RKDE: Data Compression for Time-Series Data based on Kernel Density Estimation in Reservoir Algorithm .

CAO Yanyun, XU Peng
(School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876)

Abstract: Edge-cloud collaborative anomaly detection has become the most important anomaly detection architecture. However, only in the most ideal state can the central cloud platform be fully trained with sufficient data. In the case of limited communication, we have to consider reducing the use of communication resources, but still maintain a high accuracy rate of anomaly detection. In this context, RKDE, a reservoir sampling algorithm based on kernel density estimation, is proposed to reduce the amount of data uploaded to the cloud by the edge end. By improving the probability of abnormal data being sampled, the compression pool of upload is constructed, the redundant process in gradient exchange is reduced, and the sampling process is timely fed back and adjusted according to the abnormal detection results in the cloud. At the same time, RKDE is compared with several sampling algorithms to demonstrate its performance advantages.

Key words: Kernel Density Estimation; Gradient compression; Reservoir Sampling

0 Introduction

In recent years, with the innovation and integration of the Internet of Things, cloud computing, wireless sensor network and other technologies, massive data has been generated in various fields, such as environmental monitoring, intelligent buildings, health care, industrial production and so on, which contains a wealth of useful information. Mining and mastering these hidden development laws, improving the quality of information, and improving the timeliness, accuracy and applicability of these three aspects have become the mainstream practice of using massive data at present.

Terminal devices in the Internet of Things will generate a large amount of data during operation. If all the data is uploaded to the cloud for processing, it will cause huge pressure on the cloud. In order to share the burden of the central cloud node and reduce the computing pressure, the edge node performs data calculation and storage within its own scope of responsibility. The calculation and processing capacity of edge nodes are not exactly the same, and there exists the situation that they cannot afford and complete the data mining analysis. These data that cannot be fully processed still need to be transferred from the edge node to the central cloud platform, where data analysis, mining and data sharing are carried out. Meanwhile, the algorithm model training and upgrading of the data are carried out. The upgraded algorithm is deployed to the edge node, so that the edge node can directly use the results of cloud processing, thus saving the process of data mining and statistics. Reduce data processing stress. Therefore, in order to meet the increasing demand for data processing, edge-cloud collaboration has become the best choice.

Due to the weak ability of edge to process data, all data cannot be transmitted to the cloud in the case of limited communication. It is found in the research of Tsinghua University that 99.9% of gradient exchange in distributed SGD is redundant. [1]Therefore, this paper will compress the gradient and transfer the data containing effective information to the cloud for model training.

Brief author introduction:CAO Yanyun(1997-),female,postgraduate， Intelligent big data analysis
Correspondance author: XU Peng(1977-),male,Associate professor,Cloud computing and big data. E-mail: xupeng@bupt.edu.cn

There are many differences between different users and physical devices, but the data collected from different edges have similar trends, and the gradient can describe this trend change. In this paper, gradients will be used for subsequent training and learning. But there is a problem in the process of gradient compression: in the process of gradient compression, some information must be lost. In order to pursue speed and timeliness, the existing compression algorithm, such as Top-k sparsity[2], is to find k elements with the largest absolute value in the gradient vector and upload them to the cloud. The remaining elements are accumulated locally. When the accumulation exceeds the given threshold, they are packaged and transmitted to the cloud platform. Although this approach solves some timeliness issues, it does not reduce the total amount of data uploaded.

To solve this problem, a reservoir sampling gradient compression method based on kernel density estimation is proposed. Compared with the previous gradient compression methods, this method introduces a new gradient compression idea, that is, adaptive kernel density estimation is added to the data sampling method. In this method, the gradient distribution is estimated, and a suitable kernel function is selected to estimate the kernel density, and the threshold of replacing the gradient strategy in the sampling algorithm is obtained. The gradient in the compression pool will eventually be output to the cloud in the form of a sequence for anomaly detection training.

1 Related Work

In general, when communication is limited, objective conditions do not support uploading all data, so the resource overhead in the communication process needs to be reduced. Take Federated learning as an example. Federated learning is a distributed machine learning framework that collaborates with the cloud and the edge for anomaly detection.[3] In the process of federated learning, there are usually two strategies to reduce the cost of communication resources. The first is to reduce the number of communication rounds in the training process, and the second is to reduce the amount of communication in each transmission. The most classic method to reduce communication rounds is FedAvg algorithm[4], which allows the edge to be updated locally for several rounds and then aggregated by the cloud platform. The advantage of this approach is that it can effectively reduce the number of communication rounds, but the disadvantage is that the edge must always be in a state of good network connection. In the scenario of this paper, such a condition is not satisfied.

The main purpose of the second strategy is to reduce the amount of data transferred. The gradient is compressed by quantization, sparsity and depth gradient compression. The idea of quantization is that elements are represented with low precision or mapped to a predefined set of code words to reduce the number of bits per element in the gradient tensor. [5-7] Depth gradient compression adopts four strategies: momentum correction, local gradient truncation, momentum factor hiding and preheating training. Researchers from Tsinghua University found that 99.9% of gradient switching in distributed random gradient descent training is redundant. [1] Therefore, the DGC they proposed requires top-k selection of the gradient, and the sparse coefficient of the given target is 99.9%.In addition, there are other studies using deep learning techniques for gradient compression. [8-10] Through neural network training, the communication bandwidth demand can be greatly reduced. The idea of sparse method is to upload only the gradients of important parts to update the global model. The method used to determine whether the gradient is important becomes the core focus of the method. Strom proposed using the size of the gradient to measure its importance by setting a threshold in advance and uploading it when the gradient is greater than the threshold. [11] However, in the actual situation, due to the great difference in the distribution of parameters of different network structures, we cannot choose the appropriate threshold value. In this paper, a reservoir sampling compression algorithm based on kernel density estimation is proposed, which is also a selective gradient to upload important parts. According to previous studies, most gradient exchanges are redundant. In order to improve the accuracy of anomaly detection on the central cloud platform, abnormal gradients should be highlighted as much as possible while normal gradients should be removed during sampling compression, so as to reduce data transmission.

2 Reservoir sampling algorithm based on kernel density estimation

Due to the weak ability of the edge to process data and the inability to transmit all data to the cloud under the condition of limited communication, we have to use data compression so that effective data can be transferred to the cloud for model training. The data of different users and physical devices have many differences which do not mean that they are abnormal. The data collected from different edges have similar trends, and the gradient can describe this trend change. It is easier to train a neural network to learn anomalies.

However, there exists an issue when compress gradient: In the process of gradient compression, some information is bound to be lost. In order to pursue speed and real-time performance, existed compression algorithms usually set a threshold and transmit gradients to the cloud in sections. Although this approach solves some of the timeliness problems, it does not reduce the total amount of uploaded data.

To solve this issue , we propose a reservoir sampling gradient compression method based on kernel density estimation. Compared with previous gradient compression, our method introduces a new idea of gradient compression. We add adaptive kernel density estimation into the data sampling method. This method estimates the data distribution of the gradient and selects the distribution with the best fit and selects a suitable kernel function for kernel density estimation to obtain a threshold. Threshold will be compared to the new data points. Based on the results, we select the different replacement strategies in the sampling algorithm. The gradients in the compression pool will eventually be output in the form of a sequence and sent to the cloud for training of anomaly detection.

2.1 Self-adaption kernel functions selection

The density estimation commonly used in statistics is divided into parametric estimation and non-parametric estimation. The density estimation commonly used in statistics is divided into parametric estimation and non-parametric estimation. Parameter estimation is divided into parameter regression analysis and parameter discriminant analysis. [12] In parametric regression analysis, it is assumed that the data distribution conforms to some specific law, such as linearity or exponential, and then specific solutions are found in the objective function to determine the unknown parameters in the regression model. In parametric discriminant analysis, one needs to assume that random data samples as the basis of discrimination obey a specific distribution in every possible category. However, in previous studies, there was often a large gap between the basic assumption of the parametric model and the actual physical model, and these methods could not always achieve satisfactory results. Since the collected data comes from different sensors, its overall probability density is unknown. However, the kernel density estimation method is not conducive to the prior data distribution, and does not make any assumptions about its distribution, which is more in line with the scene of real data analysis.

Assuming x_1, x_2, ... x_n are the N sample points generated by the independent identically distribution F, let f be its probability density, then the kernel density is estimated as:

$$\hat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} k\left(\frac{x-x_i}{h}\right) \tag{2-1}$$

Among them, the function k(x) is the kernel function, which satisfies the following equation:

$$\int k(x)dx = 1 \tag{2-2}$$

The estimate uses the distance from x_i to x to determine the role of the density at estimating point x. Among them, h represents the bandwidth. The smaller h is, the closer the point to x is to have an impact on the density of x.

In the data stream, there are both normal data and abnormal data. From the perspective of density estimation, when there are a lot of data points around a data point, it means that the data point is consistent with most of the data, that is, normal data. Therefore, a high-density cluster can be regarded as normal data. You can view low-density clusters as abnormal data. To make a rough distinction between the two, the separation is likely to be non-linear for real data generated in real scenarios. The purpose of using the kernel function idea is to change the possible nonlinearity into linearity, so as to make the later work easier.

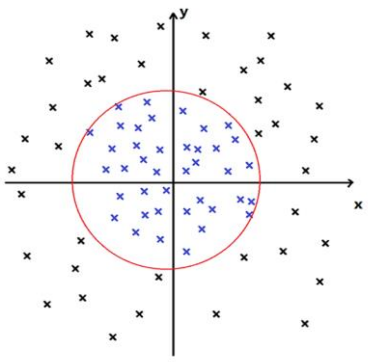

Fig. 1 Raw data density partition

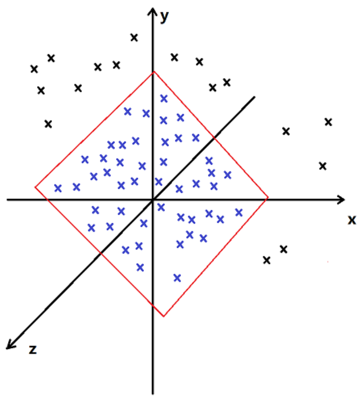

Fig. 2 Data density division after kernel transformation

Suppose that the original two-dimensional space R2 is changed into three-dimensional space R3, namely:

$$\Phi: (x_1, x_2) \rightarrow (z_1, z_2, z_3) = (x_1^2, \sqrt[2]{2}x_1x_2, x_2^2) \tag{2-3}$$

Taking Figure 1 as an example, in the original space, the dividing line separating the two types of data can be expressed as:

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \tag{2-4}$$

Transform to three-dimensional space through point-to-point mapping:

$$\frac{x_1^2}{a^2}+\frac{x_2^2}{b^2}=1 \rightarrow \frac{1}{a^2}\cdot z_1+0\cdot z_2+\frac{1}{b^2}\cdot z_3=1 \tag{2-5}$$

At this time, it can be found that in the three-dimensional space, the original elliptic dividing line becomes a plane dividing plane. By calculating the inner product:

$$\begin{aligned} &< \Phi(x_1,x_2),\Phi(x_1',x_2') > \qquad (2\text{-}6)\\ &= < (z_1,z_2,z_3),(z_1',z_2',z_3') > \\ &= < (x_1^2,\sqrt[2]{2}x_1x_2,x_2^2),(x_1'^2,\sqrt[2]{2}x_1'x_2',x_2'^2) > \\ &= \ x_1^2x_1'^2+2x_1x_2x_1'x_2'+x_2^2x_2'^2 \\ &= (x_1x_1'+x_2x_2')^2 \\ &= (< x,x' >)^2 = k(x,x') \end{aligned}$$

$k(x,x')$ is the kernel function, and it turns out that going from a lower dimensional space to a higher dimensional space has nothing to do with Phi, it just needs the kernel function. As you can see, in the raw data, if you want to separate the two types of data, you have to separate them by an ellipse, which means it's not linearly separable; But by mapping low-dimensional data to higher-dimensional data through kernel changes, the data is easily classified through a hyperplane, that is, the data becomes linearly divisible in higher-dimensional space.

Secondly，in order to match the diversity of data better, we provide three kinds of kernel functions, which have a large gap about their fit function curve, becoming the result of adaptive selection. They are Gaussian kernel, Sigmoid kernel, and Epanechnikov kernel.

The expression for the Gaussian kernel is:

$$k_{gau}(x,y) = \ exp(-\frac{1}{2\sigma^2}\parallel x-y\parallel^2) \tag{2-7}$$

The expression for the Exponential kernel is:

$$k_{exp}(x) = \ exp(-\frac{1}{2\sigma^2}\parallel x-y\parallel) \tag{2-8}$$

The expression for the Epanechnikov kernel is:

$$k_{epa}(x) = \ \frac{3}{4}(1-x^2) \tag{2-9}$$

Thirdly， in order to measure which dose curve the data more closely resembles, we compute the total log-likelihood under the model.

$$\text{SSR} = \ \sum_{i=1}^{n}(y_i-\hat{y_i})^2 \tag{2-10}$$

We calculate the difference between the data points and the corresponding positions of the fitted curve, and approximate the collected discrete data with analytical expressions. Then, calculate the residual sum of squares of the fitted curves with the three kernel functions, and select the fitting curve with the smallest residual sum of squares as the selected kernel function. This adaptive selection method can improve the validity, unbiasedness and consistency of density estimation.

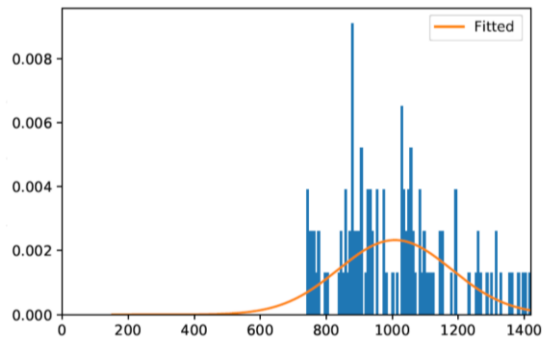

Fig. 3 Kernel function selection example

2.2 Replacement policy in reservoir sampling gradient compression

The reservoir algorithm is a classic sampling algorithm. The algorithm draws samples without replacement from a data stream of unknown size and guarantees that each sample is drawn with the same probability. [13] However, in the scenario of this paper, abnormal data accounts for a very small proportion of the data stream. In order to amplify abnormal data features, while ensuring that the generated data summaries correctly reflect the overall sample, our reservoir algorithm has been improved.

Based on the choice of kernel function, we already know the approximate distribution of the data flow. A threshold is determined based on the kernel density estimate to separate the data into high-density clusters and low-density clusters. High-density clusters are normal data, and low-density clusters are abnormal data.

First, when new data arrives, we compute the relative density of that data and classify it as either high-density clusters or low-density clusters.

Second, use different replacement strategies depending on the category you belong to. This replacement strategy will directly affect the final result of the algorithm.

The improved algorithm flow and related definitions are as follows:

Let $d(p,q)$ be the distance between point p and q.

If C is a cluster , let $d(p,C)$ be the shortest distance in the C.

$$d(p,C) = \min\{\ d(p,q) | \ q \in C\} \tag{2-11}$$

$\forall\ k \in\ N^+,\ d_k(p)$ express the distance-k^- of p

Given a distance equal to k of p, the k-nearest neighbor field N_{k-p} of p contains all objects whose distance are no more than k away from p:

$$N_{k-p}(p) = \{q \in C\backslash\{p\}\ |\ d(p,q) \le k - p\} \tag{2-12}$$

And we call q as p's k-nearest neighbors

k ∈ n^+, we call $EM(p,o)$ as effective maximum distance of p relative to o.

$$EM(p,o) = \max\{\ d(p,o), k\} \tag{2-13}$$

Especially, if the distance between p and o is too close and exceeds a certain value, we define its distance as k. We use this method to reduce statistical fluctuations.

The part density of an object p, which we call ρ, can be expressed as the number of objects within the effective maximum distance of p.

$$\rho_p = \frac{|N_{k-p}(p)|}{\sum_{o \in N_{k-p}(p)} EM(p,o)} \tag{2-14}$$

Then, the relative density between object p and object o can be expressed as:

$$\rho_{p-o} = \frac{\rho_p}{\rho_o} \tag{2-15}$$

The improved reservoir sampling data compression method is as follows:

Step 1: Insert e_1, e_2, ..., e_r elements to initialize the compression pool r.

Step 2: Calculate the SSR of the data in the compressed pool with the three fit function curves. Select the fit function with the smallest SSR as the kernel function.

Step 3: Substitute the kernel function into the kernel density estimation formula for calculation to obtain a threshold δ. Divide the data into high-density clusters and low-density clusters.

Step 4: When there is new data, calculate the relative distance between the point and the data center of the compression pool. The closer the ρ_{p-o} value is to 1, the closer the p point density is to the o point, and these two points belong to the same cluster. When the ratio is smaller, it means that the point p may be an abnormal point. If the new data belongs to a high-density set, use random sampling to enter the compression pool; if it belongs to a low-density set, with normalized $p = \frac{1}{\rho_p}$ as the sampling probability, calculate the weight, which represents the probability that the gradient is drawn and replaced into the compression pool.

Step 5: Send the final compressed pool data to the cloud.

Algorithm 1 : Gradient Compression : Based on Kernel Density Estimation

```
Input: initial data set I, initial compressed pool $C^I$, new data set X | $x_i \in$ X, density threshold δ, random sampling probability $p$, compression pool size $l$
Output: final compressed pool $C^F$
1  $\mathrm{SSR}_g = \sum_{i=1}^{n}(k_g(x_i) - \hat{y}_i)^2$
2  $\mathrm{SSR}_s = \sum_{i=1}^{n}(k_s(x_i) - \hat{y}_i)^2$
3  $\mathrm{SSR}_e = \sum_{i=1}^{n}(k_e(x_i) - \hat{y}_i)^2$
4  $k(x) = \mathrm{argmin}\{\mathrm{SSR}_g, \mathrm{SSR}_s, \mathrm{SSR}_e\}$
5  // Divide data with density estimation
6  If $\hat{f}_h(x) \geq \delta$ then
7      $x_i$ is randomly sampled according to $p$
8      put the results into $C^I$.
9      for the size of $C^I > l$
10         remove samples marked as high density from $C^I$
11     end
12 else if $\hat{f}_h(x) < \delta$ then
13     $x_i$ is sampled according to $p$ = normalized $\frac{1}{\rho_p}$.
14     put the results into $C^I$
15     for the size of $C^I > l$
16         remove samples marked as high density from $C^I$
17     end
18 End
19 $C^F$ = updated $C^I$
```

2.3 Negative feedback mechanism for edge-cloud collaboration

The reservoir sampling algorithm based on kernel density estimation is located at the edge node, and its goal is to compress the uploaded data. The neural network training algorithm is located in the cloud, and its goal is to train a model using compressed uploaded data. After the model is obtained by the cloud platform, the model parameters are sent to the edge gateway. The edge end establishes the model through the global parameters, and uses the model to detect data anomalies and alarm the abnormal values. In an ideal case, the model can be used all the time. However, the reality is that whether due to equipment aging or environmental changes, the data trend changes, then the anomaly detection model should also change with the change, and the model should be updated and iterated in time.

In order to solve the above problems, this paper proposes a negative feedback mechanism with dynamic weight adjustment as optimization. According to the results of anomaly detection and new data input, the model is updated. After receiving the parameters of the anomaly detection model trained by the cloud, the edge terminal generates the anomaly detection model locally, detects the anomaly of the data stored in the database, and alarms the anomaly. In the previous chapters, the reservoir sampling algorithm based on kernel density estimation is introduced. In the process of compression, the sampling probability is assigned to the data according to its relative density. After the completion of anomaly detection, for frequent anomaly types, the gradient generated when the anomaly data is compressed is traced back at the edge end. According to the anomaly detection results, the sampling probability weight influence factor is calculated to reduce the weight of the relevant gradient, so as to reduce the probability of this kind of data being replaced into the compression pool.

The edge end receives the anomaly detection model parameters, loads the model, and uses the model to perform anomaly detection on the data. Get results $Y = [Y_1, Y_2, Y_3 \cdots, Y_n]$, $Y_1, Y_2, Y_3 \cdots, Y_n$ said different categories of abnormal data. The results of $Y_1, Y_2, Y_3 \cdots, Y_n$ would be quantified, and it can get $N = [N_1, N_2, N_3, \cdots, N_n]$, $N_1, N_2, N_3 \cdots, N_n$ means $Y_1, Y_2, Y_3, \cdots, Y_n$ appears the number of times.

Let $\overline{N} = \frac{N_1+N_2+N_3+\cdots+N_n}{n}$, $N_k = N_{max}$;

Correction:

$$t = \frac{N_k - \overline{N}}{N_1+N_2+N_3+\cdots+N_n} \quad (2\text{-}16)$$

Updated sampling probability:

$$p' = t \cdot p \quad (2\text{-}17)$$

At this time, the edge side uses the updated probability to sample and compress the gradient, and the probability of Y_k type anomalies being transmitted to the cloud is reduced, which reduces the repetitive work. The cloud receives the data after gradient compression, which significantly reduces the communication traffic and ensures the accuracy of anomaly detection training. The edge side uses the model generated by the cloud to adjust its own gradient compression process while detecting data anomalies, which better reflects the idea of cloud-edge collaboration.

3 Experiment and Results

3.1 Experiment

In this paper, the effect of gradient compression on edge end in communication constrained scenario is verified by experiments. The experiment includes the comparison between the gradient compression algorithm designed in this paper and other algorithms, as well as the comparison between the selection of different parameters and the results of the algorithm in the longitudinal dimension. We evaluate our algorithm on five real world time series with actual anomaly events which called Numenta anomaly benchmark (NAB), covering office ambient temperature, CPU usage on Amazon Web Services (AWS) and Amazon East Coast data center servers. The internal temperature of industrial machines and the number of taxi riders in New York City.

There are two kinds of experiments designed in this paper. Firstly, use the compression algorithm proposed in this paper and the compression algorithm in related studies to compress the data set and compare the effect; second, adjust the parameters of the algorithm in this paper, and compare the effect of the algorithm proposed in this paper under different parameter Settings. In the experiment, in order to facilitate the evaluation of the algorithm's performance and the comparison with other algorithms, the experimental indicators in this paper refer to the methods in the literature, and the representability of anomalies and calculation time are used to evaluate the algorithm in this paper. For example, if there are 100 abnormal data samples in the data set and 10 abnormal data are included in the generated profile, the summary has an exception representability of 10%.The representability of the exception is that the index data set consists of A abnormal data samples, and the generated data summary contains a abnormal data samples, so the representability of the exception is $\frac{a}{A} * 100\%$. When the representability is higher, the generated data profile is more suitable for anomaly detection tasks. The computation time is the running time of the algorithm.

Firstly, the SSE of data is calculated by three fitting function curves. The one with the smallest selection value is taken as the selection kernel function, and the kernel function is substituted into the kernel density estimation formula for calculation, and the threshold value δ is obtained. The data can be classified into high density and low density categories based on the threshold value δ. In this case, when new data is incoming, the relative distance between this point and the data center of the compression pool is calculated. If the relative distance is small, the new data is low-density, while the relative distance is large, the new data is high-density. If the new data belongs to the high-density set, random sampling is used to enter the compression pool. If it belongs to the low density set, $p = \frac{1}{\rho_p}$ is taken as the sampling probability and the weight is calculated. The weight represents the probability that this point is replaced into the compression pool.

3.1.1 Adjust parameter

Table 1 and 2 shows the anomaly representability and computation time of exceptions obtained from different data sets when taking compression pools of different sizes. Where L represents the ratio of the compressed pool to the original data.

Tab. 1 The anomaly representability results base on different L

Anomaly Representability	L=5%	L=10%	L=15%	L=20%
Dataset: Ambient temperature	45.8%	63.7%	88.4%	95.5%
Dataset: CPU utilization AWS	25.4%	46.7%	86.3%	89.1%
Dataset: CPU utilization EC2	44.8%	65.7%	89.4%	97.1%
Dataset: Machine temperature	56.9%	76.7%	83.5%	96.5%
Dataset: NYC taxi	36.8%	63.7%	75.4%	96.8%

Tab. 2 The computation time results base on different L

Computation Time(MS)	L=5%	L=10%	L=15%	L=20%
Dataset: Ambient temperature	83	106	94	92
Dataset: CPU utilization AWS	67	86	84	89
Dataset: CPU utilization EC2	78	111	97	102
Dataset: Machine temperature	85	136	122	127
Dataset: NYC taxi	44	86	104	99

3.1.2 Compare with other algorithms

For our compression algorithm, it is based on reservoir sampling algorithm to carry out. Therefore, the algorithms used in comparison experiments here are all compression algorithms using sampling methods. SRS [14] represents the classical reservoir sampling algorithm. PSSR [15] sampling and MSSR [15] sampling respectively represent the pair distance and center point methods proposed in literature [15], while RKDE represents the sampling algorithm proposed in this paper. Table 3 and Table 4 show the results of anomaly representability and calculation time obtained after data compression using different sampling methods.

Tab. 3 The anomaly representability results base on different algorithm

Anomaly Representability	RKDE	SRS	PSSR	MSSR
Dataset: Ambient temperature	88.4%	4.76%	68.4%	45.9%
Dataset: CPU utilization AWS	86.3%	4.70%	59.66%	57.12%
Dataset: CPU utilization EC2	89.4%	5.87%	46.67%	23.84%
Dataset: Machine temperature	83.5%	3.91%	55.96%	68.74%
Dataset: NYC taxi	88.4%	6.43%	75.4%	38.54%

Tab. 4 The computation time results base on different algorithm

Computation Time(MS)	RKDE	SRS	PSSR	MSSR
Dataset: Ambient temperature	94	42	133	592

Dataset: CPU utilization AWS	84	33	95	423
Dataset: CPU utilization EC2	97	56	102	532
Dataset: Machine temperature	122	67	118	637
Dataset: NYC taxi	104	35	138	742

3.2 Results

Through the above experiments, experimental results are obtained in Table 1,2,3,4. According to the results shown in Table 1 and Table 2, the following conclusions can be drawn: when only abnormal representability is considered, the size L of the compression pool is positively correlated with abnormal representability. The larger L is, the more outliers it contains, and the smaller L is, the fewer outliers it contains. This is in line with our expectations. If we consider only the operation time, we find that the inflection point occurs when L=10%, and the operation time is not positively correlated with L. The reason for this situation can be analyzed as follows: when L is small, there are fewer operation cycles, and the complexity of density calculation is low, so less time is spent. However, when L is larger, such as L=15% or L=20%, the probability of being sampled in the operation process becomes larger, so the complexity becomes lower. Based on the results of Table 1 and Table 2, in the subsequent research, L=15% was used for compression, and both time and anomaly representability were taken into account, and relatively good results were obtained.

According to the results shown in Table 3 and Table 4, it can be found that in the index comparison of anomaly representability, the classical reservoir algorithm SRS cannot cover enough abnormal data, and its anomaly representability is low. The results of PSSR sampling and MSSR sampling are not the same as the RKDE sampling proposed in this paper. In terms of operation time, the classical reservoir algorithm SRS is the fastest because of its low complexity, while RKDE sampling is similar to PSSR sampling algorithm in terms of time efficiency.

4 Conclusion

In this paper , a reservoir algorithm based on kernel density estimation is proposed for edge cloud collaborative anomaly detection in communication constrained scenarios. The purpose of this algorithm is to reduce the amount of data uploaded by the edge to the central cloud platform. The main approach is to firstly classify the data by kernel density estimation; secondly, density estimation is carried out on the continuously reached time series data, and the density classification is taken as the sampling probability to generate a sampling compression pool and upload it to the central cloud platform for neural network training. RKDE can not only compress the amount of data uploaded, but also adjust the negative feedback mechanism. In this way, data uploaded to the cloud can be updated to improve the accuracy of anomaly detection. In this paper, the advantages of RKDE algorithm and its rationality are verified by parameter tuning and comparison experiments.

References

[1] Lin Y, Han S, Mao H, et al. Deep gradient compression: Reducing the communication bandwidth for distributed training[J]. arXiv preprint arXiv:1712.01887, 2017.
[2] Aji A F, Heafield K. Sparse communication for distributed gradient descent[J]. arXiv preprint arXiv:1704.05021, 2017.
[3] Li T, Sahu A K, Talwalkar A, et al. Federated learning: Challenges, methods, and future directions[J]. IEEE signal processing magazine, 2020, 37(3): 50-60.
[4] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.
[5] Dettmers T. 8-bit approximations for parallelism in deep learning[J]. arXiv preprint arXiv:1511.04561, 2015.
[6] Bernstein J, Wang Y X, Azizzadenesheli K, et al. signSGD: Compressed optimisation for non-convex problems[C]//International Conference on Machine Learning. PMLR, 2018: 560-569.

[7] Karimireddy S P, Rebjock Q, Stich S, et al. Error feedback fixes signsgd and other gradient compression schemes[C]//International Conference on Machine Learning. PMLR, 2019: 3252-3261.
[8] Stich S U, Cordonnier J B, Jaggi M. Sparsified SGD with memory[J]. Advances in Neural Information Processing Systems, 2018, 31.
[9] Rothchild D, Panda A, Ullah E, et al. Fetchsgd: Communication-efficient federated learning with sketching[C]//International Conference on Machine Learning. PMLR, 2020: 8253-8265.
[10] Haddadpour F, Kamani M M, Mokhtari A, et al. Federated learning with compression: Unified analysis and sharp guarantees[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2021: 2350-2358.
[11] Ström N. Scalable distributed DNN training using commodity GPU cloud computing[J]. 2015.
[12] Kamalov F. Kernel density estimation based sampling for imbalanced class distribution[J]. Information Sciences, 2020, 512: 1192-1201.
[13] Mahmud M S, Huang J Z, Salloum S, et al. A survey of data partitioning and sampling methods to support big data analysis[J]. Big Data Mining and Analytics, 2020, 3(2): 85-101.
[14] Kerdprasop K, Kerdprasop N, Sattayatham P. Density-biased clustering based on reservoir sampling[C]//16th International Workshop on Database and Expert Systems Applications (DEXA'05). IEEE, 2005: 1122-1126.
[15] Ahmed M. Reservoir-based network traffic stream summarization for anomaly detection[J]. Pattern Analysis and Applications, 2018, 21(2): 579-599.

基于核密度估计的蓄水池时序数据抽样压缩

曹严匀，徐鹏

（北京邮电大学计算机学院（国家示范性软件学院），北京 100876）

摘要：边云协同的异常检测成为了当今最主要的异常检测架构。然而，只有在最理想的状态下，中心云平台才可以利用充分的数据进行充分的训练。在通信受限的情况下，不得不考虑减少通信资源的使用，但依旧保持高精确率的异常检测。本文在此背景下，提出了基于核密度估计的蓄水池抽样算法——RKDE，来减少边缘端上传至云端的数据量。通过对提高异常数据被抽样的概率，来构建上传的压缩池，减少梯度交换中的冗余过程，并依据云端异常检测的结果，对抽样过程及时反馈调整。同时，将 RKDE 与多个抽样算法进行对比，展示其性能优势。

关键词：核密度估计；梯度压缩；蓄水池算法
中图分类号：TP391