A Decision Tree Candidate Property Selection Method Based on Improved Manifold Learning Algorithm

Fangfang Guo, Luomeng Chao, and Huiqiang Wang(✉)

Computer Science and Technology, Harbin Engineering University,
Harbin 150001, China
{guofangfang, wanghuiqiang}@hrbeu.edu.cn

Abstract. When the traditional decision tree algorithm is applied to the field of network security analysis, due to the unreasonable property selection method, the overfitting problem may be caused, and the accuracy of the constructed decision tree is low. Therefore, this paper proposes a decision tree selection method based on improved manifold learning algorithm. The manifold learning algorithm maps the high-dimensional feature space to the low-dimensional space, so the algorithm can acquire the essential attributes of the data source. According to this, the problems of low accuracy and overfitting can be solved. Aiming at the traditional manifold learning algorithms are sensitive to noise and the algorithms converges slowly, this paper proposes a Global and Local Mapping manifold learning algorithm, and this method is used to construct a decision tree. The experimental results show that compared with the traditional ID3 decision tree construction algorithm, the improved method reduces 2.16% and 1.626% in false positive rate and false negative rate respectively.

Keywords: Network security · Decision tree · Manifold learning algorithm

1 Introduction

Decision tree is an inductive learning algorithm that is widely used in security analysis, data mining and other fields. Because it is a heuristic algorithm, the unreasonable selection method of the property will directly result in a large deviation of the decision tree results, and it will easily lead to overfitting problem. This is also one of the key issues of the decision tree algorithm [1]. In the field of security analysis, the above problems are particularly evident, which will lead to high false negative rate and false positive rate in network security monitoring systems.

At present, the research on the decision tree mainly focuses on the following two aspects. The first is how to combine other algorithms to improve the accuracy of the algorithm [2]. The second is how to improve the performance of the algorithm by improving the property selection method in the decision tree construction process [3–5]. In the field of network security analysis, data source usually uses log data.

J. Zheng et al. (Eds.): ADHOCNETS 2018, LNICST 258, pp. 261–271, 2019.
https://doi.org/10.1007/978-3-030-05888-3_24

Because the log data has a large number of features, the complexity of the traditional decision tree algorithm is high and training model will take a long time. Therefore, the traditional algorithm cannot meet real-time requirements of the complex network environment security monitoring system represented by the cloud.

Manifold learning is an unsupervised learning algorithm, which is mainly used to reduce the dimension of high-dimensional data. The features extracted after the manifold learning process represent the important and essential features of the dataset [6]. Its characteristics can solve the problem of property selecting of the decision tree, and it can help decision tree algorithm select the proper feature to reduce the probability of overfitting. However, the manifold learning algorithm's convergence speed is slow and sensitive to noise data. Therefore, this paper proposed a Global and Local Mapping manifold learning algorithm (GALM) to improve the performance of the manifold learning.

According to this, this paper proposed a decision tree candidate property selection method based on GALM. The rest of the paper is structured as follows. In the second part of the article, the improved manifold learning method GALM is introduced; The third part introduces how to use GALM algorithm to construct decision tree, and the fourth part analyzes its advantages; In the last part, the effectiveness of the proposed algorithm is verified through experiments.

2 Global and Local Mapping Manifold Learning Algorithm

Manifold learning is an unsupervised learning method, which "manifold" represents the space homeomorphic with Euclid space in the local. Its main idea is the points in the high-dimensional observation space can be regarded as a manifold formed in the observation space by a few independent variables. Therefore, if one method can effectively find the internal main variables, it can reduce the dimension of the data set. Manifold learning can be divided into two categories: one is based on global considerations, such as Isometric Mapping [7] (ISOMAP); the other is based on local considerations, such as LLE [8] and LE [9]. LE is a locally embedded Laplacian-eigenmaps. Compared to LE, LLE algorithm tries to maintain the linear relationship among samples in the neighborhood. However, both types of algorithms have their own advantages and disadvantages. The first type of manifold learning algorithm has a slow convergence rate and it is not suitable for tasks with large data volumes. Although the convergence speed of the second manifold learning algorithm is faster, it is more sensitive to noise data. In order to solve the above problem, this article proposed Global and Local Mapping manifold learning algorithm (GALM).

The main ideas of GALM are as follows: First, local low-dimensional data representations are generated using a highly efficient local embedding method. Then, this method uses the global high-dimensional data to adjust the local low-dimensional data topology. Several definitions are given before describing the algorithm.

Definition 1 Geodesic: The geodesic distance is the shortest distance between two points on the manifold. The geodesic calculation uses Euclidean distance, the definition of Euclidean distance between two points is as follows:

$$\mathrm{G}\left(X_i, X_j\right) = \sum_{k=1}^{n} \left(X_{ik} - X_{jk}\right)^2 \tag{1}$$

where X_i and X_j represents the position of the point in space, $X_i = (X_{i1}, X_{i2}, \ldots, X_{in})$, $X_j = \left(X_{j1}, X_{j2}, \ldots, X_{jn}\right)$.

Definition 2 Harmonic average normalization: Before reducing the data dimension, it is necessary to normalized the average of geodesic. The formula is as follows:

$$\mathrm{dis}\left(X_i, X_j\right) = \frac{G\left(X_i, X_j\right)}{\sqrt{H(i)H(j)}} \tag{2}$$

where

$$H(i) = \frac{n-1}{\sum_{k=1}^{n} \frac{1}{G(X_i, X_k)}}, \quad k = 1, 2, \ldots, n \tag{3}$$

where $G\left(X_{i,}X_j\right)$ is the geodesic distance between the two points X_i and X_j, and $H(i)$ and $H(j)$ are the harmonic mean values of the two points X_i and X_j.

The main step of the improved manifold learning algorithm proposed in this paper is as follows:

① The k-order neighboring matrices are established by the neighboring rule, the Euclidean distance is used as a measure in this process, If the Euclidean distance between two points is less than ε, then define two points as neighbor, ε represents a threshold;

② For each sample point, it is reconstructed using its neighbors, and the minimum linear reconstruction weight is calculated by formula (4).

$$V_{min} = \left\| X_j - \sum\nolimits_j W_{ij} X_j \right\| \tag{4}$$

where W_{ij} is the linear reconstruction weight and the formula is shown in (5).

$$W_{i,j} = \begin{cases} e^{-\frac{\left\|X_i - X_j\right\|^2}{2\delta^2}}, & X_j \in N(i), \\ 0, & \text{else.} \end{cases} \tag{5}$$

where $N(i)$ represents the neighboring point of the point X_i. δ is a tuning parameter, it makes W_{ij} meet condition (6).

$$\sum_j W_{ij} = 1 \tag{6}$$

③ Low-dimensional embedding $\varnothing(y_i)$ of the input sample is calculated by formula (7), y_i represents the mapped node position;

$$\varnothing(y_i) = \left\| y_i - \sum\nolimits_j W_{ij} y_i \right\|^2 \tag{7}$$

④ In order to make the low-dimensional embedded geodesic lines close to the real geodesic lines, this method use global information to adjust the position of the sample after mapping. The movement of each node satisfies that $1/\theta$ of $dis\left(x_i, x_j\right)$ is the distance between the low-dimensional embedded node x_i and x_j (as shown in Fig. 1), where θ depends on the feature scaling, $x_j \in N(i)$. $1/\theta$ of $dis\left(x_i, x_j\right)$ is the distance between the low-dimensional embedded node x_i and x_j, where θ depends on the dimension scaling, $x_j \in N(i)$. D_i^{source} is the source node and $P_{i,j}$ represents neighbor node of D_i^{source}. $P_{i,j}^{new}$ is the position after mapped, which is obtained by formula (8). In order to make $P_{i,j}$ select the mapped low-dimension $\varphi(x_i)$, the top k neighboring nodes with the smallest distance loss are selected by formula (9), where π_{ij} represents the weight of each neighboring point.

$$P_{i,j}^{new} = P_{i,j} - \frac{\left(P_{i,j} - D_i^{source}\right)}{\left\|P_{i,j} - D_i^{source}\right\|} \times \frac{dis}{\theta} \tag{8}$$

$$min\theta(Y) = \sum_{i=1}^{n} \left| \varphi\left(y_i - \sum_{j=1}^{k} \pi_{i,j} \varphi\left(y_{i,j}\right) \right) \right|^2 \tag{9}$$

⑤ Reconstruct W through new neighbor node and calculate M according to formula (10), where the first d features of M represent the mapped low-dimensional space coordinates, d represents the scaled spatial dimension.

$$M = (I - W)^T (I - W) \tag{10}$$

The GALM algorithm is shown in Algorithm 1.

```
Algorithm 1: GALM
Input: Training set: X = $\{x_1, x_2, \ldots, x_n\}, x_i \in R^D$
Output: Reduced dimension dataset: Y = $\{y_1, y_2, \ldots, y_n\}, y_i \in R^d$
begin
Step1. For each $(x_i) \in$ X
            Generate $y_i$ randomly
      End For
Step2. For each $(x_i) \in$ X
         If $x_j \in X$ and $G(x_i, x_j) < \varepsilon$ then
            $N(i) \leftarrow N(i) + x_j$
            Compute $W_{ij}$ using formula (6)
         End
      End For
Step3. Get low-dimensional embedding $\emptyset(y_i)$ using formula (7)
Step4. Calculate $P_{i,j}^{new}$ where $i \in \{1,2,\ldots,n\}$
Step5. Reconstruct $W$ and get $M$ using formula (10)
return Top $d$ features in $M$
end
```

Since the geodesic distance estimated in the high-dimensional space is always larger than the geodesic distance of the low-dimensional embedding manifold, a parameter is required to dynamically adjust the distance between each embedded node and the neighboring node. Since the original data has been reconciled and averaged prior to embed, the low-dimensional embedding manifold is calculated using the distance after the reconciliation. Therefore, the embedding manifold effectively avoid the “point aggregation” problem.

3 Decision Tree Construction Method Based on GALM

The first step in the construction of the decision tree is to use the GALM algorithm to reduce the dimension. It requires that the manifold after the reduction can be spread evenly, thereby reflecting the nature of the feature.

After selecting one feature, the improved method will remove it, and it iteratively select the feature to test. However, this method is also limited, when an evenly distributed manifold cannot be found which can mapping high-dimensional data, traditional entropy calculations need to be performed on the remaining features. The decision tree construction process is shown in Algorithm 2.

Algorithm 2: Construct decision tree method

```
Input: Training set: X = $\{x_1, x_2, \ldots, x_n\}$ , $x_i \in R^D$,Feature set: F = $\{f_i, i = 1,2, \ldots, n\}$
Output: Decision tree model
begin
Step1. While Low-dimensional manifold F = $\{f_i, i = 1,2, \ldots, d\}$ not clear do
            Run GALM on F = $\{f_i, i = 1,2, \ldots, n\}$
Step2. ClassMap <key, value>←0; // randomly initialize
Step3. While F not Null do
            Compute attributesSet( ) on F
            For each attributes ∈ attributesSet( ) do
                String key←attributes.get(row, columnIndex, destination);
                if(key ∈ ClassMap)
                    classmap ← (key, value + 1);
                else
                    classmap ← (key, 1);
            End For
            F ← F – {key};
return classMap(<key, value>);
end
```

4 Complementarity Analysis of Manifold Learning Algorithm and Decision Tree Algorithm

Decision tree is an inductive algorithm, which has the advantages of strong anti-noise ability, high efficiency, etc. However, traditional decision tree generation methods often lead to overfitting. As a data dimension reduction method, the manifold learning method can help the decision tree to select important features, thereby reducing the possibility of overfitting. The comparison between the two algorithm is shown in Table 1.

Table 1. Comparison of Decision Tree and Manifold Learning

Disadvantages of decision tree	Advantages of manifold learning
① When the number of features is large, the unreasonable feature selection rules will lead to deviations in the results of the decision tree model	① Manifold learning algorithms can map high dimensions to lower dimensions, revealing the essential characteristics of the data
② Because it relies on axis-parallel segmentation, it can be difficult to model some relationships	② Manifold learning algorithm improves the efficiency of data analysis by reducing dimensions and reducing some insignificant features
③ When the sample set changes, the decision tree constructed by the algorithm will also change due to changes in the sample set	③ Manifold learning algorithm can select stable and critical feature through dimension reduction measures

5 Simulation Experiment and Performance Analysis

5.1 Experimental Design and Experimental Parameters

(1) Experimental environment

In order to analyze the performance and effectiveness of algorithm proposed in this paper, we use the Toolbox toolkit and the classic Swiss roll data source to perform experiments on the MATLAB simulation platform.

As shown in Fig. 1, using the existing equipment in the laboratory, cloud monitoring data fusion analysis system based on the log data is set up. The object of analysis is multi-source log data, the specific information of the collected data will be described in detail later. In the experiment, four infrastructures were configured in the Hadoop cluster environment. One of them is set as Master node and the others are Slave nodes.

(2) Data source

The multi-source log collection is provided by Hadoop's Flume, an acquisition component. After collection, the log data are aggregated to the log receiving server for storage. The log used for security analysis are divided into IDS log, firewall log, and DNS log. The Kali Linux penetration test is used to perform corresponding security event attacks on the target host. The attacks used in this paper include SYN Flood, ICMP Flood, TCP Flood, DNS Flood, and ARP Spoofing. This paper will use the Swiss roll data source to make usability analysis of the GALM algorithm, and to do comprehensive verification in the final analysis stage.

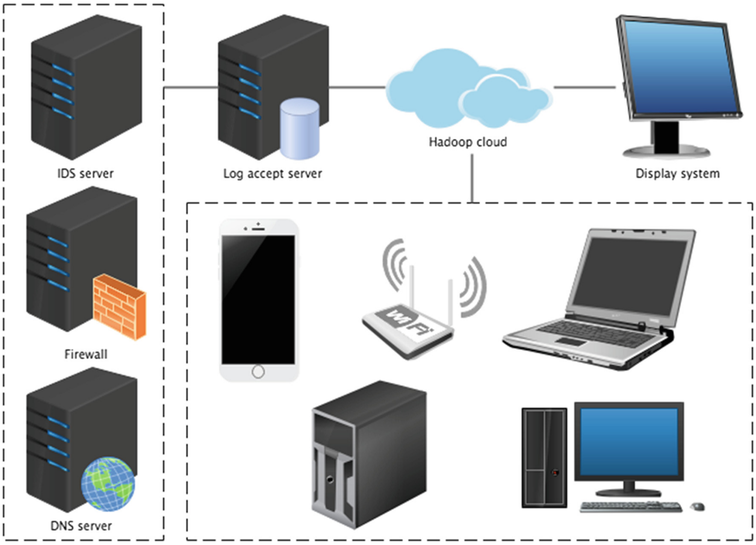

Fig. 1. Experimental environment design

The evaluation of experimental results is divided into two aspects: efficiency and accuracy. In terms of efficiency, the proposed method is compared with LLE, LE, and ISOMAP algorithms, and the improved decision tree algorithm is compared with the ID3 decision tree algorithm. In the aspect of accuracy, the improved decision tree algorithm and ID3 decision tree algorithm are compared in terms of false negative rate and false positive rate.

5.2 Experimental Results and Algorithm Performance Analysis

(1) GALM algorithm performance analysis

The selected Swiss roll data source is processed by GALM algorithm, traditional LEE algorithm, LE and ISOMAP algorithm respectively. Through the analysis of the processing results, the effectiveness of the low-dimensional manifold formed by dimension reduction through GALM algorithm is verified, then the computational efficiency of GALM is verified.

Figure 2 shows a 3-dimensional manifold image generated by a Swiss roll data source with a quantity of 2000, noise of 0.05. This experiment use GALM algorithm to reduce dimensionality of manifolds to form 2-dimensional manifold respectively. Figure 3 shows the embedded manifold Formed by GALM.

Figure 4 shows a comparison of the running time of GALM, LLE, LE and ISOMAP. It can be seen that at the beginning, the GALM runtime is close to the LLE and LE. As the amount of data increases, the running time of the GALM algorithm will gradually approach the ISOMAP.

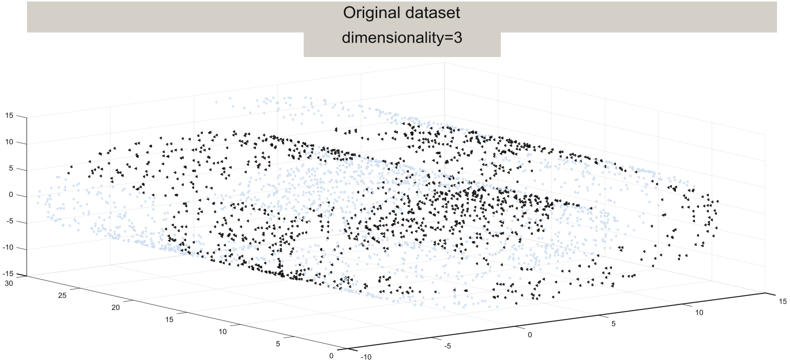

Fig. 2. Swiss roll data Source: 3-dimensional manifold formed by 2000 nodes

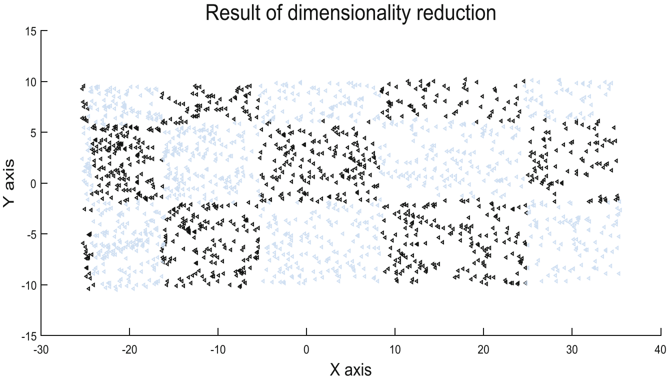

Fig. 3. 2-Dimensional Embedded Manifold Formed by GALM

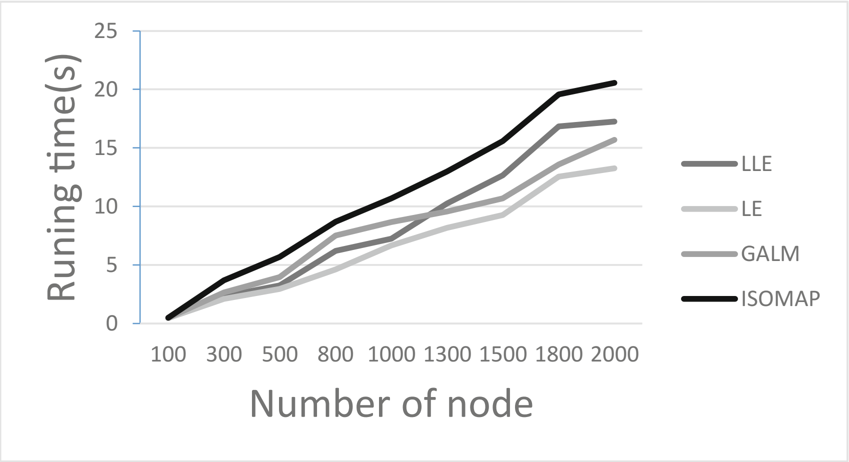

Fig. 4. Running time comparison

(2) **Algorithm accuracy rate assessment**

The log source obtained from the log receiving server is 2.1 GB. The method of constructing the decision tree based on GALM algorithm presented in this paper is compared with the ID3 decision tree construction algorithm.

Figure 5 compares the misjudgment rates of the two algorithms. The experimental results show that compared with the ID3 algorithm, the method proposed in this paper reduces the misjudgment rate by 0.323%, 0.365%, 1.079%, 0.597% and 1.128% respectively for ARP, DNS, UDPS and SYN. Especially for SYN Flood and UDP Flood, it can be seen that the decision tree construction method based on manifold learning shows a good detection effect in terms of false positive rate, so the accuracy of the improved decision tree detection algorithm has been improved overall.

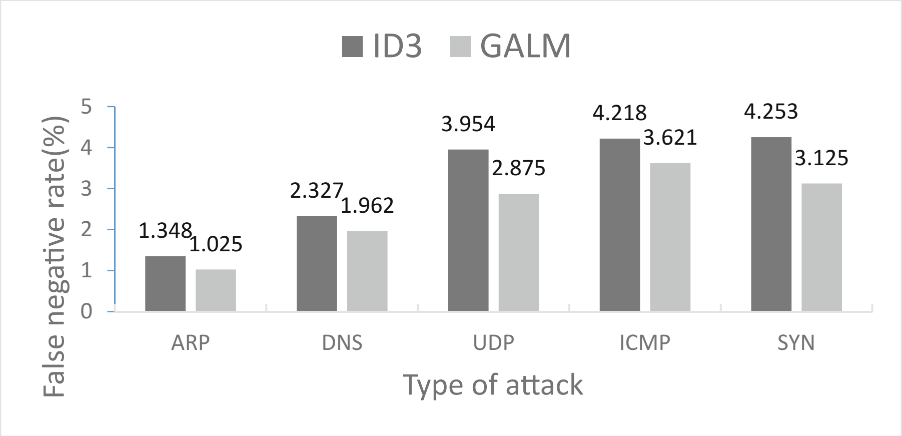

Fig. 5. Comparison chart of False negative rate

6 Conclusion

The manifold learning method has a obvious dimensionality reduction effect on non-linear data, and it can get the nature of the data. This feature can be combined with the classic decision tree algorithm to reduce the overfitting problem. Based on the above ideas, this paper proposes a GALM algorithm, it can select the nature of the data to build a decision tree. The experimental results show that the decision tree constructed using the algorithm proposed in this paper has been improved in accuracy and efficiency of model. The next step will focus on how to combine multi-manifold learning with other security analysis algorithms to effectively solve security monitoring issues in the field of network security situational awareness.

References

1. Kim, S.Y., Upneja, A.: Predicting restaurant financial distress using decision tree and AdaBoosted decision tree models. Econ. Model. **36**(1), 354–362 (2014)
2. Pai, P.F., Changliao, L.H., Lin, K.P.: Analyzing basketball games by a support vector machines with decision tree model. Neural Comput. Appl. **232**, 1–9 (2016)
3. Azad, M., Moshkov, M.: Multi-stage optimization of decision and inhibitory trees for decision tables with many-valued decisions. Eur. J. Oper. Res. **263**, 910–921 (2017)
4. Ai, X., Wu, J., Cui, Z.: Broaden the minority class space for decision tree induction using antigen-derived detectors. Knowl.-Based Syst. **137**, 196–205 (2017)
5. Cicalese, F., Laber, E., Saettler, A.: Decision trees for function evaluation: simultaneous optimization of worst and expected cost. Algorithmica **79**, 1–34 (2013)
6. Zhang, Q., Zhang, Q., Zhang, L.: Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding. Pattern Recogn. **48**(10), 3102–3112 (2015)
7. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
8. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
9. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)