A Speed-up K-Nearest Neighbor Classification Algorithm for Trojan Detection

Tianshuang Li[1](✉), Xiang Ji[2], and Jingmei Li[1]

[1] College of Computer Science and Technology,
Harbin Engineering University, Harbin 150001, China
447419029@qq.com
[2] The 73rd Institute of China Shipbuilding Industry Corporation,
Zhengzhou 450000, China

Abstract. Aiming at the problem that the traditional K-nearest neighbor algorithm has a long classification time when predicting Trojan sample categories, this paper proposes a speed-up K-nearest neighbor classification algorithm CBBFKNN for Trojan detection. This method adopts the idea of rectangular partitioning to reduce the dimensionality of the sample data. Combining the simulated annealing algorithm and the Kmeans algorithm, the sample set is compressed and the BBF algorithm is used to quickly classify the sample. The experimental results show that, the CBBFKNN classification algorithm can effectively reduce the classification time while the precision loss is small in IRIS dataset. In terms of Trojan detection, the CBBFKNN classification algorithm can guarantee higher accuracy and lower misjudgment rate and lower missed detection rate in shorter detection time.

Keywords: K-nearest neighbor algorithm · Kmeans algorithm · Trojan detection

1 Introduction

In recent years, with the progress of science and technology, there have been frequent threats to the global Internet network security, among which the Troy Trojan-horse has performed abnormally [1]. Research on Trojan detection technology cannot be delayed. Most of the communication data obtained by computers in the real world is characterized by large quantities and irregularities. How to quickly classify these communication data and determine whether it is a Trojan behavior will become the main content of the current study. The most basic classification methods are the K-nearest neighbor algorithm. The K-nearest neighbor algorithm was first proposed by Cover [2] and HART in 1967. The unclassified samples were assigned to the recent classification category according to the rules. This algorithm is a kind of supervised learning algorithm, and it has the advantages of low demand for data distribution, strong adaptability and high precision. However, it also has the disadvantages of slow convergence, poor processing of small sample and poor processing for high dimensional data. The improvement of K-nearest neighbor algorithm and its application in Trojan detection is a worthy research topic.

S. Liu and G. Yang (Eds.): ADHIP 2018, LNICST 279, pp. 214–224, 2019.
https://doi.org/10.1007/978-3-030-19086-6_24

At present, aiming at the problem of low classification efficiency of traditional K-nearest neighbor algorithm in large-scale training set, Ren [3] proposed a clustering-based accelerated K-nearest neighbor classification method. However, they neglected the similarity between the training sample and the unclassified samples. In the similarity calculation, the training samples were not weighted. The accuracy of the algorithm needs to be improved. Zuo [4] proposed an evaluation function based on improved simulated annealing Kmeans algorithm to accept poor solutions with the certain probability, but its classification effect on large-scale data was not good. Hu [5] proposed two improved K-nearest neighbors algorithms, which was an improved K-nearest neighbors algorithm based on hypersphere region partition and an improved K-nearest neighbors algorithms based on cuboid region partition. However, the improved K-nearest neighbor algorithm based on the hypersphere region partition was not suitable for datasets with large amounts of data and it had a long initial classifier construction time. The improved K-nearest neighbor algorithm based on the cuboid region partition had a short construction time but its accuracy was not high. Yao [6] proposed an improved Kmeans algorithm combed the simulated annealing clustering algorithm. The exchange mechanism of simulated annealing algorithm was used to replace the exchange mechanism of Kmeans center point to reduce the occurrence of local optimal solution. However, the mechanism of annealing was to accept disturbances with the certain probability, the probability was in the form of a random number. So it was difficult to control the results of the disturbance, and it needed to be improved.

Aiming at the problems of high time complexity [7–9] and low classification efficiency of the traditional K-nearest neighbor algorithm, this paper proposes a speed-up K-nearest neighbor classification method CBBFKNN (Clustering Best Bin First K-Nearest Neighbor) for Trojan detection. This method first uses the idea of rectangular partitioning to reduce the dimension of training samples and determine an optimal dimension. Then the Kmeans algorithm is used to cluster the samples in the training sample. In order to avoid the Kmeans algorithm from falling into the local optimal solution, the simulated annealing algorithm is used to obtain the optimal clustering set. Use the BBF (Best Bin First) algorithm to query the nearest K samples of the unclassified sample, and according to the principle of the minority obeying the majority to determine the category of the unclassified sample.

2 Speed-up K-Nearest Neighbor Classification Algorithm

The traditional K-nearest neighbor algorithm cannot efficiently calculate the similarity between the unclassified sample and a large number of training samples. This paper presents a speed-up K-nearest neighbor classification algorithm CBBFKNN.

The speed-up K-nearest neighbor classification algorithm CBBFKNN is performed as follows.

Step 1: Initialization. Assume that the labeled training sample set is $P_a = \{(x_i, y_i)\}_{i=1}^{N}$, $x_i \in X \subseteq R^h$ represents the eigenvector of the sample, R^h represents the h-dimensional space. N represents the number of training samples, x_i represents the training samples. The sample category for sample x_i is y_i, $y_i \in Y = \{c_i\}_{i=1}^{t}$, t represents the number of categories. The unclassified sample set is $P_b = \left\{\left(tx_j, ty_j\right)\right\}_{j=1}^{M}$, M represents the number of unclassified samples, tx_j represents the unclassified sample, ty_j represents the category of unclassified sample.

Step 2: Determine the division dimension. If we calculate each dimension, it will consume a lot of memories and time. So we first reduce the dimension of the samples. Assume that the initial samples dimension is h, the optimal dimension to be calculated is h'. The number of intervals to be divided in each dimension is d.

Step 2.1: Assume that the distribution of the samples is uniform, and the average number of samples in each interval in each dimension is $avg = P_a/d$. The initial value of the variable l is 1. The optimal dimension opt is initially 2.

Step 2.2: According to the sample distribution, count the actual number of training samples $q_1, q_2, \ldots, q_d$ in each interval.

Step 2.3: Calculate the distribution evaluation function W on each dimension, The W calculation formula is shown in formula (1).

$$W = \sum_{l=1}^{h} (q_l - avg)/d \tag{1}$$

Step 2.4: The calculation of α is used to determine the optimal dimension. The calculated value is stored in opt and the newly calculated α value is compared with opt. If it is bigger than opt, it is stored in opt, otherwise it is stored in opt with a probability of 0.5. The α calculation formula is shown in formula (2).

$$\alpha = \frac{1}{W^2} \tag{2}$$

Step 2.5: Set $l = l + 1$.

Step 2.6: If $l > h$, the algorithm skip to Step 2.7, otherwise the algorithm skip to Step 2.2.

Step 2.7: We output the optimal dimension opt.

Step 3: Combine the simulated annealing algorithm and the Kmeans algorithm to compress the training dataset.

Step 3.1: Select t training samples randomly as the category center of the initial t category, the initial category set is $S = \{s_c\}_{c=1}^{t}$.

Step 3.2: Calculate the distance of each training sample x_i from each category center s_c and assign the unclassified sample to the nearest category. The formula is shown in formula (3).

$$D(s_c) = \sum_{l=1}^{h'} \left(x_i^{(l)} - s_c^{(l)}\right)^2 \tag{3}$$

Step 3.3: Recalculate the center of each category. The formula is as shown in formula (4).

$$s_c^{(l)} = \frac{\sum_{i=1}^{r_c} x_i^{(l)}}{r_c} \tag{4}$$

$s_c^{(l)}$ represents the l-dimensional attribute of the category c center, r_c represents the number of training samples in category c.

Step 3.4: Take the clustering result of Step 3.3 as the initial solution s_c of the simulated annealing algorithm. The variable i is initially 0. The number of iterations is t. The objective function F is defined as shown in formula (5). The initial temperature value T is set to 10.

$$F_s = \sum_{i=1}^{r_c} \left| x_i^{(l)} - s_c^{(l)} \right| \tag{5}$$

Step 3.5: The perturbation method is to randomly change the category of a sample. The new objective function F_s' is calculated by formula (5).

Step 3.6: Calculate the value of ΔF by $\Delta F = F_s' - F_s$. Set $i = i + 1$.

Step 3.7: If $\Delta F \leq 0$, the algorithm skip to Step 3.8, otherwise the algorithm skip to Step 3.9.

Step 3.8: Accept the new solution and assign s' to s. Assign F_s' to F_s.

Step 3.9: According to Metropolis guidelines, new solutions are accepted with probability p. The formula for the probability p is shown in formula (6).

$$p = \left(e^{-\frac{\Delta F}{2T}}\right) \tag{6}$$

Step 3.10: If $i = t$, decrease the temperature T by 1° and the algorithm skip to Step 3.11, otherwise the algorithm skip to Step 3.5.

Step 3.11: It is determined whether the temperature value T is 0 or not. If T is 0, skip to Step 4. Otherwise set i to 0, skip to Step 3.5.

Step 4: Obtaining the optimal clustering result and the clustering result is $S = \{s_c\}_{c=1}^{t}$.

Step 5: In the set S, m samples are selected at random, and m samples are sorted using quick sort. Determine the median x_{avg}.

Step 6: Using the BBF algorithm to find the nearest k samples from the unclassified sample and determine the category of the unclassified sample. The median x_{avg} is the split plane of the BBF algorithm.

Step 6.1: The kd-tree is constructed with the median x_{avg} as the root node.

Step 6.2: Query the unclassified sample tx_j on the kd-tree. During the query, the distance d_i between the sample x_i and the median x_{avg} on the query path is calculated. Sort the calculated distance d_i.

Step 6.3: Backtracking checks the sample x_i with the highest priority. The priority is inversely proportional to d_i. Finally determine the k samples closest to the unclassified sample tx_j, and k samples make up the set $N_K(x)$.

According to the principle of the minority obeying the majority formula (7) is used to determine the category y_j of the unclassified sample tx_j.

$$ty_j = \arg\max_{c_j} \sum_{x_i \in N_k(x)} I(f(x_i) = c_j) \tag{7}$$

In the formula (7), I represent the indicator function, $y_i = f(x_i)$. I will be 1 when $y_i = c_j$, otherwise I will be 0.

Step 7: Repeat performs Step 6.1–Step 6.3 for each sample in the unclassified sample set P_b. The prediction category of each unclassified sample in the sample set P_b is compared with the real category of the unclassified sample, so as to test the performance of the classification algorithm.

Step 8: The algorithm ends.

3 Experimental Design and Analysis

The experiments in this paper are divided into two parts. The first part is the characteristic detection experiment of Iris based on IRIS dataset to verify the accuracy and efficiency of the proposed algorithm CBBFKNN. The second part is the detection experiment based on Trojan horse behavior sample. The algorithm CBBFKNN proposed in this paper is applied in Trojan detection aiming to verify the accuracy of the algorithm.

3.1 Characteristic Detection Experiment Based on IRIS DataSet

In order to verify the performance of the proposed speed-up K-nearest neighbor classification algorithm CBBFKNN, this paper classifies the characteristic of Iris by using CBBFKNN. The algorithm performance is compared with SVM, Naive Bayes and traditional K-nearest neighbor algorithm aiming to verify the feasibility of the classification algorithm which is proposed in this paper.

Experimental Environment. The experimental platform is Windows 10 64-bit operating system, pycharm software platform, Intel I5 CPU, 8G RAM.

This experiment uses the international standard dataset IRIS. Specific information is shown in Table 1.

Table 1. The information of IRIS dataset

Data sources	Data dimension	Number of samples	Number of categories
Iris characteristics	4	150	3

The IRIS dataset contains three species of Iris virginica, versicolor, and setosa. The four attributes of the flower are the sepal length, sepal width, petal length, and petal width.

In this experiment, the IRIS dataset was divided into two parts: the training set and the test set. Taken 70% of the IRIS dataset, 105 groups of data as the training set, and the other 45 groups of data are taken as the test set.

Experimental Parameter Configuration. The CBBFKNN classification algorithm proposed in this paper needs to determine the parameter *k* and the clustering parameter *t*. Take the clustering parameter *t* for 4 temporarily. Figure 1 shows the classification time curve of the classification algorithm when the clustering parameter *t* takes 4 and the parameter *k* takes different values. When the value of *k* is taken as 9, the classification time approaches the minimum value.

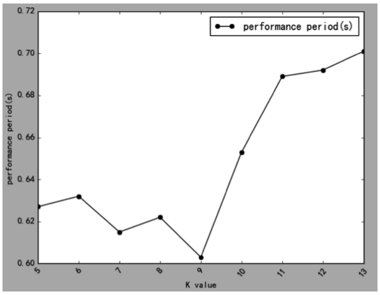

Fig. 1. The classification time curve of parameter k with different values

Figure 2 shows the classification accuracy curve of the classification algorithm when the clustering parameter *t* takes 4 and the parameter *k* takes different values. When the value of *k* is taken as 8, the classification time approaches the minimum value. Balanced the classification time and the accuracy, this paper takes the value of classification parameter *k* is 9.

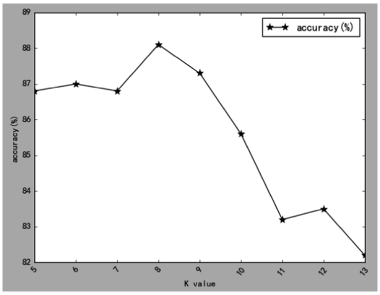

Fig. 2. The classification accuracy curve of parameter k with different values

The value of the classification parameter *k* is taken as 9, and the initial clustering parameters *t* are taken as 2, 3, 4, 5, and 6 respectively. Figure 3 shows the classification time curve of the classification algorithm when clustering parameter *t* takes different values. It shows that the classification time fluctuates little when the values of the parameters *t* are different.

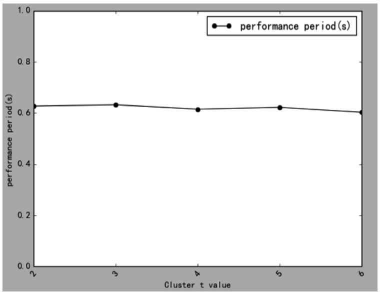

Fig. 3. The classification time curve of parameter t with different values

Figure 4 shows the classification accuracy curve of the classification algorithm when the clustering parameter *t* takes different values. When the value of the clustering parameter *t* takes 5, the loss of precision is small. Balanced the classification time and the accuracy, this paper takes the value of clustering parameter *t* is 5.

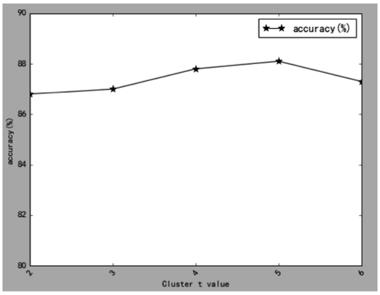

Fig. 4. The classification accuracy curve of parameter t with different values

Experimental Results. Use the CBBFKNN algorithm proposed in this paper to classify samples in the IRIS dataset and compare the algorithm performance with SVM, Naive Bayes and traditional K-nearest neighbor algorithm. The algorithm performance comparison results are shown in Table 2.

Table 2. Algorithm performance comparison table based on IRIS dataset

Algorithm	Classification accuracy/%	Classification time/s
SVM	91.8	0.932
Naive Bayes	92.5	0.853
K-nearest neighbor	93.9	0.884
CBBFKNN	93.6	0.842

It can be seen from Table 2 that the accuracy of the speed-up K-nearest neighbor classification algorithm CBBFKNN proposed in this paper is closest to that of the traditional K-nearest neighbor classification algorithm. CBBFKNN reduces the time for classification when the loss of accuracy is small.

3.2 Detection Experiment Based on Trojan Behavior Samples

In order to verify the effectiveness and practicability of the proposed speed-up K-nearest neighbor algorithm CBBFKNN in Trojan detection, this paper sets up the following experimental environment and tests the detection effect of the classification algorithm.

Experimental Environment. In order to ensure the authenticity of the data, the Trojan behavior sample in this paper is obtained in the following environment. The test environment consists of 18 hosts, of which 15 are operating normally on the Internet. Selecting 8 hosts that are surfing on the Internet to implant Trojans and extra choose one control terminal to generate Trojan corresponding traffic. Use common Trojans such as Glaciers, Broad Girls, Grey Pigeons, etc. for testing.

Sample Behavior Characteristics. Trojan communication and normal network communication have certain differences, so analyzing the Trojan's behavioral characteristics can effectively detect Trojans. The Trojan's communication process is divided into three phases: building connections, maintaining connections, and connecting interactions. (1) During the phases of building connections, Trojans will perform multiple DNS connection requests to the controller in a short period of time. (2) During the phases of maintaining connections, in order to maintain communication, the communication parties regularly send a large number of heartbeat packets. (3) During the phases of connecting interactions, a long connection is required. If the control terminal cannot communicate normally, the controlled terminal does not disconnect directly. They will send a large number of SYN packets to determine if the console is alive [10]. According to the above three phases of Trojan communication, this paper selects four characteristics to describe Trojan behavior. The characteristics are shown in Table 3.

Table 3. Behavioral characteristics table

Phases	Characteristic name
Building connections	Multiple DNS connection requests in a short time
Maintaining connections	The ratio of packets to total packets
Maintaining connections	Heartbeat packet time gap
Connecting interactions	The ratio of SYN packets to total packets

In the actual collection process, it is impossible to directly distinguish the phases of the characteristic generation, so the characteristic generation phase is not distinguished during the collection process.

Sample Collection Process

(1) We install VMWare Workstation 6.0 software for each computer in the lab. Select one computer as the controller and the virtual machine installs the Win7 operating system. Select 8 computers as the controlled machine, the virtual machine installs the WinXp operating system and installs multiple Trojans in the virtual machine. Use Wireshark software to intercept and capture the data messages generated by Trojan behavior. The control side performs the Trojan operation behavior, such as copies the files of the controlled machine, replacing the desktop wallpaper of the controlled machine. Use Wireshark software to intercept data generated by Trojan behavior.

(2) Preprocessing the collected data. The data messages generated by Trojan behavior get the category number 1, other behaviors get the category number 0.

(3) Taking the first 60% of the obtained data as the training set and the last 40% as the test set.

(4) Using the training set as the input of CBBFKNN classification algorithm, the clustering parameter *t* set as 50, the parameter *k* set as 7. And then classify the collected data.

(5) Using SVM, naive Bayes and traditional K-nearest neighbor algorithm to classify the collected data and compare the accuracy, misjudgment rate and missed rate with CBBFKNN. The evaluation criteria are defined as follows.
Accuracy = (The number of detected Trojan sessions – Misreported Trojans sessions)/Total number of sessions
Misjudgment rate = Misreported normal sessions/Total number of sessions
Missed rate = (Total number of sessions – The number of detected Trojan sessions – Misreported normal sessions)/Total number of sessions

Analysis of Results. In this experiment, the accuracy, misjudgment rate, and missed rate is used as the criteria for evaluating the performance of classification algorithms. The experimental data contains 2,805 samples of traffic characteristics, including 1,683 training samples and 1,122 test samples. Compare the algorithm CBBFKNN proposed in this paper with the SVM, Naive Bayes, and K-nearest Neighbor algorithm which is the supervised learning algorithm in the same data. Compare their classification accuracy, misjudgment, missed rate, and running time. The algorithm performance comparison results are shown in Table 4.

Table 4. Algorithm performance comparison table based on Trojan detection

Algorithm	Accuracy/%	Misjudgment rate/%	Missed rate/%	Running time/s
SVM	83.43	6.21	6.45	1357
Naive Bayes	83.68	5.42	5.98	1361
K-nearest neighbor	84.23	5.77	5.65	1285
CBBFKNN	84.33	5.65	5.60	1273

The accuracy of CBBFKNN classification algorithm proposed in this paper can reach 84.33%, the misjudgment rate is less than 6%, and the missed detection rate does not exceed 6%, which is improved compared with the traditional K-nearest neighbor algorithm. And its running time is lower than the traditional K-nearest neighbor algorithm, achieving a certain degree of speed-up. The classification method proposed in this paper has higher detection capability for some common Trojans, meanwhile the misjudgment rate and missed detection rate is controlled in a lower range, which indicates that the classification method has strong practical value.

Based on the above two experiments, the speed-up K-nearest neighbor classification algorithm CBBFKNN proposed in this paper can effectively improve the classification efficiency with a relatively high accuracy and has certain feasibility. In the detection of Trojans has a certain practical value.

4 Conclusion

Trojan detection technology has great significance in network security. How to reduce the detection time and improve the detection efficiency is an important issue at this stage. This paper presents a speed-up K-nearest neighbor classification algorithm CBBFKNN. The dimensionality reduction of training data is performed through the idea of rectangular partitioning. The Kmeans algorithm is used to compress the training sample set, and the BBF algorithm is used to quickly query the category of the sample to be measured. Experiments show that compared with the traditional classification algorithm, this algorithm can reduce the time for classification when the loss of accuracy is small in the IRIS dataset. In actual Trojan detection, CBBFKNN has improved performance in various aspects compared with traditional classification algorithms, which can ensure higher accuracy and lower misjudgment rate and lower missed rate. This algorithm has a certain practical value.

Acknowledge. This work was supported by National Key Research and Development Plan of China (No 2016YFB0801004).

References

1. Liu, H., et al.: Research on FAHP adjudgement algorithm based on the behavior of Trojan. Harbin Engineering University (2016)
2. Zhang, Q., Li, C., Li, X., et al.: Irregular partitioning method based K-Nearest neighbor query algorithm using map reduce. Comput. Syst. Appl. **9**, 186–190 (2015)
3. Ren, L.: Speeding K-NN classification method based on clustering. Comput. Appl. Softw. **10**, 298–301 (2015)
4. Zuo, N.: Application of improved K-means clustering method of simulated annealing algorithm in students' grades. Guangxi Educ. **31**, 149–152 (2017)
5. Hu, J.: Improved KNN classification algorithm based on region division. Qingdao University (2016)

6. Yao, L., Huang, H.: Rolling bearing fault diagnosis based on improved K-means simulated annealing clustering algorithm. Modul. Mach. Tool Autom. Manufact. Tech. **4**, 114–117 (2017)
7. Pan, L., Yang, B.: Study on KNN arithmetic based on cluster. Comput. Eng. Des. **30**(18), 4260–4262 (2009)
8. Lan, T., Guo, G.: Improved RSKNN algorithm for classification. Comput. Syst. Appl. **22**(12), 85–92 (2013)
9. Wang, C., Cheng, S., Yang, X.: K-nearest neighbor neural network classifier of samples reduction based on clustering. Inf. Sci. **10**, 1547–1549 (2010)
10. Li, W., Li, L., Li, J., Lin, S., et al.: Characteristics analysis of traffic behavior of remote access Trojan in three communication phases. Netinfo Secur. **5**, 10–15 (2015)