ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
UDC/UDK 004.021.052.42

INFORMATION-VALUE-BASED FEATURE SELECTION ALGORITHM FOR ANOMALY DETECTION OVER DATA STREAMS

Xiaozhen Zhou, Shanping Li, Cheng Chang, Jianfeng Wu, Kai Liu

Original scientific paper

Computer systems are becoming more and more complex, and system anomalies have a serious impact on system availability. One effective way to achieve high availability is to use anomaly detection tools to find the abnormal activities in the computer system so that they can be repaired. Because of the complexity of modern computing systems, many system metrics need to be monitored. For this reason, one major challenge of anomaly detection is multi-dimensionality. Large numbers of metrics increase the processing time of anomaly detection technology and lower the accuracy. To overcome this problem, we use information-value to ascertain the importance of features with respect to detecting anomalies. However, the information-value method does not take redundant features into account. Thus, correlations between features are evaluated to remove redundant features. This paper compares the presented method to other feature selection methods using a real system anomaly data set. Experimental results show that the presented method can learn the model more efficiently and detect anomalies more accurately.

Keywords: *anomaly detection, data stream classification, feature selection, information-value*

Algoritam za odabir karakteristika utemeljen na vrijednosti informacije u otkrivanju nepravilnosti u nizovima podataka

Izvorni znanstveni članak

Računalni sustavi postaju sve složeniji i nepravilnosti sustava uveliko utječu na raspoloživost sustava. Učinkovit način za postizanje visoke raspoloživosti sustava je primjena alata za otkrivanje anomalija kako bi se otkrile nenormalne aktivnosti u računalnom sustavu i bile popravljene. Zbog složenosti modernih računalnih sustava mnoge se matrice sustava moraju nadgledati. Zbog toga je multi-dimenzionalnost jedan od najvažnijih zahtjeva u otkrivanju nepravilnosti. Veliki broj matrica povećava vrijeme obrade tehnologije otkrivanja nepravilnosti i smanjuje točnost. Za rješavanje ovog problema mi koristimo vrijednost informacije kako bismo provjerili važnost karakteristika u odnosu na otkrivanje nepravilnosti. Međutim, metoda vrijednosti informacije ne uzima u obzir redundantne karakteristike. Zbog toga se procijenjuju korelacije između karakteristika kako bi se odbacile redundantne karakteristike. U ovom se radu prikazana metoda uspoređuje s drugim metodama odabira karakteristika primjenom niza podataka o nepravilnosti stvarnog sustava. Eksperimentalni rezultati pokazuju da prikazana metoda može učinkovitije podučiti model i točnije otkriti anomalije.

Ključne riječi: *informacija-vrijednost, klasifikacija niza podataka, odabir karakteristika, otkrivanje nepravilnosti*

1 Introduction

As computing systems become increasingly complicated, they also become more vulnerable to anomalies such as performance bottlenecks and service level objective (SLO) violations [1]. Many anomalies are inevitable and can affect the income and reputation of companies. For this reason, system administrators must detect and fix anomalies as soon as possible in order to shorten the service downtime. Anomaly detection technology can be used to find patterns that do not conform to expected behaviour. It can be used to monitor the current status of the system and alert administrators as soon as an anomaly occurs. Anomaly detection is often modelled as a classification problem in a machine-learning context. Here, system metrics data for a given period have been collected and labelled as normal or abnormal by experts. Then the classification model was learned using these data. This procedure is called training. After obtaining the classification model, the most current data is treated as the input and the classification model is run to determine the status of system. This procedure is called testing. If the status is found to be abnormal, then the anomaly detection tool can be used to alert the operation team. This may shorten the service downtime and minimize losses.

One serious problem with anomaly detection in complex computing systems is that the detection speed is slow and overload is high [19]. This is because of the multi-dimensionality of the data to be processed. In this way, it involves real-time processing requirements other than the correctness criteria. The detection method needs to be fast so that it can process huge amounts of monitoring data in real time. If the processing speed is slower than the speed at which the data are delivered, then some important information will be lost. This may cause false negatives and missed alarms. Real-time detection can help the system administrators take immediate action at the first sign of an anomaly. In this way, detection speed has become one of the most important indicators of system anomaly detection.

The issue of how to develop a lightweight anomaly detection algorithm has become a research hotspot. The sheer amount of data that must be collected and processed is one of the main causes of the slow speed and heavy overhead of anomaly detection tools. For this reason, anomaly detection tools must reduce the amount of data being processed. Many previous studies have solved this problem through data filtering and feature selection [2, 3]. Data filtering directly eliminates the data before processing, so it may remove useful data. Feature selection involves finding a subset of features that can best represent the training data. The idea behind feature selection is that some features may contain false correlations that may decrease the accuracy of the detection process; and other features may be redundant. By removing irrelevant and redundant features, both the speed and accuracy of the anomaly detection algorithm can be improved.

Many classical feature selection technologies, such as InfoGain, GainRatio, and ChiSquared calculate the amount of information in each feature, and then find the most powerful subset of features using a ranking algorithm. However, in data stream applications, such as system anomaly detection, the data distribution of each attribute cannot be constant. They are always changing and evolving. In this way, the subset of most informative features must be recalculated as the data distribution changes. As the system changes, new features emerge and old ones fade away. This is called feature evolution [10]. To address these two problems, the above-mentioned feature selection technologies must calculate the amount of information amount of all features and sort them to determine the updated feature subset. This process is always time-consuming.

In this paper, the problem of feature selection in data streams for the purpose of detecting anomalies in computing system is evaluated. A novel feature selection algorithm, here called information value is used [11]. This algorithm was first used in the financial field, especially for credit card fraud detection. It has a pervasive threshold that can be used to determine the importance of each feature. In this way, when the distribution of some features changes or when a new feature is brought in, this method does not need to calculate the amount of information in every feature and rank the features to determine a suitable feature subset. In this way, information value is more lightweight than other feature selection methods, and it is especially suitable for data stream applications such as system anomaly detection. However, this method only assesses the amount of information of each feature, but it does not consider redundancy between similar features. In this paper, the information value is improved with correlation methods to determine the relationships among streams so that these redundant features can be removed. Then feature selection is integrated with ensemble learning for data stream anomaly detection. The experiment shows that feature selection can help enhance ensemble learning. It not only improves the accuracy of classifiers by removing irrelevant and redundant features but also reduces the complexity of learning algorithm because only the most informative features are processed.

The main objective of this paper is to develop a novel feature selection algorithm suitable for detecting system anomalies from monitoring metrics. The main contributions of this paper are summarized below:

1) The information value (*InfoValue*) feature selection algorithm is enhanced through correlation, which helps to remove the redundant features.
2) Ensemble learning is integrated with information value to increase the accuracy of classification and reduce the complexity of the computation process.
3) The effectiveness of the feature selection algorithm is verified through extensive experiments on real data sets.

The rest of this paper is organized as follows. Section 2 provides the background of the data stream model and anomaly detection method. Section 3 reviews previous works. Section 4 proposes the information-value-based feature selection method. Section 5 demonstrates the experiments and provides an analysis of the results. A conclusion and prospects for future research are given in Section 6.

2 Data stream model

For the computer system being monitored, the running status of system metrics such as CPU usage and memory usage must be collected. These measurements are collected in the form of data streams. A data stream is like an infinitely N-dimensional multi-variable time series, N is the number of metrics measured. In general, the data were considered in numerical domains. Ordinal data were easily transformed to numerical data through normalization. This paper assumes that the speed with which the data arrive in the stream is constant, which means that each metric generates one datum per unit of time. In this paper, a sliding window model was used to process the data stream, and each sliding window was assumed to receive data in chunks. The data has k features, so the data stream model resembles the following:

$$S = \{D_0, D_1, D_2, ..., D_i, ...\}, \tag{1}$$

$$D_i = \{x_{n(i-1)+1}, ..., x_{in}\}, \tag{2}$$

$$x_t = \left[m_t^1, m_t^2, ..., m_t^k\right]^{\mathrm{T}}. \tag{3}$$

The data stream S is made up of a sequence of data chunks D_i, and in the present paper, each chunk was assumed to have the same size n. Eq. (2) represents the data instance in the data stream that arrives in sequence, where x_t is a k-dimension vector that arrives at time t. The feature selection method must find a subset p of k, where $p << k$. Then the classification model can be trained to recognize the reduced features set. In the present paper, the data streams were assumed to have 2 class labels: 0 and 1, where 0 means the system is running normally, and 1 means that there is an anomaly. For system anomaly detection applications, when the number of dimensions k is large, there is often only a small set of key features that are critical to building the accuracy model for classification.

3 Related work

Feature selection is an important issue in system anomaly detection applications because the monitoring streams are usually multi-dimensional. Feature selection methods can be classified into three categories: filter models, wrapper models, and hybrid models [5]. Filter methods rely on the characteristics of the training data to select features without involving any learning algorithms. The wrapper model requires a learning algorithm and uses the results of classification to determine which features must be selected. The hybrid model takes the advantage of the more useful aspects of the filter and wrapper models. The detection algorithms used in system anomaly detection applications require real-time responses. The filter model is used here because it is simpler.

In data stream applications such as system anomaly detection, the relevant importance of features is dynamic. Features that had once been more informative than others may become irrelevant, and previously rejected features may become important ones. Unfortunately, research on feature selection suitable for monitoring the evolution of features and the redundancy of features for data streams is limited.

Alec et al. proposed a novel feature selection technique which uses information value to determine the importance of each feature to the results of classification. However, the information value indicator only indicates the importance of each feature. It does not consider redundancy between similar features [11]. Nguyen et al. proposed a dynamic feature selection technique to address the feature evolution problem in data streams [6]. They used FCBF to calculate the dependencies of features and the class relevance [12]. This technique starts with the full set of features and removes irrelevant and redundant features using a backward selection technique. The algorithm stops when there are no features left to eliminate. Xindong et al. studied the problem streaming feature selection, in which not all features are available for learning [9]. They assumed that the features are generated continuously and that the instances are constant. Mohammad et al. addressed four major challenges in data stream classification: infinite length, concept-drift, concept-evolution, and feature-evolution [10]. They used ensemble learning to address the problem of concept-drift and concept-evolution. They used homogenization technology to address the feature-evolution problem.

There are already a number of studies devoted to anomaly detection. These include studies on intrusion detection, fraud detection, medical and public health anomaly detection, industrial damage detection, and system anomaly detection [13]. Computer systems always use monitoring tools to collect different types of data, such as CPU usage and memory usage data. The data are generated in a streaming mode. In this way, anomaly detection in large-scale computer systems poses a specific set of challenges. First, the detection technologies must be able to operate online. Second, they must be lightweight. Also, the data collected always includes noise and missing values. There are three main types of methods of system anomaly detection: event-driven methods, data-driven methods and symptom-driven methods. Event-driven methods directly analyse the error rate or number of reported failures and use error reports as input data to detect system anomalies. Kiciman et al. used decision trees to identify faulty components in J2EE application servers by determining whether requests had been successful or not [14]. These approaches involve the basic assumption that the behaviour of anomaly-prone systems can be identified using the characteristics of anomalies.

Data-driven methods learn from the temporal and spatial correlations among anomaly events. They aim to recognize the relationships among upcoming failures and previous failures. Liang et al. exploited these correlation characteristics on IBM's BlueGene/L [15]. They found the occurrence of a failure is strongly correlated to the time stamp and the location of others in a cluster environment. Zhang et al. proposed a hybrid detection technique which uses model checking techniques [16]. An operational model was developed to determine whether any desirable temporal property was satisfied or violated by the model itself.

Symptom-driven methods analyse some workload-related data such as input workload and memory workload in order to predict further system resource utilization. Tan et al. monitor a series of run-time metrics (CPU, memory, I/O usage, and network), use a discrete-time Markov chain to forecast the system metrics in the future, and finally predict the system state based on Naive Bayesian classification [18]. Luo et al. [23] build autoregressive model using various parameters from an Apache webserver to predict further system resource utilization; failures are detected by resource exhaustion.

4 The InfoValue feature selection algorithm

4.1 An overview of the InfoValue approach

In this section, a general framework for the detection of anomalies in system measurement data streams is proposed. Then, the details of the design and procedure of the Information Value feature selection (*InfoValue*) algorithm are explained. The key idea behind feature selection in system anomaly detection is to accelerate the learning of categorization and testing processes by reducing the dimensionality of the data. In this way, the overall performance of the anomaly detection is improved in terms of accuracy, time, and space complexity.

Fig. 1 shows the overall framework for system anomaly detection. First, the system metrics are collected by a monitoring tool, and it is assumed that these data are generated at a constant speed. Then the information-value of each metric is calculated. If this value is larger than a threshold, this metric is considered a possible important attribute. Since the information-value only indicates the importance of each feature for classification but does not consider the redundancy between features, in this way, the correlation between features can be calculated and used to determine whether any newly added feature is redundant or not. A data structure feature correlation summary (FCS) is proposed to facilitate the calculation of correlations. This is because the system measurements are generated in the form of data streams, and it is not possible to store and use all details to calculate correlation values. If the possible important attribute is not redundant, it is added into the feature subset. After the feature subset is defined, multiple classifiers are trained to recognize different types of distributions to achieve high diversity. The process is performed in a sliding window fashion, and the selected subset features are monitored to detect feature drift. When two consecutive subsets are different, it can be postulated that feature drift has occurred. Under these circumstances, the feature selection algorithm must be rerun to determine the most discriminative subset.

Feature selection involves the selection of a subset of p features from the original k features ($p << k$). Generally, the feature selection technique is needed to remove irrelevant and redundant features. The information value algorithm was used here because it is simple, fast, and effective. Its main use is to analyse the impact of each factor on the results. Currently, information values are used widely in data analysis, especially in credit rating. This paper is the first to use the information value algorithm to select features for system anomaly detection.

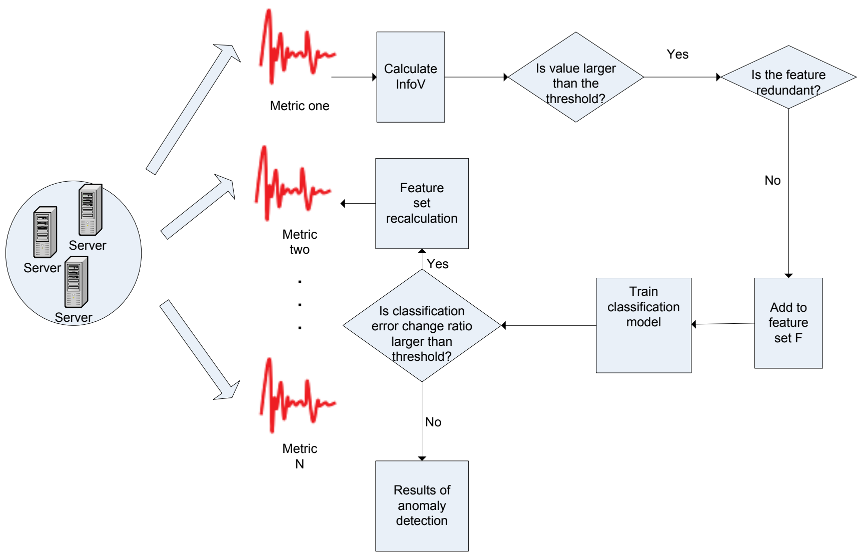
Metric one
Calculate InfoV
Is value larger than the threshold?
Yes
Is the feature redundant?
No
Feature set recalculation
Metric two
Server
Server
Server
Yes
Is classification error change ratio larger than threshold?
Train classification model
Add to feature set F
No
Metric N
Results of anomaly detection

Figure 1 InfoValue feature selection approach

4.2 Definitions

Here are some of the definitions used in this paper:

Definition 4.1. Information Value

The information value of feature X is calculated as follows:

$$InfoValue = \sum \left(X_{\text{dist}} - \overline{X}_{\text{dist}} \right) \ln \left(\frac{X_{\text{dist}}}{\overline{X}_{\text{dist}}} \right), \tag{4}$$

$$X_{\text{dist}} = f\left(Y = y_k \mid X = x_i \right), \tag{5}$$

$$\overline{X}_{\text{dist}} = f\left(Y \neq y_k \mid X = x_i \right). \tag{6}$$

As Eq. (5) shows, X_{dist} is the conditional probability density of $f(Y = y_k)$ when the feature value is $X = x_i$. In the same way, equation (6) shows that $\overline{X}_{\text{dist}}$ is the conditional probability density of $f(Y \neq y_k)$ when the feature value is $X = x_i$.

The information value can determine the importance of each feature to the results of classification. A previously published empirical rule of thumb suggests that if *InfoValue* is within the range of [0; 0,02], then this feature is not predictive; if InfoValue is within the range of (0,02; 0,1], then this feature has weak predictive power; if *InfoValue* is within the range of (0,1; 0,3], then this feature has medium predictive power; and if *InfoValue* is within the range of (0,3; +∞), then this feature has strong predictive power [11].

The information value has the following advantages: First, it can be used to assess the importance of each feature to the results of the classification process. Second, it has a pervasive threshold that can be used to determine the number of features that have power discrimination, while other feature selection technologies, such as GainRatio and principal component analysis (PCA), require a ranking algorithm to identify the most powerful features. This makes it difficult for other methods to determine the appropriate number of features. Third, the information value calculation process is more lightweight than those of other feature selection methods, so it is especially suitable for data stream applications such as system anomaly detection.

However, the information value indicator only indicates the importance of each feature. It does not consider redundancy between similar features. Correlations can be used to assess the relationships among streams. To address this problem, correlations were used to assess the similarities between different features. Correlations are statistical measurements that indicate the closeness of the relationship between two random variables. Because incoming data is much more important than older data, data must be weighted for importance using time stamps.

Definition 4.2. Correlation between variables

The correlation between two features f_i and f_j is calculated as:

$$\sigma_{XY} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{j=1}^{n} (y_j - \bar{y})^2}}, \tag{7}$$

where $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$, and $\bar{y} = \frac{\sum_{j=1}^{n} y_j}{n}$ is the average value of feature f_i and f_j. For the definition of correlation, we can see that the value of $|\sigma_{XY}| \leq 1$ the larger the value of σ_{XY}, these two features have stronger correlation; if σ_{XY}=0, then these two features are uncorrelated. As in the application of system running status monitoring, the monitoring measures are continuously coming, so the correlation between these features will also change. And in the data stream application, it is not possible to store all the past data, so we define a summary structure for the correlation calculation.

Definition 4.3. Feature Correlation Summary

Given r data streams $S_1, S_2, \ldots, S_r$, then we can define the following structure $FCS = (\vec{S}, \vec{Q}, C, t)$, where the components in FCS are defined as:

$$\vec{S}_i = \sum_{k=t-l+1} x_{ik}, \ i = 1,2,\ldots,n \tag{8}$$

$$\vec{Q}_i = \sum_{k=t-l+1} x_{ik}^2, \ i = 1,2,\ldots,n \tag{9}$$

$$C_{ij} = \sum_{k=t-l+1} x_{ik} x_{jk},\ i,j = 1,2,\ldots,n. \tag{10}$$

Theorem 4.1. Correlation between different data streams can be calculated using the feature correlation summary *FCS*.

Proof.

$$\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_j-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{j=1}^{n}(y_j-\bar{y})^2}} =$$

$$= \frac{\sum_{i=1}^{n}(x_i y_j - x_i\bar{y} - y_j\bar{x} + \bar{x}\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i^2 - 2x_i\bar{x} + \bar{x}^2)}\sqrt{\sum_{j=1}^{n}(y_j^2 - 2y_j\bar{y} + \bar{y}^2)}} =$$

$$= \frac{\sum_{i=1}^{n}(x_i y_j) - n\bar{x}\bar{y} - n\bar{x}\bar{y} + \bar{x}\bar{y}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2\right) - 2n\bar{x}^2 + \bar{x}^2}\sqrt{\left(\sum_{j=1}^{n} y_j^2\right) - 2n\bar{y}^2 + \bar{y}^2}} =$$

$$= \frac{C_{ij} - \frac{S_i S_j}{n}}{\sqrt{Q_i - \frac{S_i^2}{n}}\sqrt{Q_j - \frac{S_j^2}{n}}}.$$

Because it is only necessary to retain the statistical information of each feature, the time and space cost for the calculation of redundancy is low. This is important in the data stream applications such as system anomaly detection.

4.3 InfoValue Feature Selection Algorithm

Algorithm 1 gives the details of the *InfoValue* feature selection algorithm. At the beginning, the reduced feature set is established to be empty. Stream-based *k*-means were selected to divide the continuous data metric values into separate bins [20]. Fig. 2 shows the detail of this algorithm. When a new data emerges, the *k*-means result is updated, and C_{ik}, the number of attributes a_i with a value of y_k is determined. Line 6 shows calculation of the InfoValue for feature *f* using the equation in 4. Line 7 shows whether keep the feature as a possibly important feature. The threshold θ is set to 0,3, as suggested in a previous study [11]. The original *InfoValue* algorithm did not consider the redundancy between features, so it must select all the features with *InfoValues* larger than θ. In this way, it can be improved by using correlations to identify informative features and exclude redundant features. Line 10 shows how it can be determined whether the candidate set is empty. These features are added to the reduced feature set *F'*. Lines 12 to 16 show the process that must be performed if the candidate set is not empty. Line 13 iterate each candidate feature *f'* in *F'*, and depicts calculation of the correlation between *f'* and the latest feature, which has a value of InfoValue(*f*) ≥ θ. Line 15 shows a comparison of the correlation value to λ. If this value is less than λ, it means the two features are not correlated. Feature f is then added to the candidate feature set *F'*.

The anomaly pattern in the system is not static, so the features selected at this point do not represent the underlying system well. In this way, the concept-drift problem in the data stream is a very important aspect of feature selection technology. To address this problem, ensemble classification with weighted accuracy is used.

```
    Input: A series of infinite system metrics $x_t = [m_t^1, m_t^2, \ldots, m_t^k]^T$; InfoValue threshold $\theta$; correlation threshold $\lambda$;
    Output: Reduced feature set $F'$
1: Reduced feature set $F$ = null;
2: while X has more instances do
3:   for each metrics $m^k$ do
4:     dicresize the continuously data using stream based $k$-means;
5:     count($C_{ik}$);
6:     InfoValue($f$) = calculate info value using equation 4;
7:     if InfoValue($f$) < $\theta$ then
8:       return;
9:     else if InfoValue($f$) ≥ $\theta$ then
10:      if $F'$ is empty then
11:        $F'$.add($f$);
12:      else if F is not empty then
13:        for each feature $f'$ in $F'$ do
14:          calculate correlation ($f$,$f'$);
15:          if correlation < $\lambda$ then
16:            $F$.add($f$);
17:          end if
18:        end for
19:      end if
20:    end if
21:  end for
22: end while
```

Figure 2 *InfoValue* feature selection algorithm

In summary, the ensembles of classification can be broken down into two categories, horizontal ensemble classification and vertical ensemble classification. The horizontal ensemble classification builds classifiers using several buffered chunks, and the vertical ensemble classification builds classifiers using different learning algorithms on the current chunks.

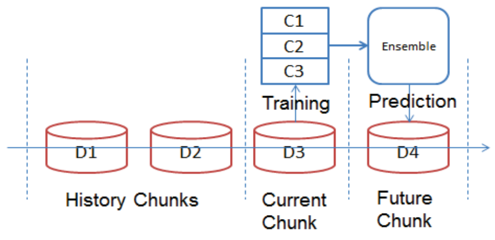

Figure 3 Vertical ensemble classification

Vertical ensemble uses different classification algorithms to build classifiers on the current chunk. It then uses the results of these classifiers to form an ensemble classification model. Vertical ensemble only uses the current chunk to build classifiers. Fig. 3 shows the vertical ensemble classification. The advantage of the vertical ensemble classification is that it uses different algorithms to build the classifier model. This can decrease the bias error between each classifier. However, vertical ensemble assumes that the data stream is errorless. As discussed earlier, the real-world data stream always contains errors. In this way, if the current chunk contains mostly noise data, severe performance deterioration may take place. The horizontal ensemble, which uses multiple history chunks to build classifiers, can be used to address this problem.

The horizontal ensemble builds the model in consecutive chunks. The data stream is separated into *n* consecutive chunks (for example, D_1 and D_2 are history chunks, and D_3 is the current chunk), and the aim of ensemble learning is to build classifiers on these n chunks and to predict data for the chunks that have yet to arrive. Fig. 4 shows the horizontal ensemble classification. The advantage of the horizontal method is that it can handle noise data in the stream because the prediction of newly arriving data chunks depends on the average of different chunks. Even if noise data causes some chunks to deteriorate; the ensemble can still produce relatively stable predictions.

The disadvantage of horizontal ensemble is that the data stream is continuously changing, and the information contained in the previous chunks may be invalid, so that use of these old-concept classifiers will not improve the overall result of prediction.

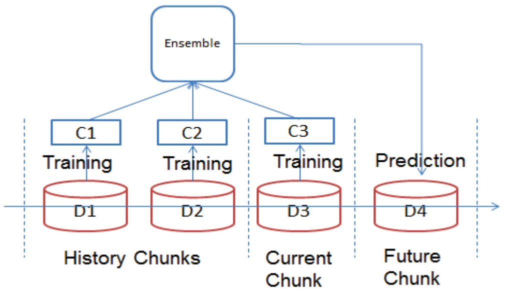

Figure 4 Horizontal ensemble classification

Because of the limitation of both horizontal and vertical ensemble, in this paper we use a novel ensemble classification which uses *k* different learning algorithms to build classifiers on *p* buffered chunks, and then train *k*-by-*p* classifier as Fig. 5 shows. By building an aggregate ensemble, it is capable of solving a real-world data stream containing both concept drifting and data errors.

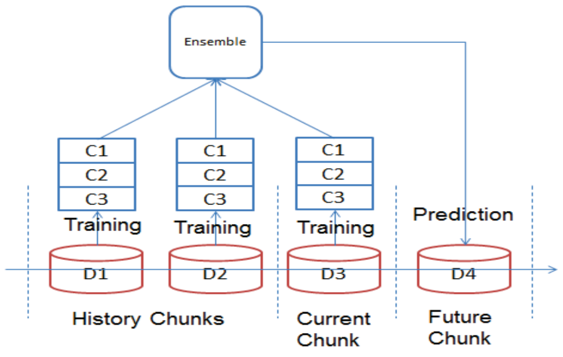

Figure 5 Aggregate ensemble classification

5 Experiments and Analysis

5.1 Experimental Setup

In this subsection, the data set used to validate the effectiveness of our *InfoValue* feature selection algorithm is described. The *InfoValue* feature selection method was evaluated using anomaly data collected from a real system PlanetLab. The PlanetLab is a global research network that supports the development of new network services [17]. We collect 66 system-level metrics, such as CPU load, free memory, and disk usage. The sampling interval is 10 s. There are 50 162 instances, among which 8 700 are labelled as anomalies.

The present experiments were conducted on a 2.6 GHz Inter Dual-Core E5300 with 4 GB of memory running Ubuntu10.4. Sliding-window-based validation was used (window size=1000 instances as authors of [21] suggested), because in real systems, the labelled instances are sorted in chronological order based on the time of collection. Cross-validation was not performed here because it randomly divides the data set into pieces without taking chronological order into account. In this approach, each data chunk is first used to assess the existing model and then used to update the model. It is finally discarded to conserve memory. In this way, sliding window validation is more appropriate for this experiment than other forms of validation.

5.2 Experimental Results

Five experiments were performed to verify the effectiveness of the *InfoValue* feature selection algorithm:

1) Feature subsets were selected using the *InfoValue* feature selection, to determine if the selection was appropriate.
2) Classification accuracy was evaluated on different subsets of features according to their information value, to demonstrate that the subset with higher information value has higher accuracy relative to the subset with lower information value.
3) *InfoValue* was compared to other feature selection techniques to demonstrate its superior performance with respect to time costs.
4) Correlation coefficient was taken into consideration, to demonstrate the subset without redundant features had higher accuracy.
5) *InfoValue* was combined with an ensemble classifier to demonstrate its usefulness in data stream classification.

5.2.1 Selection of features using InfoValue

All monitored attributes are listed in Tab. 1. The InfoValue for each attribute was calculated. There are 66 system-level metrics, including CPU load, free memory, number of live slices, uptime, and disk usage.

As mentioned above, an empirical rule of thumb shows that if *InfoValue* falls within a range of [0; 0,02], then the feature in question is not predictive; if *InfoValue* falls within a range of (0,02; 0,1], it means that this feature has weak predictive power; if *InfoValue* falls within a range of (0,1; 0,3], then the feature has moderate predictive power; and if *InfoValue* falls within a range of

(0,3; +∞), the feature has a strong predictive power [11]. After using *InfoValue* as the feature selection technology, the 11 most important features were selected and used to detecting system anomalies with respect to all 66 features.

Table 1 Features used in anomaly detection

Monitored metrics		
LOAD1	LOAD5	AVAILCPU
UPTIME	FREEMEM	FREEDISK
DISKUSAGE	DISKSIZE	MYFREEDISK
CPUUSE	RWFS	LOAD11
LOAD12	LOAD13	PUKPUKS1-10
NUMSLICE	LIVESLICE	VMSTAT1-17
SERVTEST1	SERVTEST2	BURP
CPUHOG	MEMHOG	TXHOG
RXHOG	PROCHOG	TXRATE
RXRATE	PURKS1-10	MEMINFO1
MEMINFO2	MEMINFO3	

Table 2 System metrics grouped by *InfoValue*

Information Value	Monitored metrics	
[0; 0,02]	DISKSIZE	FREEDISK
	RWFS	LOAD11
	LOAD12	LOAD13
	RXHOG	PROCHOG
	TXRATE	PURKS1-10
(0,02; 0,1]	MEMHOG	TXHOG
	RXRATE	PUKPUKS1-10
	MEMINFO2	MEMINFO3
(0,1; 0,3]	FREEMEM	NUMSLICE
	CPUUSE	BURP
	CPUHOG	VMSTAT2-17
(0,3; +∞)	LOAD1	LOAD5
	LIVESLICE	AVAILCPU
	MYFREEDISK	DISKUSAGE
	VMSTAT1	SERVTEST1
	MEMINFO1	SERVTEST2
	UPTIME	

Tab. 2 shows the attributes grouped by their *InfoValue*. The most important features are like LOAD1 and LOAD5, which are the load average for the minute and five minutes immediately preceding the detection process, respectively, in terms of how many active processes are computed by the CPU. The virtual machines are called slivers, and the set of slivers to which an account has been assigned is called a slice. Any slice that has at least one live sliver (a sliver that uses more than 0,1 % of the CPU per day) is called a live slice [17]. The number of live slices can be used to represent the use of virtual machines on the PlanetLab. AVAILCPU, VMSTAT, and DISKUSAGE are metrics that represent the CPU load, virtual memory states, and disk usage. They are important for providing the node status. CPUHOG and PROCHOG are considered resource hogs, meaning that they cost experiments on the PlanetLab cost the most CPU time and energy. This makes them less important than other features with respect to representing the overall status of one PlanetLab node.

5.2.2 Accuracy of classification of different features

In this subsection, these features are grouped into different subsets according to their *InfoValue*, and then these subsets are used to build a classifier. The first half of the data was used for training and the second half as testing. The results of the assessment of the accuracy of classification were used to validate the effectiveness of the *InfoValue* algorithm. This data set was divided into four subsets with *InfoValues* within the ranges of [0; 0,02], (0,02; 0,1], (0,1; 0,3], and (0,3; +∞). Then, the adjacent ranges were combined to produce ten subsets: [0; 0,02], [0; 0,1], [0; 0,3], [0; +∞), (0,02; 0,1], (0,02; 0,3], (0,02; +∞), (0,1; 0,3], (0,1; +∞), and (0,3; +∞).

The accuracy of the classification was evaluated using 3 criteria as in a previous study [4]: precision, recall, and *f*-measure.

Fig. 6 shows the experimental results of the performance of classification accuracy with different subsets of the PlanetLab data sets using Naïve Bayes as the classifier. The subset of (0,3; +∞) showed the highest accuracy, giving 71,5 % precision, 84,6 % recall, and a 77,5 % *f*-measure score. This result confirms the results of the attribute analysis in Subsection 5.2.1, which showed that the features with *InfoValues* larger than 0,3 have the strongest predictive power. The recall value of subsets (0; 0,02], (0; 0,1], and (0; 0,3] were almost 50 % less than the original feature set, and the *f*-measure scores were only 33 ÷ 66 % of the *f*-measure score of original feature set. This is a very bad classification result. The reason for this is that features with *InfoValue* less than 0,02 are not predictive, so these features may reduce the accuracy of the classification. The recall and *f*-measure scores of (0,02; 0,1] are 73,5 % and 72 %. These recall values within a range of (0,02; 0,3] and (0,02; +∞) were 80 % and 80,4 %. The *f*-measure scores were 77 % and 76,2 %, respectively.

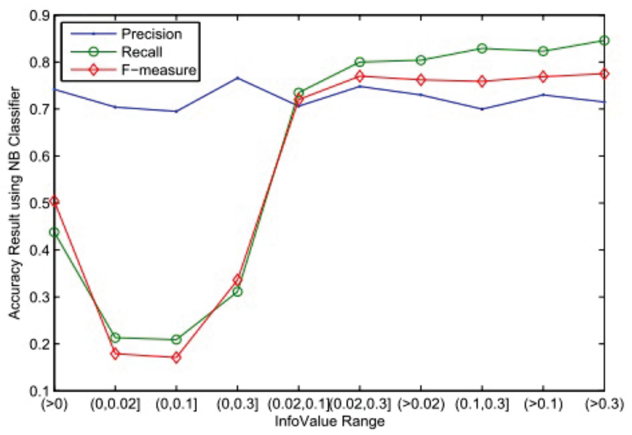

Figure 6 Accuracy of the classification process with different feature subsets

5.2.3 Comparison of InfoValue and other feature selection techniques

Sliding windows with different chunk sizes were used to evaluate different feature selection techniques. In each chunk, the number of instances was 500, 750, 1000, 1250, 1 500, 1750, 2000, 2250, or 2500. As shown in Fig. 7 as the chunk size increased, all algorithms required more time. The *InfoValue* algorithm required less than half of the time required by InfoGain, GainRatio, or ChiSquared. These three technologies rely on a ranking algorithm to decide the importance of each feature. *InfoValue* was even faster than the FCBF feature selection algorithm used in a previous study [6]. The time complexity of FCBF to calculate symmetrical uncertainty (SU) is O(N·lg$N$). For a chunk with M instances, the total cost of FCBF is O(M·N·lg$N$). In this way, FCBF required less time than InfoGain, GainRatio, or ChiSquared when the chunk size was less than 1500, and it required more and more time as chunk size increased.

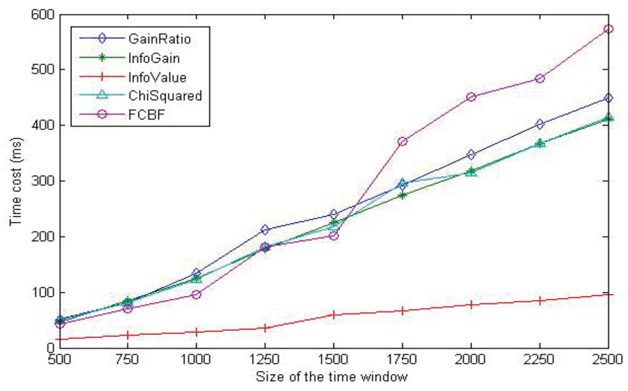

Figure 7 Time costs with different window sizes

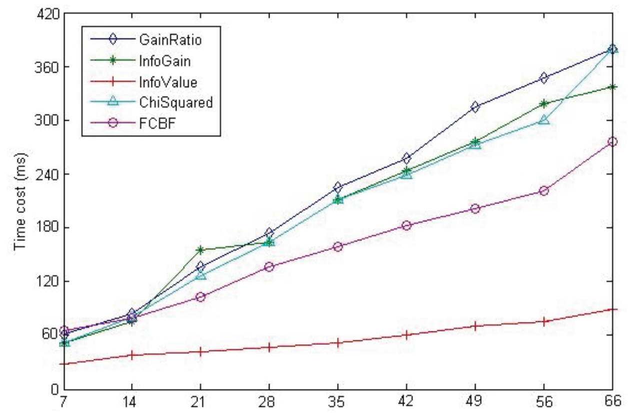

Figure 8 Time cost with different number of features

Fig. 8 shows the time cost of feature selection for features of different sizes where the chunk size is set at 1000. The full data set contains 66 attributes, and feature selection was performed using data sets with different numbers of attributes. The results show that, with smaller number of attributes, feature selection methods require less time. *InfoValue* only needs to keep the statistical summary information of the data, so it is far superior to the other algorithms evaluated here. This also shows that *InfoValue* is suitable for data stream applications.

5.2.4 Accuracy between original and redundant eliminated feature set

In this section, correlation coefficient between features was calculated. We selected the 11 features from Tab. 2 which have information value larger than 0,3 into evaluated. Tab. 3 shows the absolute value of correlation coefficient. From [22] we know that correlation coefficient less than 0,25 means little or no relationship; correlation coefficient between 0,25 and 0,50 means weak to acceptable degree of association; correlation coefficient between 0,50 and 0,75 means moderate to good association; correlation coefficient higher than 0,75 means a good level of association.

As the *InfoValue* feature selection algorithm we proposed in Section 4.3, we eliminated the features which have a correlation coefficient higher than 0,5 with these features already in the selected subset. From Tab. 3, we can see that the following pair of features have a coefficient higher than 0,5, (Fea2, Fea3) =0,76, (Fea2, Fea7) =0,68, (Fea2, Fea8) =0,84, (Fea7, Fea8) =0,76. In this way, we eliminated Fea3, Fea7 and Fea8.

Table 3 Correlation coefficient between 11 features

Correlation	Fea1	Fea2	Fea3	Fea4	Fea5	Fea6	Fea7	Fea8	Fea9	Fea10	Fea11
Fea1	1,00	0,25	0,25	0,00	0,23	0,11	0,07	0,14	0,24	0,27	0,02
Fea2	0,25	1,00	**0,76**	0,44	0,16	0,20	**0,68**	**0,84**	0,11	0,16	0,13
Fea3	0,25	**0,76**	1,00	0,24	0,09	0,15	0,38	0,37	0,08	0,17	0,07
Fea4	0,00	0,44	0,24	1,00	0,02	0,03	0,24	0,46	0,09	0,04	0,22
Fea5	0,23	0,16	0,09	0,02	1,00	0,13	0,09	0,16	0,09	0,03	0,11
Fea6	0,11	0,20	0,15	0,03	0,13	1,00	0,14	0,18	0,08	0,07	0,35
Fea7	0,07	**0,68**	0,38	0,24	0,09	0,14	1,00	**0,76**	0,05	0,06	0,07
Fea8	0,14	**0,84**	0,37	0,46	0,16	0,18	**0,76**	1,00	0,10	0,09	0,16
Fea9	0,24	0,11	0,08	0,09	0,09	0,08	0,05	0,10	1,00	0,05	0,06
Fea10	0,27	0,16	0,17	0,04	0,03	0,07	0,06	0,09	0,05	1,00	0,04
Fea11	0,02	0,13	0,07	0,22	0,11	0,35	0,07	0,16	0,06	0,04	1,00

Table 4 Accuracy between original and redundant eliminated feature set

	Original feature set	Feature set that eliminate the redundant features
Precision	71,5 %	73,8 %
Recall	84,6 %	88,9 %
f-measure	77,5 %	78,9 %

Tab. 4 showed the classification accuracy with original feature set and feature set eliminated the redundant features. We used Naïve Bayes as the classifier.

The original subset of features with information value in the range of (0,3; +∞) showed accuracy with 71,5 % precision, 84,6 % recall, and a 77,5 % *f*-measure score. And while we eliminated these redundant features, the accuracy grew into 73,8 % precision, 88,9 % recall, and a 78,9 % *f*-measure score. This result shows that while removing the redundant features, the classification accuracy can be better.

5.2.5 Comparison of InfoValue with ensemble classification to other methods of data stream classification

Classification accuracy and classification time were evaluated. Precision was defined as the proportion of successful classifications per instance in chunk D_{n+1}. Recall served as the probability of each real state to be successfully classified in the chunk D_{n+1}; and *f*-measure as the harmonic mean of precision and recall. This process is performed $n-1$ times, allowing calculation of the average precision, recall and *f*-measure. A good classifier should have high average precision, high average recall and high average *f*-measure.

Classification accuracy and classification time were evaluated. Precision was defined as the proportion of successful classifications per instance in chunk D_{n+1}. Recall served as the probability of each real state to be successfully classified in the chunk D_{n+1}; and *f*-measure as the harmonic mean of precision and recall. This process is performed $n-1$ times, allowing calculation of the average precision, recall and *f*-measure. A good classifier should have high average precision, high average recall and high average *f*-measure.

Table 5 Accuracy and time cost between different classification methods

	AWE [21]			HEFT-Stream [6]	InfoValue-EC
	C4.5	NaiveBayes	Logistic		
Precision	71,59 %	71,64 %	69,53 %	72,38 %	73,98 %
Recall	61,78 %	68,79 %	55,34 %	66,01 %	75,83 %
f-measure	59,99 %	68,16 %	57,75 %	64,76 %	73,73 %
Processing time (s)	11,625	10,407	20,391	9,563	8,891

In Tab. 5, AWE ensembles showed the worst accuracy and the largest running time. This is because AWE use single classifiers as its members and when there are concept drifts, the member classifier becomes outdated and the accuracy is degraded. In the present approach, *InfoValue* feature selection addresses these problems and employs heterogeneous classifiers so that it can adapt to any concept drift that may occur in the data streams. With the integrated *InfoValue* feature selection method, the *InfoValue* feature selection only works on the most informative feature subset, so it can improve the accuracy of classification and reduce the processing time. *InfoValue* feature selection showed the highest recall value and *f*-measure score and the lowest run time of any method evaluated here.

6 Conclusions and future work

In the present paper, we improved the information value based feature selection algorithm with correlation coefficient to remove the redundant features. This feature selection algorithm was used over ensemble classification to detect anomalies in a large-scale system, specifically PlanetLab. The results of the experiment indicate that the feature selection algorithm *InfoValue* performed better with respect to time and space than previous feature selection technologies. In this way, *InfoValue* is suited to processing large amounts of multi-dimensional data in real time. This algorithm may be further improved by the removal of redundant features using correlations. To keep this algorithm fast and lightweight, a data structure called a feature correlation summary (*FCS*) was used. In this way, the feature selection algorithm only needs to retain the statistical information of each feature, and the time and space costs of redundancy calculations are low, which is very important in data stream applications such as the detection of system anomalies.

InfoValue was integrated with ensemble classification for the detection of anomalies in a real-world system. Feature selection helped extract the most informative feature subset from the original feature set, which can accelerate the learning speed and improve the accuracy of the model. A series of extensive experiments showed that this classification system outperforms state-of-art learning algorithms with respect to data streams.

In the present paper, systems were classified only as either normal or abnormal, but in the future the model should be improved so that it may identify specific types of anomalies. *InfoValue* can be integrated with specific anomaly prediction technologies to alert users to anomalies before they occur.

Acknowledgements

This research was supported by Key Projects in the National Science and Technology Pillar Program (No. 2012BAH13F02, 2012BAH13F04), and the Ministry of Industry and Information Technology of China (No. 2010ZX01042-002-003-001).

7 References

[1] Ray, A. Symbolic dynamic analysis of complex systems for anomaly detection. // Signal Processing. 84, 7(2004), pp. 1115-1130.
[2] Michalak, K.; Kwasnicka, H. Correlation-based feature selection strategy in classification problems. // International Journal of Applied Mathematics and Computer Science. 16, 4(2006), p. 503.
[3] Chen, J.; Huang, H.; Tian, S. et al. Feature selection for text classification with Naive Bayes. // Expert Systems with Applications. 36, 3(2009), pp. 5432-5435.
[4] Han, J.; Kamber, M.; Pei, J. Data mining: concepts and techniques. // Morgan Kaufmann, 2006.
[5] Liu, H.; Yu, L. Toward integrating feature selection algorithms for classification and clustering. // Knowledge and Data Engineering, IEEE Transactions on. 17, 4(2005), pp. 491-502.
[6] Nguyen, H. L.; Woon, Y. K.; Ng, W. K. et al. Heterogeneous ensemble for feature drifts in data streams. // PAKDD'12 Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining, Kuala Lumpur, Malaysia, May 29 - June 1, 2012, pp. 1-12.
[7] Fumera, G.; Roli, F. A theoretical and experimental analysis of linear combiners for multiple classifier systems.

// Pattern Analysis and Machine Intelligence, IEEE Transactions on. 27, 6(2005), pp. 942-956.

[8] Zhao, Y.; Tan, Y.; Gong, Z. et al. Self-correlating predictive information tracking for large-scale production systems. // Proceedings of the 6th International Conference on Autonomic computing. ACM, Barcelona, Spain June 15-19, 2009, pp. 33-42.

[9] Wu, X.; Yu, K.; Wang, H. et al. Online streaming feature selection. // Proceedings of the 27nd International Conference on Machine Learning, Haifa, Israel on June 21-24, 2010.

[10] Masud, M. M.; Chen, Q.; Khan, L. et al. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. // IEEE Transactions on Knowledge and Data Engineering. 2012.

[11] Alec Cheng, Royal Bank of Scotland, NB; Moez Hababo, Royal Bank of Scotland, NB; Ray Falk, Royal Bank of Scotland, NB, Variable Selection in the Credit Card Industry, SAS Conference Proceedings, 2006.

[12] Yu, L.; Liu, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. // Machine Learning-International Workshop Then Conference. 20, 2(2003), p. 856.

[13] Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. // ACM Computing Surveys (CSUR). 41, 3(2009), p. 15.

[14] Kiciman, E.; Fox, A. Detecting application-level failures in component-based internet services. // Neural Networks, IEEE Transactions on. 16, 5(2005), pp. 1027-1041.

[15] Liang, Y.; Zhang, Y.; Xiong, H. et al. Failure prediction in IBM BlueGene/L event logs. // Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. IEEE, (2007), pp. 583-588.

[16] Zhang, P.; Muccini, H.; Polini, A. et al. Run-time systems failure prediction via proactive monitoring. // Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, (2011), pp. 484-487.

[17] Kim, W.; Roopakalu, A.; Li, K. Y. et al. Understanding and characterizing PlanetLab resource usage for federated network testbeds. // Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. (2011), pp. 515-532.

[18] Tan, Y.; Gu, X. On predictability of system anomalies in real world. // Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on. (2010), pp. 133-140.

[19] Patcha, A.; Park, J. M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. // Computer Networks. 51, 12(2007), pp. 3448-3470.

[20] Aggarwal, C. C.; Han, J.; Wang, J. et al. A framework for clustering evolving data streams. // Proceedings of the 29th International Conference on Very large data bases-Volume 29. VLDB Endowment, 2003, pp. 81-92.

[21] Wang, H.; Fan, W.; Yu, P. S. et al. Mining concept-drifting data streams using ensemble classifiers. // Proceedings of the ninth ACM SIGKDD International Conference on Knowledge discovery and data mining. ACM, 2003, pp. 226-235.

[22] Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. // The Journal of Machine Learning Research. 3, (2003), pp. 1157-1182.

[23] Luo, G.; Wu, K.-L.; Yu, P. S. Answering linear optimization queries with an approximate stream index. // Knowledge and Information Systems. 20, 1(2009), pp. 95-121.

Authors' addresses

Xiaozhen Zhou
Zhejiang University
38 Zheda Road
Hangzhou 310012, China
zxztc@zju.edu.cn

Shanping Li, Ph.D.
Zhejiang University
38 Zheda Road
Hangzhou 310012, China
shan@zju.edu.cn

Cheng Chang
Zhejiang University
38 Zheda Road
Hangzhou 310012, China
changcheng316@zju.edu.cn

Jianfeng Wu, Ph.D.
Technology Center of Shanghai Stock Exchange
528 South Pudong Road
Shanghai 200120, China
jfwu@sse.com.cn

Kai Liu
Technology Center of Shanghai Stock Exchange
528 South Pudong Road
Shanghai 200120, China
kliu@sse.com.cn