Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 179390, 10 pages
http://dx.doi.org/10.1155/2013/179390

Research Article

A Novel System Anomaly Prediction System Based on Belief Markov Model and Ensemble Classification

Xiaozhen Zhou, Shanping Li, and Zhen Ye

College of Computer Science and Technology, Zhejiang University, Hangzhou 310012, China

Correspondence should be addressed to Xiaozhen Zhou; zhouxiaozhen329@gmail.com

Received 17 March 2013; Revised 13 July 2013; Accepted 31 July 2013

Academic Editor: Yingwei Zhang

Computer systems are becoming extremely complex, while system anomalies dramatically influence the availability and usability of systems. Online anomaly prediction is an important approach to manage imminent anomalies, and the high accuracy relies on precise system monitoring data. However, precise monitoring data is not easily achievable because of widespread noise. In this paper, we present a method which integrates an improved Evidential Markov model and ensemble classification to predict anomaly for systems with noise. Traditional Markov models use explicit state boundaries to build the Markov chain and then make prediction of different measurement metrics. A Problem arises when data comes with noise because even slight oscillation around the true value will lead to very different predictions. Evidential Markov chain method is able to deal with noisy data but is not suitable in complex data stream scenario. The Belief Markov chain that we propose has extended Evidential Markov chain and can cope with noisy data stream. This study further applies ensemble classification to identify system anomaly based on the predicted metrics. Extensive experiments on anomaly data collected from 66 metrics in PlanetLab have confirmed that our approach can achieve high prediction accuracy and time efficiency.

1. Introduction

As computer systems are growing increasingly complicated, they are more vulnerable to various anomalies such as performance bottlenecks and service level objective (SLO) violations [1]. Thus, it requires the computer systems to be more capable of managing anomalies under time pressure, and avoiding or minimizing the system unavailability by monitoring the computer systems continuously. Anomaly management methods can be classified into two categories: passive methods and proactive methods. Passive methods notify the system administrator only when errors or faults are detected. These approaches are appropriate to manage anomalies that can be easily measured and fixed in a simple system. However, in nowadays dynamic and complex computer systems, detecting some anomalies may have a high cost, which is unacceptable for continuously running applications. Proactive methods take preventive actions when anomalies are imminent; thus, they are more appropriate for systems that need to avert the impact of anomalies and achieve continuous operation. Nowadays proactive methods are preferred in both academic research and real world applications.

Previous work has addressed the problem of system anomaly prediction, which can be categorized as data-driven methods, event-driven methods, and symptom-driven methods [2].

Event-driven methods directly analyze the error or failure that events report and use error reports as input data to predict future system anomaly. Salfner and Malek use error reports as input and then perform a trend analysis to predict the occurrence of failure in a telecommunication system by determining the frequency of error occurrences [3]. Kiciman and Fox use decision tree to identify faulty components in J2EE application server by classifying whether requests are successful or not. These approaches have the basic assumption that anomaly-prone system behavior can be identified by characteristics of anomaly [4]. This is why only reoccurring anomaly presented in the error report can be predicted by event-driven method.

Data-driven methods learn from the temporal and spatial correlation of anomaly occurrence. They aim at recognizing

the relationship between upcoming failures and occurrence of previous failures. Zhang and Ma use modified KPCA method to diagnose anomalies in nonlinear processes [5]. In nonlinear fault detection scenario, they utilize statistical analysis to improve the learning techniques [6], which is also applicable for large scale fault diagnosis processes [7]. Liang et al. exploit these correlation characteristics of anomaly on IBM's BlueGene/L [8]. They find that the occurrence of a failure is strongly correlated to the time stamp and the location of others in a cluster environment. Zhang et al. propose a hybrid prediction technique which uses model checking techniques; an operational model is explored to check if a desirable temporal property is satisfied or violated by the model itself [9]. To conclude, the basic idea of data-driven methods is that upcoming anomalies are from the occurrence of the previous ones.

Symptom-driven methods analyze some workload-related data such as input workload and memory workload in order to predict further system resource utilization. Tan and Gu [10] 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 Naïve Bayesian classification. Luo et al. [11] build autoregressive model using various parameters from an Apache webserver to predict further system resource utilization; failures are predicted by detecting resource exhaustion.

Efficient proactive anomaly management relies on the system monitoring data, and the metric system generated by monitor infrastructures are continuously arriving and invariably noisy, so one big challenge is to provide high accurate and good and efficient system anomaly prediction for noisy monitoring data stream. Recently, some approaches have been proposed for system anomaly prediction using discrete-time Markov chain (DTMC) [10, 12]. However, their work does not consider the issue that monitoring data may oscillate around the real value as we mentioned previously. DTMC which uses explicit state boundaries will lead to significantly different values even when the metrics oscillation around the boundaries is very slight. Soubaras [13] proposed Evidential Markov chain model which extends DTMC to overcome the noise value around explicit state boundaries problem caused by inaccuracies monitoring metrics. The problem of Evidential Markov chain is that although it works excellently in a static data scenario, it cannot be applied directly to stream data. Its fixed transition matrix is too restrictive for continuously changing stream data and brings in enormous amount of calculation.

In this paper, we present the design and implementation of an approach to solve the system anomaly prediction problem on noise data stream. We first present an improved belief Markov chain (BMC) to fit into a data stream scenario. We use a stream-based k-means clustering algorithm [14] to dynamically maintain and generate Markov transition matrix. Only information of microclusters is stored after clustering, and new comers will falls into or newly establish one of the k groups. Compared to Evidential Markov chain method, by which all the data has to be stored and recalculated every time when new one arrives to get Markov state, our approach is time efficient and more feasible in a highly dynamic and complex system. We then employ aggregate ensemble classification method [15] to determine whether the system will turn into anomaly in the future. Aggregate ensemble classification can address the incorrect anomaly mark problem in a continuously running system.

Extensive experiments on PlanetLab dataset [16] of different parameter settings show that averagely BMC achieves 14.8% smaller mean prediction error than DTMC method in various previous works [10, 12, 17, 18]. Our system anomaly prediction method (SAPredictor), which combines BMC and aggregate ensemble classification, is proved to achieve better prediction performance than other prediction models, for example, DTMC+Naïve Bayes, DTMC+KNN, and DTMC+C4.5. SAPredictor demonstrates the best performance in the three key criteria, namely, 71.6% for precision, 84.6% for recall, and 77.5% for F-measurement.

The main contributions of this paper are summarized as follows.

(1) We propose the belief Markov chain by improving the Evidential Markov model using a stream-based k-means clustering algorithm and make it more suitable for system metrics prediction on noisy data stream.

(2) We integrate belief Markov chain and aggregate ensemble classification as SAPredictor to predict system anomaly.

(3) We validate the effectiveness of SAPredictor by extensive experiments on real system data.

The rest of this paper is organized as follows. Section 2 introduces our SAPredictor method. Section 3 demonstrates the experiments and analyzes the results. Finally, we conclude and give some future research directions in Section 4.

2. Approach Overview

In this section, we present the detailed design of SAPredictor. We first describe the problem of system anomaly prediction and then propose our SAPredictor method, which is composed by the two components: belief Markov chain model and aggregate ensemble classification model. Belief Markov chain model is used to predict the changing pattern of measurement metrics; aggregate ensemble classification is a supervised learning method which employs multiple classifiers and combines their predictions. In this work, we use sliding window to partition the system metrics stream into some chunks and then train the belief Markov chain and aggregate ensemble learning model by the history. The future system status is predicted by putting future metrics as input into the classification model.

2.1. Problem Statement. For a system, we have a vector of observations at time t for the system metrics, $Y_t = [y_{1,t}, y_{2,t}, \ldots y_{n,t}]$. Y_t is a vector that contains N system metric time series at time t, namely, $y_{i,t}$ $(i = 1, 2, \ldots, n)$, i is the ith metric. We label Y_t at time t as normal (state 0) or anomaly (state 1) by monitoring the system state at time t. The system anomaly prediction problem we focus on in this paper is that

whether Y_t will fall into anomaly status in the next β steps, where $\beta > 0$ and $\beta \in N$. To solve this problem, we need to first forecast the future value of $y_{i,t+\beta}$ for each metric. Then, we train ensemble classifier EC based on a sliding window $[y_{i,t-d+1}, \ldots y_{i,t}]$ of Y_t, where d is the size of sliding window. Finally, we use EC to test on $y_{i,t+\beta}$ $(i = 1, 2 \ldots, n)$ and predict the state label of $Y_{t+\beta}$.

2.2. SAPredictor Approach. Figure 1 describes the SAPredictor system anomaly prediction approach. N measurement metrics (e.g., CPU, memory, I/O usage, network, etc.) are collected from the system continuously. Then, the collected system metrics streams are partitioned into some chunks by sliding window. The current and history chunks are used to train the belief Markov chain model and aggregate ensemble learning model. Then, the future system metrics is predicted by the belief Markov chain model, and having these metrics as input into the aggregate ensemble classification model, we can ascertain whether the system will fall into anomaly in the future. Belief Markov chain and aggregate ensemble classification will be presented in the following subsections.

2.2.1. System Metrics Value Prediction. In this section, we first introduce why the Evidential Markov chain which is based on the Dempster-Shafer theory [19] is preferred over discrete-time Markov chain in dealing with system anomaly prediction for noisy data, and then we explain the advantages of our belief Markov chain method compared to Evidential Markov method in a data stream environment.

When we build discrete-time Markov chain model, it is necessary to divide all the data into discrete states. Traditional discretion techniques used in discrete-time Markov chain include equal-width and equal-depth. Both techniques generate status with explicit boundaries using all the data. However, the system metrics being monitored are usually imprecise due to system noise and measurement error. Thus, discrete-time Markov chain which uses explicit boundary to divide the states will generate highly different prediction results even if their initial values are almost the same. Evidential Markov model [13] has made big improvement by being capable of coping with noisy data. Following is an example of explicit boundary problem in discrete-time Markov chain.

In one possible situation, we have a metric ranging in $[0, 150]$, and then we use equal-width approach to discrete the range into three bins, namely, $[0, 50)$, $[50, 100)$, and $[100, 150]$. S_1, S_2, and S_3 denote the states when metric is in $[0, 50)$, $[50, 100)$, and $[100, 150]$, respectively. The transition matrix for the metric is a $3 * 3$ matrix:

$$M = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0.1 & 0.2 & 0.7 \\ 0.1 & 0.5 & 0.4 \end{pmatrix}. \quad (1)$$

Here, each element a_{ij} in matrix M denotes the probability of transition from state i to state j. When we use discrete-time Markov chain to predict future value, a vector $\pi = [\pi_{t,S_1}, \pi_{t,S_2}, \pi_{t,S_3}]$ is needed to denote the probability of the metric in each state at time t. If we have an initial value 99 which is in state S_2, then the corresponding probability vector is $\pi_0 = [0, 1, 0]$. We can calculate the probability vector π_1 after one time unit as

$$\pi_1 = \pi_0 * M = [0, 1, 0] * \begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0.1 & 0.2 & 0.7 \\ 0.1 & 0.5 & 0.4 \end{pmatrix} = [0.1, 0.2, 0.7]. \quad (2)$$

Here, the probability vector π_1 represents that the initial value will transfer into S_3 most likely, and the predicted value after one step will be $125 = (100 + 150)/2$ as the mean of state S_3. However, if the initial value turns to be 101, then the vector π_0' will be $[0, 0, 1]$. By applying (2) again, it turns out that the prediction value will stay in state S_2 with the predicted value of $75 = (50 + 100)/2$ in the next step:

$$\pi_1' = \pi_0' * M = [0, 0, 1] * \begin{pmatrix} 0.3 & 0.2 & 0.5 \\ 0.1 & 0.2 & 0.7 \\ 0.1 & 0.5 & 0.4 \end{pmatrix} = [0.1, 0.5, 0.4]. \quad (3)$$

Note that there is only a slight difference between 99 and 101 in the initial value, yet the forecasted value after one step is in large difference from 75 to 125.

As the example shows, discrete-time Markov chain uses explicit state boundaries, and it will have very different prediction value if the original metric is around the state boundary. To solve this problem, we propose belief Markov chain based on the Dempster-Shafer theory. The Dempster-Shafer theory is an inaccurate inference theory. It can handle the uncertainty caused by unknown prior knowledge and extend the basic event space to its power set. The detailed definitions for Dempster-Shafer [19] are as follows.

Definition 1 (frame of discernment). Suppose that P is the exhaustive set of random variable X, so $P = \{x_1, x_2, \ldots x_j\}$ and the elements in P are mutually exclusive. Then, the set of all possible subsets of P is called a frame of discernment A:

$$\begin{aligned} A = &\left(\phi, \{x_1\}, \{x_2\}, \ldots, \{x_j\}, \{x_1, x_2\}, \{x_1, x_j\}, \ldots, \right. \\ &\left. \{x_2, x_3\}, \ldots, \{x_2, x_j\}, \ldots, \{x_1, x_2, \ldots, x_j\}\right) \\ = &\left(A_0, A_1, A_2, \ldots, A_j\right). \end{aligned} \quad (4)$$

We use A_k $(k \in N, k \in [0, j])$ to represent the subset in power set of P which contains k elements.

Definition 2 (mass function). Have P and A, for every subset of P; if the following statements satisfy, then the function F is called the mass function on A:

$$\begin{aligned} F(\phi) &= 0, \\ \sum F(A_k) &= 1, \\ F(A_k) &\in [0, 1], \quad A_k \in 2^P. \end{aligned} \quad (5)$$

Definition 3 (transferable belief model). Suppose that we have discernment frame A and mass function F on P. Then, the probability for each random variable X in P can be calculated by transferable belief model:

$$T(x) = \sum \frac{F(A)}{|A|}. \quad (6)$$

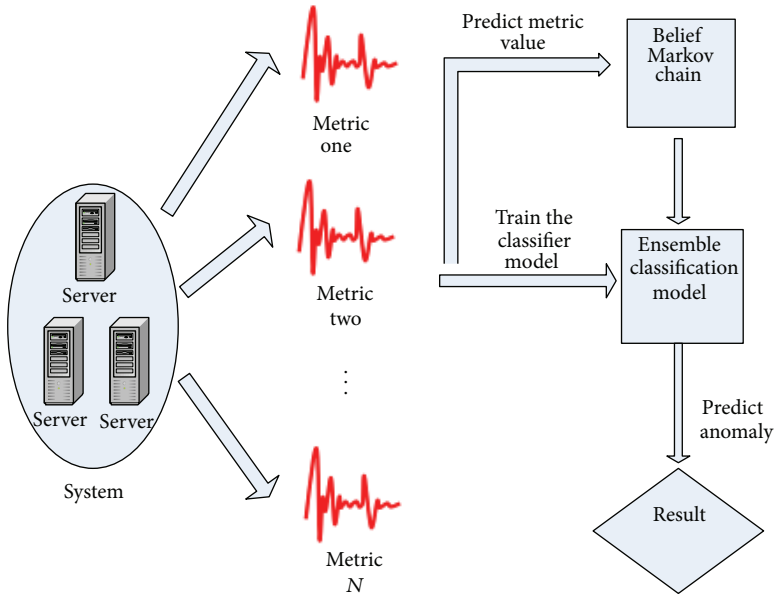

Figure 1: System anomaly prediction approach.

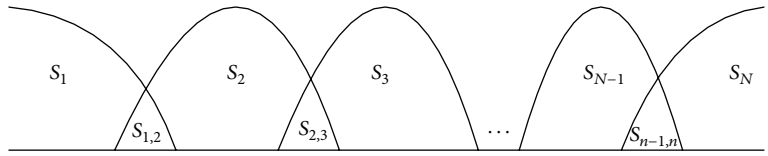

Figure 2: State-space with cross-state region.

The subset of A includes both single event set $\{x_i\}$ and multiple event combinations $\{x_i, x_j, \ldots, x_k\}$. This is why we need transferable belief model to calculate the probability of one single random variable.

Figure 2 illustrates a metric divided into N states, $S = \{S_1, S_2, S_3, \ldots, S_n\}$, and each pair of adjacent states has a state $S_{i,i+1}$ which means that the value is in cross-region between state i and state $i+1$. When using BMC model to predict, the initial metric may belong to a single state entirely or belong to the cross-region of two adjacent states. So, the discernment frame of this problem can be simplified to

$$A_{\mathrm{BMC}} = (\phi, \{x_1\}, \{x_2\}, \ldots, \{x_n\}, \{x_1, x_2\}, \{x_2, x_3\}, \ldots, \{x_{n-1}, x_n\}). \tag{7}$$

Then, we declare the mass function to assign probability to each subset in A_{BMC}. Any function that satisfies (5) can be used as mass function. The probability of each event in P can be calculated as

$$p(x_i) = \begin{cases} F(x_i) + \dfrac{F(x_i, x_{i+1})}{2} & i = 1, \\ \dfrac{F(x_{i-1}, x_i)}{2} + F(x_i) + \dfrac{F(x_i, x_{i+1})}{2} & 1 < i < n, \\ \dfrac{F(x_{i-1}, x_i)}{2} + F(x_i) & i = n. \end{cases} \tag{8}$$

At last, we need to infer the transition matrix which describes the probabilities of moving from one state to others as we did in discrete-time Markov chain. Each element p_{ij} of transition matrix M_{BMC} denotes the probability of the currently state S_i, and then it moves to state S_j. It can be calculated by

$$p_{ij} = \frac{\sum_{t=1}^{n} (p(i)_t * p(j)_{t+1})}{\sum_{j \in A_{\mathrm{BMC}}} \sum_{t=1}^{n} (p(i)_t * p(j)_{t+1})}. \tag{9}$$

However, the Evidential Markov chain needs to store all the data and recalculate the Markov state when new data arrives, this is not time efficient and feasible for the systems that need real-time response, especially for data stream applications. Thus, we improve Evidential Markov chain using stream-based k-means clustering method. The arriving data points can be mapped onto k states using data stream clustering algorithm where each cluster represents a Markov state. For each cluster i representing state s_i, we need to store a transition count vector c_i. All transition counts can be seen as a $K * K$ transition count matrix C where K is the number of clusters. As we use stream clustering, there is a list of operations for cluster: adding a new data to an existing cluster, creating a new cluster, deleting clusters, merging clusters, and splitting clusters. And we use Jaccard [20] as a dissimilarity threshold to detect clusters. Thus, the k states are adaptively changing to fit the arriving data, which

is also an advantage compared to Evidential Markov chain method.

2.2.2. System Status Classification. In this section, we first introduce why we choose ensemble classification to forecast the system status and then how the aggregate ensemble method can address the concept drift and noisy data problem in data stream. Tan and Gu [10] apply single statistical classifier on static dataset to make classification. Though this approach works well on static dataset, it is not applicable in a dynamic environment where system logs are generated continuously, and even the underlying data generating mechanism and cause of anomaly are constantly changing. To capture the time-evolving anomaly pattern, many solutions have been proposed to build classification models from data stream.

One simple model is using online incremental learning [11, 21]. The incremental learning methods deliver a single learning model to represent an entire data stream and update the model continuously when new data arrives. Ensemble classification always regards the data stream as several separated data chunks and trains classifiers based on these chunks using different learning algorithms, and then ensemble classifier is built through voting of these base classifiers. Although these models are being proved to be efficient and accurate, they depend on the assumption that data stream being learned is high quality and without consideration of data error. However, in real world applications, like system monitoring data stream and sensor network data stream, they are always containing erroneous data values. As a result, the tradition online incremental model is likely to lose accuracy in the data stream which has error data values.

Ensemble learning is a supervised method which employs multiple learners and combines their predictions. Different from the incremental learning, ensemble learning trains a number of models and gives out final prediction based on classifiers voting. Because the final prediction is based on a number of base classifiers, ensemble learning can adaptively and rapidly address the concept drift and error data problem in data stream. Based on the above reason, we choose to use ensemble classification.

In summary, the ensemble of classification can be categorized into two categories: horizontal ensemble and vertical ensemble classification [15]. The horizontal ones build classifiers using several buffered chunks, while the vertical ones build classifiers using different learning algorithm on the current chunks.

Vertical ensemble is shown in Figure 3. It uses r different classification algorithms (e.g., we simply set $r = 3$) to build classifier on the current chunk and then use the results of these classifiers to form an ensemble classification model. The vertical ensemble only uses the current chunk to build classifiers, and the advantage of vertical ensemble classification is that it uses different algorithms to build the classifier model which can decrease the bias error between each classifiers. However, the vertical ensemble assumes that the data stream is errorless. As we discussed before, the real-world data stream always contains error. So, if the current chunk is mostly containing noise data, then the result may have severe performance deterioration. To address this problem, horizontal ensemble which uses multiple history chunks to build classifiers is employed.

Table 1: Monitoring metrics used for anomaly prediction.

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	MEMINFO1-3

Horizontal ensemble is showed in Figure 4. The data stream is separated into n consecutive chunks (e.g., 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 predict data in the yet-to- arrive chunk (D_4 in this picture). The advantage of horizontal structure is that it can handle the noise data in the stream because the prediction of newly arriving data chunk depends on the average of different chunks. Even if the noise data may deteriorate some chunks, the ensemble can still generate relatively accurate prediction result.

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 these old-concept classifiers will not improve the overall result of prediction.

Because of the limitation of both horizontal and vertical ensembles, 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 Figure 5 shows. By building an aggregate ensemble, it is capable of solving a real-world data stream containing both concept drifting and data errors.

3. Experiment and Result

3.1. Experiment Setup. We evaluate our SAPredictor method on the anomaly data collected from realistic system: PlanetLab. The PlanetLab [22] is a global research network that supports the development of new network services. The PlanetLab data set [16] which we use in this paper contains 66 system-level metrics such as CPU load, free memory and disk usage, shown by Table 1. The sampling interval is 10 seconds. There are 50162 instances, and among which 8700 are labeled as anomalies.

Our experiments were conducted on a 2.6-GHz Inter Dual-Core E5300 with 4 GB of memory running Ubuntu10.4. We use sliding window (window size = 1000 instances) based validation because in real system, the labeled instances are sorted in chronological order of collecting time. The reason

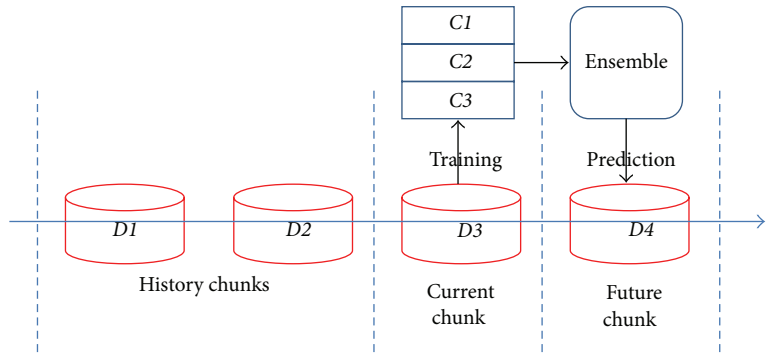

Figure 3: Vertical ensemble classification.

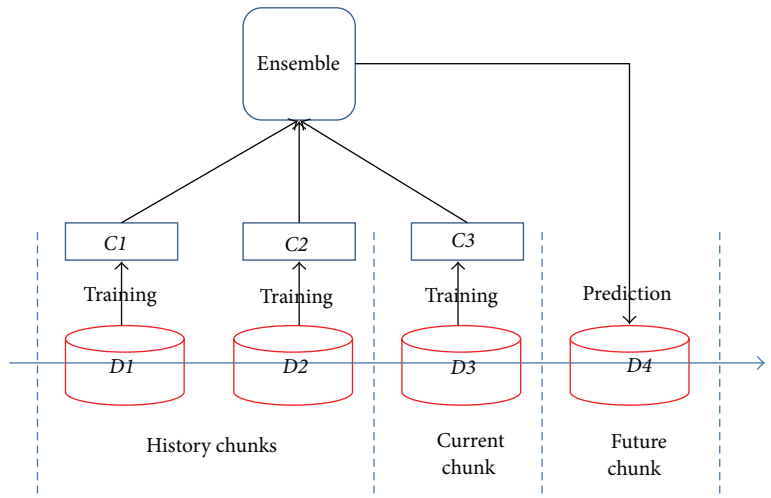

Figure 4: Horizontal ensemble classification.

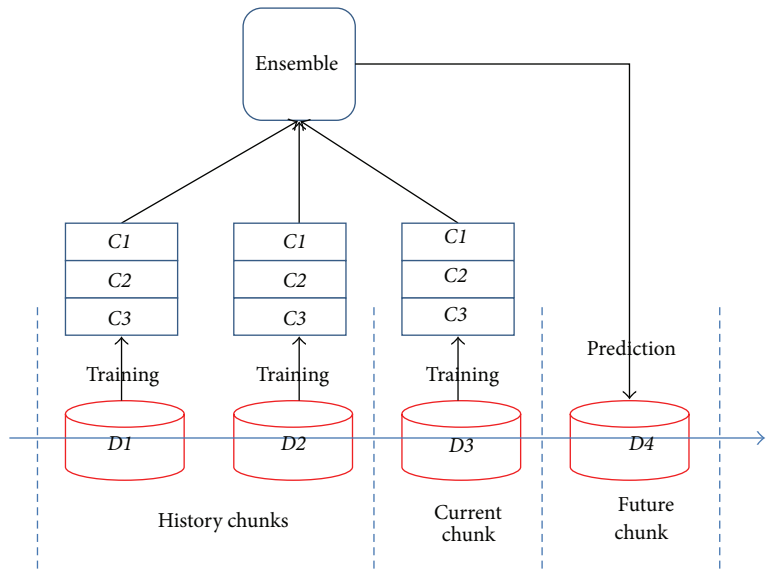

Figure 5: Aggregate ensemble classification.

Table 2: Comparison between BMC and DTMC at 10% noise.

Noise = 10%	Mean prediction error (%)		Improvement
Time units	BMC	DTMC	
1	14.04%	16.60%	15.42%
2	18.78%	20.96%	10.40%
3	22.98%	24.35%	5.63%
4	26.75%	27.83%	3.88%
5	27.57%	28.50%	3.26%

Table 3: Comparison between BMC and DTMC at 20% noise.

Noise = 20%	Mean prediction error (%)		Improvement
Time units	BMC	DTMC	
1	14.67%	17.42%	15.79%
2	19.61%	21.85%	10.25%
3	23.61%	25.23%	6.42%
4	27.66%	28.73%	3.72%
5	28.42%	29.42%	3.40%

Table 4: Comparison between BMC and DTMC at 30% noise.

Noise = 30%	Mean prediction error (%)		Improvement
Time units	BMC	DTMC	
1	15.34%	18.27%	16.04%
2	20.47%	22.77%	10.10%
3	24.36%	26.14%	6.81%
4	27.07%	28.67%	5.58%
5	29.29%	30.38%	3.59%

Table 5: Comparison between BMC and DTMC at 40% noise.

Noise = 40%	Mean prediction error (%)		Improvement
Time units	BMC	DTMC	
1	15.89%	19.15%	17.02%
2	21.43%	23.76%	9.81%
3	25.14%	27.11%	7.27%
4	27.93%	29.69%	5.93%
5	30.22%	31.43%	3.85%

Table 6: Comparison between BMC and DTMC at 50% noise.

Noise = 50%	Mean prediction error (%)		Improvement
Time units	BMC	DTMC	
1	16.38%	20.07%	18.36%
2	22.26%	24.79%	10.21%
3	25.90%	28.12%	7.89%
4	28.83%	30.75%	6.24%
5	31.23%	32.48%	3.85%

that we do not use cross-validation is that it randomly divides the dataset into pieces without considering the chronological order. Under such circumstances, it is possible that current data is used to predict past data, which does not make sense. Thus, sliding window validation is more appropriate for our experiments.

3.2. The Metrics Prediction Accuracy. Short term predictions are helpful to prevent potential disasters and limit the damage caused by system anomalies. Usually, predicting near term future is more clever and successful than long term predictions [5]. So, in our experiment, we assess system state prediction in short term.

In this experiment, we choose k-means discretion technique to create state boundary. The reason is that by k-means the state will have more adjacent data compared to the state discrete by equal-width and equal-depth, when we divide the data into k clusters, because each middle point of the cluster will be used as a state. We set the size of bins as 5, 10, 15, ... to 30, and evaluate the quality of metric prediction by mean prediction error (MPE) as the study by Tan and Gu [10]:

$$\text{MPE} = \frac{\left(\sum_D \sum_{m=1}^{i} \left(\left|x_m - \widehat{x}_m\right| / x_m\right)\right)}{(|m| * |D|)}. \tag{10}$$

D is the test dataset, and $|D|$ is the number of instances in D. $|m|$ is the number of system metrics, and x_m is the actual value of metric m. $\widehat{x}_m$ is the prediction value of metric m, which is represented by the mean value of samples in that bin. The less the value of MPE, the more accurate the predictor.

We assess the MPE in near term future (1–5 time units ahead) for different bin sizes (5, 10, 15, 20, 25, and 30) on PlanetLab dataset. Figure 6 shows the MPE of PlanetLab for time units (1–5) with bin size of 20. From these two figures, we have the following observations: (1) BMC can achieve less prediction error than DTMC from time units 1 to 5. One step prediction has the most notable advantage, and the advantage decreases slightly as time goes on, which means that our algorithm fits better when the forecast period is shorter; (2) BMC and DTMC both lose prediction accuracy as time goes by, which indicates that predict anomaly in longer term is more challenging.

Figure 7 shows the MPE of PlanetLab with different bin sizes (5, 10, 15, 20, 25, and 30) when time unit is one. From these figures, we can see that both methods have higher MPE with less number of bins. The reason is that less number of states tends to group a larger range of data into a bin. Since the mean of the bin is used as the prediction value, the gap between the prediction value and the real value will be enlarged.

In Tables 2, 3, 4, 5, and 6, we compare the mean prediction error of DTMC and BMC under different noise percentage. The noise percentage n means that the monitoring value at state i oscillates around the true value y_i in the range of $[y_{i-1} + (y_i - y_{i-1}) * n\%, y_{i+1} - (y_{i+1} - y_i) * n\%]$ as illustrated in Figure 2, where y_{i-1} is the value of the last state and y_{i+1} is the value of next state. We choose n from 10 to 50 in our experiment because the previous i will be falsely recognized as state $i - 1$ or state $i + 1$ if n is larger than 50%. Thus, in this paper, we set the noise in the percentage from 10% to 50%. The mean prediction error results in Tables 2–6 show that our proposed method BMC has better prediction quality than DTMC. Both BMC and DTMC have the smallest prediction error in one step prediction, and the error magnifies as prediction steps

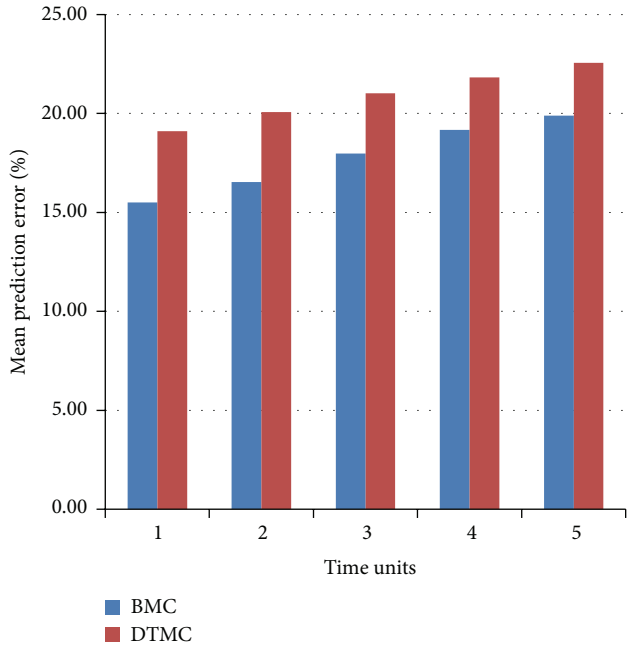

Figure 6: Prediction accuracy when bin size is 20.

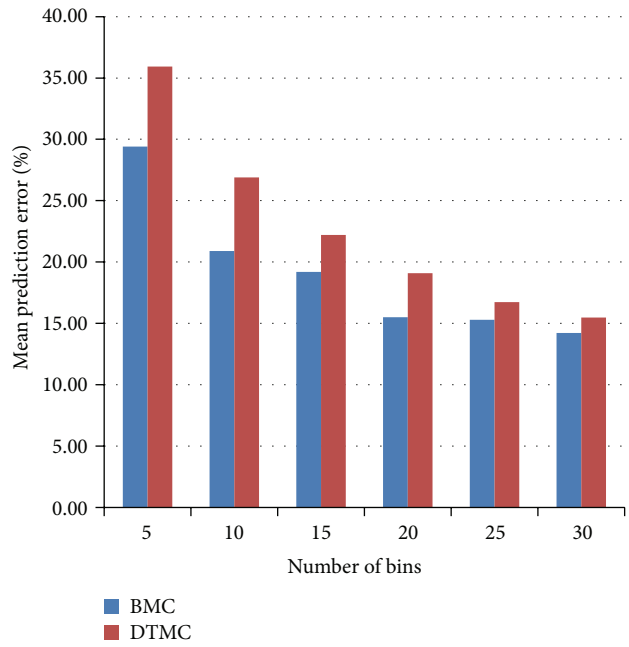

Figure 7: Prediction accuracy when time unit is 1.

become larger. BMC has the most notable advantage over DTMC in one step prediction and the advantage decreases as step goes larger. Based on the above observation, we conclude that our algorithm has better performance than DTMC in all noise ranges and fits better when we forecast imminent anomalies.

3.3. Ensemble Classification with Data Stream. In this experiment, we compare three ensemble classification methods and other classification algorithms, as decision tree and logistic regression. For ease of comparisons, we first summarize the assessment of criteria of different classification methods. Suppose that a data stream has n data chunks. We aim to build a classifier to predict all instances' label in the yet-to-come chunk. To simulate different types of data stream, we use the following approaches used in [21]: noise selection—we randomly select 20% chunks from each dataset as noise chunks and then arbitrarily assign each instance a class label which does not equal its original class label, and finally we put these noisy data chunks back into the data stream.

Table 7: The four cases that prediction belongs to.

Cases	Actually abnormal	Actually normal
Predicted as normal	True positive (TP) (true warning)	False negative (FN) (incorrectly warning)
Predicted as abnormal	False positive (FP) (missing warning)	True negative (TN) (correctly no warning)

The performance of system anomaly prediction is evaluated by 3 criteria according to [20]: precision, recall, and F-measure. We use Table 7 to help explain the definitions of these criteria, where state 0 denotes normal and state 1 denotes anomaly.

These three criteria are defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{11}$$
$$F\text{-measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

We define precision as the proportion of successful prediction for each predicted state in chunk D_{n+1}, recall as the probability of each real state to be successfully predicted in the chunk D_{n+1}, and F-measure as the harmonic mean of precision and recall.

Following the above process $n - 1$ times, we have the average precision, recall, and F-measure. Ideally, a good classifier for noise data stream should have high average precision, high average recall, and high average F-measure.

Table 8 shows the quality of classification between different classifiers. In this experiment, we choose three basic classifiers C4.5, Logistic, and Naïve Bayes as our base classifiers. And we set the sliding window size as 1000 instances. Column 2 to Column 4 in Table 8 are the classification results that employ single classifier. So, we choose the D_1 to train the model and test the model use D_2 then repeat the process by training the model using D_2 and test on D_3 and so on. HTree, HNB, and HLogist are three horizontal ensemble classification methods which use both history and current chunks to train the classifier model. So, we first use D_1 to train the model and test on D_2 and then use both D_1 and D_2 to train the model and test on D_3. Repeat this process until the end of the data stream. VerEn is the vertical ensemble

Table 8: Classification quality between different classifiers.

	Tree	NB	Logistic	HTree	HNB	HLogist	VerEn	AggEn
Precision	73.05%	72.36%	70.23%	70.04%	69.68%	69.69%	72.38%	73.25%
Recall	63.04%	69.48%	55.34%	72.18%	71.05%	51.93%	66.68%	75.08%
F-Measure	61.21%	68.85%	58.33%	68.73%	68.10%	51.76%	65.41%	72.70%

Table 9: Anomaly prediction cost.

Time	SAPredictor
Training time	452 ms
Prediction time	205 us

Table 10: Prediction results for different methods.

Methods	Precision	Recall	F-Measure
KNN + DTMC	72.7%	50.5%	57.3%
C4.5 + DTMC	74.9%	73.9%	74.4%
NB + DTMC	82.7%	26.6%	25.0%
TAN + DTMC	74.1%	80.0%	76.6%
SAPredictor	71.6%	84.6%	77.5%

model which uses all three base classifiers to train on the current chunk and test the next chunk. The last column is the aggregate ensemble which builds all base classifiers on history and current chunks.

The result in Table 8 shows that AggEn performs the best for all three measurements, the single Naïve Bayes is the second best, and VerEn is the third best. And HLogist and Logistic are listed as the last.

3.4. Anomaly Prediction System Cost. We have evaluated the overhead of our anomaly prediction model. Table 9 shows the average training time and prediction time. The training time includes the time of building BMC model and inducing the anomaly classifier. The prediction time includes the time to retrieve state transition probabilities and generate the classification result for a single data record. These results are collected over 100 experiment runs. We observe that the total training time is within several hundreds of milliseconds, and the prediction requires almost 200 microseconds. The above overhead measurements show that our approach is practical for performing online prediction of system anomalies.

3.5. SAPredictor Compared with Other Models. In this section, we compare the prediction quality between SAPredictor and DTMC combining other state-of-the-art classifiers in the machine learning literature, that is, k-Nearest Neighbor, C4.5, Naïve Bayes, and Tree-Augmented Naïve Bayesian (TAN) Network. We compare two kinds of prediction models: one is our SAPredictor which uses ensemble classification based on predicted metrics from BMC, the other is DTMC combining different single classifiers mentioned above. The performance of system anomaly prediction is evaluated by the same criteria used in Section 3.3: precision, recall, and F-measure.

Table 10 presents the experiment results of SAPredictor and other classifiers integrating DTMC on the dataset of PlanetLab. We notice that Naïve Bayes and KNN have the worst performance: its recall scores are 26.6% and 50.5%, respectively, and F-measure scores are 25.0% and 57.3%, respectively. SAPredictor receives the highest scores in recall and F-measure on this dataset, which are 84.6% and 77.5%. Thus, our SAPredictor is much more accurate than the other models.

4. Conclusions and Future Work

In this paper, we propose a novel system anomaly prediction model SAPredictor: it has clear advantages over discrete-time Markov chain which combines other classifiers. SAPredictor consists of two parts, one is belief Markov chain method which extends Evidential Markov chain by being capable of dealing with stream data, and the other is aggregate ensemble classification which identifies anomaly based on the value predicted by BMC. To conclude, SAPredictor can handle data stream from real application and systems with noise and measurement error.

Our experiments show that the BMC model achieves higher prediction accuracy than DTMC at any noise level and is especially fit for imminent anomaly prediction. SAPredictor achieves better system status prediction quality than the other popular models such as DTMC + Naïve Bayes, DTMC + C4.5, and DTMC + KNN. Our SAPredictor has small overhead, which makes it more practical for performing online prediction of system anomalies.

In the future, we plan to test and make possible improvement of SAPredictor in more real applications. In this paper, we consider the system as either normal or abnormal, while in reality the situation could have been more complicated. SAPredictor also can be improved to distinguish each kind of anomalies when making prediction and sending different level of alert. We also plan to publish a tool of SAPredictor and apply it to complex, distributed systems.

Acknowledgment

This research was supported by the Ministry of Industry and Information Technology of China (no. 2010ZX01042-002-003-001).

References

[1] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Signal Processing*, vol. 84, no. 7, pp. 1115–1130, 2004.

[2] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys*, vol. 42, no. 3, article 10, 2010.

[3] F. Salfner and M. Malek, "Using hidden semi-Markov models for effective online failure prediction," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS '07)*, pp. 161–174, October 2007.

[4] E. Kiciman and A. Fox, "Detecting application-level failures in component-based Internet services," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1027–1041, 2005.

[5] Y. Zhang and C. Ma, "Fault diagnosis of nonlinear processes using multiscale KPCA and multiscale KPLS," *Chemical Engineering Science*, vol. 66, no. 1, pp. 64–72, 2011.

[6] Y. Zhang, H. Zhou, S. J. Qin, and T. Chai, "Decentralized fault diagnosis of large-scale processes using multiblock kernel partial least squares," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 1, pp. 3–10, 2010.

[7] Y. Zhang, "Modeling and monitoring of dynamic processes," *IEEE Transactions on Neural Networks and Learning System*, vol. 23, no. 2, pp. 277–284, 2012.

[8] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM BlueGene/L event logs," in *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM '07)*, pp. 583–588, October 2007.

[9] P. Zhang, H. Muccini, A. Polini, and X. Li, "Run-time systems failure prediction via proactive monitoring," in *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE '11)*, pp. 484–487, November 2011.

[10] Y. Tan and X. Gu, "On predictability of system anomalies in real world," in *Proceedings of the 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '10)*, pp. 133–140, August 2010.

[11] G. Luo, K.-L. Wu, and P. S. Yu, "Answering linear optimization queries with an approximate stream index," *Knowledge and Information Systems*, vol. 20, no. 1, pp. 95–121, 2009.

[12] X. Gu and H. Wang, "Online anomaly prediction for robust cluster systems," in *Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE '09)*, pp. 1000–1011, April 2009.

[13] H. Soubaras, "On evidential markov chains," in *Foundations of Reasoning Under Uncertainty*, vol. 249 of *Studies in Fuzziness and Soft Computing*, pp. 247–264, Springer, Berlin, Germany, 2010.

[14] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the International Conference on Very Large Data Bases (VLDB '03)*, pp. 81–92, 2003.

[15] P. Zhang, X. Zhu, Y. Shi, L. Guo, and X. Wu, "Robust ensemble learning for mining noisy data streams," *Decision Support Systems*, vol. 50, no. 2, pp. 469–479, 2011.

[16] Y. Zhao, Y. Tan, Z. Gong, X. Gu, and M. Wamboldt, "Self-correlating predictive information tracking for large-scale production systems," in *Proceedings of the 6th International Conference on Autonomic Computing (ICAC '09)*, pp. 33–42, Barcelona, Spain, June 2009.

[17] C.-H. Lee, Y.-L. Lo, and Y.-H. Fu, "A novel prediction model based on hierarchical characteristic of web site," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3422–3430, 2011.

[18] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by Markov chains," *IEEE Wireless Communications*, vol. 16, no. 2, pp. 56–63, 2009.

[19] J. Y. Halpern, *Reasoning about Uncertainty*, MIT Press, Cambridge, Mass, USA, 2003.

[20] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2006.

[21] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 35, no. 5, pp. 905–914, 2005.

[22] Y. Zhao, Y. Tan, Z. Gong, X. Gu, and M. Wamboldt, "Self-correlating predictive information tracking for large-scale production systems," in *Proceedings of the 6th International Conference on Autonomic Computing (ICAC '09)*, pp. 33–42, ACM, June 2009.