A Review on Online Anomaly Detection Techniques

S. S. Raut[1], D. S. Randhave[2], K. V. Sonkamble[3], S. N. Deshmukh[4]

[1,2,3,4] *Department of Computer Science and IT, Dr. B. A. M. University, Aurangabad - 431004.*

Abstract

Anomaly detection has been a crucial analysis topic in data processing and machine learning. Several real-world applications like intrusion or MasterCard fraud detection need a good and efficient framework to spot deviated data instances. A good anomaly detection methodology must to be able to accurately establish many varieties of anomalies, be robust, need comparatively very little resources, and perform detection in period of time. This paper provides an outline of major technological perspective and appreciation of the basic progress of online anomaly detection and conjointly provides overview technique developed in every stage of online anomaly detection. This paper helps in selecting the technique together with their relative deserve & demerits.

1. Introduction

Anomaly (or outlier) detection aims to spot a little group of instances that deviate remarkably from the existing data. A renowned definition of "outlier" in [1]: "An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism," which gives the general idea of an outlier. The importance of anomaly detection is as a result of the actual fact that anomalies in information translate to vital, and infrequently crucial, unfair data in a very wide range of application domains. Basically, anomaly detection may be found in applications like Office of Homeland Security, MasterCard fraud detection, intrusion and business executive threat detection in cyber-security, fault detection, or malignant diagnosing. However, since exclusively a restricted quantity of labelled data accessible within the higher than real-world applications, a way to confirm anomaly of unseen data (or events) attracts attention from the researchers in data processing and machine learning communities.

1.1 Types of Anomaly

1.1.1 Point Anomalies

If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed a point anomaly. This is the simplest type of anomaly and is the focus of majority of research on anomaly detection.

1.1.2 Contextual Anomalies

If a data instance is abnormal in a very specific context, but not otherwise, then it's termed a contextual anomaly (also brought up as conditional anomaly [2]). The notion of a context is included by the structure within the data set and should be specified as a neighborhood of the problem formulation.

1.1.3 Collective Anomalies

If a group of connected data instances is abnormal with respect to the whole data set, it's termed a collective anomaly. The individual knowledge instances in an exceedingly collective anomaly might not be anomalies by themselves; however their occurrence together as a group is abnormal.

The techniques used for detecting collective anomalies are very different than the point and contextual anomaly detection techniques.

2. Anomaly Detection Techniques

2.1 Unsupervised Anomaly Detection

Techniques that operate in unsupervised mode do not need training data, and therefore most generally applicable. The techniques in this class create the implicit assumption that standard instances area unit much more frequent than anomalies within the test data. If this assumption isn't true then such techniques suffer from high warning rate. Many semi-supervised techniques may be custom-made to control in Associate in unsupervised mode by employing a sample of the

unlabeled data set as test data. Such adaptation assumes that the test data contains only a few anomalies and therefore the model learned throughout training is robust to those few anomalies.

2.2 Supervised Anomaly Detection

Techniques trained in supervised mode assume the availability of training data set that has labeled instances for traditional still as anomaly categories. A typical approach in such cases is to make a prophetical model for traditional vs. anomaly categories. Any unseen information instance is compared against the model to work out that category it belongs to. There are two major problems that arise in supervised anomaly detection. First, the abnormal instances are much fewer compared to the conventional instances within the training data. Problems that arise owing to unbalanced category distributions are addressed within the data processing and machine learning literature[3][4]. Second, getting correct and representative labels, especially for the anomaly category is typically difficult. Numbers of techniques are planned that inject artificial anomalies into a traditional data set to get a labeled training data set [5]. Away from these two issues, the supervised anomaly detection disadvantage is comparable to assembling prophetical models.

2.3 Semi- Supervised Anomaly Detection

Techniques that operate in a semi-supervised mode, assume that the training data set has labeled instances just for the traditional category. Since they are doing not need labels for the anomaly category, they're additional wide applicable than supervised techniques. As an example, in spacecraft capsule fault detection [6], Associate in anomaly situation would signify an accident, that isn't simple to model. The typical approach utilized in such techniques is to create a model for the category corresponding to traditional behavior, and use the model to spot anomalies within the take a look at test data. A restricted set of anomaly detection techniques exists that assumes availableness of only the anomaly instances for training [7]. Such techniques are not usually used; primarily as a result of it is troublesome to get a training data set that covers each potential abnormal behavior that can occur within the data.

3. Online Anomaly Detection Techniques

3.1 SCAN Statistics

SCAN statistics is a robust methodology for detecting remarkably high rates of events, conjointly referred to as anomalies. Scanning for bursts of events has several applications in numerous fields such as telecommunications, medicine, biological science, astronomy, internal control, and dependability [8]. In monitoring and management of communication networks, SCAN statistics will be used to monitor the prevalence of events in time, a degree method, like standing messages, alarms, and faults. It did not have a tendency to look for outliers, however rather uncommon bursts in events. In an online application for events occurring in time, the amount of events that have occurred in the scanning window $[t - w, t]$, where t is that the current time and w is that the scanning window size, are compared with the amount of events expected to possess occurred in that window under traditional conditions. If that variety of events is massive compared to what expected, then an alert of an abnormality will be given. SCAN statistics will be used to compute the distribution of events under traditional conditions (the null hypothesis, H_0) to see what is a significantly sizable amount (the vital value) within the scanning window, whereas properly dominant the false positive rate (FPR), that is that the chance of exceptional the critical value for any scanning window of size w within the larger time interval $[0; T)$ under H_0. A key advantage of scan statistics is that it permits for computationally easy implementation; therefore, it is possible to observe several processes at once with a tiny low process burden. In the usual treatment of scan statistics the time of events occurring within the interval $[0; T)$ are assumed to be generated by a Poisson method under H_0. In a Poisson process with rate λ over $[0; T)$, the number of events is given by Poisson (λT). The inter-arrival times are iid distributed according to Exponential λ. Conditional on there being k events, the times of the events are distributed uniformly in $[0; T)$. Assuming that λ is known, the scan statistic is defined as follows. Let $X_1, X_2, \ldots, X_N$ denote the ordered values of the events occurring in the interval $[0; T)$ and let $Y_t(w)$ be the number of points $(X's)$ in the interval $[t - w; t]$ (Extensions of scan statistics exist in discrete time and on circular data, such as time of year, but we do not focus on them here.). The scan statistic S_w is then defined as the maximum number of points to be found in any subinterval of $[0; T)$ of length w [9]. That is,

$$S_w = \max_{w \le t \le T} Y_t(w) = \max_t \sum_{i=1}^{N} ||\{t - w \le X_i \le t\}. \quad (1)$$

A related statistic is W_k, the minimum subinterval of $[0; T)$ containing k points

$$W_k = \min_{0 \le w \le T} \{w : S_w \ge k\}$$

$$= \min_{1 \le i} (X_{i+k-1-} - X_i). \quad (2)$$

The distributions of these statistics are related by $P(S_w \geq k) = P(W_k \leq w)$. Equivalently, S_w and W_k are inverses: $S_{wk} = k$ for $k \leq N$. One should also note the edge cases of S_w and W_k: $W_k = 1$ for $k > N$, $W_1 = 0$ and $S_0 = 1$ for $N \geq 1$, and $S_T = N$. The key trick in scan statistics is controlling the FPR by accounting for the overlapping multiple comparisons that are a result of the rolling scan window while maintaining more power than simple Bonferroni correction.

For a Poisson process with mean rate λ per unit time over the interval $[0; T)$, [10] gives the following approximation for the distribution $P(S_w \geq k \mid \mu, L)$, where $\mu = \lambda w$ and $L = T \mid w$ (also equal to $P(W_k \leq w \mid \mu, L)$). Let $p(k; \mu)$ be the probability of exactly k events occurring for a Poisson distribution with mean μ and $F(k; \mu)$ the cumulative distribution function (CDF) for the Poisson, then

$$P(S_w \geq k|\mu, L)1 - Q_2(Q_3/Q_2)^{L-2}. \quad (3)$$

$$Q_2 = F(k-1,\mu)^2 - (K-1)p(k;\mu)p(k-2;\mu) - (k-1-\mu)p(k;\mu)p(k - 3;\mu), \quad (4)$$

$$Q_3 = F(k-1,\mu)^3 - A_1 + A_2 + A_3 - A_4, \quad (5)$$

Where,

$$A_1 = 2p(k;\mu)F(k-1;\mu) \times [(k-1)F(k-2;\mu) - \mu F(k-3;\mu)],$$

$$A_2 = 0.5p(k;\mu)^2[(k-1)(k-2)F(k-3;\mu) - 2(k-2)\mu F(k-4;\mu) + \mu^2 F(k-5;\mu)],$$

$$A_3 = \sum_{i=1}^{k-1} p(2k-i;\mu)F(i-1;\mu)^2,$$

$$A_4 = \sum_{i=2}^{k-1} p(2k-i;\mu)p(i;\mu) \times [(i-1)F(i-2;\mu)] - \mu F(i-3;\mu)$$

To test the null hypothesis, H_0, that the background rate $\lambda = \lambda_0 = constant$ at the significance level α, find the smallest k, which we call k_{Crit}, such that

$$P(S_w \geq k_{Crit} | \mu_0, L) \leq \alpha. \quad (6)$$

Where $\mu_0 = \lambda_0 w$. For an online test with fixed w, if at time t (the current time) the number of points, k, occurring in the time interval of length w ending at t, $[t - w; t]$, is $\geq k_{Crit}$, then the null hypothesis is rejected at significance level α and an alert may be given indicating that the rate of events has likely increased. An equivalent alternative test is to determine the length of time separating the most recent k_{Crit} points, $W_{Crit} = X_i - X_i - k_{Crit} + 1$, where $X_i = t$ [11]. If $W_{Crit} \leq w$ then an alert may be given.

3.2 Incremental Local Outlier Factor

The incremental LOF technique provides equivalent detection performance as the iterated static LOF technique (applied when insertion of every data record), whereas requiring considerably less machine time. Additionally, the incremental LOF technique conjointly dynamically updates the profiles of data points. This is often a really necessary property, since data profiles might amendment over time. Incremental LOF algorithm is computationally efficient, whereas at a similar time successfully detect outliers and changes of distribution behavior in numerous data stream applications. The Local Outlier Factor algorithm [11] has been successfully applied in many domains for outlier detection in a batch mode [12, 13]. Incremental LOF algorithm is applicable for detection of outliers in data streams and this technique is the first incremental outlier detection algorithm. It provides equivalent detection performance as the static LOF technique, and has $O(n \log n)$ time complexity, where n is that the total range of data points.

In the designing of incremental LOF algorithm, two things are focused. First, the results of the incremental algorithm should be such as the results of the "static" algorithm every time t a replacement purpose is inserted into a data set. Thus, there would not be a distinction between applying progressive LOF and also the "periodic" static LOF once all data records up to time instant t are available. Second, asymptotic time complexity of incremental LOF algorithm should be like the static LOF algorithm. In order to have feasible incremental algorithm, it is essential that, at any time moment t, insertion/deletion of the data record leads to restricted (preferably small) variety of updates of

algorithm parameters. Specifically, the amount of updates per every insertion/deletion should not be captivated with this number of records within the dataset; otherwise, the performance of the incremental LOF algorithm would be $\Omega(n^2)$ wherever n is that the final size of the dataset.
Incremental LOF algorithm computes LOF value for every data record inserted into data set and instantly determines whether or not inserted data record is outlier. Additionally, LOF values for existing data records are updated if required. And hence work in two phases that are insertion and deletion.

3.2.1 Insertion

In the insertion half, the algorithm performs two steps: a) insertion of latest record, once it computes reach-dist, *lrd* and LOF values of a replacement point; b) maintenance, once it updates k-distances, reach-dist, *lrd* and LOF values for affected existing points. The evaluation of the insertion half is done on the basis of following Theorems1-4[14].

Theorem 1. The insertion of point p_c affects the $k-distance$ at points p_j that have point p_c in their $k-nearest\ neighborhood$, i.e., where $p_j \in kRNN(p_c)$. New $k-distances$ of the affected point's p_j are updated as follows:

$$k-distance^{(new)}(p_j) = \begin{cases} d(p_j,p_c), & p_c\ is\ the\ k-th\ nearest neighbor of\ p_j \\ (k-1)distance^{(old)}(p_j), & otherwise. \end{cases} \quad (7)$$

Proof. In insertion, $k-distance$ of an existing point p_j changes when a new point enters the $k-th$ nearest neighborhood of p_j, since in this case the $k-neighborhood$ of p_j changes. If a new point p_c is the new $k-th$ nearest neighbor of p_j, its distance from p_j becomes the new $k-distance\ (p_j)$. Otherwise, old $k-1th$ neighbor of p_j becomes the new $k-th$ nearest neighbor of p_j (see Fig. 1).

Corollary 1. During insertion, $k-distance$ cannot increase, i.e., $k-distance^{(new)}(p_j) \leq k-distance^{(old)}(p_j)$.

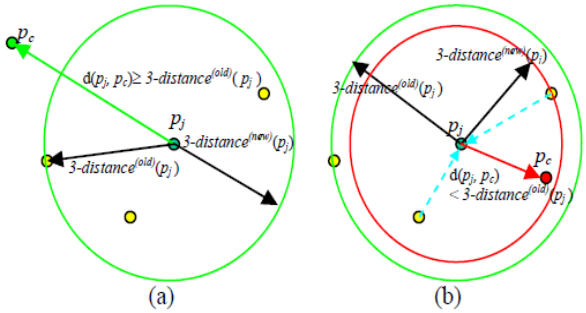

Fig. 1. Update of k-nearest neighbor distance upon insertion of a new record (k=3). a) New record P_c is not among $3-nearest\ neighbors$ of record $p_j \Rightarrow 3-distance\ (p_j)$ does not change; b) new record P_c *is* among $3-nearest\ neighbors$ of $p_j \Rightarrow 3-distance\ (p_j)$ decreases. Cyan dashed lines denote updates of reachability distances between point p_j and two old points.

Theorem 2. Change of $k-distance\ (p_j)$ may affect $reach-distk\ (p_i,p_j)$ for points p_i that are k-neighbours of p_j.

Proof.
Using $reach-dist_k(q,p) = \max\{d(q,p), k-distance(p)\}$, where $d(q,\ p)$ is Euclidean distance from q to p. $\forall\ p_i\ d\ (p_i,p_j) > k-distanc^{(old)}(p_j), \Rightarrow reach-dist^{(old)}k\ (p_i,p_j) = d(p_i,p_j)$. According to *Corollary* 1, $k-distance(p_j)$ cannot increase, hence if $d\ (p_i,p_j) > k-distanc^{(old)}(p_j), reach-dist_k^{(new)}\ (p_i,p_j) = reach-dist_k^{(old)}\ (p_i,p_j)$.

Theorem 3. *lrd* value needs to be updated for every record (denoted with p_m in [14] a general framework) for which its $k-neighborhood$ changes or for which reachability distance to one of its kNN changes. Hence, after each update of $reach-dist_k(p_i,p_j)$ we have to update $lrd(p_i)$ if p_j is among $kNN(p_j)$. Also, *lrd* is updated for all points p_j whose $k-distance$ was updated.

Proof . Change of $k-neighborhood$ of p_m affects the scope of the sum in Eq. (8) computed for all $k-neighbors$ of p_m. Change of the reachability distance between p_m and some of its $k-nearest\ neighbors$ affects corresponding term in the denominator of Eq. (8).

$$lrd(q) = \frac{1}{\sum_{p\in kNN(q)} reach-dist_k(q,p)/k}. \quad (8)$$

Theorem 4. *LOF* value needs to be updated for all data records p_m which *lrd* has been updated (since $lrd(p_m)$ is a denominator in Eq. (9)) and for those records that have records p_m in their $kNNs$. Hence, the set of data records where *LOF* needs to be updated (according to (9)) corresponds to union of records p_m and their $kRNN$.

Proof . Similar to the proof of *Theorem 3*, using (9).

$$LOF(q) = \frac{\frac{1}{k}\sum_{p \epsilon kNN} lrd(p)}{lrd(q)}. \qquad (9)$$

3.2.2 *Deletion*

The general framework for deleting the block of data records S_{delete} from the dataset S is given in [14]. The deletion of each record $p_c \in S_{delete}$ from dataset S influences the $k-distances$ of its $kRNN$. $k-neighborhood$ increases for each data record p_j that is in reverse $k-nearest\ neighborhood$ of p_c. For such records, $k-distance(p_j)$ becomes equal to the distance from p_j to its new $k-th\ nearest\ neighbor$.

The reachability distances from p_j's $(k-1)$ nearest neighbors p_i to p_j need to be updated. Observe that the reachability distance from the $k-th$ neighbor of p_j to record p_j is already equal to their Euclidean distance $d(p_i, p_j)$ and does not need to be updated (Fig. 2). Analog to insertion, *lrd* value needs to be updated for all points p_j where $k-distance$ is updated. In addition, *lrd* value needs to be updated for points p_i such that p_i is in kNN of p_j and p_j is in kNN of p_i. Finally, *LOF* value is updated for all points p_m on which *lrd* value is updated as well as on their $kRNN$. The correctness of the deletion algorithm can be proven analog to the correctness of the insertion algorithm.

3.3 Kernel Estimation Based Anomaly Detection Technique

Large backbone networks are regularly affected by a range of anomalies. This technique gives an online anomaly detection based on mathematical technique of Kernel Density Estimates (KDE) [15]. This technique sequentially and adaptively learns the definition of normality in the online applications and assumes no prior knowledge regarding the underlying distributions, and then detects anomalies. The anomaly detection threshold has been mathematically linked to the user's specified tolerance level for false alarms.

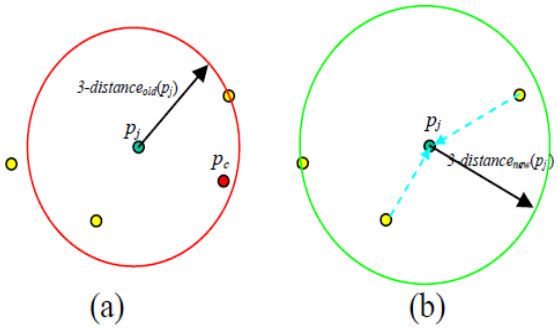

Fig. 2. Update of k-nearest neighbour distance upon deletion of record $p_j\,(k=3)$. a) Prior to deletion, data record p_c is among $3-nearest\ neighbors$ of record p_j; b) After deletion of p_c, $3-distance(p_j)$ increases and reachability distances from two nearest neighbors of p_j (denoted by cyan dashed lines) are updated to $3-distance(p_j)$.

This technique is basically generated by focusing on two problem statements:

Problem 1. Given a sequence of datapoints $\{X_i\}_{i=t-L}^{t+L} \in \mathbb{R}^D$, It need to determine if X_t is a realisation of probability distribution $P_{n,t}$ or P_a. It is considered that the points $\{X_i\}_{i=t-L}^{t+L} \in \mathbb{R}^D$ are independent observations from the mixture distribution P_t:

$$P_t = (1-\pi)P_{n,t} + \pi P_a. \qquad (10)$$

where π is the mixing faction. Here $P_{n,t}$ is slowly time-varying, while P_a is time-invariant. And needed to compute an operator-specified probability of false discovery, P_{FD}^*.

Problem 2. Need to make a preliminary decision about the underlying distribution of X_t at time t, and a final decision at time $t+\ell$ *where* $\ell \ll L$. Thus the preliminary decision is based on the data sequence $\{X_i\}_{i=t-L}^{t}$and the final decision on $\{X_i\}_{i=t-L}^{t+\ell}$, instead of on $\{X_i\}_{i=t-L}^{t+L}$.

As explicit in Problem 2, the aim is to form a preliminary decision about the underlying distribution of X_t at time t, and a final decision at time $t+\ell$. Thus the decision needs to be initially made using $\{X_i\}_{t-L}^{t}$, and eventually using $\{X_i\}_{t-L}^{t+\ell}$, as the sample from P_t

from the interval $\{X_i\}_{t-L}^{t+L}$. The subsequent technique of getting an estimated detection statistic $\hat{T}_t$ is proposed, and $\hat{T}_t$ is subsequently used as the online detection statistic that approximates the probability of false discovery achieved with the algorithm [15] for Problem 1.

At each timestep t, the KEAD first evaluates the mean squared error δ_t in representing X_t using a relatively small dictionary of approximately linearly independent elements $D_t = \{\widetilde{X}_j\}_{j=1}^{m_t}$ in the feature space defined by the kernel function. Error δ_t may be derived to be [16]:

$$\delta_t = \min_{a_t}\{a_t^T \widetilde{K}_{t-1} a_t - 2a_t \tilde{k}_{t-1}(X_t) + k(X_t, X_t)\}. \quad (11)$$

where $[\widetilde{K}_{t-1}]_{i,j} = k(\widetilde{X}_i, \widetilde{X}_j)$ and $[\widetilde{K}_{t-1}]_{i,j} = k(\widetilde{X}_i, \widetilde{X}_j)$ for $i, j = 1 \ldots m_{t-1}$. Note that here $\{\widetilde{X}_j\}_{j=1}^{m_{t-1}}$ represent those selections from $\{X_i\}_{i=1}^{t-1}$ that have been entered into the dictionary up to time $t-1$. The optimum sparsification coefficient vector a_t that minimises δ_t at time t is then:

$$a_t = \widetilde{K}_{t-1}^{-1} . \tilde{k}_{t-1}(X_t) . \quad (12)$$

The expression for error δ_t is simplified into:

$$\delta_t = k_{tt} - \tilde{k}_{t-1}(X_t)^T . a_t. \quad (13)$$

It also maintain a sliding window A_t of the optimal sparsification coefficient vectors a_t for the past L timesteps. One may then use the dictionary D_{t-1} and the matrix of past optimal sparsification coefficient vectors A_t to obtain the online detection statistic $\hat{T}_t$:

$$\hat{T}_t = \frac{1}{L}\sum_{\ell=1}^{L}\sum_{j=1}^{m_{t-1}} a_{\ell j} . k(\tilde{X}_j, X_t)$$

$$= \frac{1}{L}\sum_{i=1}^{L} A_{t-1} . \tilde{k}_{t-1}(X_t). \quad (14)$$

$\hat{T}_t$ is an approximation of T_t in two respects. First, the (sparse) dictionary sample of representative input vectors from the interval $\{t-L : t\}$ is used. The error introduced on this count is governed by the sparsification parameter ν. Second, the interval $\{t-L : t\}$ is used as representative of the interval of interest $\{t-L : t+L\}$. The error introduced on this count is governed by the window length L.

3.4 Online Oversampling Principal Component Analysis

In this a way is propose known as online oversampling principal Component analysis (osPCA) algorithm [17] to handle anomaly problem, and that aims to detect the presence of outliers from an outsized quantity of data via a online updating technique. Unlike prior principal component analysis (PCA)-based approaches, it does not store the entire data matrix or covariance matrix, and thus this approach is especially of interest in online or large-scale problems. The task is completed by oversampling the target instance and extracting the principal direction of the data, osPCA permits to work out the anomaly of the target instance consistent with the variation of the ensuing dominant eigenvector.

It uses the leave one out (LOO) strategy along with PCA. LOO strategy states that adding (or removing) a abnormal data will have an effect on the principal direction a lot of as compared with the adding or removing normal data. Using the above strategy, the principal direction of data set is calculated without considering the target instance of the initial data set. Thus, the outlierness of the targeted instance may be determined by the variation of the ensuing principal directions. A lot of exactly, the distinction between these two eigenvectors can indicate the anomaly of the target instance. By ranking the dissimilarity of all data points, one will establish the outlier instance by a predefined threshold or a preset portion of the data. While it works well for applications with moderate data set size, the variation of principal directions won't be important once the size of data sets is massive.

However, this LOO anomaly detection procedure with an oversampling strategy will markedly increase the computational load. For every target instance, one has to produce a dense covariance matrix and solves the associated PCA drawback. This may refuse the framework for real-world large-scale applications. Though the standard power technique is used to approximate PCA solutions, it needs the storage of the covariance matrix and cannot be simply extended to applications with streaming data or online settings. Therefore, online updating technique is used along with osPCA. This change technique permits to efficiently calculate the approximated dominant eigenvector while not performing eigen analysis or storing the info covariance matrix. Compared to the ability technique or different well-liked anomaly detection algorithms, the desired process prices and memory needs are considerably reduced, and so this technique is very

preferred in online, streaming data, or large-scale issues.

osPCA technique duplicates the target instance multiple times, and also the plan is to amplify the result of outlier instead of that of traditional data. Whereas it might not be adequate to perform anomaly detection merely supported the foremost dominant eigenvector and ignore the remaining ones, on-line osPCA methodology aims to expeditiously confirm the anomaly of every target instance while not sacrificing computation and memory efficiency. More specifically, if the target instance is associate outlier, this oversampling scheme permits to hyperbolize its impact on the foremost dominant eigenvector, and so it specialize in extracting and approximating the dominant principal direction in an online fashion, rather than hard multiple eigenvectors rigorously. The detailed formulation of the osPCA is given now, suppose that it oversample the target instance n times; the associated PCA can be formulated as follows:

$$\sum_{\tilde{A}} u_t = \lambda u_t , \qquad (15)$$

where $\tilde{A} = A\ U\ \{X_{t,\tilde{n}}, X_t\} \in \mathbb{R}^{(n+\tilde{n})\times P}$. The mean of $\tilde{A}$ is μ, and thus

$$\sum_{\tilde{A}} = \frac{1}{n+\tilde{n}} \sum_{X_{i\in A}} X_i X_i^T + \frac{1}{n+\tilde{n}} \sum_{i=1}^{\tilde{n}} X_i X_i^T - \mu\mu^T$$

$$= \frac{1}{r}\frac{AA^T}{n} + \frac{r}{1+r} X_t X_t^T - \mu\mu^T. \qquad (16)$$

osPCA framework, duplicates the target instance n times (e.g., 10 percent of the size of the original data set), and we will compute the score of outlierness s_t of that target instance, as defined in (17).

$$s_t = 1 - \left| \frac{\langle u_t, u \rangle}{\|u_t\| \|u\|} \right| \qquad (17)$$

If this score is above some predetermined threshold, it considers targeted instance as associate degree outlier. With this oversampling strategy, if the target instance may be traditional data, it will observe negligible changes within the principal directions and also the mean of the data. Its value noting that the employment of osPCA not exclusively determines outliers from the present data; it will be applied to anomaly detection issues with streaming data or those with on-line necessities. Clearly, the most important concern is that the computation value of scheming or changes the principal directions in massive scale issues.

4. Comparison

The overall paper describes the anomaly detection techniques that are been developed for on-line applications or data streams. In section 3 we have described briefly the various online anomaly detection techniques. The studied techniques are compared with that of the computational complexity and memory requirement. The table 1 shows the basis on which we have compared these different online anomaly detection techniques.

Table 1. Comparisons of SCAN Statistics, Incremental LOF, KEAD and osPCA for Online Anomaly Detection in Terms of Computational Complexity and Memory Requirement.

	SCAN Stat. [9]	Incret. LOF [14]	KEAD [15]	osPCA [17]
Computational Complexity	O(n)	O(nlogn)	$O(np^2+n^2)$	O(p)
Memory Requirement	O(n)	O(np)	$O(n^3)$	O(p)

Note that n and p are the size and dimensionality of data respectively.

5. Conclusion

In this paper studied a number of the anomaly detection technique developed for on-line applications or data streams. Through we've got found that the bulk of the work has been in done offline mode and a little work has been done or goes on for the online application. The technique that has been developed is in preliminary stage and does not work with the large data sets. The accuracy of this developed technique is a smaller amount compared with offline applications.

This study additionally describes the most options of many anomaly detection techniques that are presently available in brief manner. The presented information constitutes a crucial purpose to begin for addressing Research & Development within the field of online anomaly detection.

Countermeasures that are quicker and more effective are required to cope up with the data with higher dimension. We discover that the bulk of surveyed works don't meet these necessities. On the whole, the findings ensure a standard trend within the experimental technology.

6. References

[1] Hawkins, D.M. 1980. Identification of Outliers. Chapman and Hall Publication.

[2] Song, X., Wu, M., Kermaine, C., and Ranka, 2007. Conditional Anomaly Detection. IEEE Transactions on Knowledge and Data Engineering, 631-645.

[3] Joshi, M. V., Agarwal, R. C., and Kumar, V. 2001. Mining Needle in a Haystack: Classifying Rare Classes via Two-Phase Rule Induction. International Conference on Management of Data. ACM Press 2001, 631-645.

[4] Joshi, M. V., Agarwal, R. C., and Kumar, V. 2002. Predicting rare classes: can boosting make any weak learner strong? International Conference on Knowledge Discovery and Data Mining. ACM 2002, 297–306.

[5] Theiler, J. and Cai, D. M. 2003. Resampling approach for anomaly detection in multispectral images. *SPIE*. vol. 5093 , 230–240.

[6] Fujimaki, R., Yairi, T., and Machida, K. 2005. An Approach to Spacecraft Anomaly Detection Problem Using Kernel Feature Space. International Conference on Knowledge Discovery in Data Mining, 401–410.

[7] Dasgupta, D. and Nino, F. 2000. A Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection. IEEE International Conference on Systems, Man, and Cybernetics. vol. 1(Oct 2000), 125–130.

[8] Daniel, B. N. and Gregory F. C. 2010. A Multivariate Bayesian SCAN Statistic for Early Event Detection and Characterization. Machine Learning, vol. 79 (June 2010), 261–282.

[9] Ryan, T., Zoubin, G. and Steven, B. 2010. Fast Online Anomaly Detection Using Scan Statistics. IEEE International Workshop on Machine Learning for Signal Processing. 385-390

[10] Joseph, I. N. 1982. Approximations for Distributions of SCAN Statistics. Journal of the American Statistical Association, vol. 77, No. 377 (June1982), 177–183.

[11] Breunig, M. M., Kriegel, H. P., Ng, R. T. and Sander, J. 2000. LOF: Identifying Density Based Local Outliers. ACM SIGMOD Conference, Dallas, TX, 93-104.

[12] Lazarevic, L. E., Ozgur, A., Srivastava, J. and Kumar, V. 2003. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. Third SIAM International Conference on Data Mining, San Francisco, CA (May 2003), 25-36.

[13] Lazarevic, A. and Kumar, V. 2005. Feature Bagging for Outlier Detection. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, (Aug 2005), 157-166.

[14] Dragoljub, P., Aleksandar, L. And Longin, J. L. 2007. Incremental Local Outlier Detection for Data Streams. IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 504-515.