Arena: adaptive real-time update anomaly prediction in cloud systems

Shaohan Huang*‡, Carol Fung†, Chang Liu§, Shupeng Zhang*, Guang Wei*, Zhongzhi Luan*, Depei Qian*
*Sino-German Joint Software Institute, Beihang University, Beijing, China
†Computer Science Department, Virginia Commonwealth University, Richmond, Virginia, USA
‡Microsoft Research, Beijing, China
§Carnegie Mellon University, Pittsburgh, PA, USA
buaahsh@foxmail.com, cfung@vcu.edu, cliu5@andrew.cmu.edu
zspbuaa@buaa.edu.cn, weiguang0314@163.com, zhongzhi.luan@jsi.buaa.edu.cn, depeiq@buaa.edu.cn

***Abstract*—In current cloud systems, their monitoring relies strongly on rule-based and supervised-learning-based detection methods for anomaly detection. These methods require either some knowledge provided by an expert system or monitoring data to be labeled as a training set. In practice, the systems behavior changes over time. It is difficult to adjust the rules or re-train detection model for these methods. In this paper, we present an Adaptive REal-time update uNsupervised Anomaly prediction system (Arena) for cloud systems. Arena uses a clustering technique based on a density spatial clustering algorithm to identify clusters and outliers. We propose two prediction strategies to improve the ability to predict anomaly and a real-time update strategy by adding new monitoring points into Arenas model. To improve the prediction efficiency and reduce the scale of the model, we adopt a pruning method to remove redundant points. The anomaly data used in the experiments was collected from the Yahoo Lab and the component based system of enterprise T. The experimental results show that our proposed methods can achieve high prediction accuracy compared to existing methods. Real-time update strategy can improve the prediction performance. The pruning method can further reduce the scale of the model and demonstrates the prediction efficiency.**

I. Introduction

In recent years, cloud computing has become the most desired technology for the IT industry. Large enterprise data centres have been widely used in web services and applications such as E-commerce, distributed multimedia, and social networks. However, the enterprise data centres are prone to performance anomalies due to their complexity in structure and shared resources. For example, Cloud provider Amazon Web Services (AWS) reported an outage between 2:13 AM and 7:10 AM PDT on September 20, 2015, when users were unable to reach popular online services like Netflix, Tinder, Airbnb, Reddit, and IMDb[1]. The origin of the failures turned out to be a large number of errors on read and write operations for the Amazon DynamoDB service in the US-East Region.

The purpose of anomaly detection methods is to detect the outliers in enterprise data centres. The major drawback of these approaches is that they can only detect but not prevent anomalies. Comparatively, proactive methods can predict anomalies before they occur, which allows actions to be taken to prevent anomalies from happening or reduce the damage from incidents.

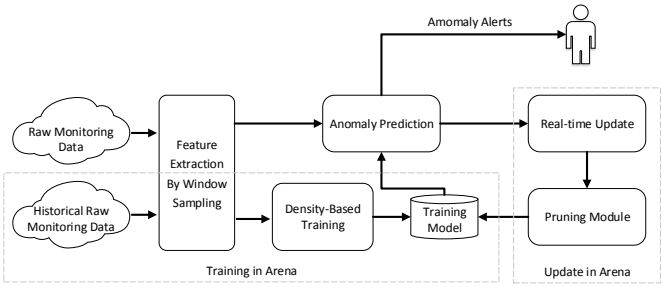

Fig. 1. High-level description of Arena system

Achieving efficient anomaly prediction through the monitoring system is a challenging problem. One reason is that the amount of monitoring data is produced by a number of nodes in the distributed enterprise system and it is impractical to obtain labelled training data from production cloud systems. Second, a large-scale enterprise data center often runs hundreds of concurrent applications. Since the system being monitored is a black box system, the anomaly detection is only based on monitored data[2]. Last, because system's behavior may change over time, the prediction model needs real-time update to prepare for the impact of new anomalies.

The Arena system uses three steps to analyze monitoring data captured in contiguous fixed-length time slots. Figure 1 depicts an overview of the Arena system. The first step is to extract different features in contiguous time slots which can capture the status of system. The second is to use all the features obtained from the last step to train the model of Arena or to predict an anomaly based on the model. In this step, outliers are identified using a robust clustering algorithm, based on a density spatial clustering algorithm. The clustering algorithm is a completely unsupervised method, which can be directly used in monitoring a system without labelled dataset. In the third and final step, after anomaly prediction the new monitoring data will be used to refine and update the model of Arena system. In this step, we propose a pruning method to reduce the redundancy. The main contribution of Arena system is its ability to work in an entirely unsupervised fashion and in a real-time manner.

We conducted anomaly prediction experiments on two system monitoring datasets: Yahoo Lab[3] and the monitoring

data from a component-based system of the enterprise T[1]. Experimental results demonstrate that for the Yahoo Lab dataset, we can achieve a prediction precision of 88.30% and F value of 0.855. For the component-based system, the Arena system achieves the highest prediction F value of 0.837 compared to other methods. Experimental results show that the pruning method can reduce the scale of the model and improve the prediction speed.

The major contributions of this paper can be summarized as follows:

- We propose Arena, an unsupervised system for anomaly prediction for cloud systems. It works in an entirely unsupervised fashion, which means that it can be directly used in any monitoring system without any labelled dataset.
- We present a real-time and pruning strategy in the Arena system. The Arena system can refine its model when it processes new monitoring data. The pruning method can reduce redundancy and improve prediction speed.
- We validate and compare the effectiveness of the Arena system through experiments on real monitoring data based on cloud systems.

The remainder of this paper is organised as following: Section 2 presents the detailed design of Arena system. In Section 3 we conduct anomaly prediction experiments on different system monitoring datasets and demonstrate the experimental evaluation results. Section 4 presents state of the art of unsupervised anomaly prediction in the literature. Finally, in Section 5, we conclude this paper and our future work.

II. System Design

In this section, we present the design details of the Arena system. We first discuss monitoring data standardisation and feature extraction. We then describe our unsupervised anomaly prediction model training algorithm that can train anomaly prediction models without a labelled dataset. Next, we present our anomaly prediction strategies. Finally, we present the real-time update strategy and pruning strategy in the Arena system.

A. Standardization and Feature Extraction

As shown in Figure 1, we first preprocess the raw monitoring data and extract a number of features from the standardized data.

There are several common preprocess functions to convert raw data into a representation which can be used as the input for the prediction models, e.g. standardisation, normalisation and binarization. For the Arena system, we adopt the standardisation as the process transformer. Standardisation is the process of re-scaling the data distribution through centering its mean to zero and scale its variance to unit value. Standardisation can unify different monitoring datasets with various means and variants and reduce the range of parameters selection, which is critical to the selection of parameters. Furthermore, the standardisation will not affect the detection of abnormal points since the outliers will still have significant distance to other points. The standardisation equation is shown in Equation 1.

$$\begin{aligned} s^2 &= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x_i})^2 \\ \dot{x_i} &= \frac{x_i - \bar{x_i}}{s} \end{aligned} \tag{1}$$

Where x_i denotes the value of the i-th attribute, and $\bar{x_i}$ represents the average value of the i-th attribute value in the initial training data. s denotes the standard deviation of the i-th attribute value. $\dot{x_i}$ represents standardisation results of the i-th attribute.

After standardisation, we use the window sampling method to extract features. The method can reduce the noise of monitoring data which helps to extract more efficient features. We compute *average value*, *amplitude*, and *amplitude rate* as our features, which are often used in anomaly prediction models[4]. We set the width of the sampling window to be w, and compute the average value using the following equation:

$$AveValue = \frac{\sum_{i=1}^{w} value_{t_i}}{w} \tag{2}$$

Where $value_{t_i}$ denotes the monitoring value at time t_i. The amplitude feature can be computed as shown in Equation 3.

$$Amplitude = max(value_{t_i}) - min(value_{t_i}) \tag{3}$$

The amplitude rate computation equation is shown in Equation 4.

$$AmplitudeRate = \frac{Amplitude}{AveValue} \tag{4}$$

The average monitoring value presents the particular value in the current period. The averaging method is used to smooth the data to reduce the impact of noise. Amplitude reflects the fluctuation in the window. When two windows get the same average monitoring value, it may be the case that a window has significant volatility, while the other window does not. The amplitude ratio is the ratio of the amplitude to the average monitoring value, presenting the magnitude of volatility compared to the window sampling. The reason that we use amplitude ratio, is that, the amplitude alone is not sufficient to determine the level of abnormality. For example, two windows may have the same amplitude. If one window has a significant larger average value than the other, then the level of its abnormality is lower.

B. Anomaly Prediction Model Training

We adopt a density-based spatial clustering algorithm, which is similar to the density-based spatial clustering of applications with noise (DBSCAN) algorithm [5], to train the anomaly prediction model in the Arena system. The DBSCAN algorithm groups together points that are closely packed together. Data points that lie alone in low-density

[1]We obscured the company name due to the double-blind review policy.

regions are marked as outliers. The prediction model training is also based on the density-based spatial clustering algorithm.

The Arena system uses the historical monitoring data as its training dataset. All historical monitoring points are classified into three categories: core points, edge points, and outliers. The prediction model training requires only two parameters: *minPts* and *eps*. A point p is a core point if at least minPts points are within the distance of eps. A point p is an edge point if at least one point is within the distance of eps. The rest points, which are not reachable from any other point, are outliers. In the training process, the core points and edge points are retained as the model of Arena system.

Algorithm 1 Anomaly Prediction Model Training

Require: D, eps, minPts
Ensure: D with Label
1: **function** TRAIN($D, eps, minPts$)
2: $List < Points > templates \leftarrow \emptyset$
3: **for** $i = 0 \rightarrow D.length$ **do**
4: $P \leftarrow D[i]$
5: **if** P is not visited **then**
6: P is visited
7: $Neighbors \leftarrow$ REGION(P, eps)
8: **if** $size(Neighbours) < minPts$ **then**
9: Label P as outliers
10: ADD_TEMPLATE($templates, context$)
11: **else**
12: Label P as core
13: EXPAND($Neighbours, eps, minPts$)
14: **end if**
15: **end if**
16: **end for**
17: **end function**
18: **function** EXPAND($Neighbours, eps, minPts$)
19: **for** $i = 0 \rightarrow Neighbours.length$ **do**
20: $P \leftarrow Neighbours[i]$
21: **if** P is not visited **then**
22: P is visited
23: $Neighbors' \leftarrow$ REGION(P, eps)
24: **if** $size(Neighbours') > minPts$ **then**
25: Label P as core
26: **else**
27: Label P as edge
28: **end if**
29: $Neighbors \leftarrow Neighbors + Neighbors'$
30: **end if**
31: **end for**
32: **end function**
33: **function** REGION(P, eps)
34: **return** all points within P's eps-neighborhood
35: **end function**
36: **function** ADD_TEMPLATE($templates, context$)
37: Add context into templates
38: **end function**

The training algorithm is explained in Algorithm 1. The function Train is the main function of training algorithm, where D represents the initial training data. eps and minPts are global parameters of the Arena model. The function Region returns all points within the Eucidean distance of eps. If the size of neighbours is less than minPts, then label point P as outliers, otherwise label point P as a core point. The training algorithm calls the function Expand when the point P is a core point. The function Expand can find more core and edge points through traversing all neighbors of point P. The function Add_template is to extract anomaly templates in the prediction strategy which is introduced in next subsection. The worst case time complexity of the training algorithm is $o(n^2)$, where the space complexity is $o(n)$.

C. Anomaly Prediction Strategy

In the training process of the anomaly prediction model, the abnormal points in the monitoring data can be identified. Also, to detect the outliers of the monitoring data, we expect the Arena model to have the ability to predict anomalies before they occur. To solve this problem, we developed two prediction strategies to enhance the accuracy of the Arena model.

First, according to the Arena training algorithm, the monitoring points are divided into three categories: the core point, edge points and outlier points. The core points and edge points are retained. We detect the status of new monitoring points using the existing core points and edge points information. Calculate the distance of the new monitoring point from the existing points. If there is a core or edge point within eps, the point is normal. If it does not exist, mark the point as an outlier. If the number of existing points within eps is more than minPts, then the new point is a core point, otherwise it is an edge point.

The first prediction strategy used in the Arena model is to judge whether there are n edge points in succession. Although the edge point is normal, the system can be considered in the borders of the normal state. So we use the continuous edge points as the standard to predict the anomalies. The Arena model can determine the status of a new monitoring point. If there are n consecutive edge points, then raise an alert. A user can adjust the prediction strategy by tuning the value of n.

The second prediction strategy of the Arena model is to use the abnormal points during the training process to extract anomaly templates. As shown in Algorithm 1, we will call the function Add_template when the point P is an outlier. The context is the fixed-length monitoring metrics before the current time. We use the context before outlier as our anomaly template. When the Arena model predicts the new monitoring data, it calculates the similarity between the context of a new monitoring point with historical templates to predict anomaly.

If we assume the length of context is 3 and current point is p_t at time t, the current context is p_{t-1}, p_{t-2} and p_{t-3}. The template (tm_{t-1}, tm_{t-2} and tm_{t-3}) is in the model's historical templates. If it meets the following conditions, we will raise an anomaly warning. dis represents the distance of p_{t-i} and tm_{t-i}.

$$dis(p_{t-i}, tm_{t-i}) < eps(i = 1, 2, 3) \quad (5)$$

As presented in Algorithm 2, we first call the function a Predict_based_stragety to predict anomaly. If it returns a normal label, then we analyze its neighbors. The Arena model can not only detect the outliers, it can use these two prediction methods to complete the abnormal prediction.

D. Real-time Update and Pruning Strategy

In this section, we introduce the core of the proposal, the real-time update and pruning strategy in the Arena system.

Real-time updates are critical to the Arena system because monitoring data changes over time. The real-time update can improve the prediction accuracy of the Arena model. The key of the Arena system is that the core points and edge points are in the model, and the new monitoring points can be classified by using the core points and edge points. We can update the core points and edge points with time to make the Arena model real-time.

The pseudo-code of update algorithm is presented in Algorithm 2. If the new point is normal, we will call the function Update to add the new point into our model. We add the new point into edge points when the size of neighour is less than minPts, otherwise they are added into core points.

Real-time update strategy brings an issue that the number of core points and edge points will increase with time and the scale of the model will become larger. If all normal points are retained as the core points or edge points, it will inevitably lead to redundancy problems. The Arena system adopts a reasonable pruning strategy to reduce the additional points and improve the efficiency of the model.

The ideal criterion for verifying a point as a redundant point is that it will not affect all predictions after the point is removed. The Arena model prunes redundant points because the density of the normal points can be much higher than needed. Fewer points can improve the computation efficiency of the model. For example, if there are a sufficient number of core points and edge points around a new point, and adding the point will not affect the result, then the point is a redundant point.

According to the standard for determine the redundancy point, it is necessary to traverse all the cases within the eps distance, which induces enormous cost. Therefore, we adopted a heuristic algorithm to determine whether a new point shall join the core points or edge points by calculating the number of surrounding points within the range of eps. Let R be the redundancy, and the redundancy can be calculated as follows.

$$R = \frac{size(Neighbours)}{minPts} \quad (6)$$

Where minPts is the parameter of the Arena model and neighbours is the number of core points and edge points within eps. If the redundancy R reaches a certain threshold, it is judged as a redundancy point.

Algorithm 2 Model Prediction and Real-time Update

Require: Model, New Point
Ensure: Label

function PREDICTANDUPDATE($Model, NewPoint$)
  $templates \leftarrow Model.templates$
  $r \leftarrow$ PREDICT_BASED_STRATEGY($P, eps, templates$)
  if r = Anomaly **then**
    return Anomaly
  end if
  $Neighbors \leftarrow$ REGION($NewPoint, Model.eps$)
  if $size(Neighbours) = 0$ **then**
    return Anomaly
  else
    UPDATE($Model, neighbor\_size, minPts, P$)
    return Normal
  end if
end function
function UPDATE($Model, neighbor\_size, minPts, P$)
  if $neighbor_size < minPts$ **then**
    Add P into Model.Edges
  else
    Add P into Model.Cores
  end if
end function
function PREDICT_BASED_STRATEGY($P, eps, templates$)
  if P is n-th consecutive edge point **then**
    return Anomaly
  end if
  if context of P in templates **then**
    return Anomaly
  end if
  return Normal
end function
function REGION(P, eps)
  return all points within P's eps-neighborhood
end function

III. EXPERIMENTAL EVALUATION

We evaluate the ability of Arena to predict anomaly on different datasets. We describe experimental setup and report empirical results in this section.

A. Experimental Setup

We evaluate our prediction method based on two real system data set: Yahoo Webscope anomaly detection dataset [6] and an anomaly data set from a component based system of enterprise T.

The Yahoo Webscope anomaly detection benchmark consists of labelled time-series representing metrics of various Yahoo services. Each time-series is replayed sequentially to emulate real life scenario. We tested our system with time-series from the A1Benchmark of the Yahoo Webscope anomaly detection dataset. A1Benchmark is based on the real production traffic to some of the Yahoo properties. The anomalies in A1Benchmark are marked by humans, and other

benchmarks in the Yahoo Webscope anomaly detection dataset are based on synthetic time-series. Therefore, we use the A1Benchmark to measure the accuracy of anomaly prediction.

The component-based system dataset is from the monitoring system of enterprise T. Enterprise T is one of the largest Internet service providers which provides mass media, entertainment, Internet mobile phone value-added services, and many other services. In the enterprise T's computer system, there are thousands of components including business components and functional components. These components are deployed on dozens of nodes distributed all over the world. The performance status of all server nodes are logged and stored in the monitoring system for days. The faults in the dataset came from the real online environment are labelled by the anomaly detector and are verified manually.

The statistics of our anomaly prediction dataset are given in Table I. The Yahoo Webscope anomaly detection benchmark contains 19,222 monitoring points, including 240 anomaly points. Enterprise T anomaly prediction dataset contains 24,236 monitoring points, including 301 anomaly points. There are multiple monitoring dimensions in the Yahoo benchmark and the enterprise T dataset.

In our experiments, we set the width of windows sampling and the number of continuous edges as 3, and redundancy of pruning as 10.

TABLE I
STATISTICS OF THE ANOMALY PREDICTION DATASET

Dataset	Total	Anomaly	Dimension
Yahoo Lab	19,222	240	14
Enterprise T	24,236	301	12

B. Evaluation Method and Baseline

We use the precision, recall, F value, false positive rate (fpr) and lead time to evaluate the anomaly prediction accuracy and performance. We define the abnormal state to be the positive class and the normal state to be the negative class. We compute the standard metrics using the formula in Equation set 7.

$$\begin{aligned} precision &= \frac{TP}{TP+FP} \\ recal &= \frac{TP}{TP+FN} \\ F &= \frac{2TP}{2TP+FP+FN} \\ fpr &= \frac{FP}{FP+TN} \end{aligned} \tag{7}$$

Where TP is the number of true positives which represents the number of times a model raises the alarm due to there is an anomaly. TN is the number of true negatives; FP is the number of false positives; FN is the number of false negatives. If the accuracy, recall and F value are higher then the performance of the model is better. The lower false positive rate represents a better result. We define the lead time to be the time duration between an alert being raised and the actual failure time. It means that if the lead time is sufficient, the anomaly management system can take an action to prevent anomalies.

To demonstrate the effectiveness of our proposed method, we compare our method with some baseline methods in this section. The baseline methods we chose include a set of commonly used anomaly prediction methods. They can be described as follows:

1) One-class support vector machines (OCSVM): It is used for novelty detection. Given a set of samples, it will detect the soft boundary of that set so as to classify new points as belonging to that set or not [7].

2) Behaviour-based Anomaly Detection (BAD): This technique exploits the knowledge of the characteristic probability density function of normal performance measurements. BAD estimates a model of the unknown PDF of a given baseline using Kernel Density Estimation[8].

3) Isolation Forest (IForest): One efficient way of performing outlier detection in high-dimensional datasets is to use random forests[9]. It isolates observations by randomly selecting a feature and then randomly choosing a split value between the maximum and minimum values of the selected feature.

For these baselines and our method, we use 10% monitoring data for training and parameter optimization; other monitoring data as incoming data are used to evaluate our system. All methods use the same feature extraction introduced in part A-standardization and feature extraction. We conduct our experiments on each dimension, then compute their average as our experiment results.

C. Experimental Results

This section demonstrates the experimental results from three aspects: prediction accuracy, prediction strategy and pruning results.

1) Prediction Accuracy: We use the precision, recall, F value and false positive rate (fpr) to evaluate the anomaly prediction accuracy.

TABLE II
ANOMALY PREDICTION ACCURACY COMPARISON FOR YAHOO

-	Precision	Recall	F value	fpr
BAD	82.31%	74.32%	0.781	4.3%
OCSVM	84.19%	79.27%	0.817	3.7%
IForest	81.43%	**85.41**%	0.834	3.3%
Arena	**88.30**%	82.79%	**0.855**	**2.7**%

As shown in Table II, the Arena system achieves higher prediction precision, F value and fpr than other methods for the Yahoo dataset. The BAD model has the worst performance of all models. The main reason for this is that, it predicts the anomalies only based on the kernel density estimation, which can't capture changes in monitoring data. At the same time, if the anomaly points have significant density, BAD will classify them as normal points.

TABLE III
ANOMALY PREDICTION ACCURACY COMPARISON FOR ENTERPRISE T

-	Precision	Recall	F value	fpr
BAD	80.21%	73.35%	0.7663	4.5%
OCSVM	82.63%	76.22%	0.793	3.9%
IForest	**86.17**%	79.21%	0.825	3.6%
Arena	85.36%	**82.18**%	**0.837**	**3.1%**

TABLE IV
LEAD TIME RESULT FOR YAHOO AND ENTERPRISE T DATASET

-	BAD	OCSVM	IForest	Arena
Yahoo	3.42	3.53	4.31	4.52
Enterprise T	29.23	34.12	33.12	36.32

We can also see that OCSVM achieves 84% accuracy which is second to the Arena model. The reason for the high accuracy is that One-class SVM needs to use normal data to train, which helps the model to obtain more normal monitoring data. Therefore, One-class SVM has a higher precision rate but a lower recall rate. SVM does not update and adjust the model itself. If the monitoring data have changed, the accuracy of the model will decline.

The recall rate of the isolation forest model is the best among the four models. Also its F value and the false positive rate are only second to the Arena model. This is because the isolated forest in the training time uses random sampling to classify the monitoring data into a number of leaf nodes, then detects the anomaly points based on their density. Therefore, in the process of random sampling, some normal monitoring points may be randomly classified into incorrect leaf nodes, which leads to the high recall rate, but not the accruracy rate.

The Arena model automatically classifies the anomaly points based on the density information. Furthermore, the Arena model can improve the accuracy of the model by using the real-time update.

Table III shows the prediction accuracy result for the Enterprise T systems using different prediction models. The Arena models also achieves a higher F value and fpr than other methods. The BAD is still the worst model of all methods. The experimental results of Enterprise T monitoring set show that the Arena model consistently demonstrates high performance compared to other models on different data sets.

Figure 3 shows the prediction precision and recall rates of the Arena system in different dimensions for the Yahoo dataset. Figure 4 demonstrates the results for the Enterprise T dataset. As shown in Figure 3 and Figure 4, the Arena system has poor performance only in few dimensions. This model is robust and it is suitable for various types of dimensions for the monitoring data.

We now present the lead time results of our experiments shown in Table IV. Yahoo Enterprise sampling interval is 1 second and T's sampling interval is 10 seconds. In both Yahoo and Enterprise T datasets, we observe that the Arena model

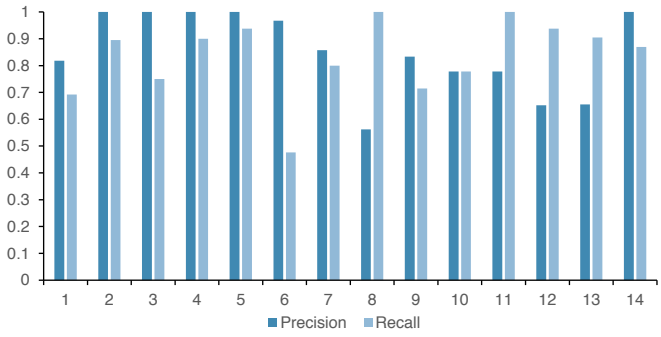

Fig. 2. Precision and recalls in different dimensions for Yahoo dataset

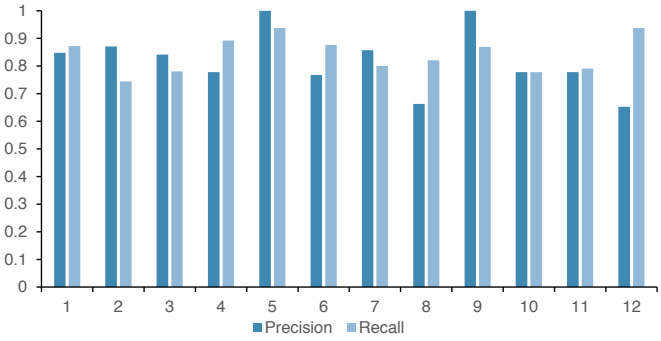

Fig. 3. Precision and recalls in different dimensions for Enterprise T dataset

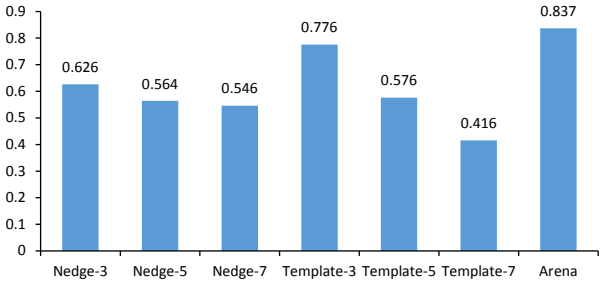

Fig. 4. Yahoo evaluation results of model component ablations

can achieve better performance than other models.

2) Prediction Strategy: We propose two strategies to predict anomalies in the Arena system. One strategy is to judge whether there are n edge points. The other strategy is to record the templates. We conducted some ablation experiments and present model analysis to help us understand prediction strategies of Arena. These strategies are described as follows:

- **Nedge**. This method only uses the first prediction strategy to raise anomaly if there are n edge points in succession. We set n as 3, 5, 7.
- **Template**. The model only uses the second strategy. It learns the templates during training and raises anomaly based on templates. We set different width of context as 3, 5, 7.

As shown in Figure 4 and Figure 5, we compute the F value for our full model and different variants on the Yahoo and the Enterprise T dataset. The N-edge strategy works best when n is 3. The results demonstrate that anomaly prediction

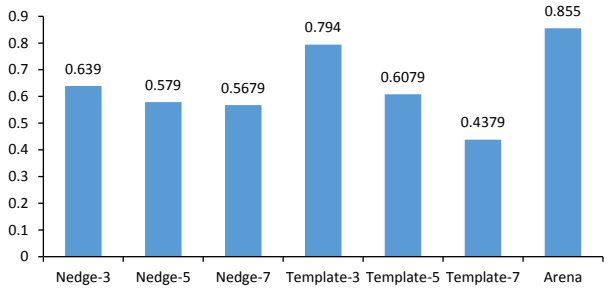

Fig. 5. Enterprise T evaluation results of model component ablations

TABLE V
F VALUE FOR REAL-TIME UPDATE

	Yahoo	T
No Update	0.816	0.812
Real-time Update	0.855	0.837

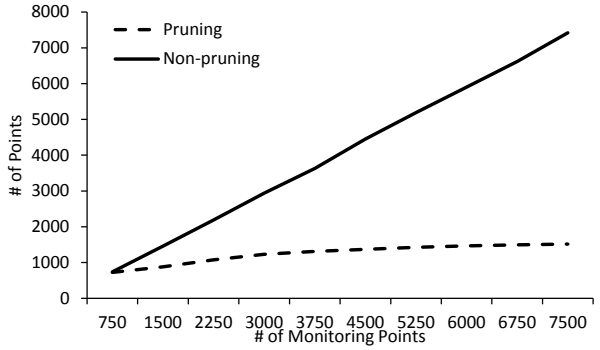

Fig. 6. Trend of the number of core points and edge points change

dose not require too many edge points. For template strategy, when the width of context is 3, the model achieves the best performance in both data sets. It brings low recall if the width of context is too large. The ablation model Nedge-3 performs worse than Template-3. This indicates that using templates is more efficient than simply rules. There are some templates in the anomaly prediction.

3) Update and Pruning results: We propose the real-time update to improve the prediction accuracy. As shown in Table V, we compute the F value to show the influence of real-time. The experiment results demonstrate that the real-time feature is essential for the Arena system. It gains improved anomaly prediction capability by updating its core points and edge points in real time.

However, the real-time update strategy also encounters a scalability issue since the number of core points and edge points increase with time and the size of the model will increase. Therefore, it becomes necessary to prune extra points by looking for redundancy. To guarantee precision, we set the redundancy to be 10, which is the a conservative configuration.

As shown in Figure 6, we change the numer of monitoring points from 750 to 7500 and observe the total used points with and without pruning. We can see from the figure that for the non-pruning model, the number of core points and edge points is linear proportional to the total data size. Using the pruning model can significantly decrease the number of data points in the model.

TABLE VI
THE EFFECT OF PRUNING ON PREDICTION TIME AND F VALUE

	Yahoo	T
No Pruning	5.733s/0.855	5.421s/0.837
Pruning	5.352s/0.849	5.135s/0.829

TABLE VII
F VALUE FOR RUBIS UNDER DIFFERENT CONTEXT WINDOW

Context Window Length	3	5	10	15
Yahoo	0.855	0.835	0.801	0.790
T	0.837	0.815	0.804	0.762

Pruning strategy reduces the number of core points and edge points, thus reducing the memory requirement of the model. At the same time, the reduction of core and edge points can also reduce the computational complexity and improve the prediction speed of the model in anomaly prediction.

We evaluated the overhead of our anomaly prediction model in both datasets. These results are the average prediction time based on over 15,000 experimental runs. In this paper, these experiments were conducted on a 3.4GHz Intel Core i7 PC machine with 8-gigabyte main memory. As shown in Table VI, we can find that the prediction time and F value is reduced when using pruning.

One observation is that the prediction time reduces 5% to 10%, which is much lower than the reduction of core and edge points. There are two main reasons: first, in addition to calculating the distance of the core points and the prediction points, there are the process of model initialization and data preprocessing, etc. These processes will not be reduced because of the decrease in the number of core points; second, the pruning process brings extra computation workload. Due to these two reasons, the reducation of prediction time is lower than the reduction of points.

We compare the effect of the pruning model and the non-pruning model on the prediction performance of Yahoo and Enterprise T data. The results are shown in Table VI. As mentioned above, to guarantee precision, we set the redundancy parameter as 10. The experimental results show that the pruning model has little effect on the prediction of the anomaly.

4) The Impact of Window Length: We have conducted experiments to study how the Arena models perform under different window length with various settings. We show results of the Arena model for the Yahoo dataset and the Enterprise T dataset under a different duration of the window.

Table VII shows the F values for both datasets under different lengths of the window. The Arena model achieves the highest score when the length of the window is 3 for both data sets. The long windows may bring more noise, which affects the anomaly prediction.

Overall, the Arena model demonstrate the best prediction accuracy on both the Yahoo dataset and the Enterprise T dataset. The pruning strategy can reduce the number of cores points and edge points efficiently.

IV. Related Work

In this section, we review the related works about anomaly prediction approaches, mainly focusing on unsupervised anomaly prediction methods.

A. Anomaly Prediction Methods

The problem of anomaly detection has been extensively studied during the last decade. Anomaly detection is a passive method that reports problematic states after they happen. Anomaly prediction is a proactive method which raises alerts when the system is still in the normal state but progressing to an anomaly state [10].

L. Cherkasova et al. proposed an integrated framework for detecting the changes and differentiation performance of application behaviours [11]. They employed a regression model to achieve statistically significant transaction types, then detected performance changes by comparing the new application signatures against the old ones. Y. Tan et al. used the Markov model for predicting the future state of the system [12]. They adopted the Tree-Augmented Naive Bayesian (TAN) network to be their anomaly classifier, where the dependencies among different metrics are considered. There is an issue when using the Markov model as the regression model, where feature values are converted to discrete states to run the model. To address this issue, many solutions have been proposed to improve the Markov model. For example, X. Zhou et al. suggested a Belief Markov Chain based on the Dempster-Shafer theory [13]. In their work, a stream-based k-means clustering method was used to improve the Evidential Markov chain, in which each cluster represents a Markov state and the arriving data points can be mapped into k states.

B. Unsupervised Anomaly Prediction Methods

In recent years, there have been many unsupervised techniques employed in anomaly detection or prediction. With a large volume of monitoring data, it is difficult to obtain labelled training data. Unsupervised anomaly prediction methods can predict anomalies without labelled training data.

D. J. Dean et al. proposed an Unsupervised Behavior Learning (UBL) system for IaaS cloud computing infrastructures [14]. UBL leverages Self-Organizing Maps to capture emergent system behaviours and predict unknown anomalies. The SOM uses three kinds of neurones(normal, pre-failure and failure) to predict anomalies. The UBL system cannot update its model and needs to set a neighborhood area size threshold by system users to differentiate normal and anomalous neurons which is integral to the accuracy of the UBL system.

Pedro introduced an unsupervised network anomaly detection algorithm for knowledge-independent detection of anomalous traffic [4]. The system used a clustering technique based on sub-space-density clustering to identify clusters and outliers in multiple low-dimensional spaces.

In [15], the author presented a predictive performance anomaly prevention system that provides automatic performance anomaly prevention for virtualized cloud computing infrastructures which integrate online anomaly prediction, learning-based cause inference, and predictive prevention actuation to minimise the performance anomaly penalty without human intervention.

V. Conclusion and Future Remarks

In this paper, we propose an adaptive real-time unsupervised anomaly prediction system (Arena), which can predict anomalies in monitoring data. We implement two prediction strategies to enhance the ability to predict anomaly in the Arena system. Our experimental results show that the prediction strategies can improve prediction accuracy. The pruning strategy is adopted to reduce redundancy. Experimental results prove that this pruning method can decrease the number of core points and edge points, and also improve the prediction speed.

In our future work, we plan further experimental evaluation of our prediction model in more cloud network environment or data centres. Additional work is to integrate our anomaly prediction system with an analysis engine that can diagnosis the cause of an alert.

Acknowledgment

This research is supported by the National Key R&D Program (Grant No. 2016YFB0201403) and the Natural Science Foundation of China (Grant No. 61361126011).

References

[1] "http://www.datacenterdynamics.com/colo-cloud-/aws-suffers-a-five-hour-outage-in-the-us/94841.fullarticle," 2015.

[2] S. Duan, S. Babu, and K. Munagala, "Fa: A system for automating failure diagnosis," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pp. 1012–1023, IEEE, 2009.

[3] W. Yan, "Toward automatic time-series forecasting using neural networks," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 7, pp. 1028–1039, 2012.

[4] P. Casas, J. Mazel, and P. Owezarski, "Unada: Unsupervised network anomaly detection using sub-space outliers ranking," in *International Conference on Research in Networking*, pp. 40–51, Springer, 2011.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[6] "Yahoo! webscope, s5 - a labeled anomaly detection dataset, version 1.0. [online]. available: https://webscope.sandbox.yahoo.com/catalog.php?datatype=s,"

[7] R. Perdisci, G. Gu, and W. Lee, "Using an ensemble of one-class svm classifiers to harden payload-based anomaly detection systems," in *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pp. 488–498, IEEE, 2006.

[8] O. Ibidunmoye, T. Metsch, and E. Elmroth, "Real-time detection of performance anomalies for cloud services," in *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*, pp. 1–2, IEEE, 2016.

[9] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pp. 413–422, IEEE, 2008.

[10] C. Wang, S. P. Kavulya, J. Tan, L. Hu, M. Kutare, M. Kasick, K. Schwan, P. Narasimhan, and R. Gandhi, "Performance troubleshooting in data centers: an annotated bibliography?," *ACM SIGOPS Operating Systems Review*, vol. 47, no. 3, pp. 50–62, 2013.

[11] L. Cherkasova, K. Ozonat, N. Mi, J. Symons, and E. Smirni, "Anomaly? application change? or workload change? towards automated detection of application performance anomaly and change," in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pp. 452–461, IEEE, 2008.

[12] Y. Tan and X. Gu, "On predictability of system anomalies in real world," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, pp. 133–140, IEEE, 2010.

[13] X. Zhou, S. Li, and Z. Ye, "A novel system anomaly prediction system based on belief markov model and ensemble classification," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[14] D. J. Dean, H. Nguyen, and X. Gu, "Ubl: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems," in *Proceedings of the 9th international conference on Autonomic computing*, pp. 191–200, ACM, 2012.

[15] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, "Prepare: Predictive performance anomaly prevention for virtualized cloud systems," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pp. 285–294, IEEE, 2012.