Locating Data Flow Anomaly With Statistical Distance

Ruoyu Wang

***Abstract*—Hackers can infiltrate networks via any attack vector, find their way into databases and applications. Not only can they steal chunks of data and hold them for ransom, but also alter key information contained in those systems, setting up traps hard to discover and causing damages unlikely to recover. Although the influence of data altering is not significant in single data record, the change can be observed and detect in a collective scale. In this paper, we present a dynamic collective anomaly detection technique based on statistical distance. The technique extracts distribution similarities among data collections, reducing collective anomalies into point anomalies. It dynamically calculate metrics which is adaptive to the slightly evolving features. This technique provides a basic solution where several components can be replaced by other scenario specific ones, further improving the performance of detection. To illustrate details of the technique and explore its efficiency, we applied the algorithm into a real world problem of click farming detection against malicious online sellers. The evaluation shows that the technique maps normal and anomalous data collections into two distinct clusters, yielding efficient classifiers. And those classifiers are sensitive enough to a much smaller magnitude of manipulation compared with real world malicious behaviours.**

I. Introduction

Major improvements in data acquisition, transmission and storage are driving increased data availability and affordability. Currently, there are totally 2.7ZB data in the digital universe [3] and the growing speed is doubling every two years. Big data analysis has been widely adopted in scientific experiments [23], electric business [4, 33, 11], healthcare [13], governments [19] and many other fields.

It has already been and will be much harder in the future to harness the exploding volumes of data which is now pervasive leading to many problems in data management and engineering, threatening trustworthiness and reliability of data flows inside working systems. Data error rate in enterprises is approximately 1% to 5%, and for some, even above 30% [27]. In 2016, according to the research by Tricentis [2], software failures cost $1.1 trillion US dollars in total and those at 363 investigated companies affected 4.4 billion customers, causing the total lost time of more than 315.5 years. And it takes more than 45% of total expense to eliminate those errors [25]. System configuration problems, as a matter of fact, also draw scratches in the messy picture. They take even dominant places in failures in large distributed clusters [37].

Those data anomalies may arise due to both internal and external reasons with respect to a certain system. On one hand, components inside the system may generate problematic source data. For example, in a sensor network, some sensors may generate erroneous data when it experiences power failure or other extreme conditions [26]. Data packages will be lost if sensor nodes fail to connect to network or some sensor hub goes down [14]. Also, human operators act as a heavily vulnerable part to bugs and mistakes. Some insiders even deliberately modify system configurations for malicious compromises [28]. A study [35] found that 65% of organizations state that human errors are the main cause of data problems.

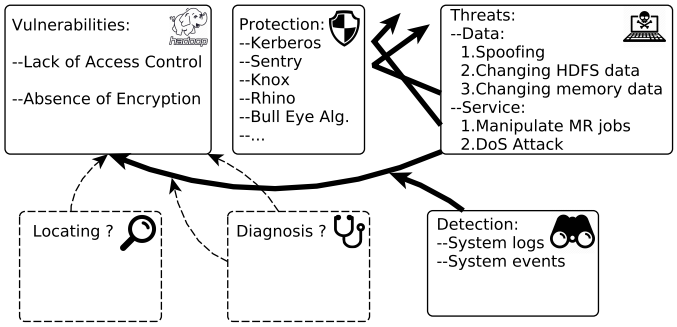

Fig. 1. Security Issues of Hadoop Clusters

On the other hand, data manipulation [17] from outside hackers composes another potential threat of data quality and reliability. *Data Manipulation* here, according to a NSA definition, refers to that "hackers can infiltrate networks via any attack vector, find their way into databases and applications and change information contained in those systems, rather than stealing data and holding it for ransom". Taken Apache Hadoop as an example. It's security issues has long been discussed within communities and industry [30, 34, 18, 31]. As is shown in Figure 1, the two basic vulnerabilities: *lack of access control* and the *absence of encryption* expose the entire cluster to dangerous threats. Data flows can be intercepted and modified; services can also be altered and blocked [15]. Although there are several frameworks(e.g. Kerberos, Sentry, Knox etc.) and algorithms providing basic protection [44, 32, 38, 42, 12], clever attackers can always bypass the barriers and sneak into the core of data pipelines. Several approaches are developed as sentinels to detect probable infiltrations. However, these approaches are not able to locate corrupted data under carefully planned manipulations. Nor can they figure out the exact reasons and recover the original records. To locate and diagnose anomalies in data pipelines under carefully planned and disguised data manipulations are still on demand by industry and academia.

According to our observation, typical data manipulations on numerical data will lead to the drift of its distribution. For example, a Taobao online seller's transaction records(see

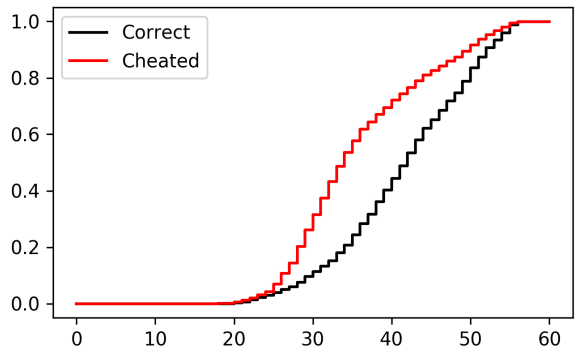

Fig. 2. Example Cumulative Distribution Function of Original And Modified Data

section V-A) can be "click farmed" to increase the volume of sales. Fig. 2 shows the sales distribution within one day. The curve for cheated data is emulated according to a popular method for click farming. It can be observed that there exists clearly a gap between these two distributions. These anomalies inside data **sets/collections** will severely affect mining and learning algorithms and further change the final decision given by the entire system.

In order to address issues caused by data manipulation, we propose a technique to detect corrupted data collections via statistical distance. As far as we know, this algorithm is the first attempt against collective data anomaly via statistical distance. Evaluations are performed on real-world and synthetic data sets, demonstrating the correctness and effectiveness of the technique.

The rest of the paper is organised as follow: Section II states related works on data anomaly detection and describes a real world problem. Section III introduces statistical distance. Details of the technique are proposed in section IV. Section V presents evaluation results and further findings of the algorithm. And all contents are concluded in section VI.

II. Related Work

A. Data Anomaly Detection

Anomaly detection, also known as outlier detection, has been studied for a long time and discussed in diverse research domains, such as fraud detection, intrusion detection, system monitoring, fault detection and event detection in sensor networks. According to a comprehensive study in [10], anomaly detection algorithms deal with input data in the form of points(or records), sequences, graphs and spatial relationships, where point data is the simplest and well studied, others are attracting more attention in new studies.

Prevalent anomalies can be classified into *Point Anomalies*, *Contextual(or Conditional) Anomalies* and *Collective Anomalies*. Point anomalies refers to an individual data instance that is considered anomalous with respect to others. But if it is anomalous only in certain circumstances or a specific context, the instance is regarded as contextual anomaly. If a group of related data(e.g. a segment of sequence) instances is anomalous with respect to other groups in the data set(e.g. the entire sequence), it is called a collective anomaly.

Detection approaches can be categorized into three types according to whether data is labeled: *Supervised*, *Semi-Supervised* and *Unsupervised* anomaly detection. As the name suggests, supervised detection methods train models on completely labeled data while unsupervised detection leverages data without any labeling. Semi-supervised detection approaches train model on data that has labeled instances for only the normal class. Supervised detection is commonly applied when both normal and anomalous data can be obtained. When it comes to the circumstances that anomalous data is hard to obtain or there exist too many diverse types of anomalies to enumerate, semi-supervised or unsupervised approaches are usually taken into consideration.

To make the final decision, detection algorithms mostly yield a score from each input instance, denoting how likely it is anomalous. The algorithm then selects top few as anomalies or compare the score with a threshold. Or, detection algorithms output a label on each instance, then decide whether each label belongs to the normal class.

Currently, distance based [7, 6] and feature evolving algorithms [21, 20, 29] algorithms seize most attention. Others adopted tree isolation [43], model based [40] and statistical methods [45] in certain applications.

To detect collective anomalies, [8] adopted the *ART(Adoptive Resonance Theory)* neural networks to detect time-series anomalies. *Box Modeling* is proposed in [9]. And *Longest Common Subsequence* was leveraged in [5] as similarity metric for symbolic sequence. Markovian modeling techniques are also popular in this domain[39, 36, 24]. [41] depicted groups in social media as combinations of different "roles" and compare groups according to the proportion of each role within each group.

B. Real World Problem: Click Farming Detection

Since the establishment of Taobao in 2003, the online business platform has been growing from the rookies in the market to the No.1 service provider in B2B and B2C markets, possessing a market share of 50.6% to 56.2% in China by 2016 [16]. Currently, there are more than 9.4 million sellers in Taobao, providing more than 1 billion different products. Under the super-pressure from massive competition, most of the sellers choose to use some cheating techniques to raise reputation and product sale volumes, then improve rankings in search lists.

The most popular approach to manipulate transaction and reputation data is *Click Farming*, where sellers use a large number of buyer accounts to create fake transaction records and give high remarks on products. Professional click farmers are usually well organized groups or companies containing thousands of people. Leaders of such groups receive orders from sellers and assign tasks to other members to create fake transactions. Some companies even develop professional applications that can be deployed on common PCs to improve productivity**[A Citation in Chinese]**.

A research performed in China showed that 81.9% of the people investigated had heard of the behaviour of click farming, 51.2% of them had seen people around them click farmed and 18.9% of them had experience of click farming themselves**[Chinese Citation?]**. American researchers reported in 2015 that over 11000 sellers were detected to have click farmed records and only 2.2% of 4000 investigaed sellers had been penalized because of the cheating attempts [22].

Current detection techniques for click farming mainly focus on user behaviours, such as browsing frequencies and periods, most common purchasing time, favourite products, remarks and whether they communicate with sellers [1]. Those techniques require the platform to keep lots of records and user features. However, the detection can be easily bypassed by trained workers and some well programmed applications.

Although it is hard to classify users as honest and malicious, we can still find clues from the sellers' aspect. For normal sellers, their customers are usually similar since choices of products are seldom changed. Therefore, the distribution of transactions in a fixed period of time, say one day, is relatively stable. No matter how much alike between the employed users or robots with honest users, the fake transaction records will always cause a bias or distortion of the original transaction distribution. Thus, if we can measure the similarity between different transaction distributions, there is still a chance for us to detect dishonest sellers.

III. Preliminaries

Statistical divergence, also called statistical distance, measures the similarity between two or more distributions. This mathematical tool can be adopted in our solution to map distributions into one dimensional points, turning collective anomaly detection problem into point anomaly detection problem. Then a concrete point anomaly detection classifier can be used to find out the cheaters.

Mathematically, statistical divergence is a function which describes the "distance" of one probability distribution to the other on a statistical manifold. Let S be a space of probability distributions, then a divergence is a function from S to non-negative real numbers:

$$D(\cdot||\cdot) : S \times S \to \mathbb{R}^+ \tag{1}$$

Divergence between two distributions P and Q, written as $D(P||Q)$, satisfies:

1) $D(P||Q) \geq 0, \forall P, Q \in S$
2) $D(P||Q) = 0$, if and only if $P = Q$

There are many ways to calculate divergence, such as f-divergences, M-divergences and S-divergences. Some of them provides better properties which brings conveniences to the design and implementation of our approach.

1) Kullback-Leibler Divergence: Let P, Q be discrete probability distributions, $Q(i) = 0$ implies $P(i) = 0$ for $\forall i$, the *Kullback-Leibler Divergence* from Q to P is defined to be:

$$KLD(P||Q) = \sum_{Q(i)\neq 0} P(i) log\Big(\frac{P(i)}{Q(i)}\Big) \tag{2}$$

For P, Q being continuous distributions:

$$KLD(P||Q) = \int_{-\infty}^{+\infty} P(x) log \frac{P(x)}{Q(x)} dx \tag{3}$$

2) Jensen-Shannon Divergence: Let P, Q be discrete probability distributions, *Jensen-Shannon Divergence* between P and Q is defined to be:

$$JSD(P||Q) = \frac{1}{2} KLD(P||M) + \frac{1}{2} KLD(Q||M) \tag{4}$$

where $M = \frac{1}{2}(P + Q)$.

A more generalized form is defined to be:

$$JSD_{\pi_1,\ldots,\pi_n}(P_1, \ldots, P_n) = \sum_{i=1}^{n} \frac{1}{\pi_i} KLD(P_i||M) \tag{5}$$

where $M = \sum_{i=1}^{n} \frac{1}{\pi_i} P_i$ and $\sum_{i=1}^{n} \frac{1}{\pi_i} = 1$.

Jensen-Shannon divergence has some fine properties:

1) $JSD(P||Q) = JSD(Q||P), \forall P, Q \in S$.
2) $0 \leq JSD_{\pi_1,\ldots,\pi_n}(P_1, \ldots, P_n) \leq log_k(n)$. If a k based algorithm is adopted.
3) To calculate $JSD(P||Q)$, it need not necessarily to be true that $Q(i) = 0$ implies $P(i) = 0$.

IV. Statistical Detection

Diverse data sets in the real world show certain structures caused by hidden patterns or relationships among records in a collection of data. For example, the traffic volume in the highway and the business transaction records, they may show a relatively stable distribution in the daily scale. Manipulation on those data(e.g. Fig 2) results in a drift or distortion of the distribution, which can be captured to trigger the alarm. Main idea of the solution is simple. But there are several detailed difficulties to be solved in the design and implementation of the technique:

1) Are the corresponding one-dimensional points distinguishable after mapping form normal and anomalous distributions? And which classifier is efficient towards those points?
2) Is there a general approach that can effectively depict different distributions in various circumstances?
3) What classifier specific metric should be adopted to filter out anomalies? Can it be further optimized?
4) Online shops are often in the process of expanding or dwindling. Thus the sales distribution will slightly change as time goes on. How to make the technique adapt to these drifting distributions?

A. Divergence-Based Collective Anomaly Detection

From section III we know that statistical divergence only provides a distance between two or more distributions. In a set of data collections, we can only draw a complete graph that each node denotes one certain collection and each edge refers to the divergence of two data collections. From the graph we

can find some points that have apparently larger distances with most of other points and return them as anomalies. This may work if anomalous ones do not compose a large proportion of all points. However the procedure will be too complicated to work out with large amounts of instances.

Otherwise we need a frame of reference and yielding absolute distances rather than the relative ones. Therefore, we should keep an evidence set E which consists of several already known correct distributions. Then the real distribution can be estimated by averaging all distributions in E. This process is similar to the parameter estimation within a certain sample set. Let M denote the estimated population distribution, then we can measure the statistical distance with respect to M for every distribution D inside or outside E, yielding absolute distances.

Fig. 7 and 8 in section V-B show the result of the above process. Red dots refer to the distances calculated from normal transaction data collections, blue and green ones are from click farmed collections. Clearly, distances of normal data collections assembles together around a small value while anomalous ones are distributed around a larger distance value. Intuitively, a simple gaussian classifier can be suitable here to discover anomalous data points. The basic framework of the algorithm is shown as follows:

Algorithm 1 Basic Classification

Input: Evidence set $E = \{D_1, \ldots, D_n\} (n \geq 2)$; New data collection D'
Output: Whether D' is anomalous
1: **for** $i \leftarrow 1$ to n do **do**
2: $P_i \leftarrow$ the distribution of D_i
3: **end for**
4: $M \leftarrow \frac{1}{n} \sum_{i=1}^{n} P_i$
5: **for** $i \leftarrow 1$ to n do **do**
6: $J_i \leftarrow JSD(P_i||M)$
7: **end for**
8: $N \leftarrow$ normal distribution estimated from $J_1, \ldots, J_n$
9: $P' \leftarrow$ distribution of D'
10: $J' \leftarrow JSD(P'||M)$
11: $p \leftarrow$ probability density of J' in N
12: **if** $p <$ threshold T **then**
13: **Return** True
14: **else**
15: **Return** False
16: **end if**

Let S be set of the daily transaction collections, if E is already given, we can run the algorithm with E and $\forall D \in S$ and discover the anomalies. Although it is convenient to compute $JSD(P_1, \ldots, P_n, P')$ instead of $JSD(P_1||M), \ldots, JSD(P_n||M), JSD(P'||M)$. It is not viable for classification. Jensen-Shannon divergence of $n+1$ distributions will dilute the affection of the abnormal one, in which case the difference between P' being normal and anomalous will become subtle when n goes very large.

According to the property of statistical divergence, we can infer that the true distribution of JSDs calculated form normal data collections are close to but not exactly a gaussian distribution $N(\mu, \sigma)$ since for each point, there are both definite upper and lower bounds instead of infinities. Therefore, μ should be slightly larger than zero. The reason why μ can not be exactly zero is hidden in line 4. M takes into consideration all existing values in every distribution sample and averages corresponding probabilities. Thus for a large $|S|$, M may highly probably contains entries that does not exist in P_i and certain probability entries in M may vary from that in P_i. For a concrete example, suppose $P_1(1) = 0.5, P_1(2) = 0.3, P_1(3) = 0.2$; $P_2(1) = 0.3, P_2(2) = 0.4, P_2(3) = 0.3$; $P_3(1) = 0.3, P_3(3) = 0.5, P_3(4) = 0.2$, then $M(1) = \frac{11}{3}, M(2) = \frac{7}{3}, M(3) = \frac{10}{3}, M(4) = \frac{2}{3}$. None of P_1, P_2, P_3 is the same with M.

B. Distribution Histogram

Not all data collections can be assumed subject to known distributions. Parameter approximation will probably lead to errors far beyond our expectation. For example, the distribution of vehicle volume in the highway may not correspond to any of the prevalent distribution model. Few assumption of parameters can be made in advance on such data. For the sake of generality, non-parametric estimation should be applied.

Surely, the kernel density estimation approaches will give a smooth, continuous distribution curve under on any sampled data. The computational cost will be much higher than applying a coarse histogram approximation. Moreover, even if we applied a distribution model upon a collection of sampled data, yielding more accurate results, the computation of divergence will be much more complex on continuous functions than discrete distributions.

Histogram methods can not only be applied to non-parametric distributions, but also to miscellaneous data types. From a general point of view, as long as there exits a function mapping the input data instance into a subset of real numbers, this data collection can be counted into a histogram. For example, date, time and single characters.

In order to generate a accurate approximation, step size is the most important parameter the algorithm should determine. If the size is too small then the resulting histogram will be over fitting; but if the size is too large then the estimation will be too coarse to depict the original shape. According to statistics theory, when dealing with a sample size of k, a step size of

$$l = c\sigma k^{-0.2} \tag{6}$$

will give a best partition, where c is a constant relative to the shape of distribution(e.g. for normal distribution, $c = 1.05$).

C. Threshold

One important factor in the algorithm is the value of threshold. A higher threshold rejects more instances, improving the sensitivity of anomalous data while increasing the number of false alarms. A lower threshold provide higher true negative rates yet neglecting more possible threats.

A naive but prevalent approach is to set a fixed value as threshold. This approach is easy to implement and may

give satisfying results in specific cases. However, a fixed threshold requires analysis of the specific problem, manual observation and tuning of parameters, which involves lots of human labor. The rule of "3σ" can be used to automatically determine a threshold. But in some cases that the anomalous data intersecting at the boundary with normal instances, the threshold may fail more often to discover those manipulated ones.

However, applying divergence as the distance metric among data collection distributions provides a fine property. That is, divergences of normal data collections assemble together, forming a quasi-normal distribution. And those of anomalous data collections lies in the right-hand-side interval on the real number axis. As a result of further observation, the divergence of anomalous data collections also assemble to be a quasi-normal distribution, since statistical distance has been restricted by a strict upper bound.

[Better give a point drawing for each JSD value]

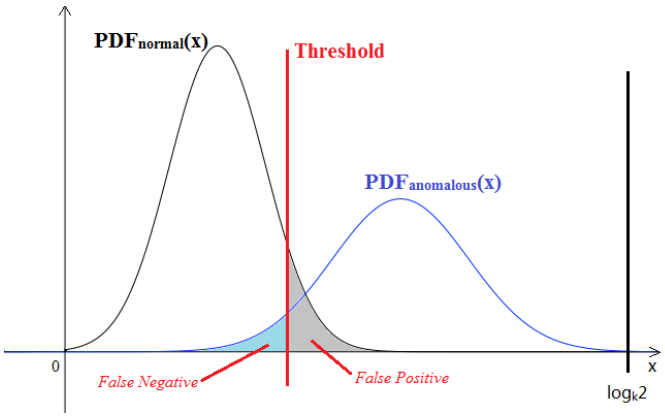

Fig. 3. Threshold Example

As shown in Fig 3, the black curve($PDF_{normal}(x)$) displays the probability density function(PDF) fitting those JSDs calculated from normal data collections; the blue curve($PDF_{anomalous}(x)$) displays the PDF derived from anomalous data collections. Threshold is chosen to minimize total errors(False Negative and False Positive).

Suppose:

$$PDF_{normal}(x) \approx N(\mu_n, \sigma_n) \tag{7}$$
$$PDF_{anomalous}(x) \approx N(\mu_a, \sigma_a) \tag{8}$$

Then the optimal threshold T is:

$$\begin{aligned} T &= \arg\min_T \int_0^T PDF_{anomalous}(x)dx + \int_T^{log_k 2} PDF_{normal}(x)dx \\ &\approx \arg\min_T \int_{-\infty}^T \frac{e^{-\frac{(x-\mu_a)^2}{2\sigma_a^2}}}{\sqrt{2\pi}\sigma_a}dx + \int_T^{+\infty} \frac{e^{-\frac{(x-\mu_n)^2}{2\sigma_n^2}}}{\sqrt{2\pi}\sigma_n}dx \\ &= \frac{\mu_n\sigma_a + \mu_a\sigma_n}{\sigma_a + \sigma_n} \end{aligned} \tag{9}$$

According to equation(9), threshold value can be automatically determined. The optimal threshold will minimize total errors, yielding an optimal outcome. The only requirement for this technique is another evidence set containing data point derived from anomalous data collections.

However, this is not accurate enough, since equation(9) implicate an assumption that the chances are the same for a new data collection to be either anomalous or not. If we can determine the probability for a new data collection to be anomalous in any segment of data sequence, the equation should be modified as:

$$\begin{aligned} T &= \arg\min_T \alpha \int_0^T PDF_{anomalous}(x)dx + (1-\alpha)\int_T^{log_k 2} PDF_{normal}(x)dx \\ &\approx \arg\min_T \alpha \int_{-\infty}^T \frac{e^{-\frac{(x-\mu_a)^2}{2\sigma_a^2}}}{\sqrt{2\pi}\sigma_a}dx + (1-\alpha)\int_T^{+\infty} \frac{e^{-\frac{(x-\mu_n)^2}{2\sigma_n^2}}}{\sqrt{2\pi}\sigma_n}dx \end{aligned} \tag{10}$$

Where α is the anomaly probability. Equation(10) is much more complicated than (9). It can be reduced to a quadratic equation. But may not have real roots in all cases.

From another point of view, equation (9) can be regarded as a weighted averaging between two mean value: μ_n and μ_a. From this aspect, we can derive some similar approximations:

$$T = \frac{\mu_n\sqrt{\sigma_a} + \mu_a\sqrt{\sigma_n}}{\sqrt{\sigma_a} + \sqrt{\sigma_n}} \tag{11}$$
$$T = \frac{\mu_n\sigma_a\sqrt{-ln(1-\alpha)} + \mu_a\sigma_n\sqrt{-ln\alpha}}{\sigma_a\sqrt{-ln(1-\alpha)} + \sigma_n\sqrt{-ln\alpha}} \tag{12}$$
$$T = \frac{\mu_n\sqrt{|ln[\sigma_a(1-\alpha)]|} + \mu_a\sqrt{|ln(\sigma_n\alpha)|}}{\sqrt{|ln[\sigma_a(1-\alpha)]|} + \sqrt{|ln(\sigma_n\alpha)|}} \tag{13}$$

Due to the restriction sample capacity, the replacement of JSD distributions by guassian distribution may lose too much accuracy. Compared with equation (11), (12)and (13), (10) may not give a best result.

D. Dynamic DCAD

In Algorithm 1, there is a hidden assumption that it regards every data collection to be sampled from a static population. That is to say the seller will never change his or her products and customer volume always remain the same. In most cases, real world data is always in the process of evolution and fluctuation. Online shops are often in the process of expanding or dwindling. Thus the sales distribution will slightly change as time goes on. Therefore, the algorithms should automatically adapt to the trend at any time. To meet the need, a sliding window technique has been applied to the basic algorithm frame work. Considering discussion in section IV-B and IV-C, a dynamic version is proposed as follow:

Algorithm 2 uses E_N and E_A as two sliding windows, keeping up with latest trend of both normal and anomalous features. Thus, the two window sizes n and m cannot be too large. Otherwise the evolving features will be flattened and aligned to the older ones. For the sake of estimation accuracy, they cannot be too small either. The size should refer to ratio of both data updates and feature evolution.

Algorithm 2 Dynamic DAD

Input: Normal evidence set $E_N = \{D_{N_1}, \ldots, D_{N_n}\}$ by given order; Anomalous evidence set $E_A = \{D_{A_1}, \ldots, D_{A_m}\}$ by given order; New data collection D'; Estimated anomalous probability α

Output: Whether D' is anomalous

1: **for** $i \leftarrow 1$ to n **do**
2: $P_{N_i} \leftarrow$ histogram of D_{N_i}
3: **end for**
4: **for** $i \leftarrow 1$ to m **do**
5: $P_{A_i} \leftarrow$ histogram of D_{A_i}
6: **end for**
7: $P' \leftarrow$ histogram of D'
8: $M \leftarrow \frac{1}{n}\sum_{i=1}^{n} P_{N_i}$
9: **for** $i \leftarrow 1$ to n **do**
10: $J_{N_i} \leftarrow JSD(P_{N_i}||M)$
11: **end for**
12: **for** $i \leftarrow 1$ to m **do**
13: $J_{A_i} \leftarrow JSD(P_{A_i}||M)$
14: **end for**
15: $J' = JSD(P'|||M)$
16: $(\mu_N, \sigma_N) \leftarrow$ estimated from $\{J_{N_1}, \ldots, J_{N_n}\}$
17: $(\mu_A, \sigma_A) \leftarrow$ estimated from $\{J_{A_1}, \ldots, J_{A_m}\}$
18: $T \leftarrow$ proper threshold derived from (μ_N, σ_N), (μ_A, σ_A) and α
19: **if** $J' < T$ **then**
20: $E_N \leftarrow E_N \backslash \{D_{N_0}\} \cup \{D'\}$
21: **Return** False
22: **else**
23: $E_A \leftarrow E_A \backslash \{D_{A_0}\} \cup \{D'\}$
24: **Return** True
25: **end if**

Line 20 and 23 update the evidence sets, replacing the most outdated one, moving the sliding window forward one step ahead. Moreover, α can also be updated according to a history sequence of return values, if necessary.

V. Evaluation

The algorithm was implemented and interpreted in Python 3.6. All experiments was tested on Ubuntu 17.04. In the following experiments, we would like to figure out:

1) Is the analysis in section II-B and IV applicable and accurate in real world data sets.
2) How to emulate different types of click farming?
3) Is the technique efficient against click farming? And How sensitive will the classifier be confronting different magnitude of click farming?
4) What is the performance of different adaptive thresholds?

A. Methodology

We adopted a data set containing Taobao online sellers' transaction records[1] provided by Alibaba Tian Chi big data competition. The data package contains seller features data set, user payments data set and user browsing behaviour data set. An example of the user payments data set is shown in Table I.

TABLE I
User Payment Record Exmaple

User ID	Seller ID	Payment Time
10523185	1629	2015-11-11 17:00:00
2	2	2
3	3	3
⋮	⋮	⋮

We randomly chose one seller(ID: 1629) and extracted transaction history of this seller, records ranging from Nov. 11th 2015 to Oct. 31st 2016. Entire transaction set was then divided into 325 collections, each containing records in one day.

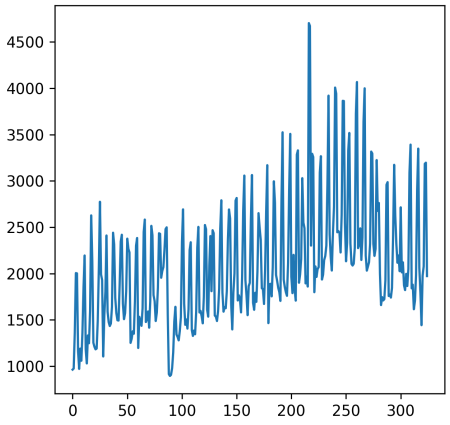

Fig. 4. Daily Transaction Volumes

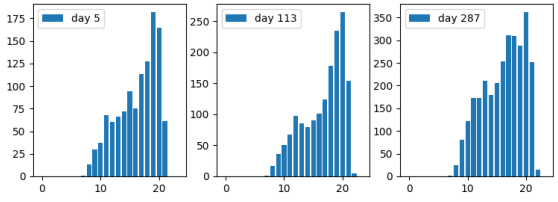

Fig. 5. Randomly selected 3 days and draw histograms by hourly sale

Fig. 4 shows the daily transaction volume of the seller. It sees a trend of daily sale which is slightly moving as the time goes. And the daily selling distributions, as is shown in Fig. 5, display a roughly similar shape.

Click farmed data was generated according to patterns described in online reports**[ref here]**. The most popular way of click farming is to call online a group of people, assign each of them an amount of transaction. And those "employees" individually buy certain products via different buyer accounts.

[1]**https://tianchi.aliyun.com/competition/information.htm?raceId=231591**.

Therefore, a significant feature of this approach is that the cheating transactions usually assemble together in a short period of time. It is called "Centralized Farming". Some other "employers" are more clever. They usually arrange a time table for each employee to create new transactions. Thus the transaction distribution may not vary too much with and without click farming. This is called "Equalized Farming".

To emulate "Centralized Click Farming", we randomly inserted some gaussian-distributed transactions in a chosen collection, where the total number of new records is the same of the original volume. To emulate a extremely clever "Equalized Click Farmer", we simply doubled each record in the chosen collection to make the new distribution exactly the same as the original one, which is harder for the online platform to discover.

To play the role of purchasing platform, we surveillance two levels of transaction distribution. The first level is simply drawing a histogram on time spans. The second level is to draw a histogram on the sub-volumes of each time span. For example, shown in Fig. 6, the first level histogram can be drawn by counting the number of hourly sales. The frequencies of each bucket in 1st level histogram is: $F = \{0, 0, 0, 0, 0, 0, 0, 1, 13, 30, 37, 68, 60, 66, 72, 94, 75, 113, 127, 182, 165, 61, 0, 0\}$. Then, counting F with a step size 20 gives the 2nd level histogram.

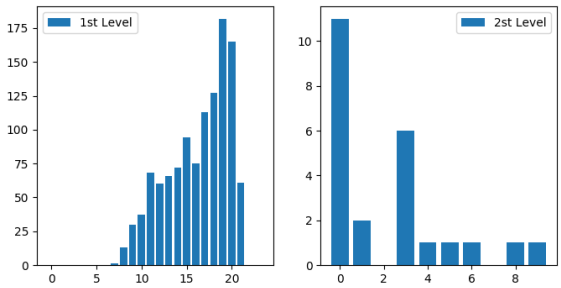

Fig. 6. Example of 1st and 2nd Level Histogram

In order to test the performance of Algorithm 2, every single collection was fed into the algorithm by the time order. Each collection input has a probability of α to be anomalous, namely, click farmed. The algorithm checks each collection with 1st or 2nd level histogram and give an answer of true and false. Then a F1 score was computed as the result of the experiment.

B. Experiment on Raw Data

We first tested Algorithm 2 on raw data set in order to see whether and why the algorithm works. Due to desensitization process, time stamps in raw data set only tell which hour the transaction was committed. Thus, the histogram was drawn by counting hourly volume of each collection. For $\alpha = 0.2$, we select first 30 days as the evidence set of normal collections; day 21st to 30th were also emulated to be click farmed, composing the evidence set of anomalous collections. Equation(12) was employed as the threshold calculator. The results are shown in Table II, where *true positive rate*, *false positive rate* and *accuracy* were recorded.

TABLE II
CLASSIFICATION RESULTS ON RAW DATA

	Centralized			Equalized		
	TPR	FAR	ACC	TPR	FAR	ACC
1st Level	100	4.05	96.72	26.64	28.03	62.49
2nd Level	88.01	7.68	91.41	83.89	15.78	84.07

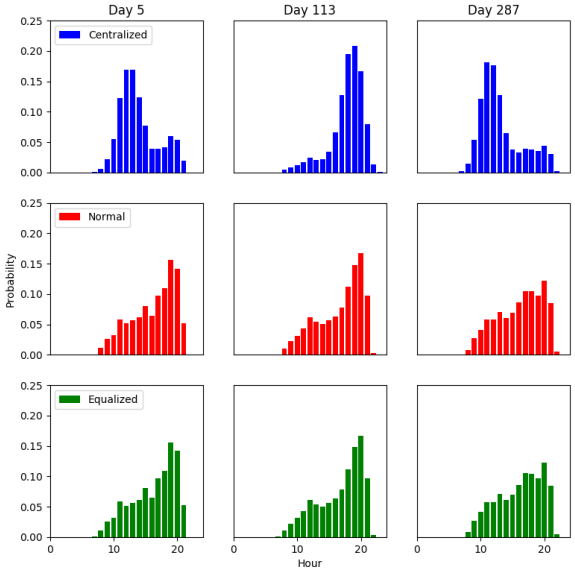

Fig. 7. 1st Level Histogram of Raw Data

When classifying toward 1st level histograms, centralized click farming behaviours can be easily discovered. As displayed in the left two columns in Fig. 7, normal collections share a similar distribution while centralized click farmed ones abruptly violates the original shape. However, as a clever click farmer, equalized click farming did not in the least distorted the distribution. Most of them escaped the security check under the perfect disguise. But when it comes to 2nd level histogram, the "clever disguise" does not work any longer. It can be clearly seen in Fig. 8 that both types of click farming shows an obvious difference from the normal ones, while the normal ones still share a similar distribution. From the overview in Fig. 9 and 10, the difference between normal and anomalous collections can be observed more intuitively.

C. Experiment on Synthetic Data

As is put in the former section, every single record in the transaction set only gives a time stamp aligned at hours. Part of the information was erased with the absence of minutes and seconds. In the following experiments, Every time stamp was assigned a random value for minutes and seconds.

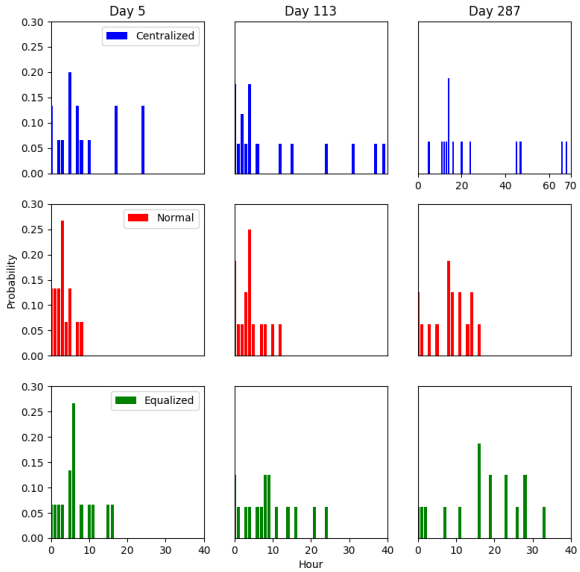

Fig. 8. 2nd Level Histogram of Raw Data

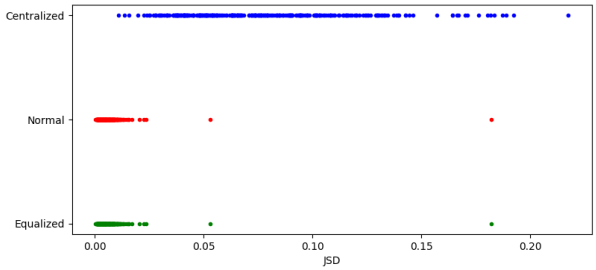

Fig. 9. JSD of 1st Level Histograms

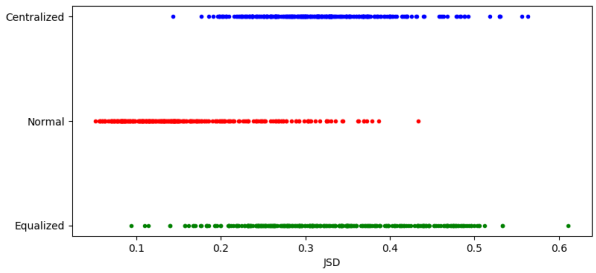

Fig. 10. JSD of 2nd Level Histogram

Therefore, the synthetic data set should be closer to the real world case. Moreover, the step size in the histograms can now be calculated more precisely. In the 1st level histogram, the constant c in equation(6) was set to 0.5 since the PDF of daily distribution was approximately a linear function. And in the 2nd level histogram, sub-volumes of every 300 seconds were counted, producing more data points in the 2nd level data collections. Then, constant c in equation(6) was also set to 0.5 given that the 2nd level histogram was also approximately a linear function. We tested algorithm performance for $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, with E_N and E_A the same configuration as before. Results were shown in Table III and IV.

TABLE III
1ST LEVEL

	Centralized			Equalized		
α	TPR(%)	FPR(%)	ACC(%)	TPR(%)	FPR(%)	ACC(%)
0.1	86.10	1.14	97.40	5.75	12.28	79.66
0.2	96.43	9.83	91.30	26.97	26.47	64.29
0.3	97.95	16.49	87.68	37.46	39.46	53.67
0.4	99.18	31.94	81.30	57.23	48.32	53.79
0.5	99.33	23.40	88.25	67.67	58.23	54.46
0.6	99.62	25.70	89.83	67.03	73.78	51.64
0.7	100.00	32.51	90.51	69.69	76.28	56.16
0.8	99.72	26.97	94.35	75.56	69.44	67.00
0.9	99.63	15.73	98.19	76.59	75.18	70.73

TABLE IV
2ND LEVEL

	Centralized			Equalized		
α	TPR(%)	FPR(%)	ACC(%)	TPR(%)	FPR(%)	ACC(%)
0.1	75.60	3.90	93.74	97.84	1.64	98..31
0.2	91.93	18.43	93.73	99.42	7.39	94.01
0.3	95.01	23.70	81.69	100.00	23.93	83.05
0.4	98.09	28.66	82.49	100.00	24.59	84.75
0.5	98.33	36.54	80.00	100.00	23.76	88.36
0.6	98.35	40.69	83.41	100.00	35.94	84.63
0.7	98.54	37.97	87.80	100.00	31.27	90.84
0.8	98.47	40.38	91.19	100.00	33.04	92.66
0.9	98.51	47.56	94.58	100.00	28.44	97.40

The result shows that the algorithm can efficiently distinguish the normal and anomalous data collections. No matter how much proportion of data is anomalous, the algorithm always gives a high accuracy. That is because the adaptive threshold was designed to minimize the number of total errors, which was irrelevant to TPR or FPR. When α was increasing, there were less data collections added into E_N. Thus the update of E_N became slower and FPR went higher. On the contrary, TPR became higher as α increased. Moreover, since $|E_N| > |E_A|$, it needs more collection instances for $|E_N|$ then that for $|E_A|$ to keep up to the trend. Therefore, it was faster of FPR to increase than TPR to increase. As a result of that, there were more false alarms emerged and ACC went lower, as α increased from 0.1 to 0.5. After that, however, although FPR was still increasing, total number of normal instances decreased dramatically and those false alarms contributed less to total errors. Therefore, ACC rose up again.

Fig. 11 shows the ROC curve($\nu = 1.0$) of the gaussian classifier dealing with centralized click farming with 1st level histograms and equalized manipulation with 2nd level histograms. The curves are pretty close to the upper left corner of the diagram and size under the curves are very large. For different αs, the classifier can efficiently distinguish normal and anomalous data collections.

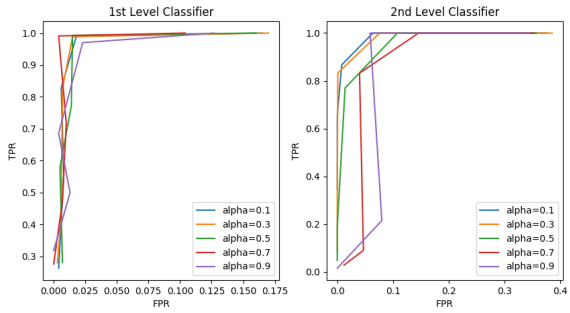

Fig. 11. ROC Curve of 1st and 2nd Level Classifier

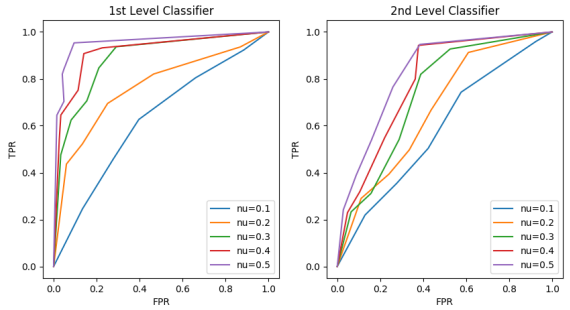

Fig. 12. ROC Curve of Different Magnitude of Click Farming

In Fig. 12, we tested the classifier with different magnitude of centralised click farming. In former experiments, the manipulation of centralized click farming always creates as many new transaction records as the original ones. Let ν denote the magnitude coefficient, then $|D_{anomalous}| = (1+\nu)|D_{normal}|$. If ν is smaller, then it will be harder for the classifier to identify anomalous data collections since the anomalous distribution will be closer to normal ones. It can be concluded from the figure that the 1st level classifier can be efficient even when $\nu = 0.3$. However the 2nd level classifier is less sensitive when $\nu = 0.5$. That is because the 1st level histogram provides more probability entries than 2nd level histograms. Data collections are compressed from 1st to 2nd level. But for real world click farming behaviours(usually, $\nu > 1$) both classifiers can be efficient enough.

D. Threshold Optimization

Section IV-C mentions only four equations for calculating adaptive thresholds. Table V gives the comparison of more equations in similar forms. We tested those calculators on synthetic data set with $\alpha = 0.1, \nu = 1$. Testing each level of classifier and each type of anomalies, we got 4 F1 scores and added them up. Equation 9 gives best performance. Then those equations were tested on large amount of gaussian distributed data instances. 90K and 10K data points were sampled from 500 pairs of random gaussian distributions which represent normal and anomalous distributions respectively, yielding 500 F1 scores for each equation. We compared the average score and discovered that equation 13 gives the best result.

Equation 13 gives best classification result for two gaussian distributions but does not outperform others in real world data. Part of the reasons are mentioned in section IV-C. Another reason is that the size of evidence sets are too restricted, that the gaussian distribution may not be well approximated.

E. Comparison with other collective classification algorithms

It would be better to have this comparison, but it requires much more experiments and the data set may not be very suitable.

VI. Conclusion & Future Work

This paper proposed an dynamic collective anomaly detection algorithm with adaptive threshold, which helps detect data manipulations in modern data pipelines and data centres. Different from existing algorithms detecting collective anomalies, our approach employs statistical distance as the similarity metric, mapping data collections to a restricted one dimensional space. We explained several technical points involved in the design of the algorithm and performed a thorough experiment to test its efficiency. It showed that the algorithm can efficiently discover anomalies within the data sequences and the classifier is sensitive enough toward real world data manipulations.

However, the modelling of the JSD distributions is not precise enough, resulting in several threshold equations unexplained. The efficiency of different adaptive thresholds should be tested on different real world circumstances to get a more complete view of their performances. Moreover, a comparison with other collective detection algorithm should be executed in order to get an performance overview of similar algorithms.

Acknowledgement

References

[1] 2015. URL: http://www.ebrun.com/20150906/147724.shtml (visited on 09/06/2015).

[2] 2017. URL: https://raygun.com/blog/cost-of-software-errors/ (visited on 03/22/2017).

[3] *Big Data Statistics And Facts for 2017*. 2017. URL: https://www.waterfordtechnologies.com/big-data-interesting-facts/.

[4] Nathan Bronson, Thomas Lento, and Janet L Wiener. "Open data challenges at facebook". In: *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE. 2015, pp. 1516–1519.

[5] Suratna Budalakoti et al. "Anomaly detection in large sets of high-dimensional symbol sequences". In: (2006).

[6] Lei Cao et al. "Multi-Tactic Distance-Based Outlier Detection". In: *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE. 2017, pp. 959–970.

[7] Lei Cao et al. "Scalable distance-based outlier detection over high-volume data streams". In: *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE. 2014, pp. 76–87.

TABLE V
F1 SCORES OF DIFFERENT THRESHOLD CALCULATORS

	Equation	Synthetic	Gaussian
1	$T = \frac{\sigma_B \mu_A + \sigma_A \mu_B}{\sigma_B + \sigma_A}$	1.0588	0.5810
2	$T = \frac{\sqrt{\sigma_B}\mu_A + \sqrt{\sigma_A}\mu_B}{\sqrt{\sigma_B} + \sqrt{\sigma_A}}$	2.1861	0.5718
3	$T = \frac{(ln\sigma_B)\mu_A + (ln\sigma_A)\mu_B}{ln\sigma_B + ln\sigma_A}$	2.2390	0.3860
4	$T = \frac{(\sqrt{\lvert ln\sigma_B \rvert})\mu_A + (\sqrt{\lvert ln\sigma_A \rvert})\mu_B}{\sqrt{\lvert ln\sigma_B \rvert} + \sqrt{\lvert ln\sigma_A \rvert}}$	2.2082	0.5515
5	$T = \frac{(ln\sqrt{\sigma_B})\mu_A + (ln\sqrt{\sigma_A})\mu_B}{ln\sqrt{\sigma_B} + ln\sqrt{\sigma_A}}$	2.1211	0.3811
6	$T = \frac{\sigma_B(1-\alpha)\mu_A + \sigma_A\alpha\mu_B}{\sigma_B(1-\alpha) + \sigma_A\alpha}$	0.7688	0.3515
7	$T = \frac{(\sigma_B\sqrt{1-\alpha})\mu_A + (\sigma_A\sqrt{\alpha})\mu_B}{\sigma_B\sqrt{1-\alpha} + \sigma_A\sqrt{\alpha}}$	0.8158	0.4510
8	$T = \frac{[\sigma_B ln(1-\alpha)]\mu_A + [\sigma_A ln\alpha]\mu_B}{\sigma_B ln(1-\alpha) + \sigma_A ln\alpha}$	1.5227	0.5523
9	$T = \frac{\mu_A\sigma_B\sqrt{-ln(1-\alpha)} + \mu_B\sigma_A\sqrt{-ln\alpha}}{\sigma_B\sqrt{-ln(1-\alpha)} + \sigma_A\sqrt{-ln\alpha}}$	**2.6013**	0.5791
10	$T = \frac{\mu_A\sigma_B ln\sqrt{1-\alpha} + \mu_B\sigma_A ln\sqrt{\alpha}}{\sigma_B ln\sqrt{1-\alpha} + \sigma_A ln\sqrt{\alpha}}$	1.4155	0.5578
11	$T = \frac{\mu_A\sqrt{\lvert ln[\sigma_B(1-\alpha)] \rvert} + \mu_B\sqrt{\lvert ln(\sigma_A\alpha) \rvert}}{\sqrt{\lvert ln[\sigma_B(1-\alpha)] \rvert} + \sqrt{\lvert ln(\sigma_A\alpha) \rvert}}$	2.2730	0.5600
12	$T = \frac{\mu_A ln\sqrt{\sigma_B(1-\alpha)} + \mu_B ln\sqrt{\sigma_A\alpha}}{ln\sqrt{\sigma_B(1-\alpha)} + ln\sqrt{\sigma_A\alpha}}$	2.0753	0.2808
13	$T = \underset{\mu_A \le T \le \mu_B}{\arg\min}\ \alpha \cdot \int_{-\infty}^{T} P_B(x)dx + (1-\alpha) \cdot \int_{T}^{+\infty} P_A(x)dx$	1.9300	**0.6072**

[8] T Caudell and D Newman. "An adaptive resonance architecture to define normality and detect novelties in time series and databases". In: *IEEE World Congress on Neural Networks, Portland, Oregon*. 1993, pp. 166–176.

[9] Philip K Chan and Matthew V Mahoney. "Modeling multiple time series for anomaly detection". In: *Data Mining, Fifth IEEE International Conference on*. IEEE. 2005, 8–pp.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), p. 15.

[11] Guoqiang Jerry Chen et al. "Realtime data processing at facebook". In: *Proceedings of the 2016 International Conference on Management of Data*. ACM. 2016, pp. 1087–1098.

[12] Jason C Cohen and Subrata Acharya. "Towards a trusted HDFS storage platform: Mitigating threats to Hadoop infrastructures using hardware-accelerated encryption with TPM-rooted key protection". In: *Journal of Information Security and Applications* 19.3 (2014), pp. 224–244.

[13] Peter Groves et al. "The'big data'revolution in healthcare: Accelerating value and innovation". In: (2016).

[14] Herodotos Herodotou et al. "Scalable near real-time failure localization of data center networks". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 1689–1698.

[15] Jingwei Huang, David M Nicol, and Roy H Campbell. "Denial-of-service threat to Hadoop/YARN clusters with multi-tenancy". In: *Big Data (BigData Congress), 2014 IEEE International Congress on*. IEEE. 2014, pp. 48–55.

[16] Iresearch. 2016. URL: http://report.iresearch.cn/content/2016/11/265551.shtml (visited on 11/25/2016).

[17] *Is Data Manipulation the Next Step in Cyber Crime*. URL: https://www.cloudmask.com/blog/is-data-manipulation-the-next-step-in-cybercrime.

[18] Masoumeh Rezaei Jam et al. "A survey on security of Hadoop". In: *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*. IEEE. 2014, pp. 716–721.

[19] Gang-Hoon Kim, Silvana Trimi, and Ji-Hyong Chung. "Big-data applications in the government sector". In: *Communications of the ACM* 57.3 (2014), pp. 78–85.

[20] Yaliang Li et al. "On the discovery of evolving truth". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2015, pp. 675–684.

[21] Mohammad M Masud et al. "Classification and adaptive novel class detection of feature-evolving data streams". In: *IEEE Transactions on Knowledge and Data Engineering* 25.7 (2013), pp. 1484–1497.

[22] The Legal Mirror. 2016. URL: http://tech.163.com/15/0405/14/AMENOJ7D00094ODV.html (visited on 11/25/2016).

[23] Frank Austin Nothaft et al. “Rethinking data-intensive science using scalable analytics systems”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM. 2015, pp. 631–646.

[24] Dmitry Pavlov. “Sequence modeling with mixtures of conditional maximum entropy distributions”. In: *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE. 2003, pp. 251–258.

[25] Kanyakumari Pawar et al. “Software Data Reduction Technique for Bug Reduction Problem”. In: *International Journal* 4.4 (2016).

[26] Murad A Rassam, Mohd Aizaini Maarof, and Anazida Zainal. “Adaptive and online data anomaly detection for wireless sensor systems”. In: *Knowledge-Based Systems* 60 (2014), pp. 44–57.

[27] Barna Saha and Divesh Srivastava. “Data quality: The other face of big data”. In: *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE. 2014, pp. 1294–1297.

[28] Felix Schuster et al. “VC3: Trustworthy data analytics in the cloud using SGX”. In: *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE. 2015, pp. 38–54.

[29] Junming Shao, Zahra Ahmadi, and Stefan Kramer. “Prototype-based learning on concept-drifting data streams”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 412–421.

[30] Ather Sharif et al. “Current security threats and prevention measures relating to cloud services, Hadoop concurrent processing, and big data”. In: *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1865–1870.

[31] Priya P Sharma and Chandrakant P Navdeti. “Securing big data hadoop: a review of security issues, threats and solution”. In: *Int. J. Comput. Sci. Inf. Technol* 5.2 (2014), pp. 2126–2131.

[32] Biplab Sikdar. “Spatio-Temporal Correlations in Cyber-Physical Systems: A Defense Against Data Availability Attacks”. In: *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM. 2017, pp. 103–110.

[33] Roshan Sumbaly, Jay Kreps, and Sam Shah. “The big data ecosystem at linkedin”. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM. 2013, pp. 1125–1134.

[34] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. “A survey on security and privacy issues in big data”. In: *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*. IEEE. 2015, pp. 202–207.

[35] TowerData. *4 Steps to Eliminating Human Error in Big Data*. 2013. URL: http://www.towerdata.com/blog/bid/113787/4-Steps-to-Eliminating-Human-Error-in-Big-Data (visited on 06/30/2017).

[36] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. “Detecting intrusions using system calls: Alternative data models”. In: *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*. IEEE. 1999, pp. 133–145.

[37] Tianyin Xu and Yuanyuan Zhou. “Systems approaches to tackling configuration errors: A survey”. In: *ACM Computing Surveys (CSUR)* 47.4 (2015), p. 70.

[38] Zhang Xu et al. “High fidelity data reduction for big data security dependency analyses”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2016, pp. 504–516.

[39] Nong Ye et al. “A markov chain model of temporal behavior for anomaly detection”. In: *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*. Vol. 166. West Point, NY. 2000, p. 169.

[40] Jianhua Yin and Jianyong Wang. “A model-based approach for text clustering with outlier detection”. In: *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE. 2016, pp. 625–636.

[41] Rose Yu, Xinran He, and Yan Liu. “Glad: group anomaly detection in social media analysis”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.2 (2015), p. 18.

[42] Xianqing Yu, Peng Ning, and Mladen A Vouk. “Enhancing security of Hadoop in a public cloud”. In: *Information and Communication Systems (ICICS), 2015 6th International Conference on*. IEEE. 2015, pp. 38–43.

[43] Xuyun Zhang et al. “LSHiForest: A Generic Framework for Fast Tree Isolation Based Ensemble Anomaly Analysis”. In: *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE. 2017, pp. 983–994.

[44] Zhiyuan Zheng and AL Reddy. “Towards Improving Data Validity of Cyber-Physical Systems through Path Redundancy”. In: *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*. ACM. 2017, pp. 91–102.

[45] Yunyue Zhu and Dennis Shasha. “Statstream: Statistical monitoring of thousands of data streams in real time”. In: *Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment. 2002, pp. 358–369.