Interactive Context-Aware Anomaly Detection Guided by User Feedback

Yang Shi, Maoran Xu, Rongwen Zhao, Hao Fu, Tongshuang Wu, and Nan Cao

***Abstract*—Automatic anomaly detection techniques have been extensively used to support decision making in abnormal situations. However, existing approaches are limited in their capacity of effectively identifying anomalies due to the complexity of the real-world environment, the uncertainty of the data input, and the unavailability of ground truth. In this paper, we propose an interactive context-aware anomaly detection algorithm framework that incorporates human judgment in searching for anomalous regions within a large geographic environment. In specific, our framework, 1) estimates a focal region and detect anomalous situations in real time, through which the user can observe and analyze suspicious entities, 2) leverages user feedback to refine results and guide further analysis, and 3) tolerates potential fault feedback provided by the users and resignal dubious anomalous points. Based on the framework, we propose two algorithm implementations, respectively, employ Bayes' theorem and metric learning. We demonstrate the effectiveness of the proposed framework and corresponding implementations through two controlled user studies and a case study with a domain expert.**

***Index Terms*—Anomaly detection, interaction techniques.**

I. Introduction

Anomaly detection refers to the problem of identifying patterns in the data that do not conform to expected behavior [1]. A variety of anomaly detection systems have been developed for different purposes such as finding environmental changes [2], improving information security on social media [3], and monitoring traffic [4].

Due to its importance, anomaly detection techniques have been extensively researched. Existing techniques primarily approach the problem through automated analysis models including classification-based [5], clustering-based [6], statistical [7], [8], and spectral methods [9]. However, the computation results of analysis models are often imprecise and even misleading due to the complexity of the real-world environment, the uncertainty of the data input, and the unavailability of ground truth [10]. Consequently, interactive strategies have also been proposed to facilitate anomaly detection such as acquiring new or updated labels from users [11]. While such human-in-the-loop approaches significantly refine the analysis models, most of them are built on the assumption that all anomalies in the environment are completely observable. This assumption, however, overlooks the impact of *environment complexity* on detecting anomalies in real-world scenarios. Here, environmental complexity measures how difficult an anomaly in an environment can be observed by humans. For example, a polluted region with a small number of monitoring stations has a low probability that its anomalous situation can be observed. When an analyst investigates such a region with high environmental complexity, he or she may incorrectly perceive that the region has not been polluted.

In response to the aforementioned limitations, we propose a more reliable and practical interactive framework that utilizes human supervision to search for anomalous regions within a large geographic environment. Our framework has the following three main advantages.

1) The framework can estimate the environment and detect anomalous situations in real time, through which a user can observe and analyze suspicious entities.
2) The framework can leverage user feedback to refine results and guide further analysis. Due to the lack of ground truth, we use user domain knowledge as the essential source for refinements, meaning that a labeling on local environment will trigger global updates and thereby guide further analysis.
3) The framework can tolerate potential false feedback by introducing environmental complexity. When the user investigates a region with high environmental complexity, he or she may provide false feedback. The fault tolerant update mechanism can re-signal dubious anomalous entities and require further investigation from the user.

Based on the proposed framework, we propose two algorithms, respectively, employ Bayes' theorem and metric learning. Bayes' theorem is selected as it can calculate the conditional probability of finding an anomaly in a specific region immediately after receiving user feedback one at a time. On the other hand, metric learning can process multiple user feedback simultaneously and apply user feedback to the data for situation update.

Manuscript received May 1, 2018; revised October 15, 2018, December 21, 2018, and April 29, 2019; accepted June 11, 2019. Date of publication July 11, 2019; date of current version November 21, 2019. This work was supported in part by the Fundamental Research Funds for the Central Universities in China and in part by the National Natural Science Foundation of China under Grant 61802283 and Grant 61602306. This paper was recommended by Associate Editor L. Rothrock. *(Corresponding author: Nan Cao.)*

Y. Shi, H. Fu, and N. Cao are with the Intelligent Big Data Visualization Lab, Tongji University, Shanghai 201804, China (e-mail: yangshi.idvx@tongji.edu.cn; fuhao@icrd.com.cn; nan.cao@tongji.edu.cn).

M. Xu is with the University of Florida, Gainesville, FL 32611 USA (e-mail: maoranxu@ufl.edu).

R. Zhao is with the University of California, Santa Cruz, Santa Cruz, CA 95064 USA (e-mail: rzhao17@ucsc.edu).

T. Wu is with the University of Washington, Seattle, WA 98195 USA (e-mail: wtshuang@cs.washington.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/THMS.2019.2925195

The results of our two user studies suggests that the framework can effectively support anomaly detection. Within the framework, the *Metric-Update* implementation outperforms the *Bayes-Update* in terms of the detection accuracy and time cost. In addition, an expert walk through in an air quality monitoring scenario suggests that our framework can provide solid supports for real use cases.

In summary, our work has the following main contributions.

1) *Algorithm framework:* We introduce an online interactive context-aware anomaly detection algorithm framework. The proposed framework provides an efficient anomaly detection mechanism that interactively adopts human judgment and includes environmental complexity into computations.
2) *Situation update algorithms:* Based on the proposed framework, we provide two algorithmic implementations that employ Bayes' theorem and metric learning, respectively, to help detect regional anomalies in the environment.
3) *Evaluation:* We conducted two controlled user studies, each having 18 participants, and a case study with a domain expert to verify the usefulness of the framework and compare the effectiveness of the two proposed algorithmic implementations.

In the rest of this paper, we first provide a brief survey of related work (see Section II). Then, we ground our motivation with a use scenario (see Section III), and then introduce the framework and two implementations (see Section IV). For evaluation, Section V describes the details and rationales of the experiment design, followed by the first study, analysis results, and discussions (see Section VI). Section VII describes the second user study and an expert walk-through based on the prototype system.

II. Related Work

Anomaly detection is a well-established field aiming at finding patterns in data that deviate from normal behavior [12]. In response to the indecisive "anomaly" definition problem, prior studies have also proposed interactive strategies to help practitioners personalize their detection models based on their own decision boundaries. Researchers have proposed multiple forms of feedback. For instance, Konijn and Kowalczyk [13] enabled users to iteratively label outliers until no more interesting outliers can be found. Krasuski and Wasilewski [14] improved the detection of outlying Fire Service's reports by discussing the features and decision boundaries with domain experts. Cao *et al.* presented TargetVue [11], a visualization system that displays the analysis results of anomalous online user behaviors and collects feedback from analysts. Liao *et al.* [15] developed GPSvas, which embeds an active learning process in its visual analysis, allowing users to input manual labels for further training. Online learning approaches has also been applied to anomaly detection. For example, Ahmad *et al.* [16] detected anomalies in streaming data using an online sequence memory algorithm. Similarly, Ozkan *et al.* [17] utilized the Markov model to detect anomaly in time series data. Bastani *et al.* [18] provided an efficient framework using sequential Monte Carlo for surveillance video. The aforementioned work helps efficiently detect anomalies in large scale, streaming data without spatial information. Laxhammar and Falkman [19] proposed the Sequential Hausdorff Nearest-Neighbor Conformal Anomaly Detector for online learning and sequential anomaly detection on geo-spatial trajectory data. Cao *et al.* [4] designed a visual interactive system, Voila, that uses Bayesian approach to detect anomalies in an urban context. Our interactive framework incorporates human supervision in searching for anomalous regions within a large geographic environment. The framework extends Voila's Bayesian approach and also introduces metric learning implementation, which is able to process multiple feedback simultaneously. Also, we use a fault tolerant mechanism by introducing environmental complexity in our algorithms to resignal dubious anomalous entities and require further investigation from the user.

III. Use Scenario

To better motivate the design of our framework, we first describe a use scenario. Suppose Alice, an urban security officer in the department of urban traffic monitoring and management, attempts to use an anomaly detection system to monitor urban traffic and find abnormal traffic incidents (e.g., the traffic flow within a region is significantly higher compared to its historical statistics). Alice first observes and analyzes the most suspicious regions ranked by the automated analysis models. However, she finds that the results are not aligned with her domain knowledge and, thus, refines the results by confirming or rejecting the anomalies detected by the system. After the refinement, the results are more accurate. However, there are tons of regions, manually checking each of them would be an onerous task. She expects that the system will use her labels as an indicator to update other regions in similar situations automatically. Unfortunately, the system fails to do so: the underlying algorithm is not designed for a spatial context, thus, all the dimensions are treated equally. Moreover, Alice notices that the system fails to consider cases when she incorrectly identifies anomalous cases and provides false feedback. The reason is that her judgment is based on the analysis of incomplete observed data, for example, traffic cameras are sparsely distributed among specific areas, resulting in difficulties in monitoring the situation for the system.

According to the use scenario, we identified the following design requirements for our system.

DR.1 *Updating analytics based on human feedback:* The system should provide an interaction mechanism that accepts feedback from users in real-time to dynamically rectify the anomaly detection model.

DR.2 *Tolerating potential false feedback:* Users might produce false feedback due to the difficulty in perceiving the environmental context. The system should include environment complexity into computations and better tolerate false feedback.

DR.3 *Allowing the integration of different anomaly detection algorithms:* To meet different detection requirements, any algorithm that yields the probability of finding anomalies in a given environment can be embedded into the framework.

IV. Algorithm Framework and Implementation

In this section, we introduce our interactive context-aware anomaly detection algorithm framework and two algorithm implementations, respectively, based on Bayes' theorem and metric learning.

Algorithm 1: Interactive Context-Aware Anomaly Detection Framework.

```
Input: $\mathbf{E} = \{r_1, ..., r_n\}$; $\mathbf{Q} = \{q_1, ..., q_n | q_i \in [0,1]\}$
1 $\mathbf{P^{(0)}} = \{p_1^{(0)}, p_2^{(0)}, ..., p_n^{(0)}\} \leftarrow Initialization(\mathbf{E}, \mathbf{Q})$
2 while true do
3     if user cannot find more anomalies then
4         break;
5     end
6     $\mathbf{J} \leftarrow GetJudgement(\{(r_{i_1}, f_{i_1}), ...(r_{j_k}, f_{j_k}) | f_{y_x} \in \{0,1\}\})$
7     $\mathbf{P^{(s)}} = \{p_1^{(s)}, ..., p_n^{(s)}\} \leftarrow update(\mathbf{J}, \mathbf{E}, \mathbf{Q}, \mathbf{P^{(s-1)}})$
8 end
```

A. Interactive Context-Aware Anomaly Detection Algorithm Framework

The goal of the framework is to help decision makers efficiently explore a large geographic environment and locate the regions that contain anomalous entities. Based on the design requirements, we achieve the goal by designing an anomaly detection mechanism that has the following characteristics:

1) interactively adopts human judgments to reflect users' decision boundary onto the global environment;
2) includes environmental contexts into computation;
3) light-weighted and independent of the underlying anomaly detection techniques.

As described in Algorithm 1, the environment E is uniformly partitioned into a set of n equal-size cells, i.e., n regions $\{r_1, r_2, \dots, r_n\}$. We assign the environmental complexity Q to each of these n regions as $\{q_1, q_2, \dots, q_n\}$. Then, the global environment E and the environmental complexity Q are used as the inputs for the probability initialization function $initialization(\cdot)$, which generates the initial probability P of finding an anomaly in each region as $\{p_1^{(0)}, p_2^{(0)}, \dots, p_n^{(0)}\}$ (line 1). Every time when a user inspects a region, a binary feedback f_i indicating whether or not he or she finds an anomaly in the region r_i is recorded. Suppose that the user labels k regions in his inspection, $(r_{i_1}, f_{i_1}), \dots, (r_{j_k}, f_{j_k})$ is used as the input to compute user judgment J (line 6). The probability update function $update(\cdot)$ receives the human judgment J, the global environment E, the environmental complexity Q, and the probability $P^{(s-1)}$ at step $s-1$ to assign the new probability $P^{(s)}$ of each region at step s (line 7). The iterative process described above continues until the user indicates that no more anomalies can be found (lines 3–5).

B. Implementation of Update Functions

We introduce two algorithm implementations based on the above-mentioned framework. The first algorithm employs Bayes' theorem [20], which handles user feedback one at a time. The second algorithm is designed based on metric learning [21], which is able to process multiple feedback simultaneously. *A priori* knowledge of the number of anomalies is required to use the first implementation while no such knowledge is required for the second implementation. Both of these implementations employ one-class support vector machines (SVM) [22] to generate initial probability values due to its benefits of unsupervised feature learning, computational efficiency, and a good performance [23].

1) Implementation I: The first implementation employs the Bayes' theorem [20] $P(A|B) = P(B|A)P(A)/P(B)$, which calculates the probability of event A given event B is observed. The Bayes' theorem updates the priori probability with the observation and yields the posterior probability. Therefore, it can be used in real world scenarios to search for regions of interest in a large environment [24]. In our case, we assume that there is only one anomaly in one of the investigation regions $\{r_1, \dots, r_n\}$. Let A_i denote the event "an anomaly exists in the ith region" and let B_j denote the event "the user thinks that the jth region does not contain an anomaly." The probability $P(B_j)$ is positively correlated with the environmental complexity of this region, q_j. Thus, $P(A_i|B_j)$ represents the probability of an anomaly exists in the region i when a user perceives the region j as normal. Note that the assumption is simplified for computation and the Bayes' theorem based method also holds true when multiple anomalies are to be observed. We first normalize the probability for each region by the total number of anomalies. Every time when an anomaly is found, the denominator of the normalization is reduced by 1 and the probability of all the remaining regions are reduced with the same rate.

Based on the Bayes' theorem, we define the probability update function $update(\cdot)$ in Algorithm 1 as

$$\text{update}(r_i, p_i, q_i, f_j = 0) = P(A_i|B_j) = \begin{cases} \frac{P(B_i|A_i)P(A_i)}{P(B_i)} = \frac{p_i q_i}{1-p_i(1-q_i)}, i = j \\ \frac{P(B_j|A_i)P(A_i)}{P(B_j)} = \frac{p_i}{1-p_j(1-q_j)}, i \neq j. \end{cases} \quad (1)$$

Here, $P(A_i) = p_i$ indicates the probability of an anomaly in the region r_i. q_i and f_i represent the environmental complexity and user feedback of this region, respectively. Note that the above-mentioned update rules only take negative user feedback ($f_i = 0$) for update, that is, it updates the global situation only when the anomaly has not been found. Here, we assume that there is only one anomaly to be detected in the investigation space. Thus, a failed attempt of finding the anomaly at one region will increase the conditional probabilities of detecting it in other regions. Once the anomaly in one of the regions has been found ($f_i = 1$), this region r_i will be removed from the investigation space by setting $p_i \equiv 1$, and the user can start to find the next anomaly in the environment.

2) Implementation II: The second algorithm implementation embeds metric learning into one-class SVM. One-class SVM [22] computes the anomaly score using the distance from data point to the decision boundary, which can be modified by users' updates. Its key component, the kernel function, computes the distance in a hyperspace and projects the distance into the original data space. We modify the kernel based on metric learning [21]. Under the new metric, points in the same class will have small distance while points in different classes will have larger distance. Next, we will explain the how one-class SVM is implemented and how metric-learning is used to refine the distance function in one-class SVM that determines the anomaly score of each data point.

3) One-Class SVM: We use a set of data points $\{x_1, \dots, x_m\}$ to represent entities to be observed in the environment E. Each data point $\mathbf{x}_i$ is associated with a multidimensional vector in the feature space. The data points are randomly distributed in

E, among which one or more data points are anomalies. The region that contains one or more anomalies is defined as an anomalous region. One-class SVM detects anomalies by finding a tight boundary in the feature space that encloses a majority of highly related data points, while points outside the boundary are identified as the anomalies. The distance between the points and the boundary indicates the anomaly score. The algorithm projects the data points into a latent hyperspace of a higher dimension, where the points can be easily separated by a hyperplane. Here, the equation representing the hyperplane is defined as

$$\sum_{i=1}^{m} w_i K(\mathbf{x}, \mathbf{x_i}) - \rho = 0 \tag{2}$$

where w_i and ρ are parameters learned from the input data, $\mathbf{x}$ and $\mathbf{x}_i$ represent the data point of interest and one of the data points in the dataset, respectively. The kernel function $K(\mathbf{x}, \mathbf{x_i})$ can be interpreted as estimating the distance between the two data points in a hyperspace. Usually, Gaussian kernel is used as it is the most frequent one used in nonlinear cases. The algorithm finds the tight boundary by ensuring the distance from a point to the hyperplane in the hyperspace to be equivalent to the distance from the same point to the boundary in the feature space. Here, the distance function is as follows:

$$\text{dist}(\mathbf{x}_i) = \sum_{j=1}^{m} w_j K(\mathbf{x_i}, \mathbf{x_j}) - \rho \tag{3}$$

where w_i and ρ are parameters learned from the input data, $\mathbf{x}_i$ and $\mathbf{x}_j$ represent the data points in the dataset. Intuitively, when a data point lies on the hyperplane, the distance is equal to zero; otherwise, the distance has a positive or negative value, indicating the point lies inside (normal) or outside (abnormal) the boundary, respectively.

4) Metric Learning: To refine the distance function in one-class SVM, we first introduce a new kernel function (K_m) based one metric learning

$$K_m(\mathbf{x}, \mathbf{x_i}) = \exp\left(-\frac{d_M^2(\mathbf{x}, \mathbf{x_i})}{2\sigma^2}\right) \tag{4}$$

where $\mathbf{x}$ and $\mathbf{x}_i$ represent the data point of interest and one of the data points in the dataset, respectively. $d_M(\cdot)$ is a distance metric defined as

$$d_M(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{(\mathbf{x_i} - \mathbf{x_j})^T M (\mathbf{x_i} - \mathbf{x_j})}. \tag{5}$$

The goal of the update is to find one optimal matrix M^* that best matches the user judgment in terms of separating the normal and abnormal situations. To this end, we then employ the least-square metric learning (LSML) [25], in which M^* can be learned based on a set of constraints $\mathcal{C}$ in the form of

$$\mathcal{C} = \{(\mathbf{x_a}, \mathbf{x_b}, \mathbf{x_c}, \mathbf{x_d}) : d_M(\mathbf{x_a}, \mathbf{x_b}) < d_M(\mathbf{x_c}, \mathbf{x_d})\} \tag{6}$$

where $\mathbf{x}_a$ and $\mathbf{x}_b$ are data points from the same class while $\mathbf{x}_c$ and $\mathbf{x}_d$ are data points in different classes. $\mathcal{C}$ ensures that the pairs of points within the same class to be closer than the pairs from different classes. $\mathcal{C}$ can be automatically generated based on user feedback f_i, the probability p_i, and the environmental complexity q_i for the ith region. Here, $\mathbf{M}(f_i, p_i, q_i)$ is used to denote the function that generates the optimal matrix M^* and Algorithm 2

Algorithm 2: Metric Learning Algorithm for $\mathbf{M}(f_i,p_i,q_i)$.

```
   Input: $f_i, p_i, q_i$, $C = \varnothing$;
1  $\mathcal{A} = \{r_j | p_j \in \mathbf{P}, p_j > threshold\}$;
2  $\mathcal{N} = \{r_k | p_k \in \mathbf{P}, p_k \leq threshold\}$;
3  for j=1 to NumOfConstraints do
4      $a, b, c \leftarrow sample(\mathcal{N})$, $d \leftarrow sample(\mathcal{A})$;
5      $C \leftarrow C \bigcup \{(a, b, c, d)\}$
6  end
7  if $f_i$ == 1 then
8      $\mathcal{A} = \mathcal{A} \bigcup r_i$;
9      for j = 1 to n do
10         $a \leftarrow sample(\mathcal{A})$, $b \leftarrow sample(\mathcal{N})$;
11         $C \leftarrow C \bigcup \{(r_i, a, r_i, b)\}$, with probability $1 - q_i$;
12     end
13 else
14     $\mathcal{N} = \mathcal{N} \bigcup r_i$;
15     for j = 1 to n do
16         $a \leftarrow sample(\mathcal{N})$, $b \leftarrow sample(\mathcal{A})$;
17         $C \leftarrow C \bigcup \{(r_i, a, r_i, b)\}$, with probability $1 - q_i$;
18     end
19 end
20 $\mathbf{M}^* \leftarrow LSML(C)$;
```

describes the metric learning algorithm for $\mathbf{M}(f_i, p_i, q_i)$. $\mathcal{A}$ denotes the abnormal training set while $\mathcal{N}$ denotes the normal training set, p_j is the probability of observing an anomaly in the region j. If p_j is greater than a *threshold*, there is a high probability that the region j contains an anomaly. These high-anomaly regions constitute set $\mathcal{A}$ while those low-anomaly regions constitute set $\mathcal{N}$ (lines 1–2). In our case, we set the threshold as 0.5. The reason is that the values of p are evenly distributed between [0, 1], so the midpoint, 0.5, is selected. Accordingly, $\mathcal{A}$ and $\mathcal{N}$ are of roughly the same size. *NumOfConstraints* is a parameter that fixes the sample size of constraints, which is empirically set to the number of regions in our case. a, b, c, and d are sampled from $\mathcal{A}$ and $\mathcal{N}$ (lines 3–6). Here, $sample(\cdot)$ is a random sampling. If the user detects an anomaly in the region r_i (line 7), r_i is appended to the set $\mathcal{A}$ (line 8). Then, the new constraints (r_i, a, r_i, b) reported by the user are added to the constraints collection $\mathcal{C}$ with probability $1 - q_i$, meaning that the user judgment on the regions with less environment complexity are more reliable and these regions are more likely to change the metric (lines 9–12). Otherwise, the region r_i is appended to the set $\mathcal{N}$ and the related constraints are added likewise (lines 13–19).

Based on the (3)–(5), the probability update function $update(\cdot)$ in Algorithm 1 is as follows:

$$\text{update}(r_i, p_i, q_i, f_i) = N\left(\sum_{j=1}^{m} w_j \exp\left(-\frac{(\mathbf{x_i} - \mathbf{x_j})^T \mathbf{M}(f_i, p_i, q_i)(\mathbf{x_i} - \mathbf{x_j})}{2\sigma^2}\right) - \rho\right) \tag{7}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ represent data points in the dataset, $\mathbf{M}(f_i, p_i, q_i)$ denotes the function that generates the optimal matrix M^*, $N(\cdot)$ normalizes the results into $[0, 1]$. σ is a free parameter to tune the fitness and smoothness of the decision boundary.

In our case, σ is set it to 1. $N(\cdot)$ is conducted on all the regions whenever the probability is updated. We use min–max normalization on each region and normalize the results into [0, 1]. The formula of normalization is: $N(r_i) = (p_i - p_m)/(p_M - p_m)$, where p_i is the anomaly score for region r_i, p_m, and p_M denote the minimum and maximum scores among the n regions, respectively.

V. Experiment Design

To evaluate the effectiveness of the algorithm framework in supporting detect anomalies and compare the performance of the two algorithm implementations under different conditions, we designed a controlled user study. In this section, we describe the details and rationales of the experiment design.

A. User Task

We design tasks that simulate real-world scenarios of anomaly detection, in which a small portion of potential anomalies needs to be found among a large collection of data points distributed in a two-dimensional (2-D) area. Users are required to find the anomalies and label the regions that contain these anomalies. Hence, the user task is described as follows.

Locate the regions that contain anomalous entities (i.e., the data points that have different feature values compared with that of other points) in a 2-D spatial environment.

In this task, the primary variable to be tested is the choice of the algorithms (i.e., *No-Update* where anomaly detection is implemented without situation update (baseline), *Bayes-Update*, and *Metric-Update*). Additionally, the study tested two more variables including 1) the scale of the investigation space, which is determined by the number of grids (denoted as g^2) and 2) the number of anomalies to be found in the investigation space (denoted as n_a). We determine the proper setting of the two variables g^2 and n_a by conducting a pilot study with 20 participants. Based on the results of the pilot study, the number of grids g^2 is set to either 10^2 (small) or 15^2 (large), and the number of anomalies n_a is set to either 3 (small) or 7 (large). In the formal user study, we defined four task including *T1 (G10-A3):* finding three anomalies in 10 $\times$ 10 grids, *T2 (G10-A7):* finding seven anomalies in 10 $\times$ 10 grids, *T3 (G15-A3):* finding three anomalies in 15 $\times$ 15 grids, and *T4 (G15-A7):* finding seven anomalies in 15 $\times$ 15 grids. Each task is repeated for three times to reduce random noise.

B. Dataset

We synthesized a testing dataset simulating all of the aforementioned task conditions. We first generated a collection of data points from multivariate Gaussian distribution with mean $\mu = (1, 1, 1, 1, 1, 1)^T$ and covariance matrix $\Sigma = \text{diag}\{1, 1, 1, 1, 1, 1\}$. These data points are grouped into the normal points (distance from the mean μ were less than 3σ) and abnormal ones (distance greater than 3σ). Each data point is associated with a 6-D vector. We then placed these points randomly in a 2-D plane. The probability p is initialized by the anomaly score output from one-class SVM with an untrained Gaussian Kernel.

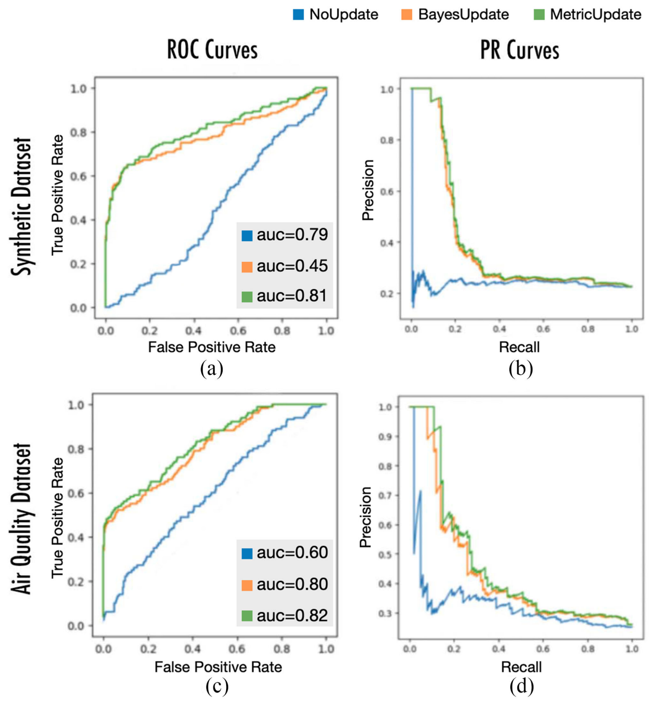


Fig. 1. Evaluation curves of the three algorithms. (a) and (b) Drawn from the testing dataset. (c) and (d) Drawn from the China air quality dataset on January 15, 2017.

C. Preliminary Test

We first ran a preliminary test to evaluate the performance of the three algorithms (*No-Update*, *Bayes-Update*, *Metric-Update*) on the testing dataset when no human interaction is involved. We let the three algorithms always pick the most anomalous region and make an update of the current situation. Receiver operating characteristic (ROC) curves were then generated by comparing the perceived labels with the original labels of these updated regions. Fig. 1(a) and (b) shows that both *Bayes-Update* and *Metric-Update* outperform *No-Update* while *Bayes-Update* is slightly better than *Metric-Update*.

D. Study Hypotheses

Our hypotheses were formed as follows.

H.1 The algorithm framework will enhance the user performance of detecting anomalous entities in the environment.

H.2 Users will achieve higher task accuracy in detecting anomalous regions using *Metric-Update* than *Bayes-Update* and *No-Update*.

H.3 User will spend less completion time in detecting anomalous regions using *Metric-Update* than *Bayes-Update* and *No-Update*.

The preliminary test suggests that *Metric-Update* outperforms *Bayes-Update* and *No-Update* when no human interaction is involved. As human might have different observations regarding our visualization and interaction design, we posed *H.2* to further compare task accuracy of the three algorithms when user interaction is involved. We hypothesized that users would spend less completion time using Metric-Update (*H.3*) as it accepts

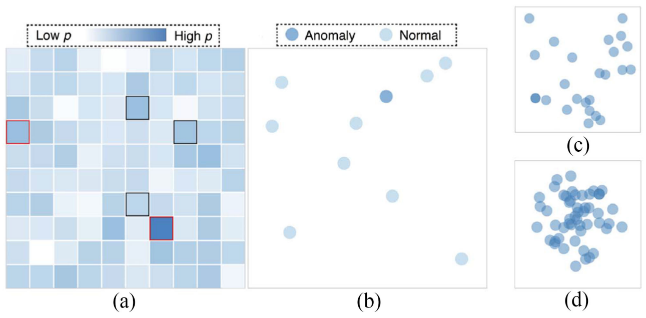


Fig. 2. User study interface. (a) Global view. (b) Detail view. The results in the detail view with different environmental complexity. (c) $q = 0.1$. and (d) $q = 0.8$.

multiple feedback simultaneously. Based on the accuracy (*H.2*) and time (*H.3*), we posed the overall hypothesis *H.1* that our algorithm will enhance user performance.

E. Task Performance Measures

To quantify user performance of detecting anomalies via different algorithms, we use task accuracy and completion time.

Task Accuracy: There are two measures of accuracy: the precision and recall rate. Our pilot study showed that the precision rate was not a proper measure to reflect the accuracy compared to the recall rate, as users rarely made mistakes in identifying anomalous grids. As a result, the recall rate was selected as the measure of accuracy in the user study.

Completion time: The completion time measures the duration starting at the time when the dataset is displayed to users, and stopping at the time when users click the "next" button to begin the next trial, which contains both the inspection time and response time. Users can click the "pause" button when having a break, during which the time recorder is held up.

F. User Interface

To evaluate how well each algorithm in our framework help users detect anomalies in the environment, we design an user interface with two coordinated views, the global view and detail view, as shown in Fig. 2.

Global view: The global view [see Fig. 2(a)] displays an overview of the anomalous information in the environment. We overlaid equal-sized grids to uniformly partition the environment to be investigated. Each grid represents a region, with the color illustrating the probability of containing an anomaly in this region. A grid with darker blue indicates that the region has a higher probability of containing an anomaly (i.e., high p value).

Detail view: The detail view [see Fig. 2(b)] shows individual entities in a specific region that the user has selected in the global view. Each data point represents an entity, with the opacity illustrating whether the entity is an anomaly. An opaque blue point indicates an anomaly while a translucent blue point encodes a normal entity. Here, we selected opacity to differentiate between anomalous and normal entities as it can be also used to show environmental complexity q through the overlapping rate of the points. That is, a set of highly overlapped translucent points will result in difficulties in identifying opaque ones. For example, by comparing Fig. 2(c) and (d), we found that the region with higher environment complexity [see Fig. 2(d)] is more difficult for users to identify the anomaly (i.e., opaque blue point).

We control the environmental complexity via the distribution of the data $N(0, 1 - q)$, where $N(\cdot)$ is the normal distribution. In the user study, the environmental complexity of each grid in each trial is randomly determined.

During the process of anomaly detection, a user can first investigate the most suspicious region in the global view by hovering on the grid that has the highest p value, that is, the region shown in the darkest blue color. Once the focal region is identified, he or she can then inspect its data points in the detail view. Based on his/her judgments of these data points, the user can label the grid in the global view; a single-click indicates the region indeed contains an anomaly, which can be canceled by a double-click. Additionally, the grids that have been hovered by the user are marked with thick black borders to avoid duplicated investigations. The user can also submit his/her judgments (i.e., the labels of grids) to the back end by a right-click whenever he or she wants, the algorithms will automatically update the p value for each grid and its corresponding color based on user feedback.

Before integrating the visual interface into the framework, the pilot study also investigates 1) whether the environmental complexity determined by $N(0, 1 - q)$ align with users' perception and 2) the "accuracy-q" correlation regarding different number of data points in a grid. The results suggest that our design of environmental complexity align with users perception and 30 points per grid is the best setting.

VI. User Study I

In this section, we first describe the study method, followed by the analysis results and discussions.

A. Method

We recruited 18 participants (9 females) with an average age of 22.06 (SD = 1.95) for the user study with the goal of evaluating and comparing three algorithms, *No-Update*, *Bayes-Update*, and *Metric-Update*, implemented in the interactive context-aware anomaly detection framework. Before the study, we conducted a 20-min tutorial session, during which the concept of anomaly detection and its application in real-world scenarios were briefly introduced. Next, we described in detail the framework with the proposed algorithms, the user interface, and the interactions. In the practice session, the participants were instructed to use the system with a sample dataset.

The study consisted of three sessions, each of which involved one of the three algorithms. In each session, participants completed four tasks, *T1 (G10-A3)*, *T2 (G10-A7)*, *T3 (G15-A3)*, *T4 (G15-A7)*). Both the task accuracy and completion time were recorded automatically for later analysis. We counterbalanced the order of the three sessions as well as the order of four tasks to avoid learning effects. Upon the completion of the three sessions, we asked the participant to complete a questionnaire. The user study took approximately 45–55 min.

This study was performed on a 13.3-in laptop computer with a display resolution of 2560×1600 and each trial was displayed in a 2000×1200 window with a white background. The size of

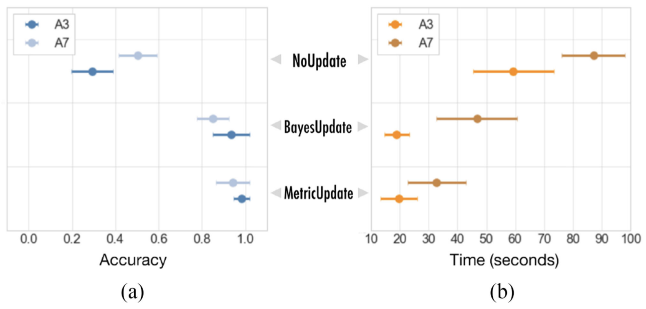


Fig. 3. (a) Accuracy and (b) completion time for 10×10 grids (small).

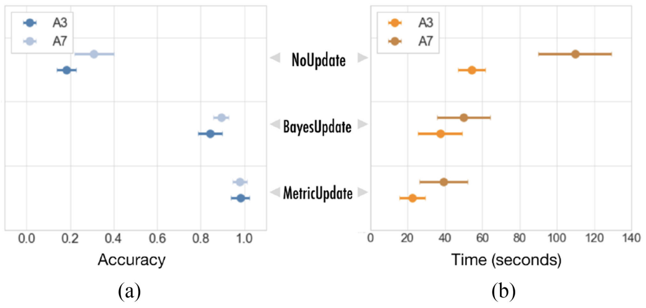


Fig. 4. (a) Accuracy and (b) completion time for 15×15 grids (large).

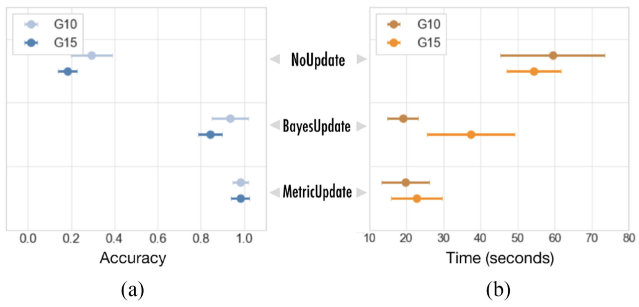


Fig. 5. (a) Accuracy and (b) completion time for three anomalies (small).

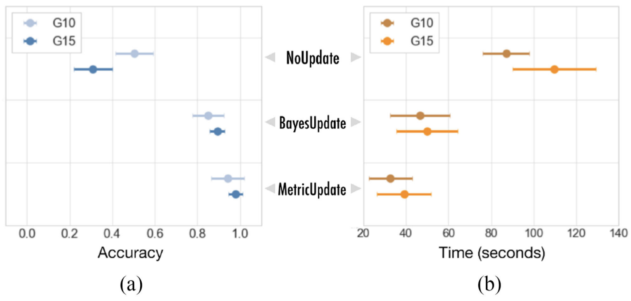


Fig. 6. (a) Accuracy and (b) completion time for seven anomalies (large).

each grid was adjusted automatically according to the number of grids g^2.

B. Result Analysis

We now report the quantitative results from the abovementioned user study. We first analyze the effect of two study variables (number of the grids and anomalies) on the task performance. We then compare the accuracy and completion time of three algorithms (*No-Update*, *Bayes-Update*, and *Metric-Update*). Finally, we show the results form the poststudy questionnaire. Repeated Measures ANOVA (RM-ANOVA) was applied to examine if there is a significant difference. Bonferroni correction was used to conduct the pairwise comparisons.

1) Validation of Variables: To evaluate the effect of the number of grids and anomalies, we analyzed user performance under different conditions.

Small grid number (10×10): RM-ANOVA shows that the number of anomalies significantly affected user performance in terms of the accuracy ($F(2, 34) = 15.81, p < 0.05$) across all three algorithms [see Fig. 3(a)]. The analysis results showed no significant difference in completion time [see Fig. 3(b)]. Compared to *No/Bayes-Update*, *Metric-Update* was the least sensitive to the change of anomaly numbers in both accuracy and time.

Large grid number (15×15): Fig. 4(a) illustrates that the accuracy was significantly lower when the number of anomalies increased ($F(2, 34) = 12.65, p < 0.05$) across all algorithms. Fig. 4(b) suggested that the completion time was also significantly influenced by the anomaly number ($F(2, 34) = 68.23, p < 0.01$). RM-ANOVA showed that *Metric-Update* and *Bayes-Update* were less influenced in both task accuracy and completion time than *No-Update*.

Small anomaly number (3): Fig. 5(a) shows that the accuracy of both *No-Update* and *Bayes-Update* significantly decreased (*No-Update:* $F(2, 34) = 19.95, p < 0.05$; *Bayes-Update:* $F(2, 34) = 10.17, p < 0.05$) when the grid number increased. The accuracy of *Metric-Update* was less sensitive to the number of the grids. In terms of completion time, only *Bayes-Update* was significantly influenced ($F(2, 34) = 63.79, p < 0.01$) [see Fig. 5(b)].

Large anomaly number (7): Metric/Bayes-Update showed no significant difference between the two grid numbers while *No-Update* was significantly influenced in terms of both accuracy ($F(2, 34) = 23.28, p < 0.01$) [see Fig. 6(a)] and completion time ($F(2, 34) = 46.36, p < 0.01$) [see Fig. 6(b)].

2) Comparison of Algorithms: We compare the task accuracy and completion time of the three algorithms under four task conditions, T1,T2,T3, and T4.

Task accuracy: When grid number was either small or large, significant differences were observed among the three algorithms in the two levels of anomaly number, as shown in Fig. 7. Moreover, post-hoc analysis showed that in most of the cases (T1 (G10-A3): $F(2, 34) = 6.05, p < 0.05$, T2 (G10-A7): $F(2, 34) = 34.11, p < 0.01$, T4 (G15-A7): $F(2, 34) = 21.36, p < 0.01$), *Metric-Update* significantly outperformed *Bayes-Update* in accuracy (*H.2* accepted).

Completion time: Fig. 8 showed a significant difference among the three algorithms. Those showing a difference were associated with grid number as well as anomaly number. Moreover, a post-hoc analysis showed that in most of these cases (i.e., T1 (G10-A3): $F(2, 34) = 79.16, p < 0.01$, T2 (G10-A7): $F(2, 34) = 98.32, p < 0.01$, T4 (G15-A7)): $F(2, 34) = 147.21, p < 0.01$, *Metric-Update* significantly outperformed *Bayes-Update* in completion time (*H.3* accepted). As a result, we verified that our framework significantly enhances user performance in terms of accuracy and time (*H.1* accepted).

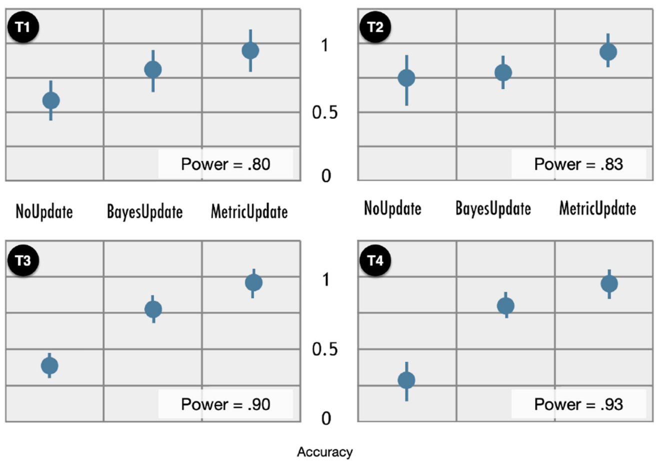


Fig. 7. Accuracy for four user tasks, including T1 (G10-A3): finding three anomalies in 10 × 10 grids, T2 (G10-A7): finding seven anomalies in 10 × 10 grids, T3 (G15-A3): finding three anomalies in 15 × 15 grids, and T4 (G15-A7): finding seven anomalies in 15 × 15 grids.

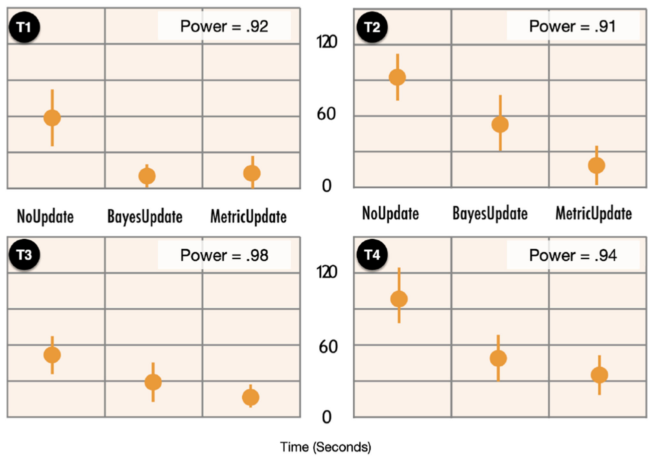


Fig. 8. Completion time for user four tasks, including T1 (G10-A3): finding three anomalies in 10 × 10 grids, T2 (G10-A7): finding seven anomalies in 10 × 10 grids, T3 (G15-A3): finding three anomalies in 15 × 15 grids, and T4 (G15-A7): finding seven anomalies in 15 × 15 grids.

3) Poststudy Questionnaire: The poststudy questionnaire was designed to qualitatively estimate the three algorithms. Questions 1–6 asked users to rate the ease of use and usefulness of each method for anomaly detection using a five-point Likert scale, as shown in Fig. 9(a). Questions 7–10 asked users to choose the method that they thought the most effective under various task conditions (i.e., larger or smaller grid number and larger or smaller anomaly number), as shown in Fig. 9(b). The results suggest that *Metric-Update* is favored by most of the participants, which supports the quantitative findings.

C. Discussion

We now discuss why and when *Metric/Bayes-Update* are useful and what are the challenges of using the framework.

Why did Metric-Update outperform Bayes-Update? According to the statistics, *Metric-Update* outperformed *Bayes-Update* on both the accuracy and completion time. *Bayes-Update* uses human judgment as an observation to update prior anomaly

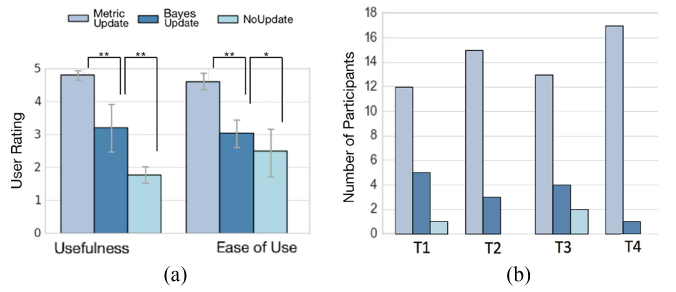


Fig. 9. Questionnaire statistics. (a) Users' rating of different algorithms with respect to their usefulness and ease of use (**: $p < 0.01$ and *: $p < 0.05$). (b) Users' preference of the three algorithms.

scores. Equation (1) also shows that *Bayes-Update* updates the anomaly scores of unchecked grids ($i \neq j$) with the same factor, resulting in the same level of color changes in grids. These changes may be difficult for users to perceive. *Metric-Update*, on the other hand, involves human judgment and data features into calculation. It uses human judgment as additional labels for training and update the hyperplane that separates abnormal and normal data points and, thus, achieves higher accuracy. In terms of completion time, users suggested that it was time consuming to update after each check when using *Bayes-Update* or to randomly guess when using *No-Update*. On the other hand, *Metric-Update* adapts a simultaneous update strategy and, thus, costs less time.

When should Metric-Update be used? Fig. 9 shows that *Metric-Update* was the most preferred method across all conditions. In terms of robustness, it was also less sensitive to the variation of the scale of the investigation space and the number of anomalies (see Figs. 3–6). Therefore, *Metric-Update* is recommended as the first choice in most real-world scenarios, especially in the complex and large-scale cases.

When should Bayes-Update be used? Fig. 9 shows that the preference for *Bayes-Update* was higher in T1 compared to T2, T3, and T4. Some participants suggested that it immediately responded to the clicks and, thus, provided good user experience. The reason is that Bayes-Update does not use feature values of data for calculation in each update, resulting in quick response. Figs. 7 and 8 also illustrate that *Bayes-Update* achieved good performances in T1 and T3 where the amount of anomalies is small. Therefore, *Bayes-Update* is applicable in simple cases due to its quick response.

VII. User Study II

To further evaluate the effectiveness of the framework and algorithms in real-world scenarios, we conducted another user study with 18 participants and an interview with an expert who is a professor in environmental engineering using a real-world dataset. Both the participants and the expert were required to monitor air quality in China and find anomalous regions where the air has been polluted.

A. Dataset

We used the air pollutant concentration dataset (http://pm25.in) collected from more than 1400 air quality monitoring stations

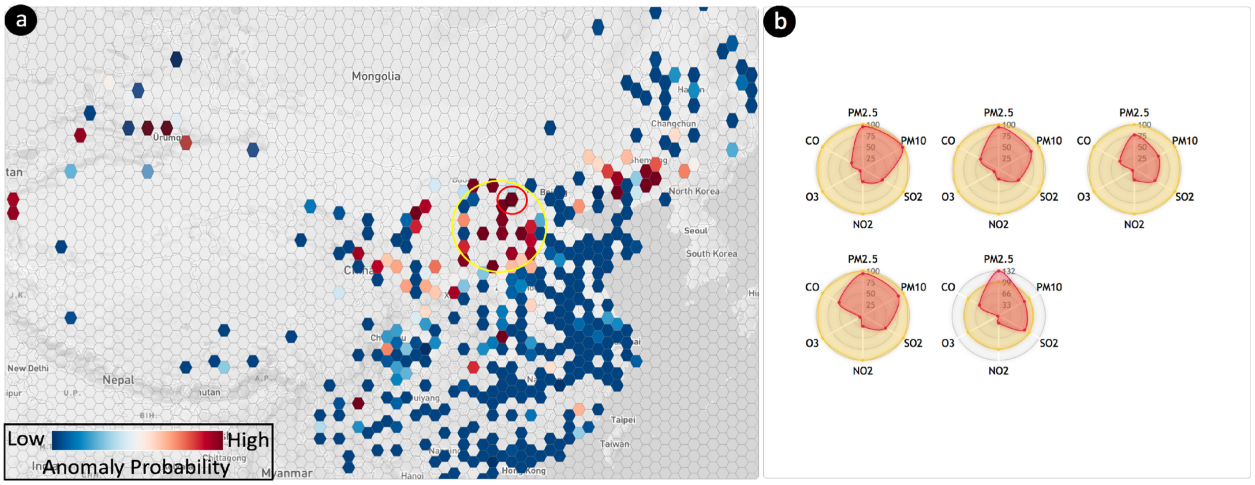


Fig. 10. User interface of the prototype system, iDetector, consists of two major views. (a) Global view. (b) Detailed view.

located in 367 urban cities as the real-world dataset. These monitoring stations record six pollutants, including nitrogen dioxide (NO_2), sulfur dioxide (SO_2), ozone (O_3), carbon monoxide (CO), and particulate matter (PM2.5, PM10).

We first standardized the value of each pollutant based on the individual air quality index (IAQI) to facilitate a comparison. Then, we used a 6-D feature vector to capture air quality in a region recorded by the local monitoring station. The feature vector indicates the average IAQI value of the six pollutants. Specifically, the likelihood of the air in a region been polluted p is calculated using one-class SVM by comparing the features of the region to that of other regions as well as the historical data. The environmental complexity q is negatively proportional to the number of air quality monitoring stations inside the region (note that q is not visualized in iDetector). Based on the air quality index (AQI) and health implications, the monitoring stations whose AQIs are above 200 are defined as anomalies.

B. iDetector

We designed an interactive anomaly detection system, iDetector, for our second user study. iDetector consists of two coordinated views, the global view and detail view. The global view [see Fig. 10(a)] display a map of China overlaid with equal-sized hexagonal grids that uniformly segment the environment. It shows an overview of anomalous regions with the color of each grid illustrating p of a region. A region's color is blending between red and blue to, respectively, encode the high and low anomaly score. Gray grids indicate that no data have been collected from those regions.

Fig. 10(b) shows the detail view of air quality in a focal region. iDetector visualizes air quality recorded by each monitoring station as a radar chart with each axis indicating a pollutant. In the glyph, the current situation of air quality is drawn in red in the foreground while the standard baseline is drawn in yellow in the background. This design facilitates a fast comparison and allows users to quickly identify an anomaly when the current score is greater than the baseline (with the red region exceeding the yellow boundary).

C. User Study and Result Analysis

The second user study follows the same protocol used in the first user study. A total of 18 participants (11 females) with an average age of 24.94 (SD = 2.60) were recruited. In each session, participants used iDetector with one of the three algorithms implemented (*No-Update*, *Bayes-Update*, *Metric-Update*) to explore the air quality dataset. Note that the performance of the three algorithms on the real-world dataset was evaluated using ROC/PR curves [see Fig. 1(c) and (d)].

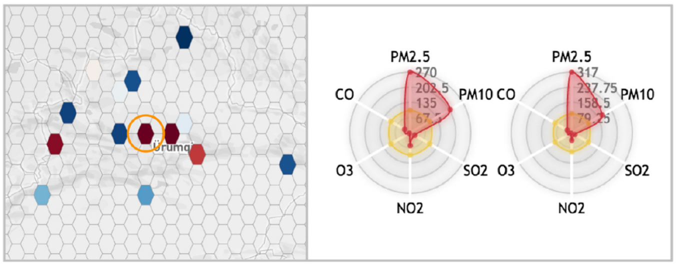
Task accuracy: The results show significant difference in accuracy ($F(2, 34) = 59.14, p < .01$). The post-hoc test suggests that participants achieved significantly higher accuracy when using *Metric-Update* than *Bayes-Update* and *No-Update* (*H.2* accepted).

Completion time: Significant difference is found in time ($F(2, 34) = 38.85, p < .01$). The post-hoc test suggests that participants spent significantly less time using *Metric-Update* than *Bayes-Update* and *No-Update* (*H.3* accepted). As a result, we verified that our framework significantly enhances user performance in terms of accuracy and time (*H.1* accepted).

D. Expert Interview

The expert interview used the air quality dataset on January 15, 2017 and iDetector with *Metric-Update* implemented. During the process, the expert first identified the most suspicious regions at first glance in the global view. "It is intuitive, I searched for the regions shown in the darkest red, and it turned out to be Henan Province based on the geographic information" [highlighted via a yellow circle in Fig. 10(a)]. He then hovered one of these regions with the highest anomaly score (highlighted via a red circle) and explored its detail view [see Fig. 10(b)].

The detail view shows radar charts that visualize the air quality recorded by the five monitoring stations in the focal region. By comparing the current situation (red) with the baseline (yellow) in each chart, the expert found that even though the values of several pollutants are high, most of them are within the normal range. He considered it as a normal situation, and thus, double-clicked to reject the result and label the region as



(a) (b)

Fig. 11. Anomalous region with high air pollution risk (a region in Xinjiang, China) revealed in (a) global view and (b) detail view.

normal and right-clicked to commit the change to iDetector for situation update. This update resulted in color changes in other regions in the global view. “I noticed that some regions turned from red to blue.” The expert explored these grids and found that the regions now turned to normal were the ones similar to the region he had labeled. “I see, it learned from what I did and made similar judgment automatically. It is smart!” The expert also noticed that the color of some grids turned to darker red. He investigated these grids in the detail view and found that the radar charts display extremely high PM2.5 and PM10 values compared to the baseline. These regions were then marked as anomalies by the expert and the global environment was update accordingly.

Another suspicious area that caught the expert’s attention is in Xinjiang Province shown in dark red in Fig. 11(a) (highlighted via an orange circle). He explored a specific region in this area in the detail view [see Fig. 11(b)]. By evaluating the pollutant values, he confirmed that this region had been polluted with high PM2.5 and PM10, and thus, single-click to mark it as the anomalous region. After situation update, he noted that “this time my judgment did not result in obvious color changes in other areas and I was wondering why.” We explained to him that his decision is of low confidence due the high environmental complexity q in this region, that is, data collected from only two monitoring stations is available for analysis. Knowing this reason, the expert was impressed by our technique and suggested “this is brilliant! I can imagine this mechanism to be used in many applications in practice.”

VIII. Conclusion

In this paper, we introduce an online interactive algorithm framework to support the analysis process of anomaly detection and two algorithm implementations based on Bayes and metric learning, respectively. The results of the two user studies indicate that the framework is useful to identify regions that contain anomalous entities and the *Metric-Update* algorithm significantly outperforms the *Bayes-Update* algorithm and the baseline in terms of accuracy and completion time. The case study with a domain expert further verified the usefulness of the proposed technique. Future work includes refining the algorithm to provide smoother update results, capturing temporal and spatial features for analysis, and applying our technique to more real-world applications.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[2] J. H. Faghmous and V. Kumar, “Spatio-temporal data mining for climate data: Advances, challenges, and opportunities,” in *Proc. Data Mining Knowl. Discovery Big Data*, 2014, pp. 83–116.

[3] J. Zhao, N. Cao, Z. Wen, Y. Song, Y.-R. Lin, and C. Collins, “# fluxflow: Visual analysis of anomalous information spreading on social media,” *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 1773–1782, Dec. 2014.

[4] N. Cao, C. Lin, Q. Zhu, Y.-R. Lin, X. Teng, and X. Wen, “Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data,” *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 23–33, Jan. 2018.

[5] V. Roth, “Kernel fisher discriminants for outlier detection,” *Neural Comput.*, vol. 18, no. 4, pp. 942–960, 2006.

[6] S. Budalakoti, A. N. Srivastava, R. Akella, and E. Turkov, “Anomaly detection in large sets of high-dimensional symbol sequences,” NASA Ames Res. Center, Mountain View, CA, USA, Tech. Rep. NASA TM-2006-214553, 2006.

[7] Z. Ju and H. Liu, “Fuzzy gaussian mixture models,” *Pattern Recognit.*, vol. 45, no. 3, pp. 1146–1158, 2012.

[8] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*, vol. 589. Hoboken, NJ, USA: Wiley, 2005.

[9] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris, “Hierarchical anomaly detection in distributed large-scale sensor networks,” in *Proc. IEEE Symp. Comput. Commun.*, 2006, pp. 761–767.

[10] D. Overby, J. Wall, and J. Keyser, “Interactive analysis of situational awareness metrics,” *Vis. Data Anal.*, vol. 8294, 2012, Art. no. 829406.

[11] N. Cao, C. Shi, S. Lin, J. Lu, Y.-R. Lin, and C.-Y. Lin, “Targetvue: Visual analysis of anomalous user behaviors in online communication systems,” *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 1, pp. 280–289, Jan. 2016.

[12] V. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, 2004.

[13] R. M. Konijn and W. Kowalczyk, “An interactive approach to outlier detection,” in *Proc. Int. Conf. Rough Sets Know. Technol.*, 2010, pp. 379–385.

[14] A. Krasuski and P. Wasilewski, “Outlier detection by interaction with domain experts,” *Fundam. Informaticae*, vol. 127, no. 1–4, pp. 529–544, 2013.

[15] Z. Liao, Y. Yu, and B. Chen, “Anomaly detection in GPS data based on visual analytics,” in *Proc. IEEE Symp. Vis. Anal. Sci. Technol.*, 2010, pp. 51–58.

[16] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[17] H. Ozkan, F. Ozkan, and S. S. Kozat, “Online anomaly detection under Markov statistics with controllable type-i error,” *IEEE Trans. Signal Process.*, vol. 64, no. 6, pp. 1435–1445, Mar. 2016.

[18] V. Bastani, L. Marcenaro, and C. S. Regazzoni, “Online nonparametric Bayesian activity mining and analysis from surveillance video,” *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2089–2102, May 2016.

[19] R. Laxhammar and G. Falkman, “Online learning and sequential anomaly detection in trajectories,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1158–1173, Jun. 2014.

[20] A. O’Hagan and J. J. Forster, *Kendall’s Advanced Theory of Statistics, Volume 2B: Bayesian Inference*, vol. 2. London, U.K.: Arnold, 2004.

[21] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, 2009.

[22] D. M. Tax and R. P. Duin, “Support vector domain description,” *Pattern Recognit. Lett.*, vol. 20, no. 11–13, pp. 1191–1199, 1999.

[23] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[24] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, “Recursive bayesian search-and-tracking using coordinated UAVs for lost targets,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2521–2526.

[25] E. Y. Liu, Z. Guo, X. Zhang, V. Jojic, and W. Wang, “Metric learning from relative comparisons by minimizing squared residual,” in *Proc. IEEE Int. Conf. Data Mining*, 2012, pp. 978–983.