Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 863545, 16 pages
doi:10.1155/2012/863545

Research Article

A Very Fast Decision Tree Algorithm for Real-Time Data Mining of Imperfect Data Streams in a Distributed Wireless Sensor Network

Hang Yang,[1] Simon Fong,[1] Guangmin Sun,[2] and Raymond Wong[3]

[1] *Department of Computer and Information Science, University of Macau, Taipa, Macau*
[2] *Department of Electronic Engineering, Beijing University of Technology, Beijing 100022, China*
[3] *School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia*

Correspondence should be addressed to Simon Fong, ccfong@umac.mo

Received 6 October 2012; Accepted 22 October 2012

Academic Editor: Sabah Mohammed

Wireless sensor networks (WSNs) are a rapidly emerging technology with a great potential in many ubiquitous applications. Although these sensors can be inexpensive, they are often relatively unreliable when deployed in harsh environments characterized by a vast amount of noisy and uncertain data, such as urban traffic control, earthquake zones, and battlefields. The data gathered by distributed sensors—which serve as the eyes and ears of the system—are delivered to a decision center or a gateway sensor node that interprets situational information from the data streams. Although many other machine learning techniques have been extensively studied, real-time data mining of high-speed and nonstationary data streams represents one of the most promising WSN solutions. This paper proposes a novel stream mining algorithm with a programmable mechanism for handling missing data. Experimental results from both synthetic and real-life data show that the new model is superior to standard algorithms.

1. Introduction

It is anticipated that wireless sensor networks (WSNs) will enable the technology of today to be employed in future applications ranging from tracking, monitoring, and spying systems to various other technologies likely to improve aspects of everyday life. WSNs offer an inexpensive way to collect data over a distributed environment that may be harsh in nature, such as biochemical contamination sites, seismic zones, and terrain subject to extreme weather or battlegrounds. The sensors employed in WSNs—which are miniatures embedded computing devices—continue to produce large volumes of streaming data obtained from their environment until the end of their lifetime. It is known that when the battery power in such sensors is exhausted, the likelihood of erroneous data being generated will grow rapidly [1]. Both uncertain environmental factors and the low cost of the sensors may contribute to an intermittent transmission loss and inaccurate measurement. Even when they seldom occur, errors and noises in data streams sensed by a large number of sensors may be misinterpreted as outliers; they frequently trigger false alarms that might either lead to undesirable consequences in critical applications or reduce measurement sensitivity.

Data classification is a popular data mining technique used to determine predefined classes (verdicts) to which unseen data freshly obtained from a WSN map, thereby providing situational information about current events in an environment covered by a dense network of sensors. At the core of the classification technique is a decision tree constructed by a learning algorithm that uses tree-like graphs to model the underlying relations of attributes characterized by the output signals of the sensors to predefined classes. Other alternative algorithms include a support-vector machine, neural network, and Bayesian network algorithms, which offer about the same ability to model nonlinear relations between inputs and outputs. However, decision trees have been widely used in WSNs because of their simplicity and the interpretability of their rules, which can easily be derived from the structure of the tree.

The huge volume and imperfect quality of data streams poses two specific issues applicable to data mining applications, especially for decision trees used in WSNs: problems surrounding model induction and predictive accuracy. A decision tree is constructed by learning from a set of training data, a process in which a local greedy partitioning method is normally used. The training data have to be stationary and bounded in size throughout the learning process. Should new learning data arrive, the learned model must be trained again by processing the whole dataset to update the underlying relations. However, although a single WSN includes a huge number of sensor nodes, each of them has only a limited storage capacity, and it is difficult to accommodate all the training data of the whole network. This implies that data mining can be carried out only at a backend base station that meets storage and computation complexity requirements. Centralized data aggregation gives rise to problems of data synchronization and data consistency, given that the data may come from different sensors randomly distributed over the whole network. Most importantly, retraining a decision tree model requires an ever-increasing degree of latency due to the tremendous volume of data needed. Even if only the latest data are used and old data are discarded, evolving data streams are nonstationary, and very frequent updates in which the model is repeatedly retrained are therefore needed to catch up with the level of prediction accuracy for the current trend.

The second issue is the imperfect quality of the data stream, which clearly affects the prediction accuracy of the decision tree. Noisy data confuse the decision tree with false relations of attributes to classes; such false relations effectively mislead the training algorithm to produce an enormous number of pseudopaths and nodes in the decision tree. Not only do such pseudopaths and nodes degrade accuracy and blunt predictive power, but they also result in problems of tree-size explosions. Though decision tree pruning is a technique commonly employed to remove redundant tree branches and nodes, it surely adds to overall computational complexity and overheads. Given the scarcity of memory space and computational power in WSNs, finding appropriate solutions to alleviate these problems has become an urgent task.

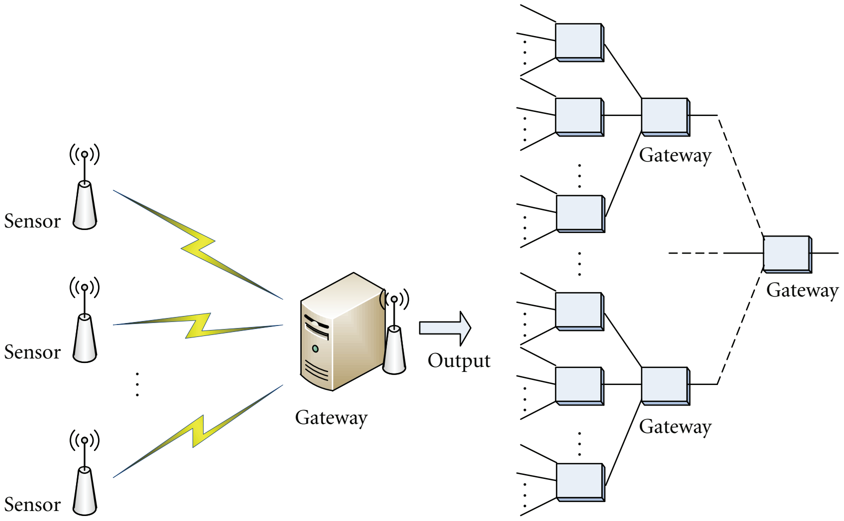
This paper proposes an alternative type of decision tree—the very fast decision tree (VFDT)—to be used in place of traditional decision tree classification algorithms. The VFDT is a new data mining classification algorithm that both offers a lightweight design and can progressively construct a decision tree from scratch while continuing to embrace new inputs from running data streams. The VFDT can effectively perform a test-and-train process each time a new segment of data arrives. In contrast with traditional algorithms, the VFDT does not require that the full dataset be read as part of the learning process, but adjusts the decision tree in accordance with the latest incoming data and accumulated statistical counts. As a preemptive approach to minimizing the impacts of imperfect data streams, a data cache and missing-data-guessing mechanism called the auxiliary reconciliation control (ARC) is proposed to function as a sidekick to the VFDT. The ARC is designed to resolve the data synchronization problems by ensuring data are pipelined into the VFDT one window at a time. At the same time, it predicts missing values, replaces noises, and handles slight delays and fluctuations in incoming data streams before they even enter the VFDT classifier.

To the best of our knowledge, this novel data mining model is the first attempt to alleviate problems of imperfect data in WSNs using a stream mining algorithm and an auxiliary control. This paper makes two key contributions to the literature: it applies stream mining techniques to WSNs by providing an ARC-cache combination to deal with imperfect data streams. The remainder of this paper is organized as follows. Section 2 reviews the technological background to data mining in WSNs and discusses existing methods of handling missing values. An imperfect data stream problem is formulated in Section 3. Section 4 gives a detailed description of our novel VFDT and ARC method and how it can be applied to WSNs. Section 5 details a set of simulation experiments performed using the VFDT and ARC method for both synthetic and real-world datasets. Section 6 concludes the paper.

2. Background

2.1. Data Mining in WSNs. Mining WSN data is said to be constrained by certain limitations and characteristics of WSNs [2]. It first depends on the topology of how a WSN is connected and the purpose of such a setup. Three main topologies have been introduced: (1) the star topology, where a central node is connected to a number of surrounding sensors; (2) the cluster topology, where different stars are interconnected by a central node and the outermost network can be expanded by adding additional stars like a cluster of clusters; (3) the mesh topology, known as an ad hoc topology where nodes and sensors are arbitrarily added and mixed without following any specific pattern. In the WSN literature, a central node known as the sink is the network component that gathers all sensor measurements. The sink usually has greater computational resources than the sensor nodes. As shown in Figure 1(a), Bahrepour et al. [3] proposed a simple "local type" star network characterized by a single sink function that serves as a gateway where collected data are aggregated and data mining is usually performed. The output of a local-type WSN is the set of data mining classification results based on measurements collected from all directly connected sensors. In this type of WSN, it is possible to perform all data mining at the sink. The other type of sink shown in Figure 1(b) is known as a fusion-type sink and resembles a hierarchy of clusters. Ensemble-style data mining is often carried out at each intermediate gateway; a voting method is used to select the best classification result with the highest level of accuracy. Alternatively, the data mining result from each intermediate gateway serves as a local optimum or answer representing its own branch of clusters; the result will then be fed into another downstream cluster as one of the inputs. From the WSN perspective, data mining is conducted at the root of the fusion-type network, with each input taken from each connected cluster that offers a representative output.

This paper focuses on the important WSN task of classification. It is applicable to almost all kinds of WSN



(a) Local type

(b) Fusion type

Figure 1: Local-type and fusion-type wireless sensor network arrangements.

applications, for example, in detecting whether a monitored biomedical patient is suffering from an illness, tracking whether a herd of cattle is moving along the normal route, determining whether a large machine is operating normally, estimating whether a rainforest is growing in balance, or ascertaining whether an anomaly of any kind has arisen in any other type of environment. A decision tree classifier makes predictions or classifications according to predefined classes based on test samples by traversing a tree of possible decisions. WSNs commonly adopt a decision tree method because the trees that represent relations between attributes and classes are informative and intuitively understood. Each path through a decision tree is a sequence of conditions that describe a class. Rules can be derived from such decision tree paths and can be used in a WSN to distinguish an outcome or phenomenon based on measurements observed from sensed data. The simplicity of a decision tree offers useful insights due to the transparent model learning process it follows. The model is learnt by first observing a complete set of training samples. Each sample has several attributes, each of which may be represented by a signal given by a sensor. A sample record may take the form (X, y), where X is a vector of (x_1, x_2, x_n) n attributes and y is a class where the classification problem is to construct a model that defines a mapping function $f : \{X\} \Rightarrow \{y\}$. In contrast, it is difficult to interpret the inner workings of alternative classification methods such as neural network, support vector machine, and regression model approaches [4]. However, one major drawback of the decision tree method is the risk of overfitting, that is, the situation in which a large number of insignificant tree paths grow, usually as a result of being mistrained with many contradicting instances due to the noise in training data. The tree grows tremendously in size and incorrect tree paths adversely confuse the classification results.

2.2. Review of Methods for Handling Missing Values. It is known that a major cause of overfitting in a decision tree is the inclusion of contradicting samples in the model learning process. Noisy data and data with missing values are usually the culprits when contradicting samples appear. Unfortunately, such samples are inevitable in distributed communication environments such as WSNs. Two measures are commonly employed to define the extent of values missing from a set of data [5]: the percentage of predictor values missing from the data set (the value-wise missing rate) and the percentage of observation records that contain missing values (the case-wise missing rate). A single value missing from the data usually indicates a transmission loss or malfunctioning of a single sensor. A missing data value record may result from a broken link between clusters, as in the fusion-type WSN in Figure 1(b). Three different patterns of missing values can occur: missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR) [6, 7]. It is only in the MCAR case that the analysis of remaining complete data could yield a valid classifier prediction according to the assumption of equal distributions [8]. The MCAR pattern occurs when the distribution of an example with a missing value for an attribute does not depend on either the observed data or the missing data. In other words, the assumption made when the MCAR pattern occurs is that the missing and complete data follow the same distribution. In this paper, we are concerned only with the value-wise MCAR type of missing data.

The simplest but not the ideal way to deal with missing values is to discard sample instances with missing values. Alternatively, when the missing values represent only a small percentage of the data set, they can be converted into a new variable. A more commonly adopted method known as imputation is to substitute missing values with analyzed or predicted values. Previous studies have compared the performance of different imputation methods in replacing missing values for a decision tree classifier. To the best of our knowledge, few studies examine how to handle missing data for stream mining types of decision tree classifiers.

An unresolved problem in stream mining research is how to detect concept changes due to noise-infested data and missing values. A streaming ensemble algorithm (SEA) [9] utilizes an ensemble approach in which a majority vote prevails, which is similar to bagging to detect and avoid concept changes in a data stream subject to noise. Another approach is the weighted classifier ensemble (WCE) method [10] in which weighs on different data partitions are carefully adjusted according to the voting decision of the ensemble. Under the WCE approach, previous data are divided into sequential chunks of a fixed size, with a classifier being built from each chunk to improve classification accuracy for the most recent chunk. The last approach of using a flexible decision tree algorithm (FlexDT) [11] facilitates robust data mining with concept drifts and guards against noise-carrying data streams by using fuzzy logic and a sigmoidal function. The drawback of this approach is clearly the longer run time and slow speed due to the use of fuzzy functions. These methods largely require complex and fundamental changes to the central decision tree algorithm; bagging requires that many other trees are grown so the one that yields the best result is solicited from the population of trees. Taking into consideration restrictions on resources and computational power in WSNs, a lightweight approach is likely to be preferred to ensemble-type methods.

3. Formulation of Imperfect Data Stream Problem

This section presents a mathematical model to address the problem of missing values in data stream mining for a sensor network. The assumptions of the model are as follows

(i) The model is defined from the perspective of a centralized data stream mining engine or base station in the WSN where aggregated sensed data are sent to a single classification tree for classification. The mining process runs continuously as the data stream in segments. The length of each data segment is equal to the width of the sliding window. A whole segment will enter the VFDT during each window of time.

(ii) The data stream is characterized by a train of data records. Each record has one or more attributes. Each attribute is assumed to hold a value given by a sensor in the case of a local-type WSN, and the value can come from the sink of a cluster of sensors in the case of a fusion-type WSN. The values for the attributes of a record are assumed to arrive in synchronization across a number of different sensors and/or sinks of clusters. A unique time stamp is added to each record. The time stamps increase in uniform intervals.

(iii) All values for the attributes of the same record are used at the same time (including missing values) for each iteration of the test-and-train process at the VFDT. The name record and instance are used interchangeably.

(iv) The attributes of data records take data of the following formats: nominal, numeric, binary, or mixed.

The original mathematical model for minimizing the overall number of missing values influencing the VFDT within a window size of timeout I? is as follows:

$$\max \sum_{p=1}^{\tau}\sum_{i=1}^{n}\sum_{j=1}^{m_i}\left(q_{ijk} \times \mu_p^i\right). \qquad (1)$$

Because the existence of all X_j^i depends on I_{-p}^i, its mathematical model is simplified as follows:

$$\max \sum_{p=1}^{\tau}\sum_{i=1}^{n}\left(q_{ik} \times \mu_p^i\right), \qquad (2)$$

subject to

$$q_{ijk} = \operatorname{Info}\left(X_j^i\right) = -\sum_{k=1}^{l} \frac{\left|C_{k,X_j^i}\right|}{\left|X_j^i\right|}\log_2\left(\frac{\left|C_{k,X_j^i}\right|}{\left|X_j^i\right|}\right), \quad \forall i, j, \qquad (3)$$

$$q_{ik} = \sum_{j=1}^{m_i} q_{ijk}, \quad \forall i, j, k, \qquad (4)$$

$$q_{ik} \geq q_{ijk} \geq 0, \quad \forall i, j, k, \qquad (5)$$

$$\mu_p^{ij} = \{0, 1\}, \quad \forall i, j, p, \qquad (6)$$

$$\mu_p^{i} = \{0, 1\}, \quad \forall i, p, \qquad (7)$$

$$\mu_p^{ij} = \mu_p^i \quad \forall i, j, p, \qquad (8)$$

$$G\left(x_u^{ij}\right) = \operatorname{Info}\left(X_j^i\right) - \operatorname{Info}_{x_u^{ij}}\left(X_j^i\right), \qquad (9)$$

$$\operatorname{Info}_{x_u^{ij}}\left(X_j^i\right) = \sum_{u=1}^{v_j^i} \frac{\left|x_u^{ij}\right|}{\left|X_j^i\right|} \times \operatorname{Info}\left(x_u^{ij}\right), \qquad (10)$$

$$\operatorname{Info}\left(x_u^{ij}\right) = -\sum_{k=1}^{l} \frac{\left|C_{k,X_u^{ij}}\right|}{\left|X_u^{ij}\right|}\log_2\left(\frac{\left|C_{k,X_u^{ij}}\right|}{\left|X_u^{ij}\right|}\right), \qquad (11)$$

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}}, \qquad (12)$$

$$M = \sum_{i=1}^{n} m_i, \qquad (13)$$

$$\Delta G\left(X_j^i\right) = G\left(x_u^{xy}\right) - G(x_u^{wz}), \quad \forall x, w \in n, \ \forall y, z \in m, \qquad (14)$$

$$\triangle G\left(X_j^i\right) > \varepsilon, \qquad (15)$$

$$\overline{r}_p = \frac{1}{n}\sum_{i=1}^{n}\left(t_p^i - t_{p-1}^i\right), \tag{16}$$

$$\rho_p = \frac{\overline{r}_p - \overline{r}}{\overline{r}}, \quad 0 \le r_p^- \le \tau, \tag{17}$$

$$E = \sum_{1}^{n} q_{ijk} e_i. \tag{18}$$

The objective function (2) represents minimization of the influence of missing values on data stream mining performance probably due to unstable network deliveries of data. Constraints (3), (4), and (5) certify the information entropy of the missing values to the VFDT. The importance of a single attribute is represented by its entropy, and the importance of all attributes in a sink comprising several sensors is the sum of the importance of each single attribute. Importance is constrained to be nonnegative. Constraints (5), (6), and (7) logically restrict the existence of missing values in the same sensor. Formulae (9), (10), (11), and (12) are parts of the VFDT learning process that use information gain (9) as a model updating method by checking the attribute splitting criteria. The Hoeffding bound (12) restricts the instances in which updating occurs; it is only when a particular pattern occurs with sufficient statistical regularity and the Hoeffding bound is exceeded that the decision tree then splits on this node, thereby expanding the tree by growing new branches. Condition (13) concerns the total number of VFDT attributes collected from distributed sensors. Constraints (14) and (15) are the conditions by which the model is updated in the heuristic evaluation. Constraints (16) and (17) are functions for estimating the average arrival rate of the data stream in segments of X collected by the distributed sensors at timestamp p. An average data rate r^- is given to estimate the percentage change in the data rate. Constraint (18) indicates the overall error rate of the ARC.

4. Design of VFDT and ARC in WSN

4.1. VFDT Algorithm. In a stream-based classification, the VFDT decision tree is built incrementally over time by splitting nodes into two using a small amount of the incoming data stream. How many samples have to be seen by the learning model to expand a node depends on a statistical method called the Hoeffding bound or additive Chernoff bound. This bound is used to decide how many samples are statistically required before each node is split. As the data arrive, the tree is evaluated and its tree nodes can be expanded. The following equations essentially depict the building blocks of the stream mining model using the Hoeffding bound. The tree they represent is generally known as the Hoeffding tree (HT), which grows by holding to the Hoeffding bound as a yardstick. The heuristic evaluation function is used to judge when to convert a leaf at the bottom of the tree into a conditional node, thereby pushing it up the tree. Given that a node split occurs when there is sufficient evidence that a new conditional node is needed, replacing the terminal leaf with the relevant decision node better reflects current conditions as represented by the tree rules.

In (19), $G(\cdot)$ denotes the heuristic evaluation function for building a decision tree based on the information gain of an attribute, $\text{Info}(A_j)$. The $\text{Info}(A_j)$ function measures the amount of information sufficient to classify a sample as a node according to the information gain theory. The merit of a discrete attribute's counts n_{ijk} represents the number of samples of class k that reach the leaf, where the attribute j takes the value i which is estimated by collecting sufficient statistics. In (20), P_i is the probability of observing the value of attribute i and $P_{i,k}$ is the probability of observing the value of the attribute i given class k

$$G\left(A_j\right) = \text{Info}\left(\text{samples}\right) - \text{Info}\left(A_j\right) \tag{19}$$

derived from (9):

$$I\left(A_j\right) = \sum_i P_i\left(\sum_k - P_{i,k}\log\left(P_{i,k}\right)\right), \tag{20}$$

$$P_{i,k} = \frac{n_{i,j,k}}{\sum_a n_{a,j,k}}, \tag{21}$$

$$P_i = \frac{\sum_a n_{i,j,a}}{\sum_a \sum_b n_{a,j,b}}. \tag{22}$$

Let us assume that we have a real-valued random variable r with a bounded range of R, which arrives at the n number of independent observations. Equation (12) shows how the Hoeffding bound is computed with a confidence level of $1-\delta$, and the mean of r is at least $r - \varepsilon$. The observed mean of the samples is r'. We assume that the range R has a probability of 1 given that the information gain of R is $\log_2$ class number. The core of the algorithm is the use of the Hoeffding bound to choose a split attribute as the decision node. Let x_a be the attribute with the highest $G(\cdot)$ and x_b be the attribute with the second-highest $G(\cdot)$, such that the difference between the pair of top-quality attributes is defined as $\Delta G = G(x_a) - G(x_b)$.

The VFDT is operated according to a simultaneous test-and-train process, meaning that when a new data segment arrives, the attribute values of the segment will pass down the tree from the root to one of the most likely leaves. In this way, the tree engages in a testing process also known as a classification or prediction exercise based on sample data. At the same pass (traversing through the tree), if the sample data carry a known class y, the model will estimate whether inclusion of the new sample has resulted in the accumulation of sufficient statistics and decide whether a new tree node should be split. This action is on a par with lightweight model learning where the decision tree uses heuristic methods (as in (9) to (14)) to estimate whether the current structure of the tree representing current knowledge needs to be updated. The workflow of the VFDT building process is shown in Figure 2. The model starts from scratch and progressively shapes the tree as new data arrive. The tree building process is essentially different from that followed by the traditional type of decision tree, which requires repeated scanning of the

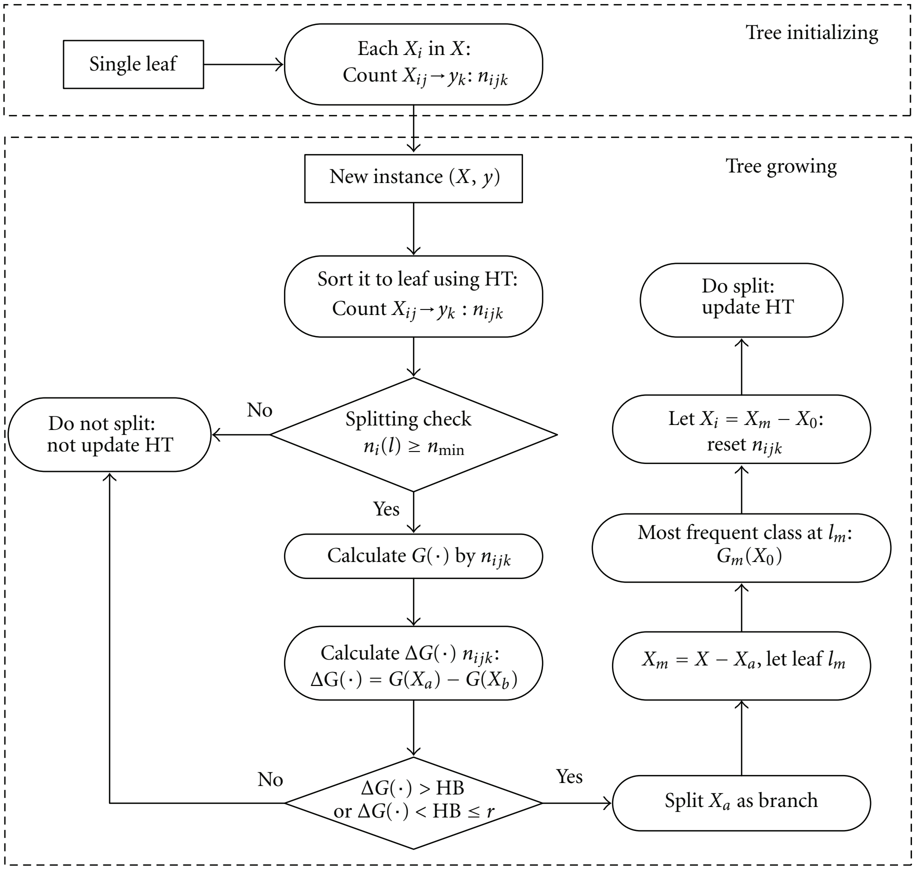

Tree initializing
Single leaf
Each X_i in X: Count $X_{ij} \rightarrow y_k$: n_{ijk}
Tree growing
New instance (X, y)
Sort it to leaf using HT: Count $X_{ij} \rightarrow y_k$: n_{ijk}
Splitting check $n_i(l) \geq n_{\min}$
No
Do not split: not update HT
Yes
Calculate $G(\cdot)$ by n_{ijk}
Calculate $\Delta G(\cdot)$ n_{ijk}: $\Delta G(\cdot) = G(X_a) - G(X_b)$
No
$\Delta G(\cdot) >$ HB or $\Delta G(\cdot) <$ HB $\leq r$
Yes
Split X_a as branch
$X_m = X - X_a$, let leaf l_m
Most frequent class at l_m: $G_m(X_0)$
Let $X_i = X_m - X_0$: reset n_{ijk}
Do split: update HT

Figure 2: A workflow representing the VFDT algorithm tree building process.

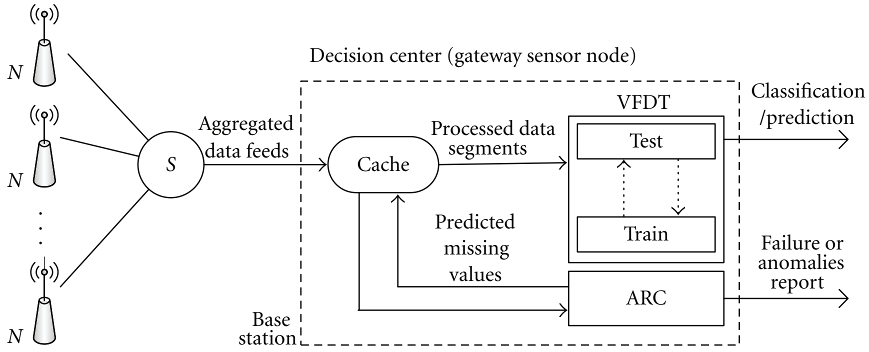


Figure 3: The workflow of the ARC and VFDT in a gateway sensor node.

whole bounded database for training and is inappropriate in a data stream mining environment. The VFDT learns by simply reading through the data stream and checking whether it should further expand its tree nodes. This unique model learning feature makes it a suitable candidate for implementing an autonomous decision maker in WSNs.

4.2. ARC Design. The ARC is a set of data preprocessing functions used to solve the problem of imperfect data streams before they enter the VFDT. The ARC can be programmed as a standalone program which may run in parallel and in synchronization with the test-and-train VFDT operation. Synchronization is facilitated by using a sliding window that allows one segment of data to arrive at a time at regular intervals. When no data arrive, the ARC and the VFDT simply stand still without any action. The operational rate of the sliding window should be no greater than the speed at which the VFDT is operated and faster than the speed at which the WSN sensors transmit data.

When data segments arrive as a stream, one segment at a time will initially be cached. The sliding window closes for a brief moment. While the window is closed, the ARC

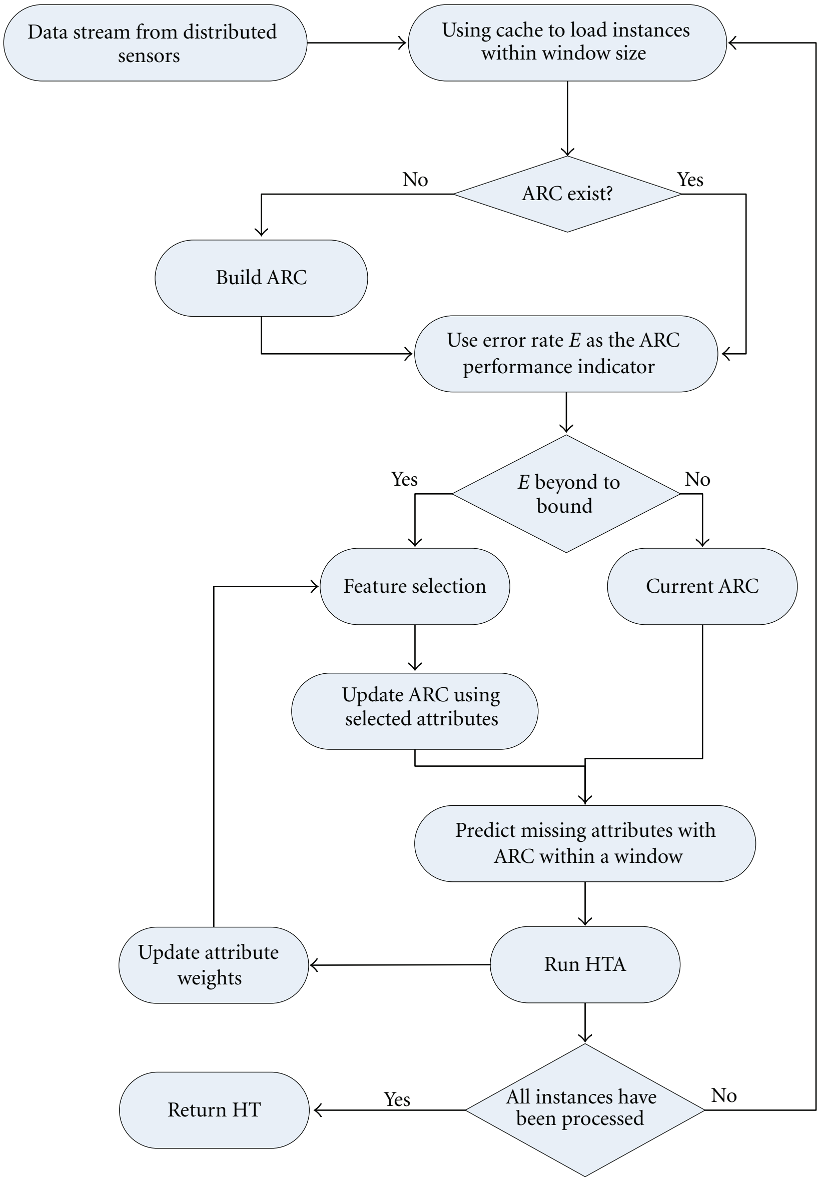

Data stream from distributed sensors
Using cache to load instances within window size
ARC exist?
No
Yes
Build ARC
Use error rate E as the ARC performance indicator
E beyond to bound
Yes
No
Feature selection
Current ARC
Update ARC using selected attributes
Predict missing attributes with ARC within a window
Update attribute weights
Run HTA
Return HT
Yes
All instances have been processed
No

Figure 4: Flow chart of ARC operations.

will attempt to correct four different types of imperfect data (if any) in the cache: missing values, noise, delayed data, and data fluctuations. The correction methods employed for each type are described in the following section. After the data have been manipulated, with missing values guessed on a best efforts basis and noise eliminated, the processed data enter the VFDT for instant testing and training. A class prediction/classification output and a failure anomaly report will then be generated by the VFDT and ARC, respectively. The end user could employ the VFDT output for subsequent decision making if implemented as a final base station, or could feed it into a further cluster of the WSN as an intermediate classification result derived from its own cluster. The failure and anomaly report contains statistics on variables such as the percentage of missing values, noise, delay, and data fluctuations as additional information about the quality of current data traffic. This information could be used as a reference indicator to gauge the reliability of the classification result based on the current quality of the data stream. It could also be used as an alarm signal to alert the network administrator to initiate repairs to the network infrastructure should the statistics in the report show a recurring problem over time. The sliding window will open again when the output results are sent, the data cache will be cleared, the VFDT will have been incrementally trained, and the gateway sensor node will be ready to receive the next incoming segment of data. Only statistics and accumulative counts remain at the ARC and VFDT throughout this continuous operation, thus providing a lightweight operating environment. No historical data need

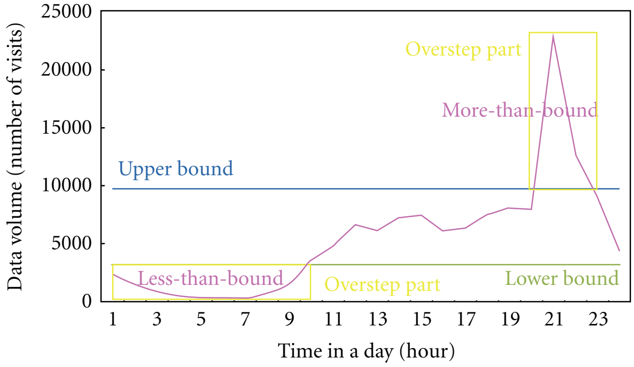


Figure 5: (a) Lateral view of bucket cache—when the ARC-cache $|W| = 5$ is used, batches of 5 data items are processed by the VFDT at a time; because no data are missing in this example, the ARC-cache operates as normal; (b) the bucket cache is used to solve the delayed data stream asynchronicity problem.

to be stored anywhere at this node. Figure 3 is a block diagram showing the major operating features of the decision center.

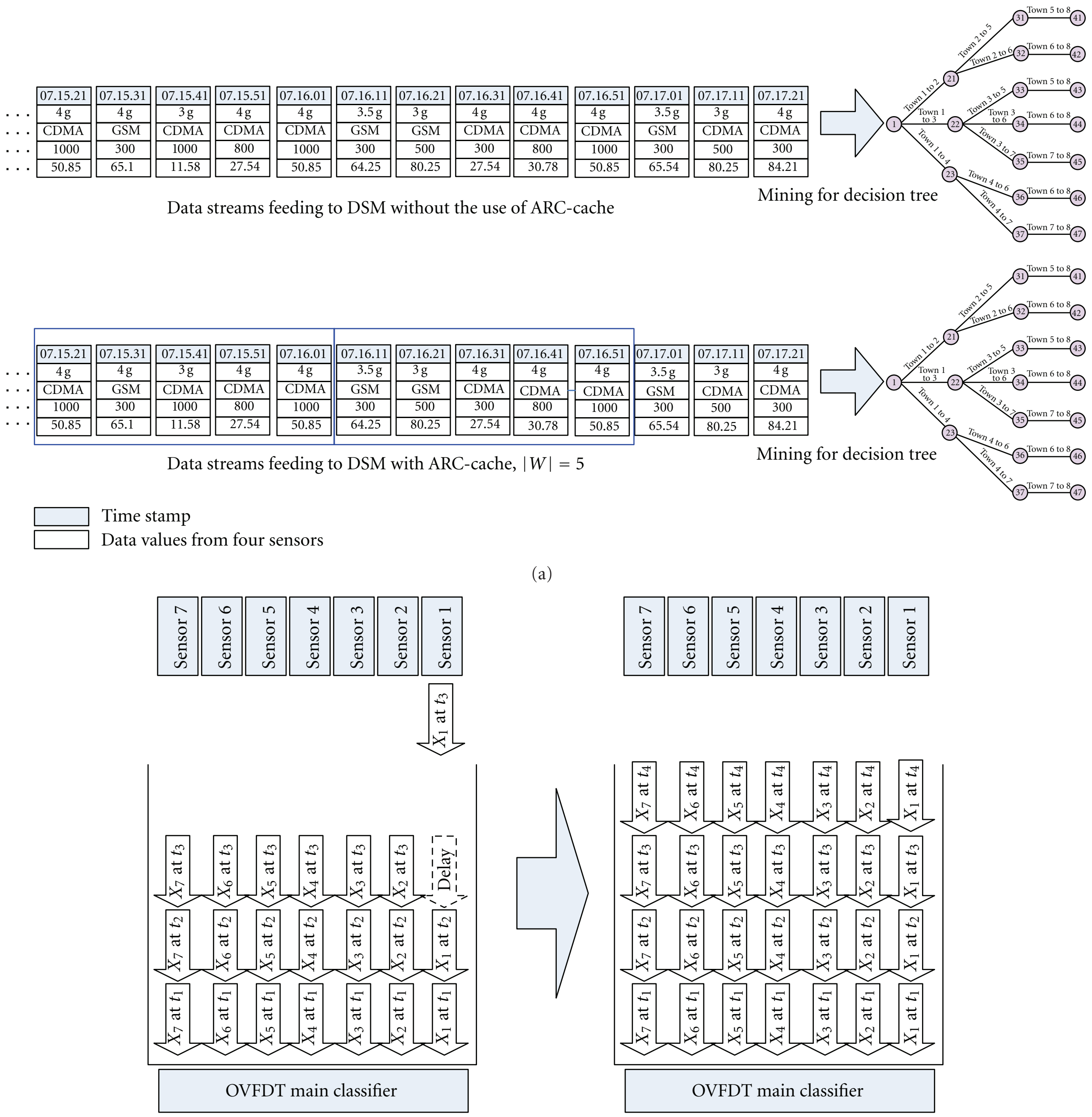
4.3. Missing Data and ARC Noise Estimation. To tackle the problem of missing values in a data stream, a number of prediction algorithms are commonly used to guess approximate values based on past data. Although many algorithms can be used in the ARC that deployed should ideally achieve the highest level of accuracy while consuming the least computational resources and time. Some popular choices we use here for simulation experiments include, but are not limited to, mean, naïve Bayesian, and C4.5 decision tree algorithms for nominal data and mean mode, linear regression, discretized naïve Bayesian, and M5P algorithms for numeric data. Missing value estimation algorithms require a substantial amount of past data to function. For example, before using a C4.5 decision tree algorithm as a predictor for missing values, a classifier must be built using statistics from a sample of a sufficient size.

To further lighten the workload induced by the ARC at the gateway sensor node, the estimation algorithm kicks on only when two conditions are met. First, sufficient statistics must be obtained from past data. Second, the trained classifier (regardless of which algorithms are used) will retrain itself only when prediction error reaches a certain threshold. The ARC therefore registers an error rate e, $0 \leq e \leq 1$, which is the average probability of the ARC misclassifying a randomly drawn test sample. On the other hand, a missing value predictor is needed for each individual attribute of the data segment. To reduce processing time, a missing value predictor is built for qualified (significant) attributes only, and their missing values are predicted on the run. As a rule of thumb, 50% of the most significant attributes contribute to the quality of VFDT classification accuracy. About half of the attributes are therefore selected for the missing value treatment, with missing values for the remaining attributes being recorded as blank to speed up the overall ARC process. The feature selection method based on a principal component analysis is known to be both efficient and effective and is therefore used to rank the most important attributes.

The mechanism adopted for handling noise in the data stream is similar to that used to estimate missing values. Noises are considered to be values far different in range from normal values. A surge or interruption in radio signals along a wireless communication link will bring such values up or down to an extreme. However, because this rarely happens in practice, noise has a low probability occurrence distribution. In our model, we can safely assume that noise is equivalent to an outlier in our data samples because both noise and outliers share the same statistical characteristics. The ARC therefore used an outlier detection algorithm instead of a missing value prediction algorithm to handle noise. However, as argued in [12], traditional outlier detection techniques are not directly applicable to WSNs because of the multivariate nature of sensor data. Janakiram et al. [13] suggest using a Bayesian belief network (BBN) model to identify local outliers in streaming sensor data. In this model, each node trains a BBN at its ARC to detect outliers based on the behavior of its neighbors' readings and its own readings. An observation is considered an outlier if it falls beyond the expected range. Given that the model was shown to work well, it is applied here to detect and replace noise. When a segment of data arrives and fills the cache, the ARC conducts a quick scan to find any outliers. A normal mean is used as a substitute for the values of any outliers in the cache. Figure 4 is a flow chart that shows how the ARC works and retrains its predictor model for the significant attributes only.

4.4. Handling Delay and Fluctuations in Data Caches. Additional buffer space is required to overcome delays and fluctuation problems in data mining in WSNs. The additional buffer is called a bucket, which is a preceding space in front of the cache. The bucket can be implemented in the same gateway sensor node at the outmost interface position between the cache and the sink connector. The function of the bucket is essentially that of a synchronized bulk transfer receptacle that regularly shuttles between the data stream inlet and the data cache. It must operate at specific intervals the frequency of which must be no lower than that of the sliding window. The concept of bucket transfer is analogous to that of a cable car or a lift in a building that carries multiple passengers in bulk. Although passengers (data) can walk into a lift (bucket) asynchronously, they must do so within a certain time limit that has to be shorter than the real-time operational requirement of the lift (VFDT). A slight time latency caused by different sensors can therefore be tolerated. Figure 5 is a visual representation of a bucket. We assume that the ARC that controls the cache is a single entity, the ARC-cache; the bucket is the outermost buffer space that first receives incoming data until it is full. The filled bucket is then loaded into the ARC-cache to fix noise and missing values.

To address the issue of data fluctuations, which is one of the requirements of our imperfect data stream handling model, we propose the use of a lower-bound β_{low} of the data rate change that constrains the data rate from dropping suddenly [14]. Meanwhile, because any dramatic rise in



. . .	07.15.21	07.15.31	07.15.41	07.15.51	07.16.01	07.16.11	07.16.21	07.16.31	07.16.41	07.16.51	07.17.01	07.17.11	07.17.21
. . .	4 g	4 g	3 g	4 g	4 g	3.5 g	3 g	4 g	4 g	4 g	3.5 g	3 g	4 g
. . .	CDMA	GSM	CDMA	CDMA	CDMA	GSM	GSM	CDMA	CDMA	CDMA	GSM	CDMA	CDMA
. . .	1000	300	1000	800	1000	300	500	300	800	1000	300	500	300
. . .	50.85	65.1	11.58	27.54	50.85	64.25	80.25	27.54	30.78	50.85	65.54	80.25	84.21

Data streams feeding to DSM without the use of ARC-cache

. . .	07.15.21	07.15.31	07.15.41	07.15.51	07.16.01	07.16.11	07.16.21	07.16.31	07.16.41	07.16.51	07.17.01	07.17.11	07.17.21
. . .	4 g	4 g	3 g	4 g	4 g	3.5 g	3 g	4 g	4 g	4 g	3.5 g	3 g	4 g
. . .	CDMA	GSM	CDMA	CDMA	CDMA	GSM	GSM	CDMA	CDMA	CDMA	GSM	CDMA	CDMA
. . .	1000	300	1000	800	1000	300	500	300	800	1000	300	500	300
. . .	50.85	65.1	11.58	27.54	50.85	64.25	80.25	27.54	30.78	50.85	65.54	80.25	84.21

Data streams feeding to DSM with ARC-cache, $|W| = 5$

(a)

(b)

FIGURE 6: Smoothing data traffic fluctuations.

the data rate may cause sensor nodes to become congested and VFDT goes timeout, any sudden surge should be suppressed. An upper-bound β_{up} of the data rate change is therefore proposed to prevent timeouts. The upper-bound value is related to the real-time constraint T_R, where $\beta_{\text{up}} \Leftarrow T_R$. If the data rate is lower than β_{low}, an empty pocket in the bucket must be replenished. For this reason, we reviewed previous research on dealing with missing data. Six methods are commonly used to deal with the problem of missing data in data analysis: probabilistic splits, the complete case method, grand mode/mean imputation, separate classes, surrogate splits, and complete variation [5]. In addition to the missing data methods listed above, we have also formulated our own solution whereby when the data rate changes beyond β_{up}, excess data will be saved in a cache and used to replenish missing data when the data rate falls below β_{low}. A snapshot of the traffic smoothing data in the bucket is taken from our experiment and shown in Figure 6. In Figure 6(a), it illustrates the process of ARC-cache that uses a window to cache the data collected from four sensors. Under the ARC-cache with a defined window size (size = 5), the ARC is constructed for every five collected data and

```
$\beta_{\mathrm{UP}}$: Upper Bound                 $\beta_{\mathrm{LOW}}$: Lower Bound
$T$: Unit Time                         $\mathrm{DR}_T$: Data rate at unit time $T$
$N_{\mathrm{CACHE}}$: Instance# saved in Bucket temporarily
$N_T$: Instance# to be used in VFDT at unit time $T$
Replenish $(N)$: return $N$ extract from $N_{\mathrm{CACHE}}$
Initialize $N_{\mathrm{CACHE}}$ = ReplenishMissingData()
FOR $(i = 1; i < T; i++)$
{
  IF $(\mathrm{DR}_i > B_{\mathrm{UP}})$
    $N_i = B_{\mathrm{UP}} \times T_i$;
              // More-than-bound data
    $N_{\mathrm{CACHE}} = (\mathrm{DR}_i - B_{\mathrm{UP}}) \times T_i$;
IF $(DR_i < B_{\mathrm{LOW}})$
    $N_i = \mathrm{DR}_i \times T_i + \mathrm{Replenish}((\beta_{\mathrm{LOW}} - \mathrm{DR}_i) \times T_i)$;
              // Less-than-bound data
    $N_{\mathrm{CACHE}} = N_{\mathrm{CACHE}} - \mathrm{Replenish}((\beta_{\mathrm{LOW}} - \mathrm{DR}_i) \times T_i)$;
ELSE
    $N_i = \mathrm{DR}_i \times T_i$;
}
```

Pseudocode 1: Pseudocode of the traffic smoothing mechanism.

then fills them to build a decision tree. Each data arrives with a timestamp. In Figure 6(b), we give an example to synchronize the delayed arrival data. A cache is like a bucket responsible for collecting those data arriving within a period, which is also referred to the window size. For example, if a piece of data is delayed and the window of bucket is still available in that period, the cache will wait for the delayed data to arrive before the bucket expires. Then the bucket with full data load will be used to update the decision tree as per usual. The corresponding pseudocode of the traffic fluctuation-smoothing algorithm is shown in Pseudocode 1.

5. Simulation Experiments

Simulation experiments are conducted to validate our theoretical model comprising the ARC-cache and the VFDT. The aim of this section is to evaluate the performance of our proposed methods in dealing with missing values in data streams. Several different types of data streams are used in the experiments to facilitate a thorough comparison, including those generated synthetically from data generators and real-life data.

For the simulation, we implement a VFDT program and extend it by incorporating the functions of an ARC model for guessing missing values in data streams. Though the estimation method employed should be generic, the following methods are used in our experiments: the mean, naïve Bayesian, and C4.5 decision tree methods for nominal data, and the mean mode, linear regression, discretized naïve Bayesian, and M5P approaches for numeric data. The simulation system is built with a JAVA open source toolkit called Massive Online Analysis (MOA) stream mining software. The software package comes with a standard data stream generator and a Hoeffding tree algorithm. The data streams used in the experiments are stored in ARFF file format. The run time environment is JAVA JDK 1.5 and WEKA 3.6, and the computing platform is a Windows 7 64-bit workstation with an Intel quad core 2.83 GHz CPU and 8 Gb of RAM.

Table 1: Synthetic datasets.

Name	Attribute number	Attribute Type	Class number	Instance number
LED7	7	Nominal	10	1,000,000
LED24	24	Nominal	10	1,000,000
SEA	3	Numeric	2	1,000,000

5.1. Synthetic Data Stream. An MOA stream generator is used to create a synthetic data stream comprising one million data records. We wrote a customized JAVA software program which randomly adds missing values according to a parameter set by the user—the missing data percentage (MDP). There is another optional control that allows the user to place missing values at either the beginning or the end of the data stream. It is well known that decision trees in stream mining models are unstable in the initial stage of learning. Inserting missing values at this early stage only lengthens the process of training the model to maturity. It may make more sense to observe the impact of missing values after the VFDT model is established and see how it responds to the imperfect stream. The MOA stream generator generates the four different synthetic datasets shown in Table 1. The two LED datasets represent a scenario whereby a WSN has a refined resolution of output classes (10 of them). For example, the data collected by the sensors can reflect a wide range of categorical phenomena. The SEA dataset demonstrates a relatively simple scenario whereby binary outputs such as true or false are derived from ternary types of sensed data.

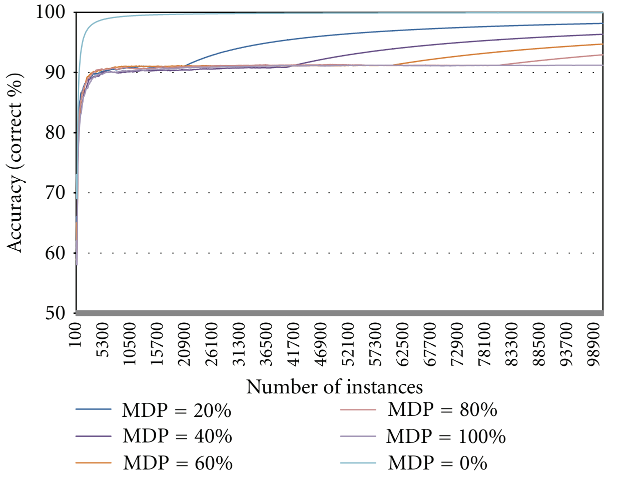


Figure 7: Comparison of accuracy of missing values in the *middle* of a synthetic LED7 data stream.

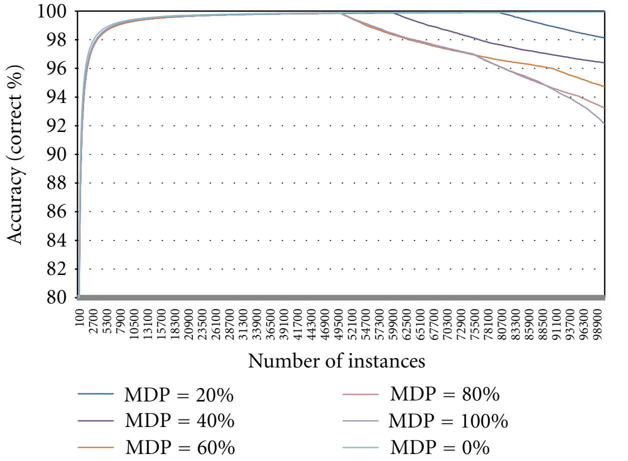


Figure 8: Comparison of accuracy of missing values at the *end* of a synthetic LED7 data stream.

LED7 is a data stream that is simpler than the rest in that it has only 7 nominal attributes. In this experiment, we configure the MDP at 20%, 40%, 60%, 80%, and 100%, which are randomly inserted missing values in defined positions in the data stream. As a result, in comparison with a perfect data stream where MDP = 0%, the higher MDP comes with a lower level of VFDT classification accuracy. Figure 7 presents the VFDT accuracy comparison when missing data are added to the middle of the data stream soon after the model learning process is completed; Figure 8 presents the same comparison but with missing values added at the end. We want to use this set of experiments to show the impact of missing values on data stream mining. Missing values lead to a dramatic reduction in accuracy if the algorithm does not have any ARC mechanism to deal with imperfect data. In other words, the results of these experiments are one of the motivations for this study suggesting stream mining algorithms should be robust to missing values, or accuracy will suffer.

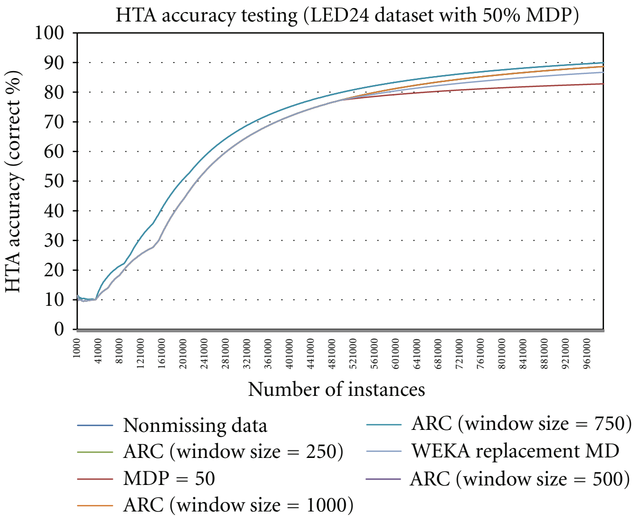


Figure 9: Accuracy of ARC-cache and VFDT in LED24 data stream, missing data added at the *end*.

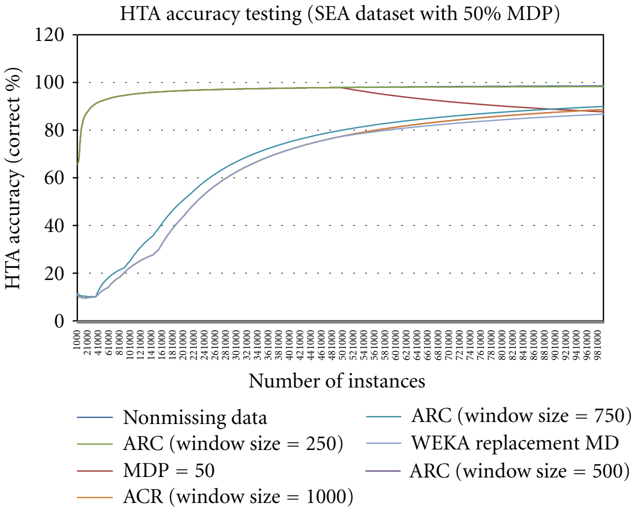
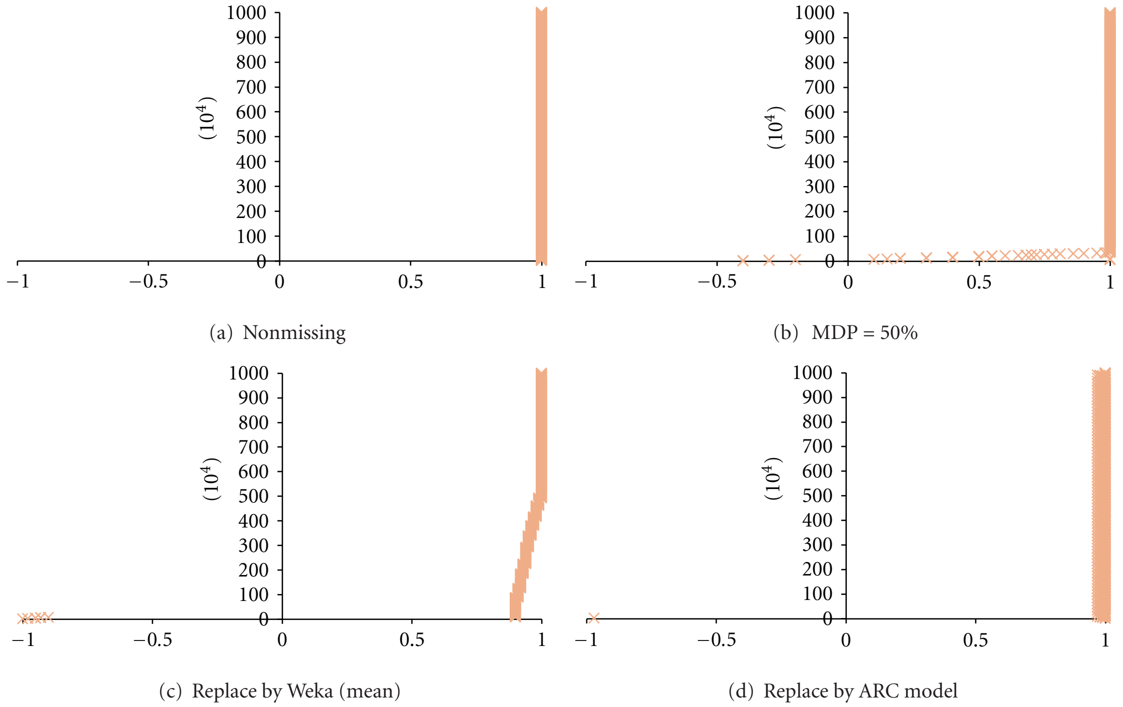


Figure 10: Accuracy of ARC-cache and VFDT in SEA data stream, missing data added at the *end*.

LED24 is a more complicated data stream with 24 nominal attributes and a total of one million instance records. We add MDP 50% to this dataset. To handle imperfect data, the ARC-cache is applied together with the VFDT with different window sizes: 250, 500, 750, and 1000. A C4.5 decision tree function in WEKA is chosen as the ARC construction method in this case. The experimental results shown in Figure 9 show there is a little difference in performance between window sizes in a very large data stream. However, it is also observed that a smaller window size gives a faster ARC-cache and VFDT computing speed. Moreover, we compare the ARC-cache and VFDT results to those of a common missing value solution in which



(a) Nonmissing

(b) MDP = 50%

(c) Replace by Weka (mean)

(d) Replace by ARC model

Figure 11: Margin curves of accuracy testing in SEA dataset.

means are used to replace the same missing values; we find that our proposed method yields better data stream mining performance. The ARC-cache provides a function whereby the missing values replacement performance is checked through a statistical report output. In comparison with the nonmissing values data stream, the results indicate how many correct/wrong number instances arise using different methods to replace the missing values (see Table 2).

SEA is a data stream consisting of 3 numeric attributes, two nominal classes, and one million instances. We also add MDP = 50% of missing values to this dataset. To handle such an imperfect data stream, we use the ARC-cache and VFDT in different window sizes of 500, 750, and 1000. The missing value predictor in the ARC is initially trained up by the M5P function in WEKA. The experimental results shown in Figure 10 show that the ARC-cache and VFDT method yields unsatisfactory results, possibly due to the processing of pure numeric attributes. Nevertheless, the level of accuracy achieved is still slightly higher than that yielded by the mean method.

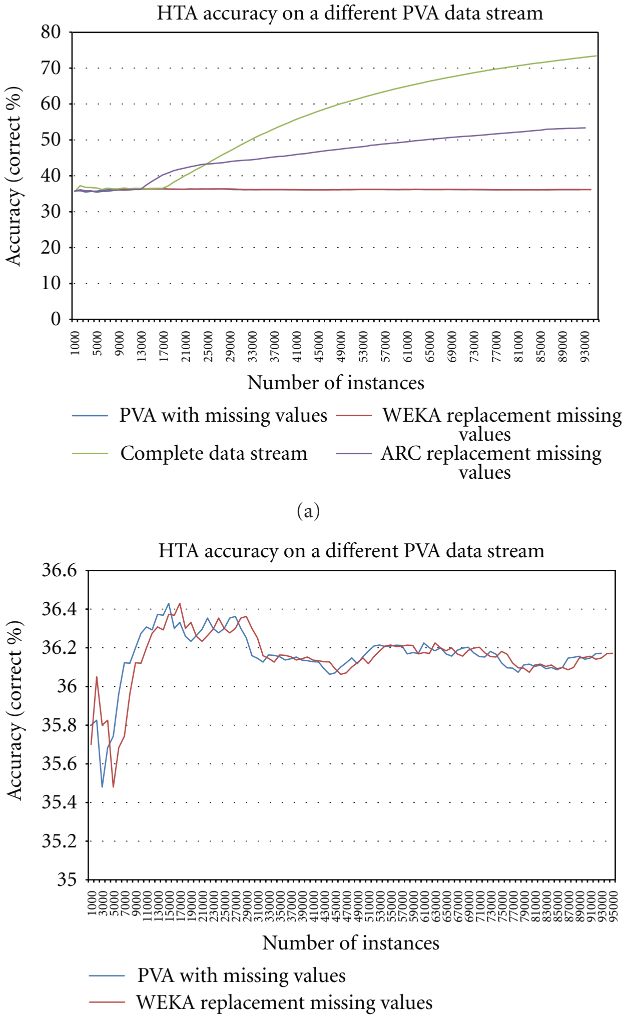
The results are loaded into a margin curve chart visualized in WEKA to evaluate the model generated by a different data stream (as shown in Figure 11). Margin is defined as the difference between the probability predicted for the actual class and the highest probability predicted for the other classes. We observe that the nonmissing data stream has the best performance in (a); while the data stream where MDP = 50% yields the worst performance, including some negative prediction results in (b). Comparing ARC missing values replacement with means computed for missing values replacement, the latter presents some negative predictions, whereas the ARC does not. As a result, we find that the ARC-cache and VFDT may have an MDP bottleneck: when MDP is higher than a certain threshold, the ARC-cache and VFDT method may perform less well. This conclusion may make sense, because when MDP is relatively high (e.g., close to 50%), the VFDT decision tree can simply no longer determine which parts of the data stream are meaningful and which are not, the same can be said for the missing value predictor in the ARC-cache.

5.2. Real-World Data Stream. In this experiment, we use a set of real-world data streams downloaded from the 1998 KDD Cup competition provided by Paralyzed Veterans of America (PVA) [15]. The data comprise information concerning human localities and activities measured by monitoring sensors attached to patients. We use the learning dataset (127 MB in size) with 481 attributes originally in both numeric and nominal form. Of the total number of 95,412 instances, more than 70% contain missing values.

In common with the previous experiment, we compare the ARC-cache and VFDT method with the standard missing values replacement method found in WEKA using means. The results of the comparison are shown in Figure 12. Considering the number of attributes is very large, we apply a moderate window size ($W = 100$) for the ARC to operate. A complete dataset given by PVA is used to test the ARC-cache and VFDT method (115 MB). The experiment results demonstrate that using WEKA mean values to replace missing data yields the worst level of VFDT classification accuracy. Although using the ARC-cache and VFDT method to deal with missing values in the dataset does not yield

Table 2: Missing values replacement performance comparison between ARC-cache and WEKA (standard function for replacing missing values).

Dataset	Wrong instance number	Wrong prediction (%)	VFDT accuracy
MDP = 0%	N/A	N/A	0.899269
MDP = 50%	N/A	N/A	0.828151
WEKA (mean)	210521	42.10%	0.867028
ARC ($W = 250$)	166712	33.34%	0.886413
ARC ($W = 500$)	166679	33.34%	0.886413
ARC ($W = 1000$)	166903	33.38%	0.886413



(a)

(b)

Figure 12: (a) Performance of AC-cache and VFDT missing values replacement method; (b) magnified version of the diagram.

results as accurate as the complete dataset without any missing values, ARC-Cache and VFDT performance is much better than that achieved using WEKA means to replace missing values. The enlarged chart in Figure 12 shows that the WEKA replacement approach has a very little effect in maintaining the level of performance because of the very high percentage of missing data (70%) in this extreme example. Our ARC model also takes a long time to lift the level of accuracy to that required.

Another dataset from the real world is introduced to the experiment. This dataset, which can be downloaded from UCI Machine Learning [16], is called "Localization Data for Posture Reconstruction." Through the observation of tag identification sensors for body location activities, the learning motivation is to classify different human activities. The original dataset has 7 attributes: sequence name (nominal) tag identifier ID (nominal) timestamp (numeric); date (DD.MM.yyyy HH:mm:ss:SSS) x-coordinate of the tag (numeric) y-coordinate of the tag (numeric) z-coordinate of the tag (numeric) and the activity label. The dataset comprises 164,860 instances. It is a typical example of a scenario in which mixed types of data are sensed. We choose linear regression as the method for predicting missing values for numeric attributes in the ARC and employ the naïve Bayesian method to do the same for nominal attributes.

A large amount of missing values is deliberately added to the dataset. The missing completely at random (MCAR) method is chosen for the use in this scenario. Forty percent of the total number of instances (records) is replaced by missing values, meaning 65,944 instances are added completely at random. The distributions of missing values for different attributes are 8,056 (person sequence name) 8,165 (tag identifier sensor) 7,921 (timestamp) 8,051 (date) 8,092 (x-coordinate) 7,943 (y-coordinate) 7,995 (z-coordinate) and 8,141 (activity target).

We observe two significant phenomena from the results shown in Figure 13. The performance gain follows approximately the same trend as the line where no missing value is applied (which is the ideal case). The other interesting finding is that in addition to that for the person-sequence-name and tag-identifier attributes, the accuracy also rises for the rest of the attributes. This result essentially means that ARC predictions for missing values have little effect on identifier-type data which basically carry no meaning other than indexing the records in the dataset. Time-series-type data with date and timestamp attributes show relatively significant improvements in ARC missing value prediction because the regression method chosen as the missing value predictor is best at predicting nonexistent values in a time-series of existing data. In summary, it may be advisable to choose an appropriate missing value predictor for each different type of attribute to obtain the best results.

Regarding the learning speed of the proposed method, we compared the ARC learning speed for each missing value

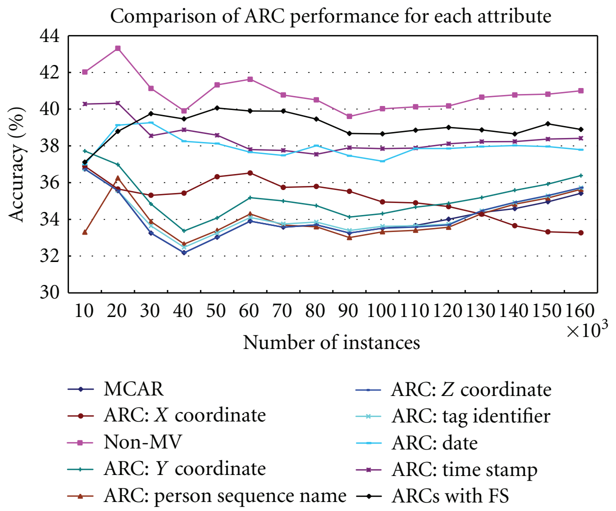


Figure 13: Percentage performance gain realized using the ARC-cache and VFDT method for different attributes in comparison with that achieved with the non-ARC-cache model.

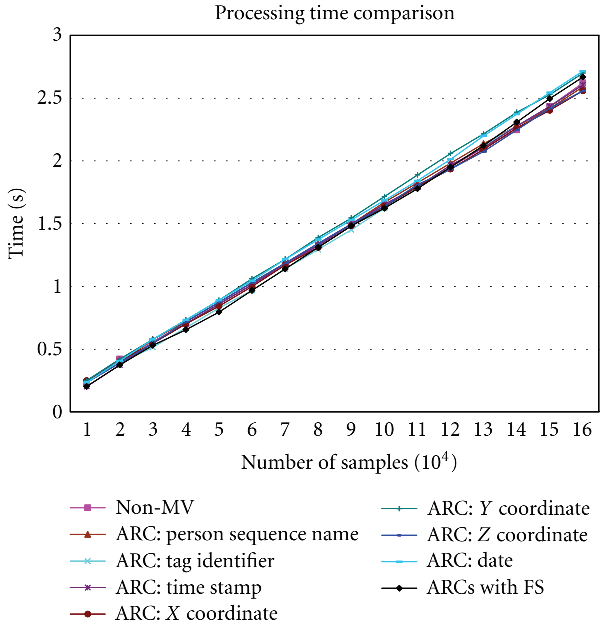


Figure 14: Learning speed comparison using the ARC-cache and VFDT method for different attributes.

in the same test. Figure 14 shows that there is a linear relationship between the learning speed and the amount of samples. With the sample size growing, the time spent on VFDT learning increases, linearly. To verify the suitability of our proposed model for real-time application with respect to learning speed, we investigate the learning time of ARCs coupled with Feature Selection. In Figure 15, suppose the cache size is 10^4 such that $X1 = X2 = X3 = X4 = X5 = X6 = X7$ and suppose Y is the leaning time taken to process each ARC-cache in the VFDT. From the theory of triangle in geometry, it is easy to observe that the $Y1 \approx Y2 \approx Y3 \approx Y4 \approx Y5 \approx Y6 \approx Y7$ approximately in the diagram. In other words, the processing speed of ARC-cache in this example goes stable hence it is suitable for some real-time applications because the increase is linear. This holds true as long as the time-critical requirement is not greater than Y. Consequently that means the real-time application allows time Y as the processing time for the proposed method so that the computing speed will satisfy the real-time requirement.

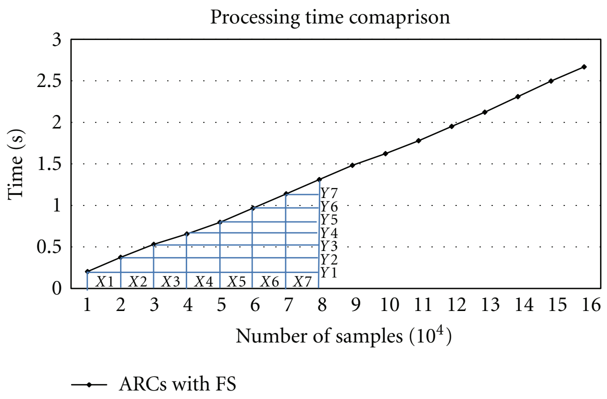


Figure 15: An example that shows the stability of processing time for ARC-cache.

6. Conclusion

The complex nature of incomplete and infinite streaming data in WSNs has escalated the challenges faced in data mining applications concerning knowledge induction and time-critical decision making. Traditional data mining models employed in WSNs work mainly on the basis of relatively structured and stationary historical data and may have to be updated periodically in batch mode. The retraining process consumes time, as it requires repeated archiving and scanning of the whole database. Data stream mining is a process that can be undertaken at the front line in a manner that embraces incoming data streams. We propose using a Very Fast Decision Tree (VFDT) in place of traditional data mining models employed in WSNs due to its benefit of lightweight operation and its lack of a data storage requirement.

To the best of our knowledge, no prior study has investigated the impact of imperfect data streams or solutions related to data stream mining in WSNs, although the preprocessing of missing values is a well-known step in the traditional knowledge discovery process. This paper proposes a holistic model for handling imperfect data streams based on four features that riddle data transmitted among WSNs: missing values, noise, delayed data arrival, and data fluctuations. The model has a missing value predicting mechanism called the auxiliary reconciliation control (ARC). A bucket concept is also proposed to smooth traffic fluctuations and minimize the impact caused by late arriving data. Together with the VFDT, the ARC-cache facilitates data stream mining in the presence of noise and missing values. To prove the efficacy of our model, a simulation prototype is implemented based on ARC-cache and VFDT theories by using a JAVA platform. Experimental results unanimously indicate that the ARC-cache and VFDT method yields a better accuracy in mining data streams in the presence of missing values than in those without. One reason for this improved performance is ascribed to the improved predictive power of the ARC in comparison with other statistical counting methods for handling missing values, as the ARC computes the information gains of almost all other attributes with nonmissing data. In future research, we will continue to investigate the impact of noisy or corrupted data and irregular data stream patterns on data stream mining in WSNs.

Nomenclature

n:	Number of sensors to which the current WSN gateway is currently connecting
m_i:	Number of attributes in sensor i
$I?$:	Timeout parameter, in unit of number of time stamps
l:	Number of class labels in the VFDT
i:	Sensor index
j:	Attribute index
u:	Attribute value index
p:	Timestamp index per complete instance being collected
k:	Class label index
v_j^i:	Number of values of attribute X_j^i
X^i:	Attributes collected by sensor i
X_j^i:	Attribute j in sensor i
C_k:	Target class label k of VFDT classification
x_u^{ij}:	Indexed value of attribute X_j^i
X:	A record X that contains a set of attributes
M:	Number of all attributes in X
t_p:	Time stamp p
$\text{Classifier}(X) \to C$:	The abstract function of VFDT, classifies X sample to class C
N:	Number of samples needed to update VFDT model
$G(\cdot)$:	Heuristic evaluation of split attribute
$\Delta\overline{G}$:	$G(\cdot)^{\text{Best}} - G(\cdot)^{\text{SecondBest}}$
ε:	Parameter to justify whether it is time to update VFDT model, when $\Delta\overline{G} > \varepsilon$
R:	Range of $G(\cdot)$
$1-\delta$:	The confidence of VFDT model update.

Decision Variables

q_{ijk}: The importance of attribute X_j^i to C_k

q_{ik}: The importance of sensor i to C_k

I_{-p}^{ij}: A binary variable taking the value of 0 if X_j^i is missing at timestamp p and 0 otherwise

I_{-p}^{i}: A binary variable taking the value of 0 if sensor i is disconnected at timestamp p and 0 otherwise.

Acknowledgments

The authors are thankful for the financial support from the research grant "Real-time Data Stream Mining," Grant no. RG070/09-10S/FCC/FST, offered by the University of Macau, FST, and RDAO.

References

[1] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Online outlier detection in sensor data using non-parametric models," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)*, pp. 187–198, Seoul, Korea, September 2006.

[2] B. Lantow, "Impact of wireless sensor network data on business data processing," in *Proceedings of the Forum Poster Session in Conjunction with Business Informatics Research (BIR '05)*, Skövde, Sweden, 2005.

[3] M. Bahrepour, N. Meratnia, Z. Taghikhaki, and P. Havinga, "Sensor fusion-based activity recognition for Parkinson patients," in *Sensor Fusion—Foundation and Applications*, pp. 171–191, InTech.

[4] K. P. Lam, M. Höynck, B. Dong et al., "Occupancy detection through an extensive environmental sensor network in an open-plan office building," in *Proceedings of the 11th International IBPSA Conference*, pp. 1452–1459, Glasgow, Scotland, July 2009.

[5] Y. Ding and J. S. Simonoff, "An investigation of missing data methods for classification trees applied to binary response data," *Journal of Machine Learning Research*, vol. 11, pp. 131–170, 2010.

[6] K. Lakshminarayan, S. A. Harp, and T. Samad, "Imputation of missing data in industrial databases," *Applied Intelligence*, vol. 11, no. 3, pp. 259–275, 1999.

[7] R. J. Little and D. B. Rubin, *Statistical Analysis with Missing Data*, Wiley, New York, NY, USA, 1987.

[8] A. Farhangfar, L. Kurgan, and J. Dy, "Impact of imputation of missing values on classification error for discrete data," *Pattern Recognition*, vol. 41, no. 12, pp. 3692–3705, 2008.

[9] W. Nick Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 377–382, San Francisco, Calif, USA, August 2001.

[10] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 226–235, Washington, DC, USA, August 2003.

[11] S. Hashemi and Y. Yang, "Flexible decision tree for data stream classification in the presence of concept change, noise and missing values," *Data Mining and Knowledge Discovery*, vol. 19, no. 1, pp. 95–131, 2009.

[12] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: a survey," *IEEE*

Communications Surveys and Tutorials, vol. 12, no. 2, pp. 159–170, 2010.

[13] D. Janakiram, V. A. Mallikarjuna Reddy, and A. V. U. P. Kumar, "Outlier detection in wireless sensor networks using Bayesian belief networks," in *Proceedings of the 1st International Conference on Communication System Software and Middleware*, pp. 1–6, 2006.

[14] Y. Hang and S. Fong, "Stream mining over fluctuating network traffic at variable data rates," in *Proceedings of the 6th International Conference on Advanced Information Management and Service (IMS '10)*, pp. 436–441, Seoul, Korea, November 2010.

[15] The Second International Knowledge Discovery and Data Mining Tools Competition, Sponsored by the American Association for Artificial Intelligence (AAAI) Epsilon Data Mining Laboratory Paralyzed Veterans of America (PVA), http://www.kdnuggets.com/meetings/kdd98/kdd-cup-98.html.

[16] A. Frank and A. Asuncion, UCI Machine Learning Repository, Irvine, Calif, USA, University of California, School of Information and Computer Science, 2010, http://archive.ics.uci.edu/ml.