Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 125781, 11 pages
http://dx.doi.org/10.1155/2015/125781

Research Article

A Lightweight Data Preprocessing Strategy with Fast Contradiction Analysis for Incremental Classifier Learning

Simon Fong,[1] Robert P. Biuk-Aghai,[1] Yain-whar Si,[1] and Bee Wah Yap[2]

[1] *Department of Computer and Information Science, University of Macau, Macau*
[2] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia*

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

Received 4 February 2014; Accepted 26 July 2014

Academic Editor: Yang Xu

A prime objective in constructing data streaming mining models is to achieve good accuracy, fast learning, and robustness to noise. Although many techniques have been proposed in the past, efforts to improve the accuracy of classification models have been somewhat disparate. These techniques include, but are not limited to, feature selection, dimensionality reduction, and the removal of noise from training data. One limitation common to all of these techniques is the assumption that the full training dataset must be applied. Although this has been effective for traditional batch training, it may not be practical for incremental classifier learning, also known as data stream mining, where only a single pass of the data stream is seen at a time. Because data streams can amount to infinity and the so-called big data phenomenon, the data preprocessing time must be kept to a minimum. This paper introduces a new data preprocessing strategy suitable for the progressive purging of noisy data from the training dataset without the need to process the whole dataset at one time. This strategy is shown via a computer simulation to provide the significant benefit of allowing for the dynamic removal of bad records from the incremental classifier learning process.

1. Introduction

Data preprocessing has traditionally referred to cleaning up the training dataset prior to sending it to the classifier construction learning process. Many researchers advocate it as an indispensable step in the data mining process because it helps to get rid of "bad" data and/or attributes (features), thus supporting a compact degree of dimensionality and enhancing the accuracy of the learned model. While feature selection is undertaken to prune redundant attributes (from the columns of a 2D dataset matrix), removing records that do not contribute to model learning is another common approach taken to maintain the quality of the learned model in terms of classification accuracy. Though it is known that the so-called "bad" records in the training dataset will subsequently affect the performance of the learned model, it is sometimes impossible to tell which records contain "bad" data until they are later reflected by a decline in the performance of the learned model, which will clearly become apparent at too late a stage, especially in critical applications.

Most engineering applications nowadays require harvesting useful insights from the so-called big data which are not only large in volume but fresh data are generated and accumulated continuously. Some typical engineering applications are vital health signs monitoring, genes analysis in biomedical engineering, real-time traffic monitoring in civil engineering, real-time fraud detection in security and data engineering, and so forth. A learnt data mining model needs to be continuously updated and refreshed whenever new records arrive. Typically, data stream mining methods that demand for both accuracy and speed in quick model induction would be useful in these applications. Although many papers can be found in literature about the learning mechanisms of incremental learning in various data stream mining models, the preprocessing techniques for incremental learning are relatively less explored. This paper aims at shedding some light on a lightweight preprocessing method for incremental classifier learning.

In data preprocessing with an objective of cleaning the training data from noise prior to model induction, it is not

easy to identify data instances as bad or good without any external judgment or intervention. This is analogous to the paradox that arises when a majority of data is wrong: the classifier model will take them as "normal." A pressing challenge facing the data mining community is how to pinpoint such noisy data when they are part of a normal statistical distribution and do not clearly fall into the categories of extreme values or outliers. In comparison, handling missing values is a relatively trivial preprocessing issue as the rows that contain any data on blank fields can be deleted or blank field scan can be filled artificially with guessed data [1].

Tagging data as noisy is a challenge when *a priori* knowledge is unavailable. One way of doing so is to compare a given dataset with the normal distribution of values for such data to determine whether it is normal or rare. This may not be a favorable approach: as it normalizes all of the data, the learning model resulting from the training process will lose the ability to make specific predictions and will be blunted by generality. The other technique commonly applied is to conduct a trial classification/test on data taken from the training dataset and check whether they lead to a satisfactory level of classification accuracy. If not, the data are deemed to be bad as they have failed the trait test; retaining them in the training dataset would only lead to a subsequent deterioration in training accuracy. Hence, the data should be purged, based on the principle that good data contribute to constructing a reliable classifier, whereas bad data only confuse the underlying cognition of the classifier model as the model is learnt in a manner similar to pattern recognition among samples from the training dataset.

In Weka, a popular collection of machine learning algorithms used for data mining with open source machine learning software, a filter function called *RemoveMisclassified()* (http://weka.wikispaces.com/Removing+misclassified+instances+from+dataset) has been made available to preemptively remove from the training dataset a subset of data that could reduce the accuracy of the training model. The function however has been developed years ago by the Weka research team and made into a standard function. Recently, it has been used by a data filtering model called PRISM (preprocessing instances that should be misclassified) [2], which eliminates certain instances from the original dataset. They are called "should be misclassified" because the filter function predicts in advance that these instances will likely be misclassified by inducing a preliminary classification model prior to the actual model induction. However, the function doubles the total model learning time at worst (involving a test run of one round of classification to filter out instances that have been misclassified + the actual building of the model based only on correctly classified instances). Therefore, it may not be an elegant solution if a user opts for a lightweight data preprocessing step that does not have to take as much time as building one dummy classifier in advance to distinguish between good and bad data. This may work for batch type data mining, but not for incremental data mining.

Section 2 outlines the contributions other researchers have made for the detection of noise in training data; few techniques have been put forward to distinguish bad data from good data. However, those techniques are all principle-based solutions, and they are likely to share the same shortcoming when it comes to data stream mining. All of these techniques need a reference model to distinguish between two groups of data; the first comprises instances useful for building a good model, and the second consists of noisy data that would not contribute to the construction of a sound model. The problem common to all of these proposed techniques is the need to use the full dataset to establish such a reference model. The decoyed classifier generated by the *RemoveMisclassified()* function, which requires the full data cache and enough time to divide up the samples, is exactly this type of reference model.

This paper proposes a novel lightweight preprocessing strategy that takes the optimal amount of time to produce a trimmed/cleansed training dataset enabling the training model to yield the highest possible degree of accuracy.

Holistically, this preprocessing strategy allows a training model to achieve a good level of accuracy with a short preprocessing time and a compact decision tree (or the minimal rules that are sufficiently significant). In practical terms, these benefits translate into a very fast process for filtering noisy instances and training a relatively accurate model at the same time. This is important in enabling incremental learning where the preprocessing component is quick and effective. The compact decision tree means that only useful classification rules are generated; technically, this means that stringent memory constraints could be met in real-time applications.

The rest of the paper is structured as follows. Section 2 highlights some commonly adopted techniques that other researchers have proposed for the detection and removal of noise from training datasets. Section 3 describes our new lightweight methodology, which includes a workflow, a calibration function for finding the optimal settings, and the details of the "contradiction analysis" mechanism used for removing misclassified instances. Section 4 reports an experiment conducted to validate our strategy. Section 5 concludes the paper.

2. Related Work

Numerous researchers have attempted to apply different techniques in detecting and removing noisy data, which are generally referred to as incorrect or erroneous data instances. These techniques all center on the common observation of how such data instances disrupt training data and thwart the pursuit of classification accuracy. In other words, they generally focus on the observation of data irregularities, how each data instance relates to the others (the majority), and how such data instances relate to classification performance. Most of these techniques can be grouped into the following three categories: statistics-based, similarity-based, and classification-based methods.

2.1. Statistics-Based Noise Detection Methods. Outliers, or data with extraordinary values, are interpreted as noise in this kind of method. Detection techniques proposed in the literature range from finding extreme values beyond a certain number of standard deviations to complex normality tests. Comprehensive surveys of outlier detection methods used

to identify noise in preprocessing can be found in [3, 4]. In [5], the authors adopted a special outlier detection approach in which the behavior projected by the dataset is checked. If a point is sparse in a lower low-dimensional projection, the data it represents are deemed abnormal and are removed. Brute force, or, at best, some form of heuristics, is used to determine the projections.

A similar method outlined by [6] builds a height-balanced tree containing clustering features on nonleaf nodes and leaf nodes. Leaf nodes with a low density are then considered outliers and are filtered out.

Recently, there are some new developments founded on sophisticated statistics models too, such as outlier detection in adaptive functional-coefficient autoregressive models based on Extreme Value Theory [7]. Extending from such statistics computation, the outlier detection evolves into detecting outliers in time-series of univariate data. It has a wide range of applications like detecting outliers from astronautical time series data [8], recovering outliers from traffic volume data using Tensor model [9], and finding outliers from a time series records of available parking spaces [10].

2.2. Similarity-Based Noise Detection Methods. This group of methods generally requires a reference by which data are compared to measure how similar or dissimilar they are to the reference.

In [11], the researchers first divided data into many subsets before searching for the subset that would cause the greatest reduction in dissimilarity within the training dataset if removed. The dissimilarity function can be any function returning a low value between similar elements and a high value between dissimilar elements, such as variance. However, the authors remarked that it is difficult to find a universal dissimilarity function.

Xiong et al. [12] proposed the Hcleaner technique applied through a hyperclique-based data cleaner. Every pair of objects in a hyperclique pattern has a high level of similarity related to the strength of the relationship between two instances. The Hcleaner filters out instances excluded from any hyperclique pattern as noise.

Another team of researchers [13] applied a k-NN algorithm, which essentially compares test data with neighboring data to determine whether they are outliers by reference to their neighbors. By using their nearest neighbors as references, different data are treated as incorrectly classified instances and are removed. The authors studied patterns of behavior among data to formulate Wilson's editing approach, a set of rules that automatically select the data to be purged.

This group of methods that are based on similarity measure is sometimes known as distance-based approach. They are also sometimes known as noise reduction algorithms [14] as they aim at detecting outliers from normal data. There are several classical contributions such as local outlier factor (LOF) [15]; a recent work on using LOF for detecting outliers in data stream mining is presented in [16]. Some variants of outlier distance based algorithm are called Enhanced Class Outlier Distance Based Algorithm (ECODB) [17] and repeated edited nearest neighbor (RENN) [18]. Some latest works reported in 2014 include angle based outlier detection method in preprocessing [19], automatic outlier detection using inter-quartile-range [20], and a deviation-based approach for real-time wireless sensor network data [21], just to name a few.

2.3. Classification-Based Noise Detection Methods. Classification-based methods are those that rely on one or more preliminary classifiers built as references for deciding which data instances are incorrectly classified and should be removed.

In [22], the authors used an n-fold cross-validation approach to identify mislabeled instances. In this technique, the dataset is partitioned into n subsets. For each of the n subsets, m classifiers are trained on the instances in the other $n - 1$ subsets and the instances in the excluded subset are classified. Each classifier tags an instance as misclassified if it is classified incorrectly. Majority voting or a consensus approach can be used in the filtering process.

Another team of researchers [23] presented a robust decision tree method for the removal of outliers. In this method, a pruning tree is built on the training dataset and is used to classify the training data. Instances the pruned tree classifies incorrectly are removed from the training dataset. These processes are repeated until the pruned tree correctly classifies all instances in the training dataset.

In the study reported in [24], the researchers innovatively used a genetic algorithm (GA) to create a set of suspicious noisy instances and select a prototype to identify the set of actual noisy instances. The fitness function of the GA is a generic classifier built in advance, and the GA uses it to search heuristically for misclassified instances.

There are recently other similar contributions which centered on the idea of building a preliminary classifier in advance; from the performance of the early model, the user judges which data instances are deemed to be removed prior to constructing the main classification model. These methods have this principle in common, though the underlying algorithms that they use may vary, such as [2, 25]. Nevertheless, they were designed for traditional data mining where a full dataset needs to be loaded in for batch learning, rather than for incremental learning.

3. Our Proposed Incremental Preprocessing Model (IPPM)

All of the techniques reviewed above were designed for preprocessing in batch mode, which requires a full set of data to determine which instances are to be deleted.

We argue that, for real-time data stream mining, traditional preprocessing approaches may cause a reduction in performance simply due to the different nature of operations and the requirements of a lightweight mechanism. The unique data preprocessing and model learning approach proposed here are different from all those outlined in Section 2.

Preprocessing has traditionally been seen as a standalone step which takes place before model learning starts. The dataset is fully screened at least once to determine which instances should be removed because they would cause misclassification at a later stage. The enhanced training set, which is usually a subset of the original dataset, is then fed

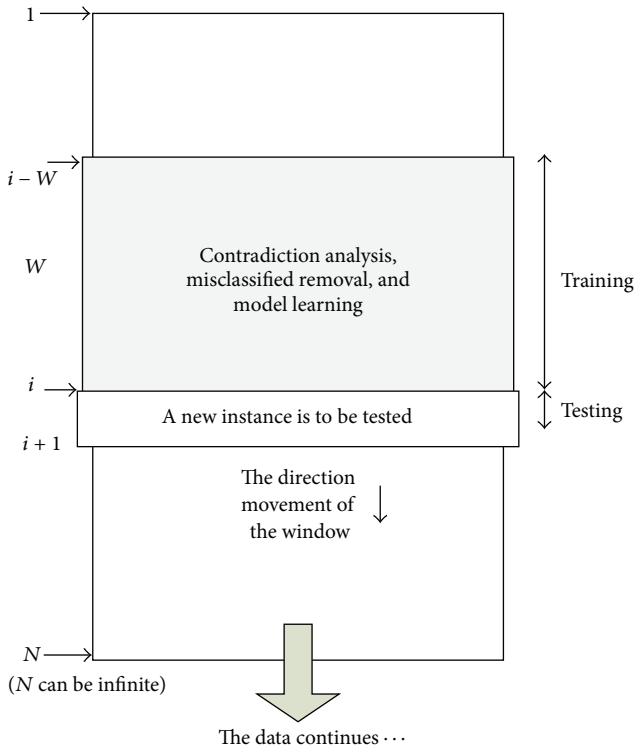

Figure 1: Illustration of how the IPPM works.

into the learning process in the hope that it will facilitate noise-free learning.

In contrast, our incremental preprocessing model (IPPM) is embedded in the incremental learning process, and all of the steps—noise detection, misclassified data removal, and learning—occur within the same timeframe. In this dual approach, preprocessing and training are followed by testing work as the data stream flows in. By way of illustration, Figure 1 shows a window of size W rolling along the data stream. Within the window, the data are first subject to contradiction analysis (for noise detection), then to misclassified data removal and training (model building).

After the model is duly trained, incoming instances are tested. Because this allows for intermediate performance results to be obtained, the average performance level can also be calculated at the end of the process based on the overall performance results.

3.1. Workflow of the Preprocessing and Incremental Learning Model. The full operational workflow of the IPPM is shown in Figure 2.

Both preprocessing and training occur within the same window, which slides along the data stream from the beginning and is unlikely to require all available data. In data mining jargon, this is an anytime method, which means the model is ready to use (for testing) at any time. Whenever new data come in, the window progressively covers the new instances and fades out the old (outdated) instances, and as the analysis kicks in again, the model is updated incrementally in real time.

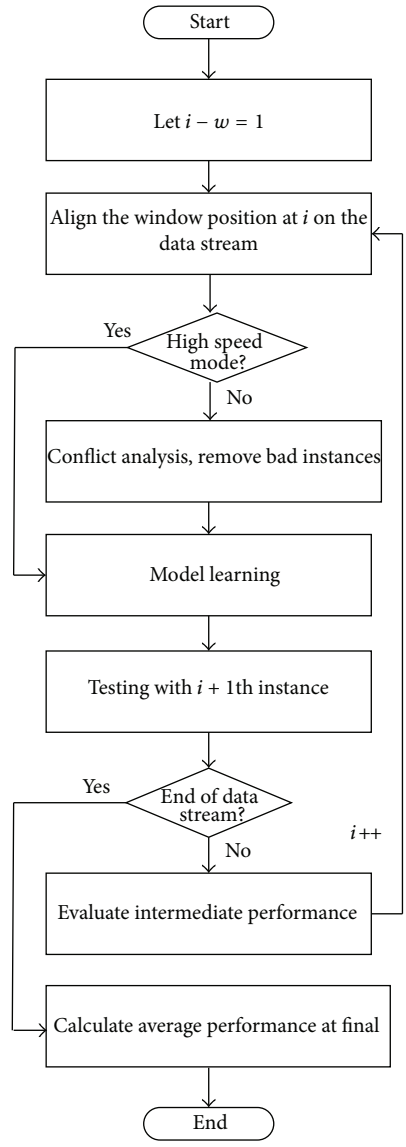

Figure 2: Workflow of the incremental learning strategy.

Through this approach, there is no need to assume the dataset is static and bounded, and the advantages of removing misclassified instances persist. Each time the window moves on to fresh data, the training dataset framed in the window W is enhanced and the model incrementally learns from the inclusion of fresh data within W. Another benefit of the proposed approach is that the statistics retained by the rolling window W can be cumulative. By accumulating statistics on the contradiction analysis undertaken within each frame of the window as it rolls forward, the characteristics of the data are subtly captured from a long-run global perspective. Contradiction analysis can be improved by employing such global information, and it can possibly become more accurate in recognizing noisy data. In other words, the noise detection function becomes more experienced (by tapping into cumulative statistics) and refined in picking up noise. Noise is, of course, a relative concept, the identification of which requires an established reference.

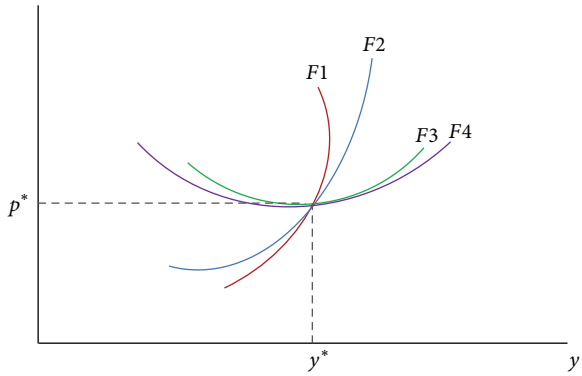

Figure 3: Illustration of optimal p^* and y^* that can be derived among several performance variables.

3.2. Calibration for Optimal Settings. Crucial factors for the IPPM to work well include the choice of window size W and the algorithms used for contradiction analysis and incremental model learning.

There are more than a dozen choices of algorithms, and there is no universal algorithm that fits datasets of all sizes and natures and the theme of the application. For example, a support vector machine, which is relatively fast and has a lower false alarm rate, would be recommended or the rapid detection of malicious attacks in cyber security. For applications where accuracy is more important than speed and relations between attributes and classes are of a complex nonlinear nature, a neural network would be ideal.

For the choice of W, there is always a compromise between time taken and accuracy. In general, the larger the window, the greater the amount of data included in the contradiction analysis and for subsequent enhancement of the training data used for model learning. Hence, accuracy is slightly increased because sufficient data are available. However, what degree of sufficiency is required? How large should the window be to ensure an optimum balance between accuracy and time taken, which is proportional to the amount of data included in W?

Therefore, it is suggested that, before the actual work begins, IPPM operations be initialized with respect to different data and applications. This is why a calibration step is advocated, an exercise that could be used to find the optimal W that produces the performance variables desired. Different algorithms can also be test driven in this step so that the user can gauge the goodness or suitability of each candidate algorithm. Figure 3 shows an example of the optimal point that projects the right W size where all performance variables can be maximized.

The curves $F1, F2, \ldots$, and $F3$ represent factors that influence the values of p^* and y^* simultaneously. The intersection of these curves is an optimum balance by which the corresponding values on the x-axis and y-axis could be mutually maximized without jeopardizing one from another. In our case here, p^* may represent the percentage of performance that could be achieved by our learning model given y^* which is the percentage of the amount of training data. Intuitively, the more the data available for model training, the better the performance that can be yielded. Two factors, time and accuracy, however, are often in opposition—high accuracy usually comes with more data and longer processing time.

In this paper, we adopt a theory of economic order quantity (EOQ), which is also known as the Barabas EOQ model [26], for finding the optimal point that produces a good balance of accuracy and time such that y^* can be deduced from the intersect; hence, the optimal W is obtained.

Given that W is the window size that holds a certain quantity of training samples at a time, W^* is the optimal window size we want to find. N is the total number of samples or instances in the whole dataset. The efficiency gain, E, is defined as the degree of efficiency which is the inverse of the required processing time for processing each instance in average. A, then, is the accuracy cost (loss) that drops per piece of instance that is left out from the training set. Hence, we want to optimize the following:

$$\begin{aligned}\text{Total cost (TC)} = {} & \text{Data size cost} + \text{time consumption cost} \\ & + \text{accuracy cost},\end{aligned} \tag{1}$$

where data size cost is the variable cost of data which is the data unit cost × the total quantity in the dataset. Time consumption cost is the cost of preprocessing the data; each data has an assumed fixed cost E, and we need to spend N/W rounds of iteration; the time consumption cost is then $E \times N/W$. For the average window size is about $W/2$, the accuracy cost is thus $A \times N/W$. Consider

$$\begin{aligned}\text{Hence TC} &= N \times D + \frac{N \times E}{W^2} + \frac{A \times W}{2} \\ D &= -\frac{N \times E}{W^2} + \frac{A}{2}.\end{aligned} \tag{2}$$

Solving for W gives W^* which is the optimal amount of sample in the window. One has

$$W^2 = \frac{2 \times N \times E}{A}; \quad \text{therefore } W^* = \sqrt{\frac{2 \times N \times E}{A}}. \tag{3}$$

The optimization is not limited to two factors; the equation can be extended to multiple factors such as ROC, precision and recall, $F1$ measure, Kappa's statistic, tree size, and model complexity for a multidimensional optimization.

3.3. Contradiction Analysis. For contradiction analysis, a modified pair-wise based classifier (PWC) is used that is based on the dependencies of the attribute values and the class labels. PWC is similar to instance-based classifier or lazy classifier which only gets activated for testing an instance and incrementally trains at most in one round a classifier when the instance arrives. PWC has several incentives over other methods despite the fact that it is very fast in processing which is a prerequisite for lightweight preprocessing. The advantages include simplicity in merely computing the support and confidence values for estimating which target label one instance should be classified into, no persistent tree structure or trained model needs to be retained except small registers

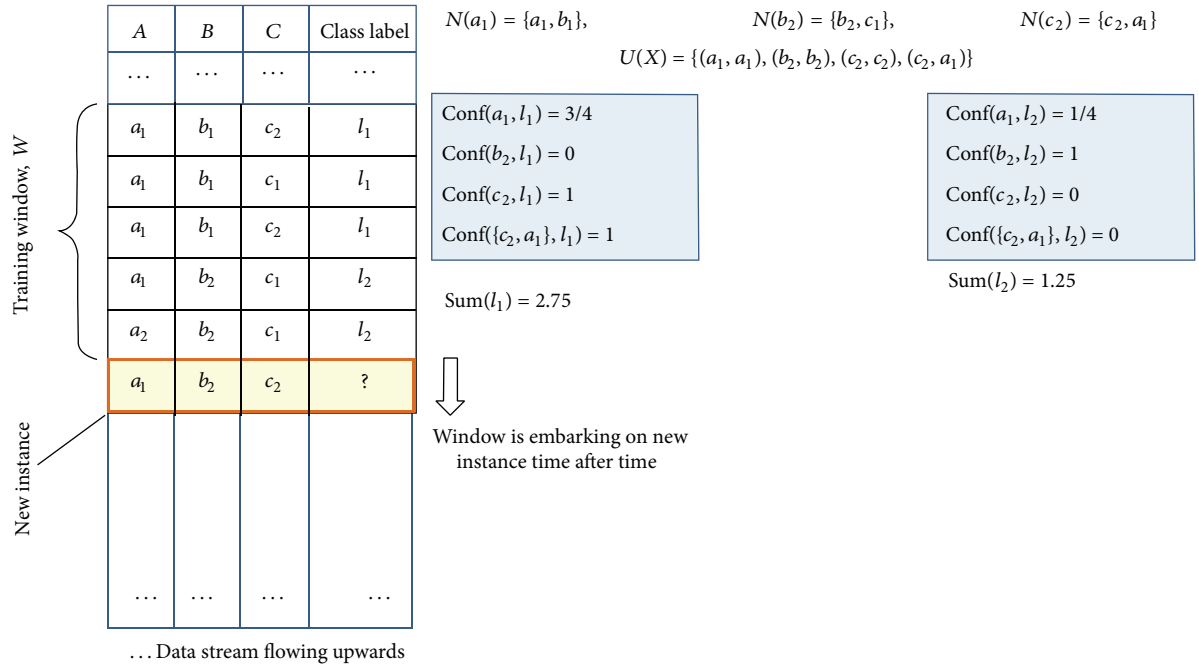

Figure 4: Illustration of a rolling window from which contradiction analysis takes place.

for statistics, and the samples (reference) required for noise detection can scale flexibly to any amount ($\leq W$).

One example that is inspired by [27] about a weighted PWC is shown in Figure 4.

In each time, where current position is i, the sliding window contains W potential training samples over three attributes (A, B, and C) and one target class. X is the new instance at the position $i + 1$ which is just ahead of the preceding end of the window, and it has a vector of values $\{a_1, b_2, c_2\}$ as an example. Assuming $k = W/2$ which rounds up to 2, the neighborhood sets for each attribute value of X are shown in the upper-right area of the figure. For example, $N(a_1) = \{a_1, b_1\}$ because $\text{conf}(a_1, a_1) = 1$ and $\text{conf}(a_1, b_1) = 0.75$ are the highest two for a_1. The resulting $U(X)$ set is found below. For instance, associated with a_1 in $U(X)$ is only a_1 itself, forming the pair (a_1, a_1). It does not belong to X and will be excluded from $U(x)$ although $b_1 \in N(a_1)$. The same applies to c_1 with respect to $N(b_2)$, whereas both c_2 and a_1, which belong to $N(c_2)$, are included in $U(X)$ associated with c_1. For each member of $U(X)$, PWC examines the confidence values against the two target classes l_1 and l_2. For instance, for (a_1, l_1), we calculate a $\text{conf}(a_1, l_1) = \text{support}(\{a_1, l_1\})/\text{support}(a_1) = 3/4 = 0.75$, which checks a first-order dependency between a_1 and l_1. In contrast, for pair (c_2, a_1), we examine a second-order dependency by calculating $\text{conf}(\{c_2, a_1\}, l_1) = \text{support}(\{a_1, c_2, l_1\})/\text{support}(a_1, c_2) = 2/2 = 1$. Taking the sum of confidence values for each class, we obtain $\text{Sum}(l_1) = 2.75$ and $\text{Sum}(l_2) = 1.25$; therefore, the new instance should belong to class l_1. In this way, contradiction is determined by observing whether the calculated class membership matches the actual class label of each new instance. If the new instance is in agreement with the PWC calculation, no contradiction is assumed and the window proceeds forward by one row, leaving out the last row and including the new instance in the training set. If the class label of the new instance contradicts the result of the calculated class label, the new instance is purged.

One modification we made in our process is the use of the other version of neighbour sets. The current neighbor sets store only the most updated confidence values of the pairs within the current window frame, which are called local sets. The information in the local sets gets replaced (recomputed) by the new results every time when the window moves to a new position with inclusion of a new instance. In our design, a similar buffer called global sets is used that do not replace but accumulate the newly computed confidence values corresponding to each pair in the window frame.

Of course, the contradiction analysis can be implemented by similar algorithms. It can be seen that, by using PWC, the required information and calculation are kept as minimal as possible, which implies fast operation.

4. Experiment

The objective of the experiment is to verify the efficacy of the proposed IPPM preprocessing strategy. In particular, we want to see how IPPM works when coupled with different classification algorithms which we briefly divided into two groups—those that were founded on greedy search method and instance-based method. A total of eight algorithms were put under test of IPPM preprocessing strategy. For greedy search, representative algorithms include decision tree (J48), naive Bayesian (NB), neural network (NN), and support vector machine (SVM). Instance-based classifiers include K*—an instance-based learner using an entropic distance measure (KStar), K-nearest neighbours classifier

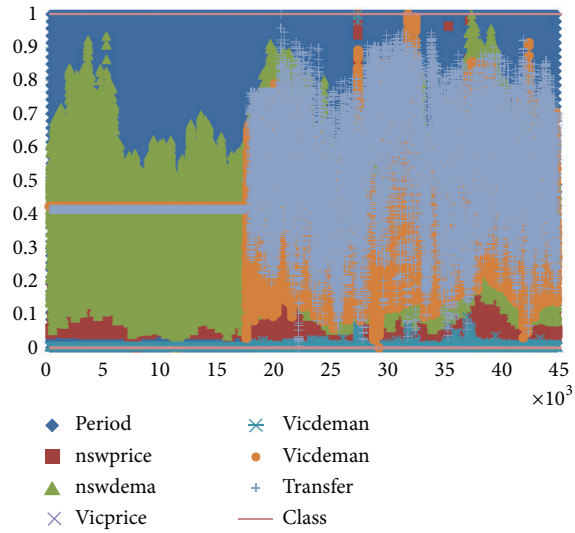

Figure 5: Visualization of the electricity dataset.

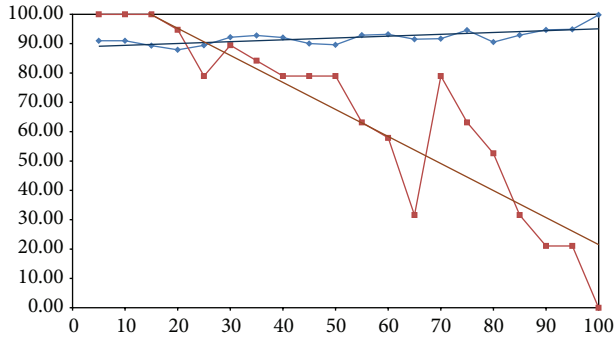

Figure 6: Performance curves for J48.

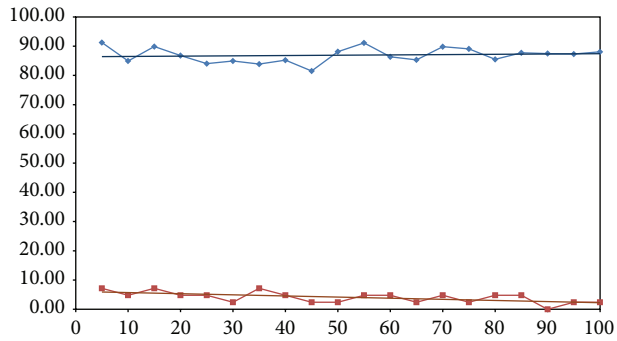

Figure 7: Performance curves for NB.

(IBK), locally weighted learning (LWL), and decision table (DT). The experiment is conducted in a Java-based open source platform called Weka which is a popular software tool for machine learning experiments from University of Waikato. All the aforementioned algorithms are available as either standard or plug-in functions on Weka which have been well documented in the Weka documentation file (which is available at http://www.cs.waikato.ac.nz/ml/weka). Hence, their details are not repeated here. The hardware used is Lenovo laptop with Intel Pentium Dual-Core T3200 2 GHz processor, 8 Gb RAM and 64-bits Windows 7.

The test dataset used is called electricity (http://moa.cs.waikato.ac.nz/datasets) which is popularly used for testing data stream mining algorithms. The data was collected from the Australian New South Wales Electricity Market. In this market, prices are not fixed and are affected by demand and supply of the market. They are set every five minutes. The dataset contains 45,312 instances in 7 attributes. The class label identifies the change of the price relative to a moving average of the last 24 hours, whether it goes up or down as class label. Besides, it was widely used in data stream mining; this particular dataset has a property of slight concept drift. It means the statistical properties of the target class the model is trying to predict change over time without any prior clue. This usually deters the accuracy of a classification model; the predictions that are based on previously trained data become less accurate as time passes. This feature is useful for testing the two modes of the local memory and the global memory in our contradiction analysis. A visualization of the electricity dataset is shown in Figure 5; one can see that the variables *vicprice* and *vicdemand* suddenly vary their range and dominate the data distribution at approximately instance 17,000 onwards. To make the problem more challenging for stress testing the efficiency of the preprocessing method, additional random noises are perturbed at 30% of the dataset.

To start the experiment, the dataset is first subject to a collection of eight algorithms in the calibration stage for testing out the optimal size of W. In practice, only a relatively small sample will be used, and calibration could repeat periodically or whenever the performance of the incremental learning drops, for fine-tuning the window size.

For each of the eight algorithms, the accuracy curve is plotted versus the time efficiency, in the same range between 0 and 100, as described in Section 3.2. It is assumed that the optimal point between accuracy that is defined as the percentage of correctly classified instances and time efficiency is wanted. An averaged trend line is regressed on each curve in order to rectify the fluctuations. The line in blue is for accuracy, and the one in red is for time efficiency. They are obtained from running the algorithm over a sample of training data, which is being scaled down from 100% to 5% with a decrement of 5% each time. Reservoir sampling technique is applied here in extracting a certain percentage of the sample so that the selection of data for exclusion would be fair among the data. Being on the same scale at the y-axis, the lines may intersect at a Pareto optimum (Pareto optimal) that reveals the appropriate training size on the x-axis that will just be enough for producing an optimal level of accuracy at the minimal cost of time efficiency. The optimization charts are shown in Figures 6, 7, 8, 9, 10, 11, 12, and 13, respectively, for the eight algorithms.

It can be observed that, interestingly, optimization of the two performance characters (accuracy versus time) over some algorithms produces a cross-point. For instance, the intersection points for J48 and DT fall at the range between 20% and 30% of sampling data. IBK interests at approximately

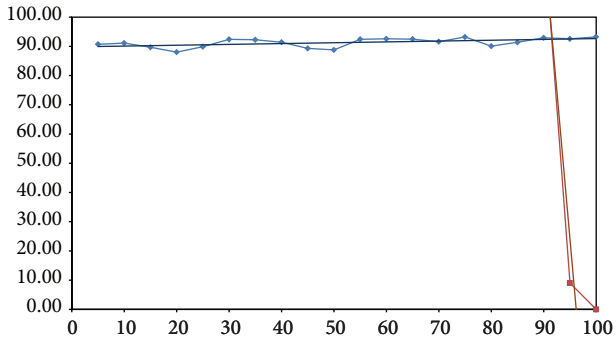

Figure 8: Performance curves for NN.

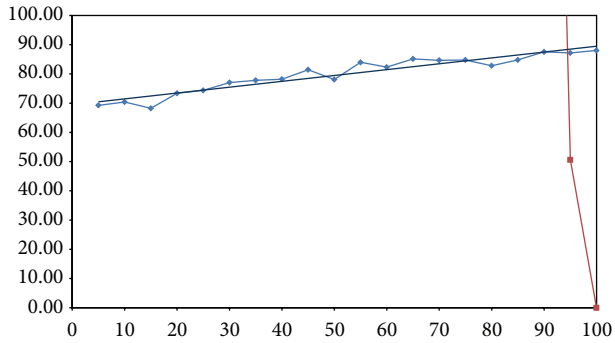

Figure 9: Performance curves for SVM.

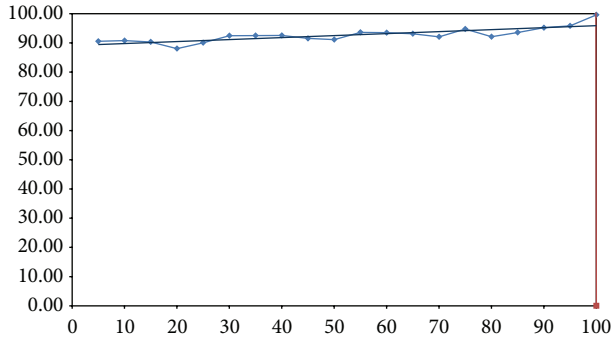

Figure 10: Performance curves for KStar.

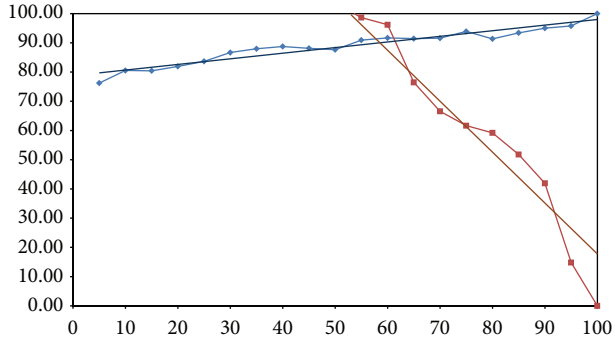

Figure 11: Performance curves for IBK.

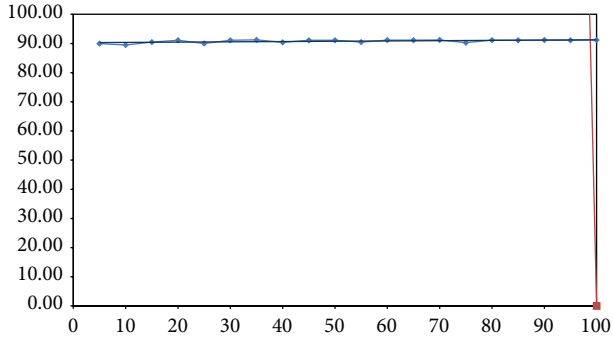

Figure 12: Performance curves for LWL.

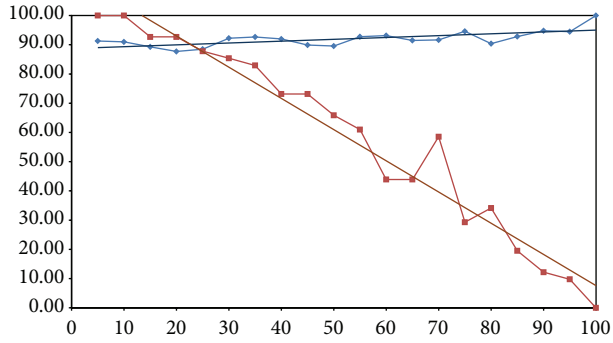

Figure 13: Performance curves for DT.

60%. That means commonly J48 and DT algorithms require about 25% of sampling data in order to achieve an optimal balance or compromise between good accuracy and time efficiency. IBK needs 2.4 times larger the size of the sample. All the achievable accuracies rest on approximately 90% which is not too bad. In contrast, the rest of the algorithms like neural network, SVM, KStar, and LWL require almost the full set of data for reaching an acceptable level of accuracy and speed. This can be explained by the fact that they generally require hundreds times longer the time than those others (e.g., LWL trains and tests for 5052 seconds; compared to IBK that only needs 72 seconds for processing 100% amount of the data). NB is an outlier which is executed extremely fast, but the accuracy fluctuates to the greatest extent among all.

After all, the choice of W and the algorithm could be subjective by the users. From the results, it seems like 25% of window size W is a logical choice. One may argue that if the data stream is potential infinite, how do we know relatively the ideal percentage of the required window size? In such a case, an absolute number of instances would be used in lieu of a relative percentage. The number of training instances would be reduced gradually while observing the degeneration of accuracy, and it will stop when the accuracy falls onto an acceptable minimum threshold.

Next, the whole dataset is subject to the dual process of IPPM preprocessing and incremental learning, after an appropriate W is derived. The dataset now serves a data stream that is being tested and model-trained segment by segment via a sliding window. Again, the eight algorithms

Table 1: Performance comparison of algorithms under batch learning mode.

Algorithm	Batch learning					
	No preprocessing			Preprocessing		
	Accuracy	Time	Gain ratio	Accuracy	Time	Gain ratio
J48	62.5110	33.0000	1.8943	99.7983	9.0000	11.0887
NB	55.0207	3.0000	18.3402	88.0629	4.0000	22.0157
NN	60.0702	1083.0000	0.0555	93.2245	613.0000	0.1521
SVM	58.1060	3607.0000	0.0161	88.0324	803.0000	0.1096
Kstar	91.7483	7970.0000	0.0115	99.6041	7860.0000	0.0127
IBK	100.0000	258.0000	0.3876	100.0000	72.0000	1.3889
LWL	60.5204	16858.0000	0.0036	91.1804	5082.0000	0.0179
DT	61.5731	37.0000	1.6641	100.0000	42.0000	2.3810

Table 2: Performance comparison of algorithms under incremental learning mode.

Algorithm	Incremental learning								
	No preprocessing			PP. with respect to local reference			PP. with respect to global reference		
	Accuracy	Time	Gain ratio	Accuracy	Time	Gain ratio	Accuracy	Time	Gain ratio
J48	56.7105	12.0000	4.7259	71.6697	5.3333	13.4381	69.2177	12.5000	5.5374
NB	55.7100	2.6667	20.8913	68.5534	4.6667	14.6900	84.5540	2.0000	42.2770
NN	53.7179	834.0000	0.0644	66.2704	521.3333	0.1271	73.4672	984.0000	0.0747
SVM	55.8189	844.0000	0.0661	69.6478	238.6667	0.2918	76.5615	1040.0000	0.0736
Kstar	55.9808	3485.3333	0.0161	74.2809	1300.6667	0.0571	78.4271	3262.5000	0.0240
IBK	52.8808	63.3333	0.8350	69.1130	24.0000	2.8797	66.2719	61.5000	1.0776
LWL	59.5147	2562.6667	0.0232	73.4821	911.3333	0.0806	83.6198	4102.0000	0.0204
DT	56.9724	9.3333	6.1042	71.4792	4.6667	15.3170	69.0701	9.0000	7.6745

are used in enabling the model learning. The models by the eight algorithms are updated and refreshed every time when the window progresses by one instance. Specifically, the two modes of contradiction analysis, local reference and global reference, are tested with the given data stream. In data streaming, a significant difference is that the temporal pattern (a.k.a. data evolution pattern or concept drift) can be taken into effect. That is contrary to batch training where the whole data is lumped up and the overall data distributions could only be observed—the change of data distribution in time cannot be seen.

The batch mode learning is still conducted as a comparison benchmark to the incremental learning. Supposedly, the batch mode learning has the luxury of accessing the full set of data; thereby, accuracy can be possibly made to the highest with the full provision. The ultimate objective of this experiment is to compare and judge if incremental learning can be on par with traditional batch learning. In both cases, they are done with and without preprocessing for all the eight algorithms. In batch mode, the training and testing of the data by a learning model are done over the full length of data; the testing results are validated by 10-fold validation; in incremental mode, the process is carried out according to IPPM as specified in Section 3.1.

The performance results in terms of accuracy and total processing time in seconds are shown in Tables 1 and 2 for batch learning and incremental learning, respectively. A gain ratio is added that is simply the accuracy value divided by the time taken value, for easy comparison. The higher the accuracy, the shorter the time taken, and the better the gain ratio. The value of gain ratio, if close to zero, means this algorithm under this configuration of learning mode and preprocessing method is not favorable. Otherwise, it is a good candidate for the combo of algorithm and learning environment when the gain ratio is large.

In the batch learning plus no preprocessing, in Table 1, the gain ratios for those algorithms (NN, SVM, KStar, IBK, and LWL) are consistent with the optimization results in Figures 6 to 13 carry a small gain ratio. Those algorithms that require long training are unfavorable and their accuracies are poor (with exceptions of KStar and IBK). NB has the highest gain ratio; J48 and DT achieve a gain ratio of greater than 1. With preprocessing in batch mode, the trend persists but the gain ratio scaled up significantly. The same PWC was used in all preprocessing methods for consistency.

Some interesting phenomenon is observed when comparing batch learning with no preprocessing and incremental learning with no preprocessing. While the gain ratios of most of the algorithms remain about the same, the gain ratios for J48 and DT increased severalfold in the incremental learning mode. The gain ratios increase even more remarkably when preprocessing applies in incremental learning. The gain ratio for J48 increases from 4.7 (no preprocessing) to 13.4 (preprocessing with local reference) and to 5.5 (preprocessing with global reference). The gain ratio for DT escalates from 6.1 (no preprocessing) to 15.3 which is more than double

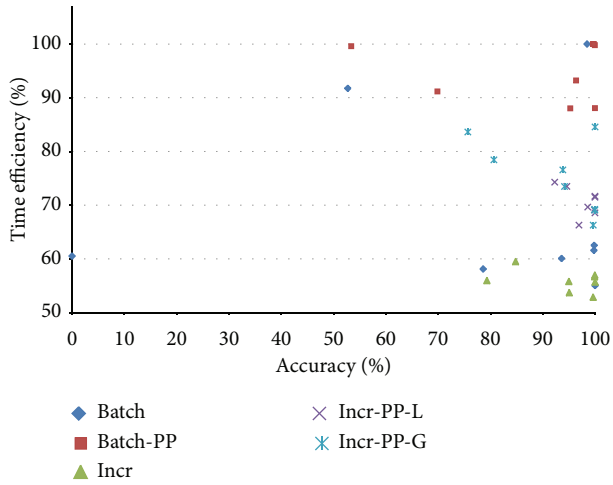

Figure 14: Performance comparison of groups of learning and preprocessing modes at a glance.

the gain (preprocessing with local reference) and to 7.7 (preprocessing with global reference). The other algorithms have their share of increase in gain ratio in varying degrees too, when preprocessing applies in incremental learning.

In general, it can be shown that incremental learning for the eight popular algorithms has an edge over batch learning especially when the lightweight preprocessing methods applied. It may provide an alternative methodology of data mining over the traditional preprocessing and then training followed by testing approach, in batch mode.

In incremental learning, almost all the algorithms perform better in preprocessing with local reference than global preprocessing with local reference, except only NB. In fact, the gain ratio of NB in preprocessing with local reference is even worse than no preprocessing in incremental learning. The increase in local reference over global reference in preprocessing attributes to the effect of concept drift experienced from the given dataset. Keeping check of the information in local context can effectively adapt to the new data distribution caused by the concept drift. New (local) data are more relevant than the aged data, so making use of which maintains good accuracy. NB in opposite, works well in preprocessing with global reference as NB is purely probability-based and it is known to provide accurate result by digesting all the instances of data and enumerating all the probabilities of outcomes and variables. Constraining NB by a local window may thwart its pursuit of counting sufficient instances for probabilities estimation, hence it may result in poor accuracy.

The performance of all the algorithms in terms of accuracy and time is visualized in several groups. They are visualized in Figure 14. The time efficiency is scaled into a range of 0% to 100%. The best performance is at the upper-right corner of the chart. We can easily see a general pattern that the performance of batch learning is leading. Incremental learning without preprocessing ranks at the bottom. However, incremental learning equipped with preprocessing methods follows the performance of batch learning. Out of the two preprocessing methods, the one with local reference is superior to global reference except for the case of NB.

5. Conclusion

Noise is known to be a cause of confusion in the construction of classification models and thus as a factor leading to a deterioration in accuracy. We regard noise as simply a contradicting instance that does not agree with the majority of data; this disagreement causes the establishment of erroneous rules in classification models and disrupts homogenous meta-knowledge or statistical patterns by distorting the training dataset. Other authors refer to noise as outliers, misclassified instances, or misfits, all of which are data types, the removal of which will improve the accuracy of the classification model. Though this research topic has been studied for over two decades, techniques previously proposed for removing such noise assume batch operations requiring the full dataset to be used in noise detection.

To the best of our knowledge, this paper is the first to propose the novel preprocessing strategy incorporated into our model, the IPPM, which is of particular relevance to incremental classification learning. The chief advantage of the IPPM lies in its lightweight mechanism, which provides optimal speed and accuracy. Its design is also suitable for mining moving data streams. The IPPM is extremely simple to use in comparison with other more complex techniques such as those outlined in Section 2. Our experiment validates its benefits in terms of its very high speed and its efficacy in providing a noise-free streamlined training dataset for incremental learning. In future work, the IPPM will be tested with datasets that have different types and numbers of attributes. More importantly, sample datasets with concept-drift characteristics will be used to test our model. Concept drift is a phenomenon whereby the underlying meaning of the dataset changes over time. Because the contradiction analysis statistics retained by the model are of two types—those obtained from a window frame only and those obtained by accumulating the values calculated during the run as a whole—they represent local optima and global optima, respectively. Whereas local statistics are akin to local memory, global statistics are like global memory. It would be interesting to see whether the latter adapt better to the evolving dataset than local see-and-forget statistics. With its unique lightweight capability of empowering incremental learning with the benefit of fast data preprocessing, it is anticipated that the IPPM will contribute to a wide range of data stream applications [28] in which speed and accuracy are equally important.

Conflict of Interests

The authors of this paper do not have a direct financial relationship with the commercial identities mentioned in this paper that might lead to a conflict of interests.

Acknowledgments

The authors are thankful for the financial support from the Research Grant "Adaptive OVFDT with Incremental Pruning and ROC Corrective Learning for Data Stream Mining," Grant no. MYRG073(Y3-L2)-FST12-FCC, offered by the University of Macau, FST, and RDAO.

References

[1] H. Yang and S. Fong, "Aerial root classifiers for predicting missing values in data stream decision tree classification," in *Proceedings of the SIAM International Conference on Data Mining (SDM '11)*, pp. 1–10, Mesa, Ariz, USA, April 2011.

[2] M. R. Smith and T. Martinez, "Improving classification accuracy by identifying and removing instances that should be misclassified," in *Proceedings of the International Joint Conference on Neural Network (IJCNN '11)*, pp. 2690–2697, San Jose, Calif, USA, July-August 2011.

[3] X. Zhu and X. Wu, "Class noise vs. Attribute noise: a quantitative study," *Artificial Intelligence Review*, vol. 22, no. 3, pp. 177–210, 2004.

[4] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[5] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 37–46, May 2001.

[6] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *Procceedings of the Conference of Management of Data (ACM SIGMOD '96)*, pp. 103–114, 1996.

[7] P. Chen, L. Dong, W. Chen, and J.-G. Lin, "Outlier detection in adaptive functional-coefficient autoregressive models based on extreme value theory," *Mathematical Problems in Engineering*, vol. 2013, Article ID 910828, 9 pages, 2013.

[8] W. Hu and J. Bao, "The outlier interval detection algorithms on astronautical time series data," *Mathematical Problems in Engineering*, vol. 2013, Article ID 979035, 6 pages, 2013.

[9] H. Tan, J. Feng, G. Feng, W. Wang, and Y. Zhang, "Traffic volume data outlier recovery via tensor model," *Mathematical Problems in Engineering*, vol. 2013, 8 pages, 2013.

[10] Y. Ji, D. Tang, W. Guo, P. T. Blythe, and G. Ren, "Detection of outliers in a time series of available parking spaces," *Mathematical Problems in Engineering*, vol. 2013, Article ID 416267, 12 pages, 2013.

[11] A. Arning, R. Agrawal, and P. Raghavan, "A linear method for deviation detection in large databases," in *Proceedings of the International Conference on Data Mining and Knowledge Discovery (KDD '96)*, pp. 164–169, Portland, Ore, USA, 1996.

[12] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar, "Enhancing data analysis with noise removal," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 304–319, 2006.

[13] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining and Knowledge Discovery*, vol. 6, no. 2, pp. 153–172, 2002.

[14] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 427–438, 2000.

[15] M. M. Breuniq, H. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.

[16] S. Fong, Z. Luo, B. W. Yap, and S. Deb, "Incremental methods for detecting outliers from multivariate data stream," in *Proceedings of the 13th IASTED International Conference on Artificial Intelligence and Applications (AIA '14)*, Innsbruck, Austria, February 2014.

[17] M. K. Saad and N. M. Hewahi, "A comparative study of outlier mining and class outlier mining," *Computer Science Letters*, vol. 1, no. 1, 2009.

[18] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, no. 6, pp. 448–452, 1976.

[19] M. M. Tavakoli and A. Sami, "Robust preprocessing for improving angle based outlier detection technique," in *Proceedings of the Iranian Conference on Intelligent Systems (ICIS '14)*, pp. 1–6, February 2014.

[20] L. Sunitha, M. BalRaju, J. Sasikiran, and E. Venkat Ramana, "Automatic outlier identification in data mining using IQR in real-time data," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3, no. 6, pp. 7255–7257, 2014.

[21] M. Govindarajan and V. Abinaya, "An outlier detection approach with data mining in wireless sensor network," *International Journal of Current Engineering and Technology*, vol. 4, no. 2, pp. 929–932, 2014.

[22] C. E. Brodley and M. A. Friedl, "Identifying and eliminating mislabeled training instances," in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 799–805, AAI Press, Portland, Ore, USA, August 1996.

[23] G. H. John, "Robust decision tree: removing outliers from databases," in *Proceedings of the 1st International Conference Knowledge Discovery and Data Mining*, pp. 174–179, 1995.

[24] B. Byeon, K. Rasheed, and P. Doshi, "Enhancing the quality of noisy training data using a genetic algorithm and prototype selection," in *Proceedings of the International Conference on Artificial Intelligence (ICAI '08)*, pp. 821–827, Las Vegas, Nev, USA, July 2008.

[25] P. Chen, L. Dong, W. Chen, and L. Jin-Guan, "Outlier detection in adaptive functional-coefficient autoregressive models based on extreme value theory," *Mathematical Problems in Engineering*, vol. 2013, Article ID 910828, 9 pages, 2013.

[26] A. Hax and D. Candea, *Production and Operations Management*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1984.

[27] A. Nanopoulos, A. N. Papadopoulos, Y. Manolopoulos, and T. Welzer-Druzovec, "Robust classification based on correlations between attributes," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, 14 pages, 2007.

[28] S. Fong and H. Yang, "The six technical gaps between intelligent applications and real-time data mining: a critical review," *Journal of Emerging Technologies in Web Intelligence*, vol. 3, no. 2, pp. 63–73, 2011.