Integrated Learning Method for Anomaly Detection Combining KLSH and Isolation Principles

Hongchun Qu
College of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China
hcchyu@gmail.com

Zonglan Li
College of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China
1217292567@qq.com

Jingjing Wu
College of Automation
Chongqing University of Posts and Telecommunications
Chongqing, China
2734801096@qq.com

***Abstract*—Aiming at the problem that the Isolated Forest (iForest) has low local anomaly detection accuracy in high-dimensional and massive data sets, this paper proposes an anomaly detection method that combines locality-sensitive hashing algorithm based on Gaussian Kernel Function (KLSH) and means-optimized iForest algorithm. In this method (KLSH+iForest), the kernel function is used to map the data from the linearly indivisible data space to the linearly separable feature space, and local anomalies are converted into global anomalies. Based on above, iForest is constructed to perform anomaly detection on the Kernelized data sets. To solve the problem of how to select the optimal segmentation attributes and values for iForest, this paper proposes a mean optimization strategy. While maintaining the ability of iForest to detect global anomalies, KLSH+iForest also improves the accuracy of local anomaly detection. We compare KLSH+iForest with the LOF algorithm and the improved algorithms based on LSH on public data sets. Experimental results show that KLSH+iForest has significantly improved the accuracy and efficiency of anomaly detection in high-dimensional and massive data sets.**

Keywords-component; anomaly detection; isolated forest; locality-sensitive hashing; kernel function

I. INTRODUCTION

With the development of information technology, data sets with large-volume, high-dimensional, heterogeneous, geographically distributed pose considerable challenge on the anomaly detection field in the current big data era. There are many anomaly detection algorithms, anomaly detection based on clustering [1], anomaly detection based on distance [2], anomaly detection based on density [3] and anomaly detection based on angle [4]. One can refer to [5] for more details. Anomaly detection has become an important research direction in the fields of data mining and machine learning and has led to numerous applications in a wide range of domains, such as detecting fraud in bank transaction data[6]; detecting intrusion protocol problems in network security data[7]; monitoring patients' vital signs in the abnormal detection framework of medical wireless sensor network[8]; detecting abnormal vehicle trajectories in traffic trip data[9]; detecting abnormal high temperature data to prevent forest fires in environmental sensing data[10]; detecting merchants' violations in the e-commerce sales data[11]; detecting the factors that affect blueberry pollination efficiency in the agricultural data[12] etc. Although researches on anomaly detection have started earlier and various methods have emerged, the time complexity of the algorithms in these methods is relatively high. Therefore, it is necessary to accurately and quickly detect anomalies from high-dimensional massive data sets.

In order to detect abnormal points quickly, Liu [13], [14] proposed the iForest based on the idea of isolation. Among many anomaly detection algorithms, iForest has low time complexity and good detection effect. However, this algorithm also has some problems: (1)IForest uses a global anomaly score that is not sensitive to local data distribution of the dataset to measure the degree of global anomaly. When the local outliers are close to the normal cluster, iForest cannot effectively detect local anomalies. (2)IForest builds itree based on randomization technology. It is unclear how to choose the appropriate segmentation attributes and segmentation values [15]. This may cause the attributes related to the outliers to be missed, which reduces the reliability of iForest.

Facing the above problems of iForest, this paper proposes an anomaly detection method that combines KLSH and means-optimized iForest. The contribution of this paper is as follows: (1)The isolation method works better with a small sample size. Massive data sets can reduce iForest's ability to isolate anomalies.Therefore, the technique of random non-repeating subsampling provides a favorable environment for the good work of iForest. (2)In order to improve the detection accuracy of local anomalies in iForest, this paper uses the LSH algorithm based on the Gaussian kernel function to map data sets from the data space to the feature space, and build an isolated forest by using kernelized sub-sampling data sets. (3)Different from iForest's method of randomly selecting segmentation attributes and segmentation values to construct itree, this paper proposes a mean optimization strategy to select appropriate segmentation attributes and segmentation values.Through the above three methods, the detection accuracy and efficiency of the iForest are significantly improved.

The remainder of this paper is organised as follows. Section II reviews the related research work. In Section III, we explained iForest and explained our approach in detail. We give experimental comparisons in Section IV and analyze the results. We conclude his paper in Section V.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. Related Work

Aiming at the two problems of iForest, there are many articles to optimize it. For example, in order to improve the local anomaly detection rate, Aryal [16] proposed to replace the global ranking method based on path length with a local quality method based on relative quality. Ding [17] proposed iForestASD, which used a sliding window frame to process stream data and considered the concept drift phenomenon. The algorithm can effectively detect abnormal instances of stream data. Shen [18] proposed EGITree based on three heuristic ideas by choosing the dimension with the smallest entropy as the segmentation attribute. Bandaragoda [19] proposed an isolation-based KNN method, which was more efficient than an anomaly detection algorithm based on distance or density and can detect local anomalies. However, the calculation efficiency of this algorithm is lower than iForest due to the calculation of hypersphere. Marteau [20] proposed HiForest, which used the unsupervised nature of iForest to improve the effectiveness of anomaly detection. Liu [21] used hyperplane to effectively detect local anomalies. Due to the hyperplane calculation, the calculation efficiency of this algorithm is lower than iForest. Yu [22] proposed an algorithm combining iForest and LOF, which improved the accuracy and stability of the algorithm. Xu [23] improved the calculation efficiency by deleting some itrees with poor detection performance, and used the fast convergence of the simulated annealing optimization algorithm to improve the efficiency of anomaly detection. Zhang [24] proposed an isolation-based LSHiForest algorithm, which used different distance measurements to construct different LSH forests to satisfy anomaly detection under different data distributions. Liao [25] proposed E-iForest, which introduced dimensional entropy as the basis for selecting segmentation attributes and segmentation points, and adopted three isolation strategies to construct itree. Experiments proved that this algorithm was more stable than iForest.

Based on the above analysis, this paper proposes an anomaly detection method that combines KLSH and iForest. Under the premise of maintaining the global anomaly detection capability of iForest, use kernel functions to map data to feature space to effectively divide local anomalies. In order to improve the detection accuracy and efficiency of the algorithm, a random non-repeating subsampling technique and a mean optimization strategy are used. Experimental analysis shows that our method can effectively solve the two problems of iForest.

III. Isolated Forest and Improvement

A. Isolated Forest (iForest)

Definition 1: Isolation tree (iTree). Given a sample of data $X = \{x_1, x_2, ..., x_n\}$ of n instances from a d variate distribution, to build an isolation tree(iTree), recursively divide X by randomly selecting an attribute q and a split value p. Divide the data of $x_{i_{(q)}} < p$ into the left subtree T_l, and the data of $x_{i_{(q)}} \geq p$ into the right subtree T_r. Where $x_{i_{(q)}}$ represents the i-th sample point on the segmentation attribute q in dataset X. Tree building stops until any one of the following conditions is met: (1) the tree reaches a limited height; (2) there is only one sample on the node; (3) all data in X have the same values [21].

Definition 2: Path Length. $h(x)$ is the number of edges that the sample point x traverses from the root node of the itree to the leaf nodes. Given a dataset containing n samples, the average path length of the tree is:

$$c(n) = 2H(n-1) - 2(n-1)/n \tag{1}$$

where $H(i)$ is the harmonic number and it can be estimated by $In(i) + 0.5772$ (Euler's constant). As $c(n)$ is the average of $h(x)$ given n, we use it to normalise $h(x)$ [13].

Definition 3: The anomaly score s of the sample point x is defined as:

$$s(x,n) = 2^{-E(h(x))/c(n)} \tag{2}$$

where $E(h(x))$ is the expected path length of the sample x in a batch of itrees. When $E(h(x)) \to c(n)$, $s \to 0.5$: it is unclear whether x is an outlier. When $E(h(x)) \to 0$, $s \to 1$: x is determined to be abnormal. When $E(h(x)) \to n\text{-}1$, $s \to 0$: x is determined to be normal.

B. Related Algorithm

Locality-Sensitive Hashing (LSH): Compared with data instances that are far away, LSH hashes data instances that are close to each other into the same bucket with a high probability to build indexes for similarity search [26].

Definition 1: A set of data points based on the distance function DI. For any data point $p, q \in R^d$, the function family $H = \{h_1, h_2, ...h_n\}$ is called (r_1, r_2, p_1, p_2)-sensitive to the distance function $DI(\|p-q\|)$, the following conditions:

$$1).\ d(p,q) < r_1 \Rightarrow Pr_H[h(q) = h(p)] \geq p_1;$$
$$2).\ d(p,q) < r_2 \Rightarrow Pr_H[h(q) = h(p)] \geq p_2.$$

In order to make a family of Locality-Sensitive Hashing functions available, the condition $r_1 < r_2, p_1 > p_2$ must be satisfied. When p, q are sufficiently similar, the probability of mapping to the same hash value is greater. A hash function that meets the above two conditions is called (r_1, r_2, p_1, p_2)-sensitive.

The process of hashing a dataset to generate one or more hash tables is called Locality-sensitive Hashing(LSH). LSH can arbitrarily pull p_1 and p_2 far away while keeping r_1 and r_2 unchanged, which shows that LSH can improve the quality of similarity search as much as possible [27], [28].

Kernel function: The kernel function can map data set to a high-dimensional feature space, thereby improving the computing power of machine learning algorithms [29], [30].

Definition 2: Given a data set $X = \{x_1, x_2, ..., x_n\}$. Let H be the feature space. If there is a mapping from X to H: $\phi(x): X \to H$, for all $x_i, x_j \in X$, $K(x_i, x_j) = \langle \phi(x_i) \cdot (x_j) \rangle$. $K(x_i, x_j)$ is called the kernel function, and $\phi(x)$ represents the mapping function from the original input space to the feature space. Where $\langle \phi(x_i) \cdot (x_j) \rangle$ is the inner product of $\phi(x_i)$ and $\phi(x_j)$.

Gaussian kernel function: The following is the formula of Gaussian kernel function. The adjustable parameter γ controls its width.

$$K(x_i, x_j) = \exp(-\gamma \left\| x_i - x_j \right\|^2) \quad (3)$$

C. Details of The Improvement Process

Like iForest, KLSH+iForest is divided into two phases: in the training stage, nt number of KLSH trees are generated by Algorithm 1; in the evaluation phase, the abnormal score of the data instance after kernelized is calculated by Algorithm 4.

Algorithm 1: KLSH_iForest(X, S, nt, ts)

Input: X - dataset; S - sub_sample sets; nt - number of KLSH ITrees; ts - size of train data.
Output: A forest of KLSHtrees $\{\text{itree}_i | 1 \le i \le nt\}$.
1: $X \leftarrow$ klsh format the X
2: $avg_i \leftarrow$ Calculate the average of the dimensions of X
3: **for** i ← 1, nt **do**
4: $S_i \leftarrow$ variable_subsamping(X);
5: $H \leftarrow 2\log_2(sub\_size)+0.8327$, sub_size is size of S_i;
6: $S_i \leftarrow$ klsh format the S_i;
7: $train\_data \leftarrow$ Select ts from S_i
8: $S_i \leftarrow$ Calculate_kernel($train\_data$, $S_i T$)
9: itree$_i$ ← build ITree: KLSH_ITree(S_i, J, H, avg), J is current height limit;
10: return {itree$_i$}.

Algorithm 1 describes the process of recursively constructing KLSH iForest by kernelized subsamples $S_i \in X$. Line 4 uses the subsampling technique [31]. Different from [31], we use a random non-repeated sampling technique. The specific process is: randomly and non-repetitively generate nt (nt = 100) subsets S_i from X. The size of S_i is sub_size, $sub\_size \in [len(X)/2, len(X))$. In order to make the itree height distribution balanced and reduce the calculation cost, this paper uses a uniform distribution as $F(log_{10}(0.1/d), log_{10}(1.0/d))$. In line 5, H is the height limit, and H can save computing costs during the construction of KLSH itree. The impact of H on the division of normal instances is not significant, because the depth of normal instances is generally deeper than that of abnormal instances. LSH will hash two close data instances into the same bucket. Therefore, H must be long enough to fully separate the two instances. This paper cites $H = 2\log_2(sub\_size) + 0.8327$ in [24] and compared it with the height limit in [21] in experiments. We found that the segmentation results of H in [24] were more accurate. Line 8 is the calculation of the kernel function, see Algorithm 3. Line 9 is the itree building process, see Algorithm 2.

Algorithm 2: KLSH_ITree(S_i, J, H, avg)

Input: S_i - sub_sample set; J - current height limit; H - height limit; avg_i- the average of the dimensions of X.
Output: A tree of KLSH.
1: **If** length of $S_i = 0$ **then**
2: *return null*;
3: **else if** $J >= H$ or length of $S_i = 1$ **then**
4: *return itree{*left_itree ← T_l, right_itree ← Tr*}*;
5: $sub\_avg_i \leftarrow$ Calculate the average of the dimensions of S_i
6: *index* ← Select the dimension with the greatest difference between avg_i and sub_avg_i
7: v ← avg$_{index}$, v is the split property value
8: **for** row ← S_i **do**
9: **if** row_{index} < v **then**
10: left_itree{*row*}
11: **else**
12: right_itree{*row*}
13: KLSH_ITree(itree, $J+1$, H,avg_i)

Different from iForest in selecting segmentation attributes and segmentation values based on randomization technology, we propose a mean optimization strategy to construct KLSH itree. The specific steps are as follows: (1)Calculate the dimensional mean of the kernelized dataset X, and record it as: $avg_i\ (1 \le i \le d)$. (2)Randomly sample nt subsets S_i from the kernelized data. (3)Calculate the mean of the dimensions of each sample subset, written as: sub_avg_i. (4)Calculate the difference between avg_i and sub_avg_i : $diff_i = |avg_i - sub\_avg_i|$. (5)Select the i dimension corresponding to the maximum value of $diff_i$ as the segmentation attribute, sub_avg_i as the segmentation value. (6)Compare x_{ji} $(j \in len(S_i), S_i \in X)$ with sub_avg_i, if $x_{ji} < sub\_avg_i$, put the j-th sample vector into the left subtree, otherwise, into the right subtree. (7)When the sample subset S_i is divided, an itree is constructed. Repeat the above steps for the nt sample subsets in order to obtain nt itrees. nt itrees form an iForest.

In order to improve the accuracy of iForest to detect local anomalies, this paper uses a Gaussian kernel function, referring to (3), where the tunable parameter γ refers to [24]. The rationale is that local anomalies in the original space can be mapped as global anomalies in the kernelised space and become more susceptible to be isolated and detected [24]. Kernel functions are used for the first and sixth lines in Algorithm 1, and the first line in Algorithm 5.

Algorithm 3: Calculate kernel function(X_t, Y)

Input: X_t, Y - train_data.
Output: The kernel matrix K.
1: squared_diff ← Calculate the square difference
2: X_0 ← a matrix of 1×n_Y, all of element are 1, n_Y is size of Y
3: $part\_X \leftarrow X_2.\mathrm{T}\cdot X_0$, X_2 ← Calculate the summation of the dimensions of $\mathrm{X}.T^2$
4: Y_0 ← a matrix of n_X×1, all of element are 1, n_X is size of X
5: $part\_Y \leftarrow \mathrm{X}_0\cdot \mathrm{Y}_2.T$, Y_2 ← Calculate the summation of the dimensions of $\mathrm{Y}.T^2$
6: $squared\_diff = part\_X + part\_Y - 2\cdot \mathrm{X}\cdot \mathrm{Y}.T$
7: *kernel_kwds* ← Calculate the uniform distribution
8: $kernel\_kwds \leftarrow 10^{\mathrm{F}}$, $\mathrm{F} \in [\log_{10}(0.1/d), \log_{10}(1.0/d))$, d is dimension of Y
9: $K \leftarrow (kernel\_kwds \cdot squared\_diff)\text{^}0.5$

We refer to the kernelized LSH method [32] and create nt KLSH instances. The kernel-based LSH function is defined as:

$$h(\phi(x)) = sign(\sum_{i=1}^{s_i} \omega(i) k(x, x_i)) \qquad (4)$$

Where $\phi(x)$ is the mapping function and $k(x, x_i)$ is the kernel function, $\omega = \hat{\mathrm{K}}^{-1/2} e_\zeta$, e_ζ is an $s_i \times 1$ vector, $\hat{\mathrm{K}}$ is the $s_i \times s_i$ centred kernel matrix. In order to construct a set of LSH functions, a sample subset is randomly sampled from the dataset X_t , and the sample size is s_i , $s_i \in \min(\sqrt{len(X_t)}, 300)$. Since the running time of KLSH increases significantly with sample size s_i , s_i takes the minimum value. ζ is a random projection, $\zeta \in \min(s_i / 4, 30)$ [32].

Algorithm 4: Calculate Anomaly Scores(X, ts, S)

Input: X - dataset; ts - size of train data; S - sub_sample sets.
Output: scores.
1: *avg_size* ← Calculate the average of S's size
2: **for** row ← X **do**
3: *avg_ehx* ← Calculate the average of *PathLength*
4: $c_n \leftarrow 2(\ln(avg\_size-1)+0.5772)-2(avg\_size-1)/avg\_size$
5: index ← avg_ehx / c_n
6: $score_{row} \leftarrow 2$ ^ -index
7: scores ← $\{score_{row}\}$

In the evaluation phase, the abnormal score of the data instance after kernelized is calculated by Algorithm 4. In the training phase, KLSH iforest is completed, and we further evaluate the abnormal scores of the kernelized data instances.

In the first line of Algorithm 5, the kernelized data instance is put into each itree, and the length of the path that each data instance traverses all the itrees is recorded. In Algorithm 4, the average path length is calculated in the fourth line, referring to (1), and the fifth and sixth lines are calculating the abnormal score, referring to (2).

Algorithm 5: Calculate Path Length(R, S,*itrees*,*nt*)

Input: R - a row of dataset; S - sub_sample sets; *itrees*; *nt* - *number of KLSHTrees.*
Output: *Path Length.*
1: row ←Calculate_kernel*(train_data,* Zrow.*T) , train_data*
←select *ts* from *Si , Zrow* ← Convert R to a matrix
2: **for** *itree* ← *itrees* **do**
3: **while** itree = null **do**
4: **if** row[itree.splitAttrIndex] < itree.splitAttrValue **then**
5: *return PathLength(s_i, left_itree, path_length+1)*
6: **else** row[itree.splitAttrIndex] >= itree.splitAttrValue
7: *return PathLength(s_i, right_itree, path_length+1)*
8: **end if**

IV. Experimental Results and Analysis

All the code running environment of this paper is IntelliJ IDEA, the programming language is python3, the operating system is Micros Windows 10, the hardware environment CPU is Intel (R) Core (TM) i7-6700HQ CPU @ 2.6GHz, and the RAM is 4.00GHz. To explore the performance of the proposed KLSH+iForest, we conduct an experimental study which using Yoga dataset selects from the UCR repository[33] and other datasets select from the UCI machine learning repository as shown in Table I. All datasets are normalized by the z-score normalization method, where n is the sample size, d is the dimension, *rate* is the anomaly rate, and φ represents the abnormal score determination threshold. Table II shows the average AUC (Area Under Curve) of all algorithms running 10 times. In this paper, the iterative calculation method is used to put the dataset into the model to calculate the model parameters with the highest AUC value, such as the number of isolated trees, the size of subsamples, and the size of the judgment threshold of abnormal score. Then run all the datasets under the optimal parameter 10 times to get the average of AUC value. For each dataset, the leading values are highlighted. The algorithms with highlighted values can be deemed to have similar performance as their AUC difference is less than 5%.

TABLE I. Basic Description of Datasets.

Name	n	d	Outlier vs. Inlier Labels	*rate*	φ
Glass	214	9	class 6 vs. others	4.2%	0.58
Yoga	3000	400	class 1 vs. class 2	46.4%	0.59
Wilt	4339	5	class w vs. class n	1.68%	0.59
Har	5744	561	class 3 vs. others	13.8%	0.58
Musk	6598	166	39 molecules were classified as musk vs. others	15.4%	0.59

TABLE II. AUC of All Methods (%).

Methods:	KLSH+ iForest	iForest	LOF	ALSH	L1SH	L2SH	KLSH
Glass	80.40	62.67	59.67	**88.83**	79.67	77.12	**83.46**
Yoga	**53.35**	50.05	52.11	50.11	53.26	52.33	52.28
Wilt	55.48	52.23	**68.59**	**60.12**	51.07	51.99	50.72
Har	**64.48**	54.87	51.09	64.04	**69.30**	52.16	50.33
Musk	**66.00**	54.73	53.07	52.69	59.15	64.95	63.54

Analyzing the relationship between dimensions d and AUC. Compared with iForest, the AUC of KLSH+iForest is higher than iForest in the five datasets, which indicates that KLSH+iForest has higher detection accuracy than iForest. The reason is that KLSH+iForest is based on random subsampling technology and mean optimization strategy to construct itree. At the same time, combined with the Gaussian kernel function, the detection accuracy of local anomalies of iForest is improved. Compared with LOF, low-dimensional dataset Wilt has higher AUC in LOF than KLSH+iForest, but in KLSH+iForest, other datasets have good detection results. The reason is that LOF is a distance-based anomaly detection method. Therefore, *Wilt* has better detection accuracy in LOF. In contrast, KLSH+iForest is more ideal for detecting high-dimensional data. Compared with ALSH, the AUC of the low-dimensional datasets *Wilt* and *Glass* in ALSH are 60.12% and 88.83%, respectively. The AUC of other high-dimensional datasets performs well in KLSH+iForest. The reason is that ALSH is a locality-sensitive hashing algorithm based on angle measurement. This algorithm has better detection effect in low-dimensional datasets where the spatial distribution angle is easy to divide. Compared with L1SH, the AUC of high-dimensional dataset *Har* in KLSH+iForest is 64.48%, which is only 5% lower than L1SH. In the high-dimensional datasets *Musk* and *Yogo*, the AUC of KLSH+iForest is higher than that of L1SH. Overall, KLSH+iForest has better detection accuracy than L1SH based on Manhattan distance. Compared with L2SH, KLSH+iForest has higher AUC than L2SH. The reason is that L2SH is based on Euclidean distance metric, while KLSH+iForest is based on Gaussian kernel function LSH. Kernel functions can improve the accuracy of anomaly detection. Therefore, KLSH+iForest is better than L2SH in detecting anomalies. Compared with KLSH, the AUC of *Glass* in KLSH+iForest is 80.40%, which is only 3% lower than KLSH. In the high-dimensional datasets *Musk*, *Yogo* and *Har*, KLSH+iForest has higher AUC than KLSH. Although KLSH and KLSH+iForest also use the kernel function, KLSH+iForest uses a mean optimization strategy to build the KLSH itree. KLSH only builds the LSH tree based on the isolation idea, and doesn't optimize the selection of the optimal segmentation attributes and segmentation values. Therefore, KLSH+iForest has better detection accuracy than KLSH.

Analyzing the relationship between sample size n and AUC: In 5 datasets, although the AUC of *Glass* in KLSH is higher than KLSH+iForest, KLSH+iForest is only 3% lower than KLSH. The AUC of the other 4 datasets in KLSH+iForest are all higher than KLSH, indicating that with the increase of sample size n, the detection effect of KLSH+iForest is better than KLSH. **To sum up,** with the increase of sample size n and dimension d, the AUC of KLSH+iForest is generally higher than these reference algorithms, which indicates that KLSH+iForest can be applied to anomaly detection with a larger number of sample points and higher dimensions. Experiments show that the threshold value φ of the abnormal score of the proposed method is basically around 0.58-0.59, and the variation range is 0.01. It shows that KLSH+iForest can provide users with an ideal threshold reference range for different datasets.

In KLSH+iForest, the Gaussian kernel function maps datasets to the feature space, and then uses iForest to detect the abnormal points in the feature space. The execution time will increase significantly with the number of dimensions. While ALSH, L1SH, and L2SH [24] have similar execution times, they all construct LSH forests based on different distance measures in the data space. Therefore, the running time of KLSH+iForest is longer than ALSH, L1SH and L2SH. Different from the process of KLSH constructing LSH forest, KLSH+iForest is to build iForest based on random non-repeated subsampling technology and mean optimization strategy. Random non-repeated subsampling technology reduces the sample size of the constructed itree. The mean optimization strategy improves the efficiency of KLSH+iForest by purposefully selecting segmentation attributes and segmentation values. At the same time, iForest has logarithmic time complexity. Compared to KLSH, KLSH+iForest runs more efficiently.

V. CONCLUSION

IForest cannot effectively detect local anomalies in high-dimensional and massive data sets. In this paper, we propose an anomaly detection method (KLSH+iForest) that combines locality-sensitive hashing algorithm based on Gaussian Kernel Function (KLSH) and means-optimized Isolated Forest algorithm. This method uses kernel functions to map data sets to a high-dimensional feature space, converts local anomalies into global anomalies, and then uses iForest to perform anomaly detection on the kernelized data sets. In order to quickly find the optimal segmentation attributes and segmentation values, a mean optimization strategy is proposed. Experiments on KLSH+iForest, LOF algorithm and the improved algorithms based on LSH on public data sets. Experimental results show that KLSH+iForest outperforms iForest. Compared with KLSH, the detection accuracy and efficiency of KLSH+iForest are better than KLSH in most cases. The experimental analysis concludes that KLSH+iForest can be applied to anomaly detection of data sets with more sample points and higher dimensions.

ACKNOWLEDGMENT

Funding was received from the National Natural Science Foundation of China (61871061) and Chongqing Research Program of Basic Research and Frontier Technology (cstc2017jcyjAX0453).

REFERENCES

[1] Lian Duan, Lida Xu, Ying Liu, and Jun Lee, "Cluster-based outlier detection," Annals of Operations Research 168.1 (2009): 151-168.

[2] Sridhar Ramaswamy, Rajeev Rastogi, Kyuseok Shim, and Taejon Korea, "Efficient algorithms for mining outliers from large data sets," ACM SIGMOD Records, vol. 29, no. 2, pp. 427–438, 2000.

[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," ACM SIGMOD Record, vol. 29, no. 2, pp. 93–104, 2000.

[4] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek, "Angle-based outlier detection in high-dimensional data," Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2008.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys (CSUR), vol. 41, no. 3, p. 15, 2009.

[6] Bao-Hua Jiang, "Research on Withdrawal Behavior Pattern of Bank Card Withdrawal Anomaly Detection Algorithm Based on Rule," Computer Knowledge and Technology (2015).

[7] Hongchun Qu, Zeliang Qiu, Xiaoming Tang, Min Xiang, and Ping Wang, "Incorporating unsupervised learning into intrusion detection for wireless sensor networks with structural co-evolvability," Applied Soft Computing 71 (2018): 939-951.

[8] Osman Salem, Alexey Guerassimov, Ahmed Mehaoua, and Anthony Marcus, "Anomaly Detection in Medical Wireless Sensor Networks using SVM and Linear Regression Models," International Journal of E-Health and Medical Communications 5.1(2014):20-45.

[9] Daqing Zhang, Nan Li, Zhi-Hua Zhou, Chao Chen, Lin Sun, and Shijian Li, "iBAT : detecting anomalous taxi trajectories from GPS traces," International Conference on Ubiquitous Computing ACM, 2011.

[10] Yashwant Singh, Suman Saha, Urvashi Chugh, and Chhavi Gupta, "Distributed Event Detection in Wireless Sensor Networks for Forest Fires," Uksim International Conference on Computer Modelling & Simulation IEEE, 2013.

[11] Jagdish Ramakrishnan, Elham Shaabani, Chao Li, and Mátyás A. Sustik, "Anomaly Detection for an E-commerce Pricing System," (2019).

[12] Hongchun Qu and Frank Drummond, "Simulation-based modeling of wild blueberry pollination," Computers and Electronics in Agriculture 144(2018):94-101.

[13] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "Isolation forest," 2008 Eighth IEEE International Conference on Data Mining. IEEE, 2008.

[14] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "Isolation-based anomaly detection," ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012): 3.

[15] Zhen Liu, Xin Liu, Jin Ma, and Hui Gao, "An optimized computational framework for isolation forest," Mathematical Problems in Engineering 2018 (2018).

[16] Sunil Aryal, Kai Ming Ting, Jonathan R. Wells, and Takashi Washio, "Improving iforest with relative mass," Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham, 2014.

[17] Zhiguo Ding and Minrui Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," IFAC Proceedings Volumes 46.20 (2013): 12-17.

[18] Yanhui Shen, Huawen Liu, Yanxia Wang, Zhongyu Chen, and Guanghua Sun, "A novel isolation-based outlier detection method," Pacific Rim International Conference on Artificial Intelligence. Springer, Cham, 2016.

[19] Tharindu Bandaragoda, Kai Ming Ting, David W. Albrecht, and Fei Tony Liu, "Isolation-based anomaly detection using nearest-neighbor ensembles:iNNE." Computational Intelligence 34.4 (2018): 968-998.

[20] Pierre-Francois Marteau, Saeid SOHEILY-KHAH, and Nicolas Béchet, "Hybrid Isolation Forest-Application to Intrusion Detection," arXiv preprint arXiv:1705.03800 (2017).

[21] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, "On detecting clustered anomalies using SCiForest," Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin, Heidelberg, 2010.

[22] Xiao Yu, Lu An Tang, and Jiawei Han, "Filtering and refinement: A two-stage approach for efficient and effective anomaly detection," 2009 Ninth IEEE International Conference on Data Mining. IEEE, 2009.

[23] Dong Xu, Yanjun Wang, Yulong Meng, and Ziying Zhang, "An Improved Data Anomaly Detection Method Based on Isolation Forest," 2017 10th International Symposium on Computational Intelligence and Design (ISCID). Vol. 2. IEEE, 2017.

[24] Xuyun Zhang, Wanchun Dou, Qiang He, Rui Zhou, Christopher Leckie, Kotagiri Ramamohanarao, et al, "LSHiForest: a generic framework for fast tree isolation based ensemble anomaly analysis," 2017 IEEE 33rd International Conference on Data Engineering (ICDE). IEEE, 2017.

[25] Liefa Liao and Bin Luo, "Entropy Isolation Forest Based on Dimension Entropy for Anomaly Detection," International Symposium on Intelligence Computation and Applications. Springer, Singapore, 2018.

[26] Indyk, Piotr, and Rajeev Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," Proceedings of the thirtieth annual ACM symposium on Theory of computing. ACM, 1998.

[27] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji, "Hashing for similarity search: A survey," arXiv preprint arXiv:1408.2927, 2014.

[28] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," Proceedings of the twentieth annual symposium on Computational geometry. ACM, 2004.

[29] Shutao Li, Kunzhong Zhang, Puhong Duan, and Xudong Kang, "Hyperspectral Anomaly Detection With Kernel Isolation Forest," IEEE Transactions on Geoscience and Remote Sensing (2019).

[30] K R. Müller, S. Mika, G. Ratsch,K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," IEEE transactions on neural networks 12.2 (2001): 181-201.

[31] Kollios G, Gunopulos D, Koudas N, and Berchtold S, "Efficient biased sampling for approximate clustering and outlier detection in large data sets," IEEE Transactions on Knowledge and Data Engineering 15.5 (2003): 1170-1187.

[32] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," IEEE TPAMI, vol. 34, no. 6, pp. 1092–1104, 2012.

[33] Anthony Bagnall, Anh Dau, Jason Lines, and Michael Flynn, "The UCR Time Series Archive," (2018).