Article

A Novel Sensor Data Pre-Processing Methodology for the Internet of Things Using Anomaly Detection and Transfer-By-Subspace-Similarity Transformation

Yan Zhong ^{1,†}, Simon Fong ^{2,*,†}, Shimin Hu ^{2,†}, Raymond Wong ^{3,†} and Weiwei Lin ^{4,†}

¹ Department of Big Data and Cloud Computing, Zhuhai Institutes of Advanced Technology of the Chinese Academy of Sciences, Zhuhai 519000, China; mb75500@um.edu.mo

² Department of Computer and Information Science, University of Macau, Taipa 999078, Macau; yb77401@um.edu.mo

³ School of Computer Science & Engineering, University of New South Wales, Sydney 2052, Australia; wong@cse.unsw.edu.au

⁴ School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China; linww@scut.edu.cn

* Correspondence: ccfong@umac.mo; Tel.: +853-8822-4460; Fax: +853-8822-2426

† These authors contributed equally to this work.

Received: 15 August 2019; Accepted: 2 October 2019; Published: 18 October 2019

Abstract: The Internet of Things (IoT) and sensors are becoming increasingly popular, especially in monitoring large and ambient environments. Applications that embrace IoT and sensors often require mining the data feeds that are collected at frequent intervals for intelligence. Despite the fact that such sensor data are massive, most of the data contents are identical and repetitive; for example, human traffic in a park at night. Most of the traditional classification algorithms were originally formulated decades ago, and they were not designed to handle such sensor data effectively. Hence, the performance of the learned model is often poor because of the small granularity in classification and the sporadic patterns in the data. To improve the quality of data mining from the IoT data, a new pre-processing methodology based on subspace similarity detection is proposed. Our method can be well integrated with traditional data mining algorithms and anomaly detection methods. The pre-processing method is flexible for handling similar kinds of sensor data that are sporadic in nature that exist in many ambient sensing applications. The proposed methodology is evaluated by extensive experiment with a collection of classical data mining models. An improvement over the precision rate is shown by using the proposed method.

Keywords: Internet of Things; sensor data; preprocessing; subspace similarity

1. Introduction

The infrastructure of the Internet of Things (IoT) is establishing rapidly recently, with the hype of smart cities over the world. Many ambient-sensing applications subsequently were developed that tap into the maturity of IoT technology [1]. From these applications and the proliferation of sensor equipment and ubiquitous communication technologies, we are able to collect a huge amount of useful data, which was not possible before [2,3].

As the Figure 1 shows, there are some problem in the internet of things. For example, a typical and challenging ambient sensing application namely human activity recognition (HAR) collects data about

human activities, such as walking, running and standing, in a confined space [4]. HAR tries to make sense of a massive amount of data that is collected continuously for analyzing what a person is doing over a long period of time. Another typical application of ambient sensing focuses on detecting unusual environmental occurrences by using many outdoor sensors such as atmospheric sensors [5]. Although sensors have become ubiquitous and their applications are prevalent covering every aspect of our life, the data mining component in an IoT system has to keep up its effectiveness with the sheer volume of dynamic data. Sensor data are data feeds which exhibit some patterns when they are zoomed out and viewed longitudinally. The patterns are somewhat different from the structured datasets that are used traditionally for supervised learning. Earlier on, some researchers advocated that this kind of sensor data has unique characteristics, and it is unsuitable for use directly in data analysis [6]. First, the sensor data collected by the sensor are usually in numerical values. Second, the frequency of collection is relatively quick, and a lot of data can be collected within a few seconds at a time, depending on the sampling rate. Third, the adjacent data cells along a sensor data sequence may be very similar to each other without any significant changes in values over a period of time. For instance, a sensor that is tasked to monitor the activity of a sleeping baby or a sitting phone operator: their body postures change relatively little in slow or still motions; the same goes for monitoring the humidity of a forest or agriculture in fine and stable weather [7]. Fourth, sensor data may be corrupted by noise and losing useful information due to various transmission errors or malfunctioning sensors [8]. It is not uncommon for sensors that collect bogus data due to unreliable medium or external interference in a large scale IoT network deployed in harsh environment.

From the perspective of data mining, the process may not know whether a piece of data is noisy or a plain outlier, until and unless the fault is pinpointed. For example, when a sensor malfunctions, it sends incorrect readings to the server. The wrong data will not be discovered though they are more than useless for training a data mining model. As a result, the induced model is degraded by junk data. The wrong prediction outputted from the data mining model would propagate to throughout the IoT network and eventually to the final [9].

Figure 1. Kinds of problem in the Internet of Things.

Basically, the characteristic of the IoT data collected by sensors are sequential and huge. The data values are largely repetitive and noises that have irregular values may easily go unnoticed in the data. Many users would aggregate the data for deriving descriptive statistics such as mean and distributions. Relatively they pay less attention to individual pieces of data at the micro-scale. It is known that sensor data are often generated at narrow intervals at the high sampling rate. Some problematic data that have gone unnoticed will cause the classifiers drop in performance during operation. At the data level, it is difficult for the model induction process to know whether an incoming training data will be counter-productive

to the supervised learning. This may require post-processing or feedback-learning. A simpler method is pre-processing, which is usually fast and lightweight suitable for dynamic IoT scenarios.

Some popular pre-processing methods, such as Principle Component Analysis (PCA) [10] only change the original data by reducing the attribute dimension. Dimension reduction may not be so effective here because sensor data may have only a few attributes about the sensor readings. Sensor data are time-series that come as training data feed in sequential manner. Some regularization methods [11] only change the scope of the data, and they cannot effectively isolate outliers from a data feed of mediocre values. Changing the time interval of the instance by using a simple sliding window and statistical methods is possible and straightforward, but it could be further improved. Some efficient and effective pre-processing mechanism is required for upholding the performance of a trained model. For solving the problem of “repetitive and redundant data” in an IoT situation, a novel pre-processing methodology is proposed in this article. In this paper, the proposed pre-processing part of a classification model calculates the probability between the subspace of source sequential data and the target data. This model transforms the subsequent input data into probability data in a period by the length of sliding windows that controls the time interval hence the resolution. It focuses on the data features while maintaining the data original structures.

The contributions of this paper are as follows: (i) a pre-processing method suitable for sensor data that may have persistent and redundant data values is proposed. The method converts the original data values into probabilities that are computed based on the similarity of subspace; (ii) The pre-processing method could set the size of the subspace and the length of the sliding window, and it can effectively combine the needs of the time segment analysis in the real task. (iii) The advantages of proposed pre-processing mechanisms can combine perfectly with different models, reducing the sensitivity to noisy data and redundancy problems. The precision of the classification model will be improved by using this pre-processing method, rather than directly using classification algorithms alone. The source code of this new pre-processing methodology can be downloaded for testing and verification at <https://github.com/Ayo616/TBSS>.

In Section 2, some related work about the pre-processing methods that are applicable for sensor data are introduced. In Section 3, the proposed pre-processing methodology is explained in detail. In Section 4, validation experiment is designed for evaluating the performance of our method. Empirical datasets are used in the comparison of other methods. In Section 5, the work is concluded.

2. Literature Review

With the advancement of communication technologies and, electronic sensing devices are increasing, sensor-centric IoT technologies and applications experienced rapid growth. From recent statistics [12], by 2017 the total value of the IoT industry reached 29 billion U.S. dollars. This huge market has attracted attention from both practitioners and academic researchers.

Over the years, some companies have been developing and building IoT smart systems such as smart homes, smart transportation, and smart security [13–15]. These technologies are aimed at solving the specific problem for a relatively pure dataset in Internet of Things.

The massive deployment of IoT devices helps people to obtain large amounts of sensory data. How to tap valuable information from this vast amount of data and form knowledge to serve life more effectively is an important issue [16]. Some researchers have tried to use current data mining technology in the development of the Internet of Things to make the Internet of Things more intelligent [17–20]. These methods under the IoT framework could usefully solve some parts of the RFID mining task problem.

Clustering is commonly used in data mining of the Internet of Things. The most common clustering method is K-means [21]. K-means is very mature in traditional data mining. It divides a dataset into several clusters. K-means have been widely used in unsupervised classification task. The distribution of

Internet of Things data in some cases is a clustering problem [22,23], but the classification results presented by clustering are only similar data and cannot be judged. If people are unfamiliar or unclear with the collected data, they cannot rely on the clustering results to precisely dig out effective knowledge.

In supervised learning, people often use decision tree algorithms for data mining of the Internet of Things [24,25]. In addition, probabilistic models are also widely used, such as the Naive Bayesian model [26,27]. In machine learning, there is also a simple and efficient classification method that is SVM [28], combining with the kernel function can linearly separate data in high-dimensional space. SVM methods are suitable for small dataset due to its long learning time. Finding a hyperplane in SVM is time-consuming because it iteratively tries and searches for the most appropriate non-linear division among the data.

However, the traditional classification model shows an unstable performance on the actual sensor data. That is because of the unique nature of the Internet of Things data. Through analysis and observation, all collected sensor data have highly repetitive characteristics and often contain noisy data. The source of these noise data may be due to sensor detection errors. That leads to too many samples of negative instances in the classification process, and the accuracy of the training model will decrease. In addition, in real life, people pay more attention to the results of a period, which is inconsistent with the phenomenon of collecting data in seconds.

Some research [10] used pre-processing of the dimension to deal with the sensor data problem. It is advantageous for tackling the dimension explosion problem. However, the useless features will increase the amount of calculation and it is not beneficial for the model's precision. So the performance of these methods is not very good in sensor data mining because this type of methods only affects the feature of instance [11]. Sensor data have a time-series feature, so some research uses sliding windows as pre-processing methodology to adjust the time interval and do statistical computation within the windows. This way is straightforward and easy to implement but still affected by noisy data. Its contribution to the performance of the model is limited [29,30] because it depends on the data purity and special requirement of task.

Our intuition for this method is that a subspace is corresponding to a behavior state. No matter how the redundant in data, the corresponding space is limited. We could detection the similarity of space to identify the status. Therefore, we propose a new type of data pre-processing methodology called Transfer by Subspace Similarity (TBSS). TBSS constructs subspace from the initial dataset. Our proposed TBSS methodology combines an anomaly detection algorithm that derives a probability table, and finally using just a traditional classification method for classification. TBSS pre-processing method can significantly improve the precision of classification. Moreover, this pre-processing methodology can be combined with various classification methods and anomaly detection methods, as well as it is flexible for real-time activity detection.

3. Methodology

3.1. Problem Definitions

Suppose we have a training dataset D_{train} , $d_i^j = (x_i^j, c_n)$, $d \in D_{train}$, it includes original sequence training data. These data are all from the sensors.

The training dataset includes different labels. The collection of these labels is $C = \{c_1, c_2, \dots, c_n\}$, $c_n \in C$. These labels indicates the the status of object at a moment.

Each instance in dataset has a set of features. $x_i = \{x_i^1, x_i^2, \dots, x_i^j\}$, $x_i \in X$. The collection of these features without label is $X = \{x_1, x_2, \dots, x_i\}$.

Our task is to design an algorithm to deal with these sequence data in order to improve the data mining quality in the Internet of things. So we need to transfer the dataset D to the $\hat{D}$, $\hat{d} = (x_e^n, c_n)$, $\hat{d} \in \hat{D}$.

$$D \xrightarrow{TBSS} \hat{D}$$

3.2. Reconstruct Training Data Table

In this step, we deal with the original training data table. We could divide some group from the original dataset according to the different labels. We get collection τ based on different class.

$$\tau = \left\{ T_0^{C_m} \right\}_{m=1}^n \quad (1)$$

Next we construct a sub-dataset Δ from the τ and the sample size of the Δ is z_Δ . The τ represent the datum of a space.

$$\Delta = \left\{ ST_0^{C_m} \right\}_{m=1}^n \quad (2)$$

We construct a sub-dataset Γ from the τ , and the sample size of Γ is z_Γ . The Γ is used to compare with the τ .

$$\hat{\Gamma} = \left\{ L_0^{C_m} \right\}_{m=1}^n \quad (3)$$

We also construct a noisy dataset $\phi = \{ Nos^{C_m} \}_{m=1}^n$, and we add this ϕ to the Γ .

$$\Gamma = \hat{\Gamma} + \phi \quad (4)$$

All this sampling method does in the above operation is randomly select data. When we constructed these sub-datasets, we could compute some information between them as new features. Here we define some definitions. First we define TBSS function (transfer by subspace similarity).

Definition 1. Function $TBSS_{train}$ is defined to generate a new training dataset T_1 from T_0 with the setting of z_Γ (the length of $L_0^{C_m}$) and z_Δ (the length of $ST_0^{C_m}$).

$$TBSS_{train} : SP \times IR \times IR \rightarrow SP'$$

$$TBSS_{train}(\tau, z_\Delta, z_\Gamma) = \hat{D}_{train}$$

The new table is combined as follows, and the $\hat{x}_e$ is new instance for a new table.

$$T_1 = \{ \hat{x}_1, \hat{x}_1, \dots, \hat{x}_e \}$$

The features of instance in new table is computed by the function TDT.

$$\hat{x}_e = (\{ [TDT]_{e,m} \}_{m=1}^n, c_i)$$

$\hat{x}_e$ means we construct a new training data in $\hat{D}_{train}$. The new attributes are a group of non-isolation rates and class of this instance is c_i . A group of non-isolation rates is computed by function ITR () for whose input is a combination of $(L_0^{C_i}, ST_0^{C_j})$, $1 \leq i \leq n$ and $1 \leq j \leq n$ and $z_\Gamma < z_\Delta$.

Definition 2. Construct a train data table $TDT \in [[0,1]^{n \times n} \mid C^{n \times 1}]$ and the element in the i_{th} row and j_{th} column of TDT is computed by function $ITR(L_0^{C_i}, ST_0^{C_j})$ and the element in the k_{th} row and the $(n+1)_{th}$ column is $C_k, 1 \leq i \leq n$ and $1 \leq j \leq n$:

$$\begin{aligned} [TDT]_{i,j} &= ITR(L_0^{C_i}, ST_0^{C_j}) \\ [TDT]_{k,n+1} &= C_k \quad \text{for } 1 \leq k \leq n \end{aligned} \quad (5)$$

Δ and Γ are sub-datasets from the τ , so the Δ and Γ are subspaces of the τ . We use non-anomaly attribute to represent the similarity of two spaces.

Definition 3. Function $ITR()$ used X as standard case to train an isolation forest which is an algorithms created by Prof. Zhi Hua Zhou [31] for detecting the isolation point and then put the Y sample set (detection case) into the isolation forest model to classify whether there are isolation points or not in the Y . Finally, computing the rate P of data in Y that normally obeys the distribution in X . In other words, this function is to compute the non-isolation rate of Y .

$$ITR : SP \times SP \rightarrow [0,1], \quad ITR(Y, X) = P \quad (6)$$

3.3. Reconstruct Test Data Table

Suppose we have a test dataset. We only use the label of test dataset in evaluation process. We extract the features from each instance. The collection of test dataset is $\hat{d} = (y_i^j, c_n), \hat{d} \in \hat{D}_{test}$.

We still apply some transfer processes to the D_{test} , transforming $\hat{d} = (w_r^j, c_n)$, where $\hat{d} \in \hat{D}_{test}$.

A sliding window is used to construct the sub-dataset. The length of the sliding window is p . p is same as z_Γ . So the original dataset could be transferred to the $w_z = \{y_{1+z}, y_{2+z}, \dots, y_{p+z}\}$, where r is determined by the length of sliding window p and length of test dataset.

Definition 4. Construct a test data table $TDT_1 \in [0,1]^{r \times n}$:

$$[TDT_1]_{t,j} = ITR(w_t, ST_0^{C_j}) \quad (7)$$

where $ST_0^{C_j} = Sam_{md}(T_0^{C_j}, z_\Delta), 1 \leq i \leq n$ and $1 \leq t \leq T$. w_t is the t_{th} sliding window, while the element in the t_{th} row and j_{th} column of TDT_1 is computed by function $ITR(w_t, ST_0^{C_j})$.

With the help of TDT_1 the final high level function of step two can be defined as follows.

Definition 5. Function $TBSS_{test}$ is defined for transforming testing dataset D_{test} into new testing dataset $\hat{D}_{test}$ which has same categories of attributes as T_1 . But part of computing input changes because we no longer use τ to gain sample set $L_0^{C_i}$ whose size is z_Γ but using sliding windows of D_{test} while the class of new testing data is replaced by the Maj_{w_t} which is major class of a sliding window.

$$TBSS_{test} : SP \times SP \times IR \rightarrow SP''$$

$$TBSS_{test}(\tau, z_\Delta, p) = \hat{D}_{test}$$

where

$$\hat{D}_{test} = \{w_1, w_2, \dots, w_r\}$$

and for $1 \leq t \leq r$,

$$w_t = (\{[TDT_1]_{i,j}\}_{j=1}^n, Maj_{w_t}) \quad (8)$$

3.4. Step 3: Model Learning

After using the pre-processing method to generate the new training dataset D_{train} and new testing dataset D_{test} , user could apply different algorithms to make the prediction with the help of D_{train} and D_{test} . Then a group of high level equations that represent these processes is defined as follow:

Definition 6. $\{algo_i\}_{i=1}^5$ is defined for a group of high level equations that use training data set (such as D_{train}), testing dataset (such as D_{test}) as well as a group of parameters for training the model and testing the model. Finally the performance evaluation index pf is obtained.

$$algo_i : SP' \times SP'' \times Pre \rightarrow IR$$

$$algo_i(D_{train}, D_{test}, P) = pf \quad (9)$$

where Pre is a collection of all possible specific parameters with respect to the demand of user and $P \in Pre$, pf is a performance evaluation index which is a combination of several statistical parameters of model such as accuracy, recall and $f1$ – score. In this paper, there are five classical classification algorithms being tested, they are SVM, logistic regression, C4.5, Bayes classifier and KNN.

In TBSS, the probability between each subspace similarity is calculated. If the subject is walking, of course the data collected under walking status belong to the statistical distribution of walking training dataset. It is known that walking is a homogenous activity without any strange behaviour. So we should obtain a same data space from the walking test data. Time-consumption depends on the time spent on constructing the subspace and the distribution of the original space. If the original space is large (for example, we can possibly have different styles of lying down behavior), the sampling times will be large too because we need to ensure the coverage rate for the original space. If the original space is small (for example we only have a simple kind of walking behavior), the sampling times will be short and the cost time will be low too. Figures 2 and 3 show the TBSS's process details as well as we introduced the pseudo code of TBSS Algorithm 1 in detail.

After comparing the performance between before and after pre-processing, we found that this new method actually could improve the accuracy of the prediction. The following section will show the experiment results in detail.

Figure 2. Process of transferring raw training dataset to new training dataset.

Figure 3. Process of transferring raw test dataset to new test dataset.

Algorithm 1: TBSS Algorithm

Input: $D_{train}, d_i^j = (x_i^j, c_n), d \in D_{train}, D_{test}, w_i^j = (w_i^j, c_n), w \in D_{test}$ and Parameters are Γ and Δ

Output: $\hat{D}, \hat{d} = (x_e^n, c_n), \hat{d} \in \hat{D}$

- 1 Preliminary Work For Construct Training Data Table And Test Data Table;
- 2 **for** each instance in D_{train} , identify class **do**
- 3 $\tau = \{T_0^{C_m}\}_{m=1}^n$;
- 4 **for** Each Epoch select subspace in τ , according to the size of space z_Δ **do**
- 5 $\Delta = \text{Sample}(D_{train}, z_\Delta) \Delta = \{ST_0^{C_m}\}_{m=1}^n$;
- 6 **for** Each Epoch select subspace in τ , according to the size of space z_Γ **do**
- 7 $\hat{\Gamma} = \text{Sample}(D_{train}, z_\Gamma) \hat{\Gamma} = \{L_0^{C_m}\}_{m=1}^n$;
- 8 To strengthen algorithm's robust, define the ϕ noisy;
- 9 $\phi = \{Nos^{C_m}\}_{m=1}^n$;
- 10 $\Gamma = \phi + \hat{\Gamma}$;
- 11 Training Data Table Process;
- 12 **for** $i = 1, i \leq n, i++$ **do**
- 13 $\Delta = \{ST_0^{C_m}\}_{m=1}^n$, according to the z_Δ ;
- 14 **for** $j = 1, i \leq n, j++$ **do**
- 15 $\Gamma = \{L_0^{C_m}\}_{m=1}^n$, according to the z_Γ ;
- 16 $\hat{D}_{train} \leftarrow \text{calculateSimilarity}(\Gamma, \Delta)$;
- 17 Test Data Table Process;
- 18 **for** $i = 1, i \leq n, i++$ **do**
- 19 $\Delta = \{ST_0^{C_m}\}_{m=1}^n$, according to the z_Δ ;
- 20 **for** $w = 1, i \leq n, w++$ **do**
- 21 $\hat{d} = (w_r^j, c_n), \hat{d} \in \hat{D}_{test}$, according to the $p = z_\Gamma$;
- 22 $\hat{D}_{test} \leftarrow \text{calculateSimilarity}(\hat{d}, \Delta)$;
- 23 **return** $\hat{D}_{train}$ and $\hat{D}_{test}$;

4. Experiment

In this section, the performance of the proposed method is evaluated through extensive experiments. We chose three datasets which are generated from some IoT applications, and they are all related to the sensors and ambient assisted environment. The performance of five chosen classification algorithms that function without any pre-processing, serves as a baseline. The baseline is then compared with three pre-processing methods including our new method.

4.1. Datasets

Three datasets are prepared in this experiment. All these data are related to the sensor applications, which monitor human activities and environment. The description of each dataset is tabulated in Table 1. These datasets inherently consist of redundancy problem but lack of error information. To simulate the real situation in IoT where faulty sensor can exit, we inject a controlled level of noisy data when we construct the training and testing datasets. As the Figure 4 shows, we analyze some feature of datasets at first. The source datasets are all from the UCI datasets website.

Table 1. Description of various datasets.

Dataset	Instances	Features	Labels	Noisy
LD	164860	8	8	10000
HHAR	43930257	16	6	100000
AReM	42240	6	4	1000

Figure 4. Visualization of AReM dataset which is a graph of time-series of five feature values: it shows that all the time-series are stationary without any significant and distinctive shape, fluctuating within a limited range of values.

Heterogeneity Activity Recognition Data Set: The Heterogeneity Human Activity Recognition (HHAR) dataset from Smart phones and Smart watches is a dataset devised to benchmark human activity recognition algorithms (classification, automatic data segmentation, sensor fusion, feature extraction, etc.) in real-world contexts; specifically, the dataset is gathered with a variety of different device models and use-scenarios, in order to reflect sensing heterogeneities to be expected in real deployments.

Localization Data for Person Activity Data Set: Data contains recordings of five people performing different activities. Each person wore four sensors (tags) while performing the same scenario five times.

Activity Recognition system based on a Multisensor data fusion (AReM) Data Set: This dataset contains temporal data from a Wireless Sensor Network worn by an actor performing the activities: bending, cycling, lying down, sitting, standing, walking. This dataset represents a real-life benchmark in the area of Activity Recognition applications. The classification tasks consist of predicting the activity performed by the user from time-series generated by a Wireless Sensor Network (WSN), according to the EvAAL competition technical annex

4.2. Comparison of Pre-Processing Methods

This section chooses two ways to compare the performance of TBSS. First, five traditional methods were selected as baseline comparisons. There are KNN, Logistic Regression, Naive Bayes, Decision Tree and Support Vector Machine. We try to use the algorithms directly to get the results, then compare with the result that is after TBSS. Second, some pre-processing methods were selected as comparisons.

- K-Neighbors Classifier
- Logistic Regression
- Gaussian Naive Bayes
- Decision Tree
- Support vector machine

Here we select three well-known comparing pre-processing methods as comparisons. They are PCA, Incremental PCA and Normalize Method. Two methods are related to reducing dimension (PCA and Incremental PCA) and one method (Normalize Method) is related to changing the range of the data values. These methods are useful in traditional data mining because they could deal with redundancy problem and dimension explosion.

- PCA
- Incremental PCA
- Normalize Method

4.3. Evaluation Criteria

Three evaluation indicators are used for evaluating the pre-processing methods. There are precision, recall and F1-score. These evaluation criteria are very commonly used in classification tasks so we chose these evaluation criteria as a baseline evaluation. In the experiment result, we could focus on the precision criteria. Our proposed methodology performs well in the precision indicator.

- Precision
- Recall
- F1 score

4.4. Parameters Setting

For setting the parameters of the comparing methods, default values that are given by the Sklearn package are used. For a fair evaluation of the experiment, the program code is run for ten times and obtain the average precision and recall, etc. The length of the sliding window could be changed to suit different tasks. Further experimentation is planned as future work for exploring the optimal sliding window size. The amount of noise we injected into the original dataset is proportional to the amount of the original dataset.

4.5. Result and Analysis

First, we use some original classification algorithms as a baseline to test our datasets. As we could see the results from Tables 2–4, the performance of the model that is trained by raw data is not excellent, because the raw data include so much noisy data, and we found that the running time of the whole process is long because the model repeatedly calculated the redundant data.

Table 2. The performance of TBSS and other pre-processing methods on localization recognition dataset.

Model	Precision	Recall	F1 Score
Localization Dataset ($z_{\Gamma} = 10, z_{\Delta} = 20$)			
KNN	0.1721 ± 0.018	0.3015 ± 0.011	0.2134 ± 0.019
KNN+PCA	0.1154 ± 0.016	0.2125 ± 0.013	0.1363 ± 0.014
KNN+Normalize	0.1422 ± 0.012	0.2614 ± 0.015	0.1832 ± 0.009
KNN+IPCA	0.0134 ± 0.017	0.1683 ± 0.029	0.1035 ± 0.027
KNN+TBSS	0.0467 ± 0.028	0.0454 ± 0.014	0.0335 ± 0.018
DT	0.1811 ± 0.015	0.2436 ± 0.025	0.2021 ± 0.019
DT+PCA	0.1043 ± 0.018	0.1653 ± 0.029	0.1137 ± 0.025
DT+Normalize	0.1457 ± 0.028	0.2134 ± 0.014	0.1612 ± 0.018
DT+IPCA	0.0854 ± 0.035	0.1210 ± 0.023	0.0814 ± 0.037
DT+TBSS	0.3264 ± 0.016	0.0978 ± 0.015	0.1248 ± 0.026
SVM	0.1321 ± 0.035	0.3014 ± 0.029	0.1731 ± 0.013
SVM+PCA	0.0934 ± 0.019	0.2426 ± 0.028	0.1367 ± 0.018
SVM+Normalize	0.0823 ± 0.031	0.2464 ± 0.012	0.1134 ± 0.019
SVM+IPCA	0.1012 ± 0.027	0.2810 ± 0.036	0.1521 ± 0.021
SVM+TBSS	0.1634 ± 0.022	0.0767 ± 0.010	0.0936 ± 0.019
LG	0.1311 ± 0.029	0.3023 ± 0.019	0.1763 ± 0.016
LG+PCA	0.0325 ± 0.026	0.0312 ± 0.028	0.0853 ± 0.019
LG+Normalize	0.0821 ± 0.009	0.2516 ± 0.008	0.1274 ± 0.007
LG+IPCA	0.1234 ± 0.020	0.2346 ± 0.010	0.1474 ± 0.012
LG+TBSS	0.2668 ± 0.025	0.1121 ± 0.019	0.1453 ± 0.016
NB	0.1333 ± 0.022	0.2745 ± 0.018	0.1624 ± 0.017
NB+PCA	0.0929 ± 0.014	0.2535 ± 0.018	0.1374 ± 0.019
NB+Normalize	0.1153 ± 0.013	0.2074 ± 0.018	0.1136 ± 0.018
NB+IPCA	0.1167 ± 0.012	0.2963 ± 0.011	0.1526 ± 0.019
NB+TBSS	0.2442 ± 0.011	0.0947 ± 0.013	0.1163 ± 0.029

Table 2 shows that the performance of TBSS and other pre-processing methods on localization recognition dataset. The TBSS pre-processing method is useful to improve the performance of the model. The parameters are set as $z_{\Gamma} = 10, z_{\Delta} = 20$ and $epoch = 50$. The KNN method does not perform very well in this dataset. Compared with other methods, the TBSS get high precision. Especially the combination model of TBSS and decision tree get the about 0.3264 precision in this dataset.

Table 3. The performance of TBSS and other pre-processing methods on AReM recognition dataset.

Model	Precision	Recall	F1 Score
AReM Dataset ($z_{\Gamma} = 10, z_{\Delta} = 20$)			
KNN	0.1743 ± 0.016	0.1945 ± 0.014	0.1324 ± 0.012
KNN+PCA	0.3053 ± 0.016	0.2631 ± 0.018	0.2734 ± 0.016
KNN+Normalize	0.1784 ± 0.014	0.1982 ± 0.028	0.1335 ± 0.014
KNN+IPCA	0.3033 ± 0.101	0.2613 ± 0.018	0.2743 ± 0.026
KNN+TBSS	0.3042 ± 0.026	0.2634 ± 0.025	0.2733 ± 0.018
DT	0.1832 ± 0.010	0.2421 ± 0.015	0.2053 ± 0.015
DT+PCA	0.3453 ± 0.013	0.2442 ± 0.018	0.2042 ± 0.016
DT+Normalize	0.4935 ± 0.024	0.2674 ± 0.032	0.2434 ± 0.018
DT+IPCA	0.1845 ± 0.012	0.1534 ± 0.012	0.1057 ± 0.014
DT+TBSS	0.3037 ± 0.019	0.3353 ± 0.018	0.3073 ± 0.015
SVM	0.1335 ± 0.012	0.3073 ± 0.022	0.1753 ± 0.017
SVM+PCA	0.2675 ± 0.010	0.2443 ± 0.018	0.2123 ± 0.016
SVM+Normalize	0.3554 ± 0.017	0.3943 ± 0.012	0.2532 ± 0.012
SVM+IPCA	0.2642 ± 0.010	0.2421 ± 0.018	0.2122 ± 0.016
SVM+TBSS	0.2844 ± 0.013	0.2142 ± 0.013	0.2242 ± 0.009
LG	0.1334 ± 0.013	0.3034 ± 0.009	0.1732 ± 0.016
LG+PCA	0.2586 ± 0.010	0.2394 ± 0.028	0.1975 ± 0.018
LG+Normalize	0.2832 ± 0.020	0.2212 ± 0.042	0.2523 ± 0.032
LG+IPCA	0.2212 ± 0.010	0.1863 ± 0.038	0.1524 ± 0.036
LG+TBSS	0.3234 ± 0.029	0.2413 ± 0.014	0.2523 ± 0.021
NB	0.1313 ± 0.022	0.2726 ± 0.025	0.1613 ± 0.027
NB+PCA	0.3086 ± 0.010	0.2426 ± 0.018	0.2111 ± 0.016
NB+Normalize	0.3132 ± 0.023	0.1713 ± 0.009	0.0239 ± 0.010
NB+IPCA	0.2845 ± 0.015	0.2234 ± 0.027	0.2132 ± 0.023
NB+TBSS	0.3477 ± 0.020	0.2663 ± 0.017	0.1856 ± 0.012

Table 3 shows that the performance of TBSS and other pre-processing methods on AReM dataset. In this evaluation of dataset, parameters are set as $z_{\Gamma} = 10, z_{\Delta} = 20$ and $epoch = 50$. As we can see, the TBSS pre-processing is successful in improving the performance of the model by combining with a classification algorithm. The precision of Bayesian algorithm with TBSS is up to about 0.3477, and it is higher than original precision. Table 4 shows that the performance of TBSS and other pre-processing methods on activity recognition dataset. In this dataset, parameters are set as $z_{\Gamma} = 20, z_{\Delta} = 30, epoch = 50$, and we adjust the interval of data in order to improve higher precision. The TBSS still perform well compared with other methods.

The TBSS gets good results in these datasets, although these datasets have noisy and redundancy problem, because the TBSS focuses on the similarity of space and a subspace, which is not easy to affect by single error instances. We use non-anomaly detection and statistics representing the similarity of each instance. These mechanisms strengthen the model's robustness and precision.

Table 4. The performance of TBSS and other pre-processing methods on activity recognition dataset.

Model	Precision	Recall	F1 Score
Activity Recognition ($z_{\Gamma} = 20, z_{\Delta} = 30$)			
KNN	0.1734 ± 0.027	0.3023 ± 0.011	0.2113 ± 0.019
KNN+PCA	0.1543 ± 0.016	0.1434 ± 0.014	0.1665 ± 0.025
KNN+Normalize	0.1932 ± 0.028	0.2075 ± 0.024	0.2157 ± 0.023
KNN+IPCA	0.1374 ± 0.016	0.1834 ± 0.014	0.1656 ± 0.015
KNN+TBSS	0.0934 ± 0.015	0.1074 ± 0.013	0.0455 ± 0.013
DT	0.2664 ± 0.014	0.2767 ± 0.015	0.2568 ± 0.014
DT+PCA	0.2964 ± 0.016	0.2221 ± 0.018	0.2468 ± 0.025
DT+Normalize	0.2635 ± 0.013	0.2867 ± 0.013	0.2434 ± 0.024
DT+IPCA	0.3168 ± 0.024	0.2878 ± 0.023	0.2767 ± 0.027
DT+TBSS	0.3441 ± 0.027	0.2342 ± 0.039	0.3084 ± 0.024
SVM	0.1345 ± 0.022	0.3066 ± 0.012	0.1753 ± 0.037
SVM+PCA	0.3105 ± 0.026	0.2368 ± 0.046	0.2323 ± 0.013
SVM+Normalize	0.3243 ± 0.023	0.2163 ± 0.035	0.2435 ± 0.033
SVM+IPCA	0.2532 ± 0.016	0.2463 ± 0.046	0.2513 ± 0.014
SVM+TBSS	0.3021 ± 0.035	0.2147 ± 0.033	0.2373 ± 0.013
LG	0.1924 ± 0.013	0.2432 ± 0.010	0.1703 ± 0.010
LG+PCA	0.2523 ± 0.029	0.2154 ± 0.024	0.2163 ± 0.014
LG+Normalize	0.2935 ± 0.026	0.1964 ± 0.016	0.2243 ± 0.035
LG+IPCA	0.2626 ± 0.037	0.2172 ± 0.013	0.2025 ± 0.015
LG+TBSS	0.2834 ± 0.016	0.2053 ± 0.014	0.2153 ± 0.015
NB	0.2653 ± 0.024	0.2753 ± 0.025	0.3026 ± 0.014
NB+PCA	0.2923 ± 0.014	0.2845 ± 0.015	0.2624 ± 0.053
NB+Normalize	0.3129 ± 0.012	0.2965 ± 0.023	0.3024 ± 0.012
NB+IPCA	0.3123 ± 0.023	0.2754 ± 0.023	0.2636 ± 0.014
NB+TBSS	0.3534 ± 0.017	0.2623 ± 0.034	0.2753 ± 0.027

For a realistic simulation, we design other experiments in which the length of the sliding window (z_{Γ}, z_{Δ}) is a variable. When we change z_{Γ}, z_{Δ} , the whole model will generate different performance. As Table 5 shows, we fix the value of epoch and set $z_{\Gamma} = 10, z_{\Delta} = 20, z_{\Gamma} = 20, z_{\Delta} = 30, z_{\Gamma} = 20, z_{\Delta} = 40$, and we use a localization dataset to test the TBSS. It is clear that the setting of $z_{\Gamma} = 20, z_{\Delta} = 40$ is better. We analyze this dataset and we found that its classification interval is suitable for the twenty to thirty seconds (which is approximately $z_{\Gamma} = 20$). So this result indicated that TBSS methodology is robust with regards to pre-process the sensor data according to the interesting time interval.

The final experiment is to test the influence of epoch. TBSS will receive more instances when the epoch is large because the times of sampling decide the coverage of the subspace to the original space. As Table 6 shows, we set the parameters $z_{\Gamma} = 10, z_{\Delta} = 20, z_{\Gamma} = 20, z_{\Delta} = 30, z_{\Gamma} = 10, z_{\Delta} = 30$. We try to use 10 to 3 epochs to run the model. We found that the more epochs there are in a range, the greater the improvement in the precision of the model. The epoch does not affect the model's performance out of a range because the subspace similar is overlapping.

Table 5. Different size of subspace and its performance.

Localization Dataset			
Model (TBSS)	Precision	Recall	F1 Score
$z_{\Gamma} = 10, z_{\Delta} = 20$			
KNN	0.0612 ± 0.012	0.0723 ± 0.011	0.0353 ± 0.012
DT	0.3134 ± 0.032	0.0713 ± 0.017	0.1462 ± 0.016
SVM	0.2052 ± 0.013	0.0746 ± 0.013	0.1134 ± 0.032
LG	0.2852 ± 0.038	0.1153 ± 0.013	0.1524 ± 0.013
NB	0.2626 ± 0.011	0.1163 ± 0.013	0.1323 ± 0.012
$z_{\Gamma} = 20, z_{\Delta} = 30$			
KNN	0.0253 ± 0.010	0.0623 ± 0.011	0.0352 ± 0.011
DT	0.2353 ± 0.022	0.1275 ± 0.041	0.1253 ± 0.023
SVM	0.3125 ± 0.012	0.1035 ± 0.012	0.1153 ± 0.012
LG	0.1834 ± 0.042	0.0734 ± 0.014	0.0835 ± 0.014
NB	0.1953 ± 0.031	0.0922 ± 0.021	0.1073 ± 0.011
$z_{\Gamma} = 20, z_{\Delta} = 40$			
KNN	0.0134 ± 0.007	0.0324 ± 0.008	0.0132 ± 0.005
DT	0.2035 ± 0.056	0.1652 ± 0.022	0.1764 ± 0.012
SVM	0.4212 ± 0.010	0.1662 ± 0.024	0.1823 ± 0.032
LG	0.1626 ± 0.033	0.1023 ± 0.012	0.1043 ± 0.042
NB	0.3017 ± 0.028	0.0845 ± 0.012	0.1115 ± 0.051

Table 6. Different iteration times to the performance of the classification in LG dataset.

Parameters	Epoch	Precision	Recall	F1 score
$z_{\Gamma} = 10, z_{\Delta} = 20$	10	0.2952 ± 0.025	0.0942 ± 0.012	0.1343 ± 0.022
	20	0.3045 ± 0.032	0.1084 ± 0.032	0.1253 ± 0.022
	30	0.3264 ± 0.023	0.1311 ± 0.023	0.1442 ± 0.014
	40	0.3323 ± 0.022	0.1434 ± 0.013	0.1503 ± 0.035
	50	0.3353 ± 0.015	0.1334 ± 0.024	0.1452 ± 0.024
$z_{\Gamma} = 20, z_{\Delta} = 30$	10	0.3035 ± 0.010	0.0823 ± 0.013	0.1224 ± 0.022
	20	0.3130 ± 0.012	0.1026 ± 0.012	0.1164 ± 0.009
	30	0.3364 ± 0.024	0.1282 ± 0.034	0.1033 ± 0.013
	40	0.3464 ± 0.013	0.1374 ± 0.024	0.1079 ± 0.031
	50	0.3457 ± 0.022	0.1275 ± 0.015	0.1135 ± 0.023
$z_{\Gamma} = 10, z_{\Delta} = 30$	10	0.4223 ± 0.010	0.1452 ± 0.025	0.1653 ± 0.024
	20	0.4232 ± 0.030	0.1674 ± 0.024	0.1854 ± 0.022
	30	0.4423 ± 0.012	0.1534 ± 0.016	0.1734 ± 0.033
	40	0.4564 ± 0.020	0.1241 ± 0.015	0.1477 ± 0.032
	50	0.4542 ± 0.028	0.1537 ± 0.025	0.1652 ± 0.025

5. Conclusions

This paper reports a subspace similarity detection model based on the subspace-attribute probability calculation, and the computational process uses the anomaly detection method. The proposed methodology is to be used as a pre-processing method that transforms the fine granular time scaled dataset (that has frequent intervals) into a probability dataset in a period, hence better classification model training. The TBSS pre-processing method can effectively solve the problem of repeatability and noise that exist in the sensor data. TBSS is able to smoothly combine with anomaly detection and traditional classification

algorithm. Therefore it should be flexible for most of the IoT machine learning applications in real life. Through the different aspect of experiments, we observe that this model can effectively improve the performance of traditional machine learning classification algorithms in data mining of the Internet of Things. Although TBSS improves the precision in the sensor data mining task, it has a shortcoming. The time-consumption is relatively long, because the sampling time incurs certain overhead for covering the whole search space in each step of window sliding. In the future work, we will look into formulating a low-complexity design suitable for fast sampling for TBSS. The codes of TBSS should be optimized for leveraging GPU computation for fast running.

Author Contributions: Conceptualization, Y.Z. and S.F.; methodology, Y.Z.; software, Y.Z.; validation, S.H., R.W. and W.L.

Funding: The financial support was funded by the following four research grants: (1) Nature-Inspired Computing and Metaheuristics Algorithms for Optimizing Data Mining Performance, Grant no. MYRG2016-00069-FST, by the University of Macau; (2) A Scalable Data Stream Mining Methodology: Stream-based Holistic Analytics and Reasoning in Parallel, Grant no. FDCT/126/2014/A3, by FDCT Macau; and 2018 Guangzhou Science and Technology Innovation and Development of Special Funds, (3) Grant no. EF003/FST-FSJ/2019/GSTIC, and (4) EF004/FST-FSJ/2019/GSTIC.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dhakar, L.; Tay, F.; Lee, C. Skin based flexible triboelectric nanogenerators with motion sensing capability. In Proceedings of the 2015 28th IEEE International Conference on Micro Electro Mechanical Systems (MEMS), Estoril, Portugal, 18–22 January 2015; pp. 106–109.
2. Li, T.; Liu, Y.; Tian, Y.; Shen, S.; Mao, W. A storage solution for massive iot data based on nosql. In Proceedings of the 2012 IEEE International Conference on Green Computing and Communications, Besancon, France, 20–23 November 2012; pp. 50–57.
3. Madakam, S.; Ramaswamy, R.; Tripathi, S. Internet of Things (IoT): A literature review. *J. Comput. Commun.* **2015**, *3*, 164. [[CrossRef](#)]
4. Nyan, M.; Tay, F.E.; Murugasu, E. A wearable system for pre-impact fall detection. *J. Biomech.* **2008**, *41*, 3475–3481. [[CrossRef](#)] [[PubMed](#)]
5. Dan, L.; Xin, C.; Chongwei, H.; Liangliang, J. Intelligent agriculture greenhouse environment monitoring system based on IOT technology. In Proceedings of the 2015 International Conference on Intelligent Transportation, Big Data and Smart City, Halong Bay, Vietnam, 19–20 December 2015; pp. 487–490.
6. Shi, F.; Li, Q.; Zhu, T.; Ning, H. A survey of data semantization in internet of things. *Sensors* **2018**, *18*, 313. [[CrossRef](#)] [[PubMed](#)]
7. Xie, S.; Chen, Z. Anomaly detection and redundancy elimination of big sensor data in internet of things. *arXiv* **2017**, arXiv:1703.03225.
8. Pérez-Penichet, C.; Hermans, F.; Varshney, A.; Voigt, T. Augmenting IoT networks with backscatter-enabled passive sensor tags. In Proceedings of the 3rd Workshop on Hot Topics in Wireless, New York, NY, USA, 3–7 October 2016; pp. 23–27.
9. Nesa, N.; Ghosh, T.; Banerjee, I. Outlier detection in sensed data using statistical learning models for IoT. In Proceedings of the 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.
10. Jolliffe, I.T.; Cadima, J. Principal component analysis: A review and recent developments. *Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci.* **2016**, *374*, 20150202. [[CrossRef](#)] [[PubMed](#)]
11. Patro, S.; Sahu, K.K. Normalization: A preprocessing stage. *arXiv* **2015**, arXiv:1503.06462.
12. M&M Research Group. *Internet of Things (IoT) & M2M Communication Market: Advanced Technologies, Future Cities & Adoption Trends, Roadmaps & Worldwide Forecasts 2012–2017*. Technical Report; Electronics. ca Publications: Kirkland, QC, Canada, 2012.
13. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [[CrossRef](#)]

14. Miorandi, D.; Sicari, S.; De Pellegrini, F.; Chlamtac, I. Internet of things: Vision, applications and research challenges. *Ad Hoc Netw.* **2012**, *10*, 1497–1516. [\[CrossRef\]](#)
15. Bandyopadhyay, D.; Sen, J. Internet of things: Applications and challenges in technology and standardization. *Wirel. Pers. Commun.* **2011**, *58*, 49–69. [\[CrossRef\]](#)
16. Xu, L.; Guo, D.; Tay, F.E.H.; Xing, S. A wearable vital signs monitoring system for pervasive healthcare. In Proceedings of the 2010 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology, Petaling Jaya, Malaysia, 20–21 November 2010; pp. 86–89.
17. Cantoni, V.; Lombardi, L.; Lombardi, P. Challenges for data mining in distributed sensor networks. In Proceedings of the 18th International Conference on Pattern Recognition, Hong Kong, China, 20–24 August 2006; Volume 1, pp. 1000–1007.
18. Keller, T. Mining the Internet of Things-Detection of False-Positive RFID Tag Reads Using Low-Level Reader Data. Ph.D. Thesis, University of St. Gallen, St. Gallen, Switzerland, 2011.
19. Masciari, E. A Framework for Outlier Mining in RFID data. In Proceedings of the 11th International Database Engineering and Applications Symposium, Banff, AB, Canada, 6–8 September 2007; pp. 263–267.
20. Bin, S.; Yuan, L.; Xiaoyi, W. Research on data mining models for the internet of things. In Proceedings of the 2010 International Conference on Image Analysis and Signal Processing (IASP), Zhejiang, China, 9–11 April 2010; pp. 127–132.
21. MacQueen, J. Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Oakland, CA, USA, 21 June–18 July 1965; Volume 1, pp. 281–297.
22. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data clustering: A review. *ACM Comput. Surv. (CSUR)* **1999**, *31*, 264–323. [\[CrossRef\]](#)
23. Xu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Safavian, S.R.; Landgrebe, D. A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man, Cybern.* **1991**, *21*, 660–674. [\[CrossRef\]](#)
25. Friedl, M.A.; Brodley, C.E. Decision tree classification of land cover from remotely sensed data. *Remote Sens. Environ.* **1997**, *61*, 399–409. [\[CrossRef\]](#)
26. McCallum, A.; Nigam, K. A comparison of event models for naive bayes text classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization Citeseer, Madison, WI, USA, 26–27 July 1998; Volume 752, pp. 41–48.
27. Langley, P.; Iba, W.; Thompson, K. An analysis of Bayesian classifiers. *Aai* **1992**, *90*, 223–228.
28. Hearst, M.A.; Dumais, S.T.; Osuna, E.; Platt, J.; Scholkopf, B. Support vector machines. *IEEE Intell. Syst. Appl.* **1998**, *13*, 18–28. [\[CrossRef\]](#)
29. Danita, M.; Mathew, B.; Shereen, N.; Sharon, N.; Paul, J.J. IoT Based Automated Greenhouse Monitoring System. In Proceedings of the 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 14–15 June 2018; pp. 1933–1937.
30. Akbar, A.; Khan, A.; Carrez, F.; Moessner, K. Predictive analytics for complex IoT data streams. *IEEE Int. Things J.* **2017**, *4*, 1571–1582. [\[CrossRef\]](#)
31. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.