feature selection algorithm, specially designed for choosing an optimal subset from a huge hyperspace is called swarm search-feature selection (SS-FS) Model [22]. SS-FS is wrapper-based feature selection model which retains the accuracy of each trial classifier built from a candidate feature subset, picks the highest possible fitness, and deems the candidate feature subset as the choice output. The workflow of the SS-FS Model is shown in Figure 6. It can be seen that the operation iterates starting from a random selection of feature subset, continues to refine the accuracy of the classification model by searching for a better feature subset, in stochastic manner. The flow enables the classification model and the chosen feature subset finally converges.

The wrapped classifier is used as a fitness evaluator, advising how useful the candidate subset of features is; the optimization function searches for candidate subset of features in stochastic manner. This approach if run by brute-force testing out all the possible subsets, it will take a very long time. For there are 50 features in the sensor data, there are $2^{50} \approx 1.1259 \times 10^{15}$ possible trials of repeatedly building the wrapped classifier. While the increase in data features goes by O^2, the high computation costs intensify proportional to the amount of instances; in the case data stream mining, the sensor feed may amount to infinity!

In this regard, a search strategy called Swarm Search is used. Instead of testing on every possible feature subset, the Swarm Search which is enabled by multiple search agents who work in parallel would be able to find the most currently optimal feature subset at any time. In order to shorten the search process, a speed-up is implemented in our proposed model by incorporating a speed-up in the initialization step in the Swarm Search, called Accelerated Particle Swarm Optimization (APSO) [23].

PSO searches the space of an objective function by adjusting the trajectories of individual agents, called particles,

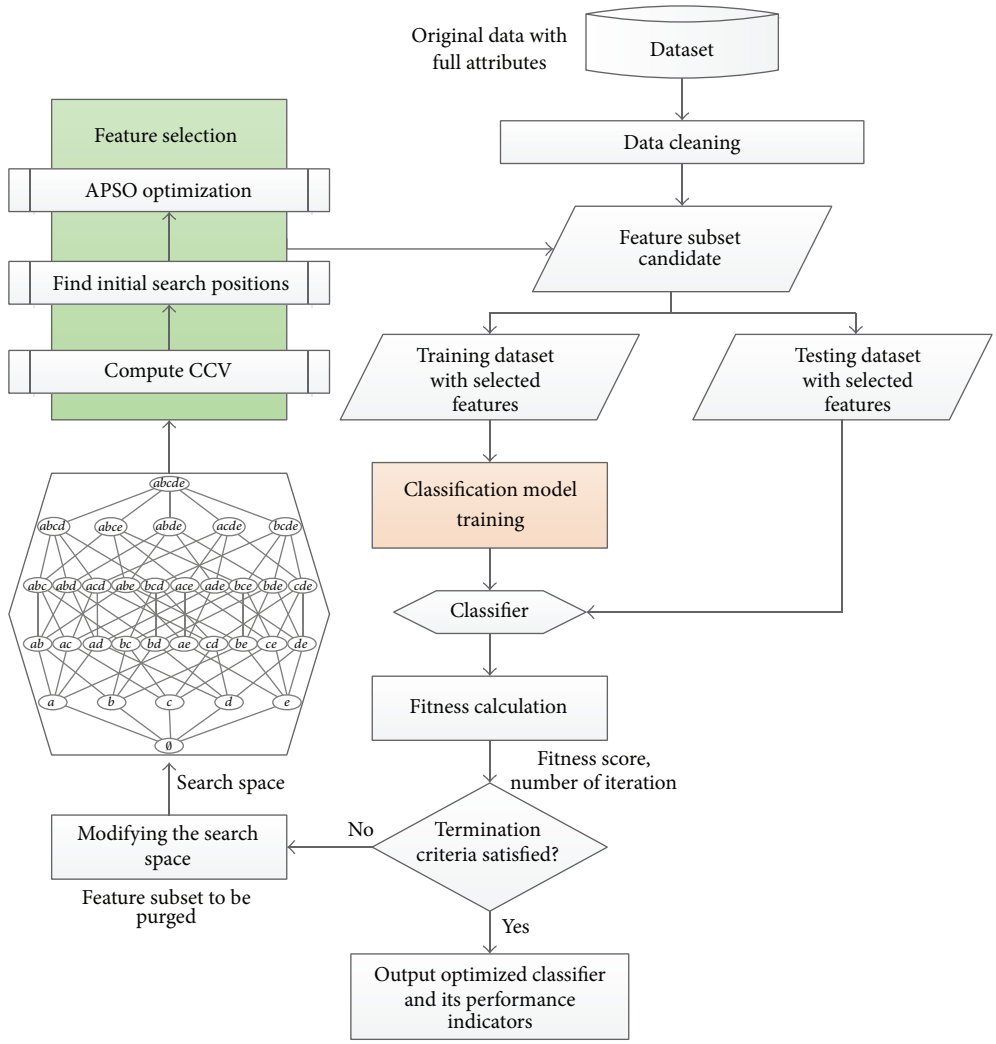


Figure 6: Workflow of swarm search feature selection model.

as the piecewise paths formed by positional vectors in a quasistochastic manner. The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted towards the position of the current global best g^* and its own best location o_i^* in history called "individual best", while at the same time it has a tendency to move randomly. Let o_i and v_i be the position vector and velocity for particle i, respectively. The velocity vector is defined by

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1\left[g^* - o_i^t\right] + \beta\epsilon_2\left[o_i^* - o_i^t\right], \quad (7)$$

where ϵ_1 and ϵ_2 are two random vectors and each entry takes the values between 0 and 1. The parameters α and β are the learning parameters for accelerating the particles with typical value of $\alpha = \beta = 2$. One noticeable improvement is the use an inertia function $\theta(t)$ so that v_i^t is replaced by $\theta(t)v_i^t$ where the velocity vector with the inertia function is defined by

$$v_i^{t+1} = \theta v_i^t + \alpha\epsilon_1\left[g^* - o_i^t\right] + \beta\epsilon_2\left[o_i^* - o_i^t\right], \quad (8)$$

where $\theta \in [0, 1]$ with a typical value of 0.5. This is similar to introducing a virtual mass to stabilize the motion of the particles, so the swarm search can converge more quickly.

The reason of using the individual best is primarily to increase the diversity in the quality solutions; however, this diversity can be simulated using some randomness. A simplified version which could accelerate the convergence of the algorithm is to use the global best only. Thus, in this version of APSO the velocity vector is generated by a simpler formula. Consider

$$v_i^{t+1} = v_i^t + \alpha\epsilon_n + \beta\left[g_i^* - o_i^t\right], \quad (9)$$

where ϵ_n is drawn from $N(0, 1)$ to replace the second term. The update of the position now becomes simply,

$$o_i^{t+1} = o_i^t + o_i^{t+1}. \quad (10)$$

In order to speed up the convergence sooner, we can define the update of the location in a single step,

$$o_i^{t+1} = (1 - \beta)o_i^t + \beta g^* + \alpha\epsilon_n. \quad (11)$$

This simpler version of position update will deliver the same order of convergence. Typically, $\alpha = 0.1L{\sim}0.5L$ where L is the scale of each variable, while $\beta = 0.1{\sim}0.7$ is sufficient for most cases. It is worth indicating that velocity does not appear in (11), and there is no need to deal with initialization of velocity vectors except the starting positions must be set appropriately.

In order to set the initial starting positions for APSO, some feature ranker function that must be quick and simple should be applied. In our proposed data stream mining model, a very simple and efficient feature selection called clustering coefficients of variation (CCV) is used for finding the ideal starting positions for APSO. CCV is based on a very simple principle of variance-basis that finds a subset of features useful for optimally balancing the classification model induction between generalization and overfitting. CCV is founded on a basic belief that a good attribute in a training dataset should have its data vary sufficiently wide across a range of values, so that it is significant to characterize a useful prediction model. The coefficient of variation (CV) is expressed as a real number from $-\infty$ to $+\infty$ and it describes the standard deviation of a set of numbers relative to their mean. It can be used to compare variability even when the units are not the same. In general CV informs us about the extent of variation relative to the size of the observation, and it has the advantage that the coefficient of variation is independent of the units of observation. The coefficient of variation, however, will be the same over all the features of a dataset as it does not depend on the unit of measurement. So you can obtain information about the data variation throughout all the features, by using the coefficient of variation to look at all the ratios of standard deviations to mean in each feature. Intuitively, if the mean is the expected value, then the coefficient of variation is the expected variability of a measurement, relative to the mean. This is useful when comparing measurements across multiple heterogeneous data sets or across multiple measurements taken on the same data set – the coefficient of variation between two data sets, or calculated for two attributes of measurements in the case of feature selection, can be directly compared, even if the data in each are measured on very different scales, sampling rates or resolutions. In contrast, standard deviation is specific to the measurement/sample it is obtained from, that is, it is an absolute rather than a relative measure of variation. In statistics, it is sometimes known as measure of dispersion, which helps compare variation across variables with different units. A variable with higher coefficient of variation is more dispersed than one with lower CV.

Let X be a training dataset with n instances of vector whose values are characterized by a total of m attributes or features. An instance is an m-dimensional tuple, in the form of $(x_1, x_2, \ldots, a_m)$. For each x_a where $a \in [1, \ldots, m]$, can be partitioned into subgroups of different classes where $c \in C$ is the total number of prediction target classes. So that $x_a \in \{x_a^1, x_a^2, \ldots, x_a^c\}$. Consider

$$v_a = \sum_{c=1}^{c} \frac{\sqrt{\left[\sum_{n=1}^{n} \left(x_n^c - \overline{x_a^c}\right)^2\right]/n}}{\overline{x_a^c}}. \tag{12}$$

$\overline{x_a^c}$ is the mean of all the ath feature values that belong to class c. v_a is the sum of all coefficients of variation for each class c where $c \in [1, \ldots, C]$, for that particular ath feature. The coefficient of variation is expressed as a real number from $-\infty$ to $+\infty$. The subsequent step required in CCV after calculating the CV is to find a threshold in order to decide which features and how many features are to be retained. The underlying concept behind this task is Bia-Variance dilemma. Some recent studies stated that the decomposition of a supervised learner's error into bias and variance terms can provide considerable insight into the prediction performance of the classifier learner.

Assume a target function: $t(x) = g(x) + \varepsilon$. Then the expected squared error over fixed size training sets D drawn from $P(X, T)$ can be expressed as sum of three components:

$$\begin{aligned}
&\sum_D \left[\int_x \int_t (h(x) - t)^2 \, p(t \mid x) \, p(x) \, dt \, dx\right] \\
&\quad = \sigma^2 + \text{bias}^2 + \text{variance}, \\
&\qquad \sigma^2 = \text{unavoidable Error}, \\
&\text{bias}^2 = \int \left(\sum_D [h(x)] - g(x)\right)^2 p(x) \, dx, \\
&\qquad \overline{h}(x) = \sum_D [h(x)], \\
&\text{variance} = \int \sum_D \left[\left(h(x) - \overline{h}(x)\right)^2\right] p(x) \, dx.
\end{aligned} \tag{13}$$

Our goal is to minimize the expected loss, which we have decomposed into the sum of a (squared) bias, a variance, and a constant noise term. As we shall see, there is a trade-off between bias and variance, with very flexible models (which can possibly overfit) having low bias and high variance, and relatively rigid models (under fit) having high bias and low variance. In order to achieve this optimum equilibrium between bias and variance, a simple K-means clustering technique is employed. It tries partition the data point into two clusters: one to be retained and the other one to be removed. The goal is to assign membership of a cluster to each data point. Clustering algorithm helps to find the ideal cluster positions μ_i, $i = 1, \ldots, k$ of the clusters that minimize the distance from the data points to the cluster centroids, with the following objective function:

$$\begin{aligned}
\arg &= \min_c \sum_{i=1}^{2} \sum_{x \in c_i} d(x, \mu_i) \\
&= \arg\min_c \sum_{i=1}^{2} \sum_{x \in c_i} \|x - \mu_i\|^2,
\end{aligned} \tag{14}$$

where c_i is the set of points that belong to cluster i. The clustering algorithm uses the square of the Euclidean distance $d(x, \mu_i) = \|x - \mu_i\|^2$.

Given a data set $X = \{x_1, x_2, \ldots, x_n\}$, the estimation function is $f(x) = a_0 x_{0+} a_1 x_{1+} a_2 x_{2+} \ldots = \vec{a}\vec{x}$. As it was

mentioned before, adding more parameters into the model as features, the complexity of the model rises, so does the variance while bias steadily falls. The function of K-means is to divide the data set into two groups according to the values of coefficient of variation. The values of variance-bias are different for the data points in different clusters that reflect the complexity of model. It is known the more a complex model, the more bias it is, and vice versa. So we are reducing the complexity of model by choosing some valuable attributes by separating the variance. The total error of two groups:

$$\text{group}_1 = \text{bias}^2 \uparrow + \text{variance} \downarrow + \sigma^2, \quad (15)$$

$$\text{group}_2 = \text{bias}^2 \downarrow + \text{variance} \uparrow + \sigma^2. \quad (16)$$

One of the two groups (15) and (16) with data points representing the combinations of variances and biases is to be chosen as the optimal feature subset. A quick and effective division method call HyperPipes [24] is utilized for this task. HyperPipes is a probabilistic learning tool that is very similar to Naïve Bayes, except for the fact that it does not record the frequency count of how attributes correspond to classes. In essence, an attribute either corresponds to a hypothetical class or it does not, regardless of how many times this is the case. The learner will record all of the attributes and their correspondence with the class in a table of Booleans. The learner will determine the class based on the score of the attributes added up (0 for if it does not exist and 1 for if it does).

4. Mining Sensor Data Streams

4.1. Evaluation Method. The experiment contains two parts: firstly, we compare two groups of classification learning methods, traditional batch learning and incremental learning pertaining to their classification performance such as accuracy, kappa, precision and recall, and so forth. The names of the classification learning algorithms, together with a short description are shown in Table 2. The choices of algorithms for both groups are popular methods that have been used widely in the literature. The data stream mining algorithms which are put under test here are mainly inherited from the Hoeffding principle in growing a decision tree. In addition, two nondecision tree type of incremental learning such as Updateable Naïve Bayes and KStar are tested in the comparison. Secondly the timing performance is evaluated for the two groups of classification, in relation to the cost-benefit of accuracy improvement at the price of extra running time.

The experiment was conducted on the computing platform of a Dell Precision T7610 PC with Intel Xeon Processor E5-2670 v2 (Ten Core HT, 2.5 GHz Turbo, 25 MB) and 128 GB RAM. The programming environment is Java Development Kit 1.5. For the algorithms, they are implemented on MOA platform. Default parameter values are set for all experimentation runs. For a fair evaluation over the efficacy of the algorithms, 10-fold cross-validation is used for obtaining an unbiased estimate of the accuracy performance of the classification models. The data is divided into 10 subsets of equal portion; the models by the same algorithm are built 10 rounds, each round sparing out one of the 10 subsets from training the model, as unseen data for performance validation.

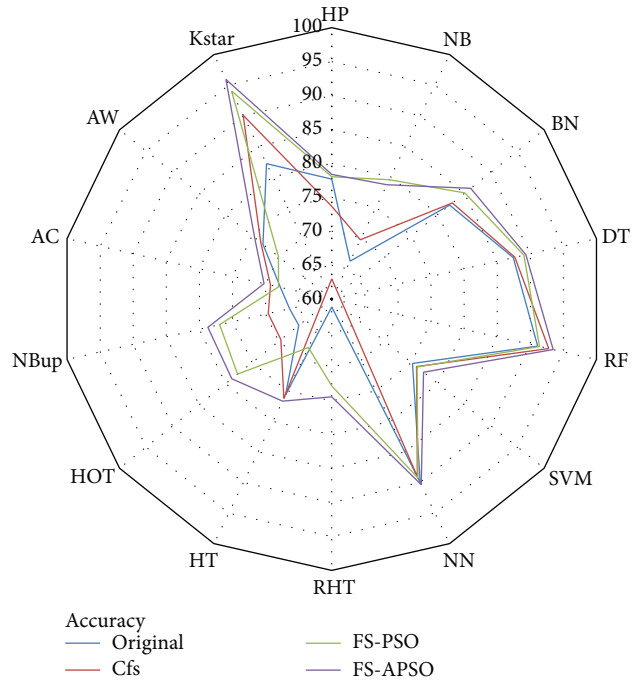


Figure 7: Radar chart of sensor data classification performance in accuracy.

4.2. Sensor Data Classification. The sensor data that is subject to the experiment of performance evaluations is treated with 4 types of preprocessing methods for feature selection. The first preprocessing does no feature selection we simply call it "Original" meaning the sensor data is in its original form as collected from the video sensor; the second method is preprocess with Correlation-based feature selection, namely Cfs which is a popular approach in data mining, the third preprocessing is done with swarm search feature selection using PSO, called FS-PSO; and the fourth preprocessing is the same as the third method except standard PSO is replaced by Accelerated PSO, called FS-APSO.

The experiment conducted over a number of combinations of feature selection preprocessing methods and classification algorithms, from both traditional and incremental learning types. The performance results are harvested in terms of Accuracy, Kappa (Kappa statistics), True Positives rate, false positive rate, precision, recall, F-measure, model building time per run, preprocessing time, and number of features selected. The results are tabulated in Tables 3 and 4 for traditional classification algorithms and incremental classification algorithms respectively. Some selected important performance indicators such as Accuracy, Kappa, True Positive rate, False Positive rate, Time and size of selected feature subset are graphed in radar charts respectively in Figures 7–12.

4.3. Discussion of the Results. The radar charts are laid out by placing the 7 traditional classification algorithms on the right side of the chart, and the 7 incremental algorithms on the left, for easy comparison. The accuracy measure is defined

Table 2: Classification algorithms used in experiments.

Traditional classifier	Description
HyperPipe (HP)	For each category a HyperPipe is constructed that contains all points of that category (essentially records the attribute bounds observed for each category). Test instances are classified according to the category that "most contains the instance"
Naviie Bayes (NB)	Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an Updateable Classifier (which in typical usage is initialized with zero training instances)
BayesNet (BN)	Bayes network learning using various search algorithms and quality measures. Base class for a Bayes network classifierprovides data structures (network structure, conditional probability distributions, etc.) and facilities common to Bayes Network learning algorithms like $K2$ and B
Decision tree (DT)	Generating a pruned C4.5 decision tree
Random forest (RF)	Constructing a forest of random trees
Support vector machine (SVM)	A wrapper class for the libsvm tools (the libsvm classes, typically the jar file, need to be in the classpath to use this classifier)
Neural network (NN)	A classifier that uses backpropagation to classify instances. The nodes in this network are all sigmoid (except for when the class is numeric in which case the output nodes become unthresholded linear units)
Incremental classifier	**Description**
Random Hoeffding tree (RHT)	Random decision trees for data streams
Hoeffding tree (HT)	Very fast decision tree implementation using Hoeffding bound
Hoeffding option tree (HOT)	Hoeffding tree: Single tree that represents multiple trees
NBUpdateable (NBup)	This is the updateable version of Naive Bayes. This classifier will use a default precision of 0.1 for numeric attributes when buildClassifier is called with zero training instances
Active (AC)	Active learning classifier for evolving data streams
Adwin (AW)	ADaptive sliding WINdow method. This method is a change detector and estimator. It keeps a variable-length window of recently seen items, with the property that the window has the maximal length statistically consistent with the hypothesis "there has been no change in the average value inside the window"
Kstar	K^* is an instance-based classifier; that is, the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function

by the number of correctly classified instances over the total instances in the sensor data. In the case of batch learning by traditional algorithms, the accuracy is the ratio of correctly classified instances over all the 99,000 instances. In the case of incremental learning, the accuracy is measured by averaging all the intermediate accuracies resulted from each data segment over a series of tests. In Figure 7 the overall accuracy by the traditional classification algorithms is slightly higher than those by the incremental algorithms: average accuracy 82.98296% for traditional versus 74.08409% for incremental. The top performers are Neural Network and KStar. The performance in general for the preprocessing methods of original and Cfs is out-performed by FS-PSO and FS-PSO. Generally Cfs consistently offered improvement in accuracy for traditional algorithms, though marginally. For incremental algorithms however Cfs do not always have enhance the accuracy. This may be due to the fact that the calculation of correlation between targets and attributes in the incremental mechanism does not work well with nonstationary data, and vice versa. The swarm search type of feature selections (FS) unanimously outperformed Cfs. The improvement by FS is most obvious for NB, RHT, HOT, NBup, and KStar algorithms. These algorithms have a phenomenon in common as their model structures are loosely represented by a large set of numeric variables. Like HOT and RHT for example, the decision trees are in multiple forms, gathering a pool of possible model candidate during the induction process. NB, NBup, and KStar are represented by a large number of conditional probabilities and statistical variables. These models are relatively loosely defined; hence the stochastic search by PSO is appropriate and effective in finding the optimal feature subsets leading to a big leap in performance improvement. The proposed new version of APSO for Swarm Search, namely FS-APSO nevertheless shows its superior respective to performance improvement over the standard PSO version by FS-PSO. FS-APSO is better than FS-PSO in all cases except NB. Moreover, for HT, PSO has very poor performance in upholding the accuracy whereas APSO solved the problem. By far, FS-APSO has shown the maximum accuracy improvement compared to original and Cfs, indicating that FS-APSO would be a feasible feature selection scheme for the other family members of Hoeffding tree. When it comes to performance indicators like Kappa and True Positive rate, the algorithms show similar patterns as described above in Figures 8 and 9

Table 3: The performance results of classifying sensor data using traditional algorithms.

Traditional slassifier/feature selection	Accuracy	Kappa	TP	FP	Precision	Recall	*F*-measure	Model building time (s)	Preprocessing time (s)	# selected features
HyperPipe (HP)										
Original	77.6822	0.6666	0.777	0.084	0.803	0.777	0.772	0	0	50
Cfs	73.6661	0.6159	0.737	0.078	0.817	0.737	0.733	0	0	23
FS-PSO	78.0264	0.672	0.78	0.083	0.814	0.78	0.774	0	2	36
FS-APSO	78.3706	0.6764	0.784	0.084	0.817	0.784	0.777	0	2	32
Naive Bayes (NB)										
Original	66.2077	0.5257	0.662	0.085	0.732	0.662	0.68	0.08	0	50
Cfs	69.7074	0.569	0.697	0.08	0.744	0.697	0.707	0.01	0	23
FS-PSO	79.5181	0.6953	0.795	0.076	0.791	0.795	0.789	0.01	9	15
FS-APSO	78.7149	0.6835	0.787	0.079	0.785	0.787	0.78	0	9	12
BayesNet (BN)										
Original	82.2146	0.7417	0.822	0.052	0.835	0.822	0.828	0.37	0	50
Cfs	82.7309	0.7459	0.827	0.057	0.828	0.827	0.827	0.04	0	23
FS-PSO	85.0832	0.78	0.851	0.05	0.849	0.851	0.849	0.05	17	25
FS-APSO	86.1733	0.7952	0.862	0.049	0.859	0.862	0.86	0.05	12	19
Decision tree (DT)										
Original	87.4355	0.8147	0.874	0.044	0.875	0.874	0.874	0.2	0	50
Cfs	87.6649	0.8178	0.877	0.044	0.876	0.877	0.876	0.11	0	23
FS-PSO	89.0419	0.8379	0.89	0.042	0.889	0.89	0.89	0.11	36	23
FS-APSO	89.3287	0.842	0.893	0.041	0.891	0.893	0.892	0.07	20	13
Random forest (RF)										
Original	91.1073	0.8654	0.911	0.048	0.911	0.911	0.904	0.21	0	50
Cfs	92.7711	0.8916	0.928	0.036	0.927	0.928	0.924	0.09	0	23
FS-PSO	91.3941	0.87	0.914	0.046	0.913	0.914	0.908	0.08	34	21
FS-APSO	93.4596	0.9024	0.935	0.032	0.934	0.935	0.933	0.11	31	13
Support vector machine (SVM)										
Original	75.2151	0.5961	0.752	0.155	0.678	0.752	0.659	2.05	0	50
Cfs	75.961	0.6123	0.76	0.143	0.742	0.76	0.678	0.96	0	23
FS-PSO	76.0757	0.6145	0.761	0.141	0.738	0.761	0.682	0.39	349	14
FS-APSO	77.2806	0.6359	0.773	0.133	0.748	0.773	0.706	0.42	296	13
Neural network (NN)										
Original	90.0172	0.8512	0.9	0.042	0.898	0.9	0.897	24.44	0	50
Cfs	89.0419	0.8369	0.89	0.041	0.889	0.89	0.888	8.35	0	23
FS-PSO	89.2714	0.8403	0.893	0.043	0.889	0.893	0.89	14.71	5212	35
FS-APSO	90.3614	0.8565	0.904	0.04	0.901	0.904	0.901	12.49	4685	32

respectively. False positive rate which is also known as false alarm rate is an undesirable feature in machine learning. Figure 10 shows that RHT with Cfs incurred the highest false alarm rate, inferring the unsuitability of correlation-based feature selection for data stream mining especially when many random trees are being generated during runtime. FS-APSO managed to subside the false alarm rate in all cases. KStar in particular works extremely well with FS-APSO being able to maintain the lowest false alarm rate of all.

The amounts of features that are selected as an optimal subset by different combination of algorithms are shown in Figure 12. It can be seen that FS-APSO is capable of maintaining only the minimum amount of features which are significant enough for inducing classification models of the highest accuracies in most of the cases. Followed by FS-PSO the standard version of APSO, likewise can select fewer features than Cfs in all except NN, BN and HP. Less number of features to be selected may imply simpler deployment of classification or prediction in sensor data, without the need of using a full array of features, each of these features may require certain processing resource and sensing abilities. In other words, it would be cost-effective if fewer features were to be required yet being able to attain a good level of accuracy in sensor data classification.

Table 4: The performance results of classifying sensor data using incremental algorithms.

Incremental classifier/feature selection	Accuracy	Kappa	TP	FP	Precision	Recall	*F*-measure	Model building time (s)	Preprocessing time (s)	# selected features
Random Hoeffding tree (RHT)										
Original	61.1589	0.4226	0.612	0.158	0.607	0.612	0.589	0.02	0	50
Cfs	57.0281	0.3233	0.57	0.245	0.522	0.57	0.523	0.01	0	23
FS-PSO	72.8055	0.5749	0.728	0.143	0.658	0.728	0.68	0.01	5	24
FS-APSO	74.4119	0.5923	0.744	0.142	0.665	0.744	0.681	0.01	5	16
Hoeffding tree (HT)										
Original	75.6741	0.6309	0.76	0.113	0.725	0.76	0.735	0.35	0	50
Cfs	76.2478	0.6402	0.763	0.103	0.735	0.763	0.745	0.08	0	23
FS-PSO	67.9289	0.5569	0.719	0.153	0.669	0.719	0.677	0.03	16	16
FS-APSO	76.7068	0.6457	0.767	0.104	0.739	0.767	0.747	0.03	15	15
Hoeffding option tree (HOT)										
Original	66.2077	0.5262	0.662	0.085	0.732	0.662	0.68	0.2	0	50
Cfs	69.5927	0.5674	0.696	0.081	0.742	0.696	0.706	0.04	0	23
FS-PSO	77.7395	0.6657	0.777	0.09	0.763	0.777	0.762	0.02	11	13
FS-APSO	78.8296	0.6856	0.788	0.077	0.788	0.788	0.782	0.02	10	13
NBUpdateable (NBup)										
Original	66.3798	0.5281	0.664	0.085	0.733	0.664	0.682	0.07	0	50
Cfs	69.5927	0.5676	0.696	0.08	0.743	0.696	0.706	0.02	0	23
FS-PSO	76.9363	0.6588	0.769	0.082	0.768	0.769	0.764	0.01	9	15
FS-APSO	78.7149	0.6835	0.787	0.079	0.785	0.787	0.78	0.01	12	12
Active (AC)										
Original	67.7567	0.5387	0.678	0.092	0.719	0.678	0.694	0.17	0	50
Cfs	69.2484	0.5554	0.692	0.092	0.718	0.692	0.702	0.12	0	23
FS-PSO	68.0436	0.5342	0.68	0.105	0.703	0.68	0.689	0.17	55	18
FS-APSO	70.2238	0.5647	0.702	0.095	0.72	0.702	0.709	0.14	50	20
Adwin (AW)										
Original	72.9776	0.6035	0.73	0.091	0.735	0.73	0.731	1.55	0	50
Cfs	73.3792	0.6101	0.734	0.088	0.74	0.734	0.732	0.44	0	23
FS-PSO	70.0516	0.5566	0.701	0.118	0.71	0.701	0.698	0.47	176	13
FS-APSO	74.5267	0.619	0.745	0.099	0.743	0.745	0.732	0.38	142	10
Kstar										
Original	82.1572	0.734	0.822	0.085	0.834	0.822	0.818	0	0	50
Cfs	90.0746	0.8536	0.901	0.038	0.902	0.901	0.9	0	0	23
FS-PSO	94.0333	0.9117	0.94	0.026	0.94	0.94	0.94	0	2121	12
FS-APSO	95.9266	0.94	0.959	0.014	0.959	0.959	0.959	0	1817	9

Lastly, the factor of runtime is considered together with other accuracy performance. Figure 11 shows a comparison of preprocessing times incurred by different mixes of feature selection and classification algorithms. Cfs takes almost no-time which is indeed a benefit although Cfs is underperforming in accuracy and other performance indicators. By comparing only FS-PSO and FS-APSO which are stochastic in nature and they do need to take time to search for the optimal feature subset, it is interesting to observe which is more efficient. FS-APSO with the benefits of precalculating the qualified features, very quickly by CCV, as initial starting search position, shortens the runtime in all cases (except NBup) when compared to FS-PSO. HP is amazingly quick with both FS-PSO and FS-APSO, followed by RHT which completes the preprocessing in a relatively short time. KStar, NN, SVM, and AC however require relatively the longest preprocessing in both types of FS methods. By glancing over the results on Figure 11 it can be seen that the preprocessing in the traditional group of classification algorithms takes slightly longer than the incremental group of algorithms in data stream mining. This could be explained by the nature of the traditional classifiers which are embedded in the swarm search as a fitness evaluation function is time-consuming over the stationary data. On the other hand, the incremental

Table 5: Comparison of gain in accuracy per preprocessing second for different algorithms.

	FS-PSO	FS-APSO
Traditional Classifier		
HyperPipe (HP)	0.1721	0.3442
Naive Bayes (NB)	1.4789333	1.3896889
BayesNet (BN)	0.1687412	0.3298917
Decision tree (DT)	0.0446222	0.09466
Random forest (RF)	0.0084353	0.0758806
Support vector machine (SVM)	0.0024659	0.006978
Neural network (NN)	−0.0001431	$7.347E-05$
Average:	**0.2678793**	**0.3201961**
Incremental classifier		
Random Hoeffding tree (RHT)	2.32932	2.6506
Hoeffding tree (HT)	−0.484075	0.0688467
Hoeffding pption tree (HOT)	1.0483455	1.26219
NBUpdateable (NBup)	1.1729444	1.027925
Active (AC)	0.0052164	0.049342
Adwin (AW)	−0.016625	0.0109092
Kstar	0.0055993	0.0075781
Average:	**0.5801037**	**0.7253416**

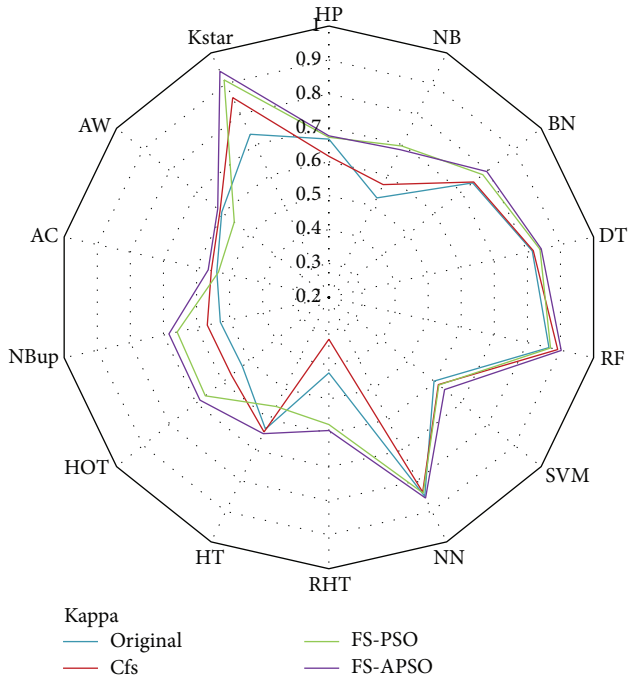


Figure 8: Radar chart of sensor data classification performance in Kappa.

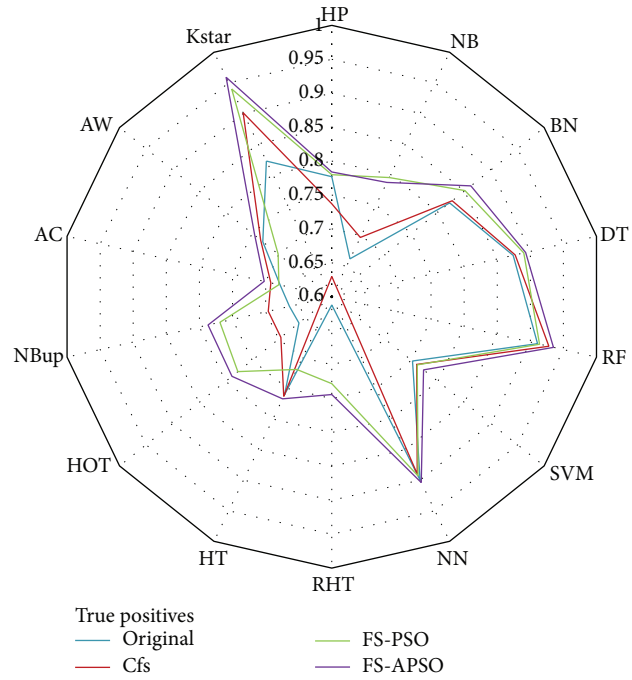


Figure 9: Radar chart of sensor data classification performance in true positive.

algorithms that mine along the data stream when being used as the fitness function performs much quicker because of its incremental nature.

In order to have a fairness comparison with respect to time, a new indicator called gain is proposed in this paper. Gain is simply the performance increase factor, considering the increment of accuracy (accuracy % with feature selection, accuracy % of original) over the number of seconds consumed in preprocessing. Ideally we favour a combination of algorithms with a high-gain, meaning it can yield the highest increase in accuracy while incurring the shortest preprocessing time. For the comparison of different combinations of algorithms, in view of this gain indicator, a gain value for each of them is computed and tabulated in Table 5.

As it is shown in Table 5, the highest average gain is the group of incremental learning algorithms coupled with FS-APSO (0.7253), followed by the same incremental group with FS-PSO (0.5801) and then traditional algorithms with FS-APSO (0.3202) and traditional group with FS-PSO comes last (0.2679). Individually the top performance in gain is by RHT combined with FS-APSO. Both RHT and FS-APSO employed a lot of randomization functions, yet they are complimenting each other in operation. NN in turn has the least gain for it has a rigid mechanism in machine learning by adjusting its internal weights and activation function.

To sum up, it is most feasible to utilize our newly proposed FS-APSO for data stream mining, particularly for RHT algorithm. Alternatively, NB, DT, and RF are good choices considering their relatively high accuracy and moderate amount of preprocessing times.

5. Conclusion

As long as a sensing device is operating, it collects a large amount of data streams all the time. Fresh data are being generated at all times that it requires an incremental computation which is able to monitor large scale of data dynamically. As a result, the algorithm design of data mining sensor application shall consider a lightweight incremental algorithm that is

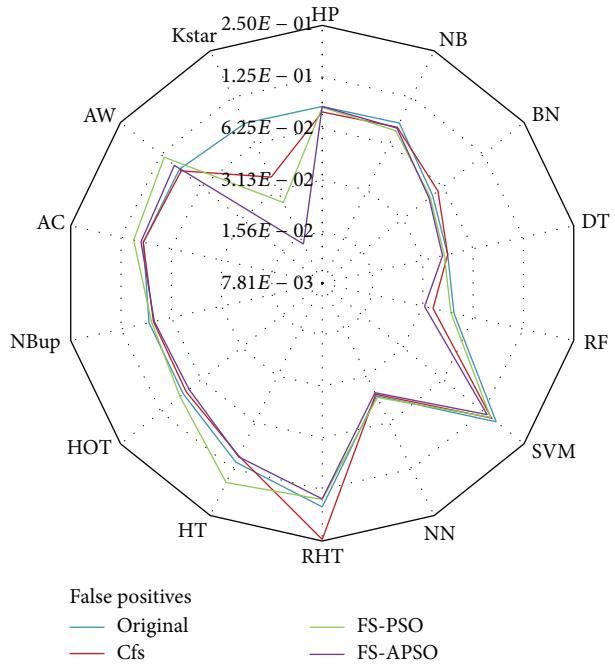


Figure 10: Radar chart of sensor data classification performance in false positive.

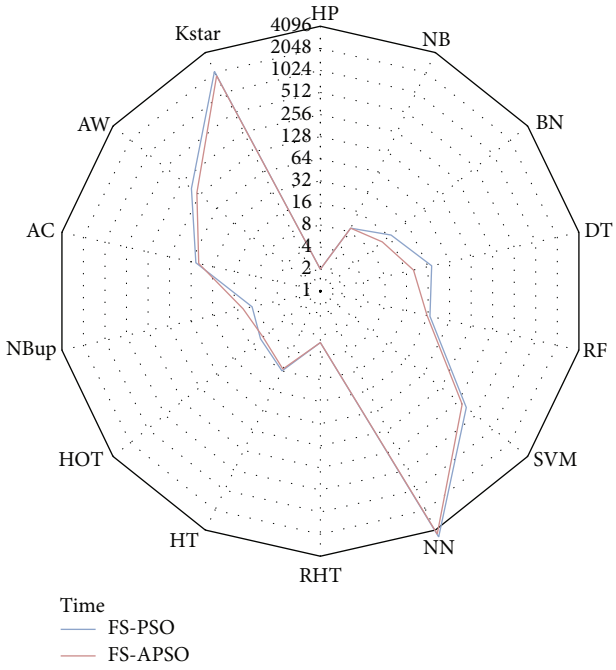


Figure 11: Radar chart of sensor data classification performance in time.

capable of robustness, high accuracy and minimum preprocessing latency. In this paper, we investigated the possibility of using a group of incremental classification algorithm for classifying the collected data streams from video sensor for gesture recognition. As a case study empirical data stream was used that was donated by a research team from the

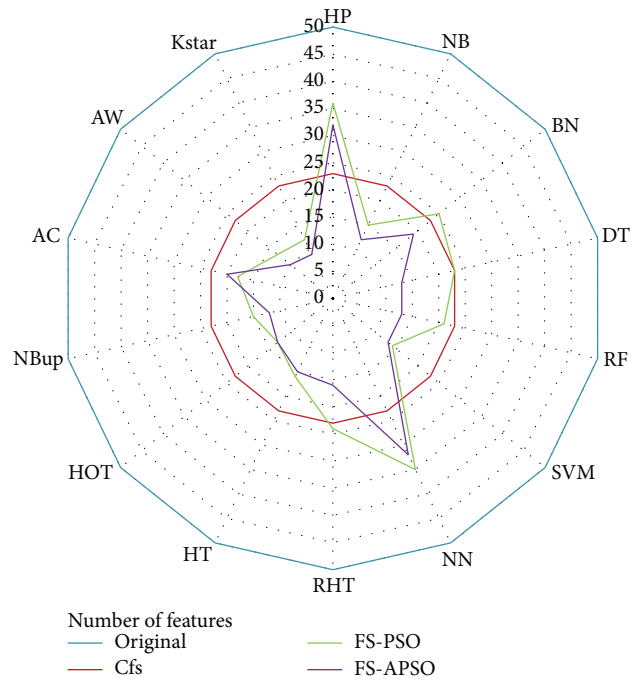


Figure 12: Radar chart of sensor data classification performance in number of selected features.

research team of Madeo et al. at University of Sao Paulo, Brazil. The data collected are visual feeds of composed video captured by Microsoft Kinect sensor. The video data are transformed into 50 numeric variables extracted from videos with people gesticulating, aiming at studying Gesture Phase Segmentation initial. We compared the traditional classification model induction and their counter-part in incremental inductions. In particular we proposed a novel lightweight feature selection method by using Swarm Search and Accelerated PSO, which is supposed to be suitable for data stream mining. The evaluation result showed that the incremental method obtained a higher gain in accuracy per second incurred in the preprocessing. The contribution of this paper is experimental insights for anybody who wishes to design a similar gesture recognition application from video sensors in choosing the appropriate decision support algorithms especially in scenario of mining activity patterns that are temporal and streaming in nature. In the future, we will want to extend the data stream mining of such sensor data with extra capabilities of sensing more complex gestures for richer information in the experimentation.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] N. Kiryati, T. R. Raviv, Y. Ivanchenko, and S. Rochel, "Real-time abnormal motion detection in surveillance video," in

Proceedings of the 19th International Conference on Pattern Recognition, pp. 1–4, December 2008.

[2] J. Ruiz, Y. Li, and E. Lank, "User-defined motion gestures for mobile interaction," in *Proceedings of the 29th Annual CHI Conference on Human Factors in Computing Systems (CHI '11)*, pp. 197–206, Vancouver, Canada, May 2011.

[3] K. M. Culhane, M. O'Connor, D. Lyons, and G. M. Lyons, "Accelerometers in rehabilitation medicine for older adults," *Age and Ageing*, vol. 34, no. 6, pp. 556–560, 2005.

[4] J. W. Lee, A. Helal, Y. Sung, and K. Cho, "Context-driven control algorithms for scalable simulation of human activities in smart homes," in *Proceedings of the 10th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC '13) and 10th IEEE International Conference on Autonomic and Trusted Computing (ATC '13)*, pp. 285–292, December 2013.

[5] L. Cheng, S. Hailes, D. Leung, F. Fan, Y. Yang, and Z. Cheng, "An experimental study on a motion sensing system for sports training," in *Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN '08)*, Bologna, Italy, 2008.

[6] Y. Sung and K. Cho, "Collaborative programming by demonstration in a virtual environment," *IEEE Intelligent Systems*, vol. 27, no. 2, pp. 14–17, 2012.

[7] K. Cho, H. Cho, and U. Kyhyun, "Inferring stochastic regular grammar with nearness information for human action recognition," in *Proceedings of the International Conference on Image Analysis and Recognition (ICIAR '06)*, pp. 193–204, 2006.

[8] Microsoft, "'Project Natal' 101," 2009.

[9] H. Yang, S. Fong, G. Sun, and R. Wong, "A very fast decision tree algorithm for real-time data mining of imperfect data streams in a distributed wireless sensor network," *International Journal of Distributed Sensor Networks*, vol. 2012, Article ID 863545, 16 pages, 2012.

[10] H. Yang, S. Fong, R. Wong, and G. Sun, "Optimizing classification decision trees by using weighted naïve bayes predictors to reduce the imbalanced class problem in wireless sensor network," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 460641, 15 pages, 2013.

[11] R. C. B. Madeo, C. A. M. Lima, and S. M. Peres, "Gesture unit segmentation using support vector machines: segmenting gestures from rest positions," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC '13)*, pp. 46–52, March 2013.

[12] R. C. B. Madeo, P. K. Wagner, and S. M. Peres, "A review of temporal aspects of hand gesture analysis applied to discourse analysis and natural conversation," *International Journal of Computer Science and Information Technology*, vol. 5, pp. 1–20, 2013.

[13] R. C. B. Madeo, *Support vector machines and gesture analysis: incorporating temporal aspects [M.S. thesis]*, Universidade de Sao Paulo, Sao Paulo Researcher Foundation, São Paulo, Brazil, 2013, (Portuguese).

[14] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers—a survey," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 35, no. 4, pp. 476–487, 2005.

[15] C. C. Aggarwal, Ed., *Data Streams: Models and Algorithms*, vol. 31, Springer, 2007.

[16] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, Calif, USA, 1993.

[17] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00)*, pp. 71–80, ACM, New York, NY, USA, 2000.

[18] B. Pfahringer, G. Holmes, and R. Kirkby, "New options for hoeffding trees," in *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pp. 90–99, Gold Coast, Australia, December 2007.

[19] J. G. Cleary and L. E. Trigg, "K*: an instance-based learner using an entropic distance measure," in *Proceedings of the 12th International Conference on Machine Learning*, pp. 108–114, 1995.

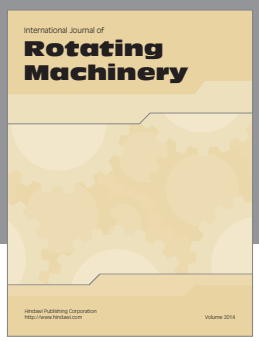
[20] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 443–448, 2007.

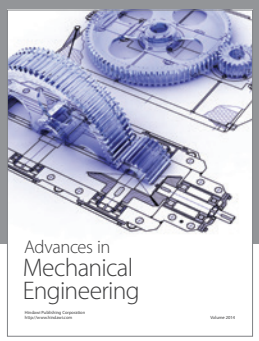
[21] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with evolving streaming data," in *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD '11), Athens, Greece, September 5–9, 2011*, vol. 6913 of *Lecture Notes in Computer Science*, pp. 597–612, Springer, Berlin, Germany, 2011.

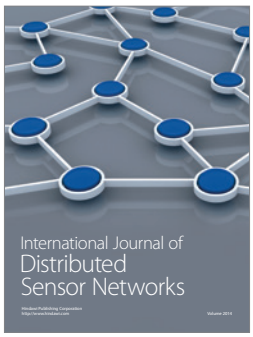
[22] S. Fong, S. Deb, X.-S. Yang, and J. Li, "Metaheuristic swarm search for feature selection in life science classification," *IEEE IT Professional Magazine*, vol. 16, no. 4, pp. 24–29, 2014.

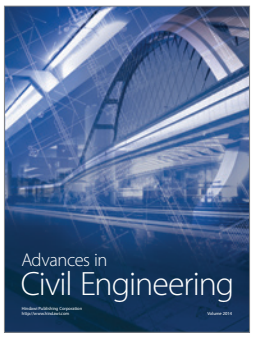
[23] X.-S. Yang, S. Deb, and S. Fong, "Accelerated particle swarm optimization and support vector machine for business optimization and applications," in *Networked Digital Technologies: Third International Conference, NDT 2011, Macau, China, July 11–13, 2011. Proceedings*, vol. 136, pp. 53–66, Springer, Berlin, Germany, 2011.

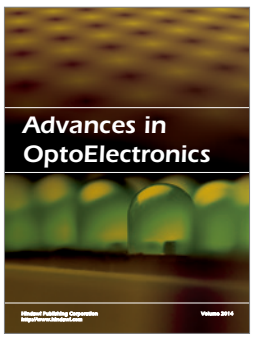
[24] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 2005.

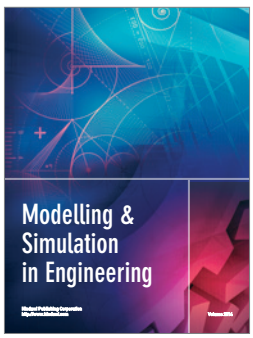


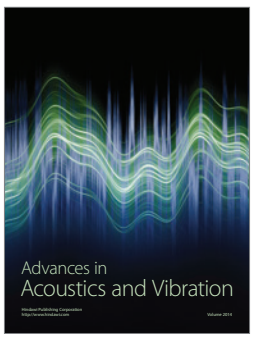


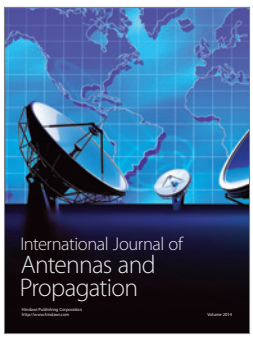


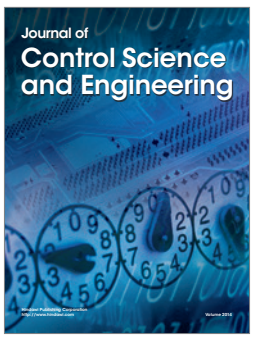


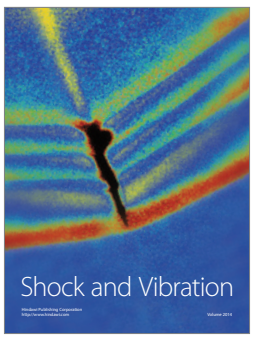












Hindawi

Submit your manuscripts at
http://www.hindawi.com