Deep Learning-Based Network Intrusion Detection System

Authors: Li Zexian

Date: 2024-01-03T00:00:00+00:00

Abstract

Please provide the Chinese academic text requiring translation and cleanup. I am prepared to:

- Remove XML/HTML tags and watermark artifacts
- Ensure proper academic paragraph structure with complete paragraphs (typically 3-8 sentences)
- Preserve all LaTeX math expressions (...,

...

, […]) and citations ([], **??**, (**??**))
- Maintain section headers (## for major sections, ### for subsections) and formatting
- Preserve all figure/table markers ([FIGURE:N], [TABLE:N]) without modification
- Deliver fluent, readable English suitable for an academic audience
- Organize content logically with clear topic sentences and transitions

Once you provide the text, I will process it according to these specifications and return a clean, properly formatted academic translation.

Full Text

Student ID: 2023200815
Course Report: Deep Learning, Beijing University of Chemical Technology
Title: Network Intrusion Detection System Based on Deep Learning
Major: Computer Science and Technology
Class: Xinyan 2305

December 31, 2023

Chapter 1 Introduction

With the development of computer technology, new opportunities have emerged continuously, yet a wide variety of network attacks have also negatively impacted the growth of the Internet. The overall number of network attack methods such as DoS attacks, phishing attacks, Trojans, and botnets continues to rise. In the face of increasingly complex and diverse network attacks, static defense measures such as firewalls and antivirus software alone are insufficient.

Intrusion Detection Systems (IDS) can be categorized based on data sources and detection techniques. As shown in [Figure 1: see original paper]-1, early IDS predominantly employed misuse-based techniques to detect network attacks. However, misuse-based intrusion detection systems rely heavily on existing signature knowledge bases, resulting in poor identification of unknown attacks.

With the support of deep learning, the accuracy and efficiency of network intrusion detection have improved. To further reduce the false positive rate, deep learning network models require continuous parameter refinement during training and overfitting prevention during prediction. This means that the number of parameters and structural complexity of our network models must far exceed those of previous approaches. This situation places higher demands on our choice of algorithms and models to a certain extent. Therefore, while ensuring prediction accuracy, how to simplify network structures, improve computational efficiency, and reduce computational load and power consumption has become a critical issue to address for practical deployment.

Chapter 2 Theoretical Foundation

2.1 Intrusion Detection

Intrusion Detection Systems (IDS) differ from firewalls in that they can actively monitor the entire network and detect intrusion behaviors in real time. If a firewall is compared to the lock on a shopping mall's main entrance, then intrusion detection is the mall's surveillance system. The front door lock can prevent some thieves from entering the mall, while the surveillance system can detect situations and issue warnings. As an important supplement to firewalls, intrusion detection has become a crucial component in building network security defense systems, helping to overcome the limitations of traditional defense mechanisms. In a network, intrusion detection is typically positioned between switches and hubs.

The workflow of intrusion detection is shown in [Figure 2: see original paper]-1. The first step involves information collection by the host, primarily focusing on current system or user-related behaviors such as system logs, application logs, network data, and audit records. The second step is intrusion analysis, which compares the collected data against a knowledge base in the database. This knowledge base is derived from historical network behaviors or downloaded traffic features. Analysis methods are divided into anomaly detection and misuse

detection. Using a trained detection model, the system evaluates the information obtained in the previous step. If the information does not match known normal behaviors in the model, it is judged as an attack. Data analysis is the core component of intrusion detection. The final step is alarm response: when the system identifies a behavior as an attack, it first saves information about this behavior in the database, then takes measures such as issuing warnings to the system or implementing manual intervention to counter the attack.

2.2 Deep Learning-Based Network Intrusion Detection Algorithms

Traditional network intrusion detection suffers from poor accuracy, low efficiency, long processing times, and high false positive rates, yielding suboptimal results for large datasets. The emergence of deep learning-based network intrusion detection algorithms has significantly improved these issues.

Deep learning is a machine learning algorithm for training deep neural networks. Its essence lies in constructing multi-layer network models that leverage high-level features from multiple layers to represent the abstract semantic information of data, using large amounts of training data for feature learning to ultimately improve classification or prediction accuracy. Deep learning and traditional machine learning are similar in data preprocessing, both requiring operations such as data cleaning, normalization, denoising, and dimensionality reduction. However, compared to deep learning, traditional machine learning relies primarily on manual feature extraction, which is efficient and interpretable for simple tasks but lacks generality. Deep learning, by contrast, employs automatic feature extraction, offering relatively better classification performance but poorer interpretability.

2.2.1 K-Nearest Neighbors Algorithm The KNN algorithm, or K-Nearest Neighbor algorithm, is instance-based learning and represents lazy learning. The KNN algorithm has broad application scenarios and is widely used in spam identification, image content recognition, and text analysis. The core idea is that given a training dataset, the algorithm finds the K instances in the training dataset that are nearest to the input data. It classifies new instances based on the majority class among these K instances and solves regression problems by taking their weighted average, as shown in [Figure 2: see original paper]-2.

In [Figure 2: see original paper]-2, the size of the circle represents the value of k. Whichever category has the most instances within the circle becomes the predicted value. When k=1, the classification is assigned to the red square; when k=5, since there are more gray triangles within the circle, the classification is assigned to the gray triangle. We conclude that the final prediction value depends on two factors: the size of the k value and the distance metric. The commonly used Euclidean distance is the most intuitive representation of distance between two or multiple points. As shown in [Figure 2: see original paper]-3, it can be simply understood as the length of the line connecting two points. For example,

the Euclidean distance between A(x1,y1) and B(x2,y2) is d(A,B) = √((x1-x2)^2 + (y1-y2)^2).

2.2.2 BP Neural Network The BP (Back-propagation) neural network is the most traditional and simplest neural network, and its parameter training philosophy makes it the most widely used neural network. The so-called error backpropagation involves using the chain rule to differentiate the loss function with respect to weight values (each weight parameter can be viewed as a parameter of the loss function). Since the opposite direction of the gradient is the direction in which the function value decreases fastest, we can use gradient descent to optimize the loss function. Through these two mechanisms, weight values are updated until the error function is minimized and model training concludes.

As shown in [Figure 2: see original paper]-9, in a three-layer neural network with two inputs and one output, each neuron performs two operations: first, it computes the product of weight coefficients and input signals, and second, it passes this product into the neuron's activation function.

To train the neural network model, we need to prepare a training set that includes input signals (x1,x2) and corresponding targets (desired output) z. Network training is an iterative process where each iteration uses new data from the training dataset to modify the nodes' weight coefficients. This modification is completed through the algorithm shown in [Figure 2: see original paper]-10, with each step requiring two input signals from the training set. When propagating signals through the network, we can determine the output signal values of each neuron in each network layer, where the symbol w(xm)n denotes the connection weight between network input xm in the input layer and neuron n, and the symbol yn denotes the output signal of neuron n.

The method of adjusting network parameters to make the network output as close as possible to the true value is crucial. Here we first introduce the concept of the loss function. The so-called loss is the error between the network's output value yout and the true value y, expressed as |yout-y|. The function that calculates this loss is the loss function. Commonly used loss functions include the mean squared error function and cross-entropy loss. A better network model is equivalent to minimizing the loss function, and the method to minimize the loss function is gradient descent. The gradient is obtained by differentiating the curve formed by the loss function under different parameters. A smaller gradient indicates that the loss function is decreasing faster, thereby obtaining the minimum value of the loss function. Optimizing network model parameters is achieved by adjusting the weight coefficients of each neuron's input, and we can update each neuron's parameters (weight coefficients for different inputs) using Formula 2-10. The term df(e)/de in the formula represents the derivative of each neuron's activation function.

The so-called network training involves the model continuously performing for-

ward propagation and backward propagation to optimize model parameters and approach optimal performance. Theoretically, continuous training can eventually achieve the optimal model, but in practice, the final classification effect may actually deteriorate after training to a certain extent, which has also become a hot research topic in recent years.

Chapter 3 Training Set and Framework for Deep Learning-Based Network Intrusion Detection Algorithms

3.1 Network Training Dataset

This experiment uses the KDD CUP 99 dataset, which is the most commonly used dataset in intrusion detection. It originated from the 1998 DARPA intrusion detection evaluation project. Later, Wenke Lee et al. performed feature extraction on the raw data, which was collected from the US Air Force local area network over nine weeks of network traffic collection—seven weeks of traffic for the training set and the remaining two weeks for the test set—containing approximately 7 million traffic records. The test set also includes some attack types not present during training to test the intrusion detection model's ability to identify unknown attacks.

The training set categorizes network traffic as normal or abnormal, with types indicated in traffic labels. As shown in [Figure 3: see original paper]-1, abnormal traffic is divided into four major categories containing a total of 39 attack types, but only 22 attack types appear in the training set, with the remaining 17 types appearing only in the test set. The ability to identify unknown network attacks is an important metric for evaluating network intrusion detection systems.

Each traffic record in the KDD CUP 99 dataset contains 41 fixed feature attributes and one class identifier, among which nine feature attributes are symbolic (discrete) types while the others are continuous.

Chapter 4 Experiments and Results Analysis

This graduation project experiment was implemented on a PC platform. This chapter introduces the hardware and software configuration of the PC platform and describes the data preprocessing stage for the KDD CUP 99 dataset. Finally, it presents and analyzes the training and prediction results of the network intrusion detection system.

4.1 Dataset Preprocessing

The KDD CUP 99 dataset used in this graduation project contains symbolic data attributes that cannot be processed directly by neural network models. To facilitate better experimentation, we need to preprocess the data. Dataset preprocessing generally consists of three steps: (1) converting character data in the dataset to numeric data, (2) numeric standardization, and (3) numeric normalization.

4.1.1 Converting Symbolic Data to Numeric Data Among the 41 features in the KDD CUP 99 dataset, three are symbolic features: the protocol type feature (protocol_{type}), the target network service feature (service), and the connection status feature (flag) indicating whether the connection is normal. For these three symbolic features, we can use attribute mapping methods to convert character data to numeric data [23].

The protocol type feature represents the transport layer protocol of network traffic and has three types: TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and ICMP (Internet Control Message Protocol). These are encoded as 0-2, or in binary as TCP-00, UDP-01, ICMP-10. The target network service feature has 70 types, such as http, ftp, IRC, link, login, mtp, etc., which are encoded as 0-69. The connection status feature (flag) has 11 types, such as OTH, REJ, RSTR, etc., which are encoded as 0-10. Finally, attack types are encoded with normal traffic as 0, DoS-1, Probe-2, R2L-3, and U2R-4.

4.2 Experimental Process

This experiment first splits the dataset into training and test sets, allocating 20% of the KDD CUP 99 dataset as the test set and the remainder as the training set. Data preprocessing follows, converting symbolic data to numeric data through one-hot encoding, then performing Z-Score standardization and Max-Min normalization. Next, the training target is determined; for binary classification, data labels are examined and set to 1 if the label indicates an anomaly. The KNN algorithm is then applied with K values of 3, 5, 7, 9, 11, 13, and 15 using the training set and corresponding categories. For the test set prediction, Euclidean distances between the input point and neighboring points are calculated and sorted, and the top k points are selected. The category with the most instances among these points becomes the input point's category. Predictions are compared with labels to calculate accuracy, and functions generate accuracy rates for different k values. The workflow diagram is shown in [Figure 4: see original paper]-1.

After identifying the k value corresponding to the highest accuracy, the BP neural network, Naive Gaussian Bayes, and Decision Tree algorithms are applied to predict on the test set. ROC curves, P_R curves, and confusion matrices are generated for comparison with other models.

4.3 Experimental Results and Analysis

This experiment uses the KNN algorithm for training and prediction, then compares it with Gaussian Bayes, BP neural network, and Decision Tree algorithms by outputting classification results. For the KDD CUP 99 dataset, results and analysis are presented for both binary classification (normal vs. abnormal) and five-class classification (normal and four major attack categories). Due to the large original dataset and limited host performance, to accelerate experimental

progress, the number of folds in cross-validation is set to 5, and only 20% of the original dataset (approximately 80,000 packets) is sampled as the training and test sets, with the test set accounting for 20%.

(1) Binary Classification To prevent accuracy degradation caused by insufficient data and to estimate the model's accuracy in real-world data applications, this experiment uses cross-validation.

In the KNN algorithm, I selected K parameter values between 3 and 15, incrementing by 2 each time (resulting in values of 3, 5, 7, 9, 11, 13, and 15), and calculated the average accuracy for each corresponding k value, as shown in [Figure 4: see original paper]-2.

All accuracies under various k values exceed 99%, meeting pre-experimental expectations. Among these, the average accuracy is highest when K=3, so K=3 is selected as the K value for the KNN algorithm in binary classification.

The KNN algorithm is then compared with other algorithms. First, the Decision Tree algorithm is selected, with the max_{depth} parameter range set from 10 to 30. The accuracy under each value is shown in [Figure 4: see original paper]-3.

The accuracy is highest when the value equals 26 and 30; here we select 26. Under this setting, accuracy is tested for values ranging from 2 to 20, as shown in [Figure 4: see original paper]-4.

The min_{{{samples}}_{{split}}} is ultimately set to 2.

Performance Evaluation: First, examine the confusion matrices for each model in [Figure 4: see original paper]-5. [Figure 4: see original paper]-6 shows the accuracy, precision, recall, and F1-Score for each model (here the results are multiplied by 100). Finally, the ROC curves and P_R curves are presented in [Figure 4: see original paper]-7.

In the ROC curves, the AUC area for each model is 1 when rounded to two decimal places, with minimal differences between the four models. All metrics are above 94%, indicating good classification results. In the P_R curves, it is also difficult to compare the relative merits of each model. However, observing the confusion matrices, we find that KNN performs the best among these models. Nevertheless, all four models exhibit cases where normal packets are identified as abnormal packets.

(2) Five-Class Classification In the five-class KNN algorithm, I selected the same K parameter values as in binary classification and calculated the average accuracy for each corresponding K value, as shown in [Figure 4: see original paper]-8.

As the results demonstrate, the K value setting for the five-class KNN algorithm is consistent with that in binary classification at K=3.

Similarly, the Decision Tree algorithm is selected for comparison. The Decision Tree max_{depth} parameter range is 10 to 30, with accuracy shown in [Figure 4: see original paper]-9.

When the parameter reaches 27, the accuracy basically remains at 99.84, so max_{depth} is set to 27. Based on this, min_{{{samples}}_{{split}}} is tested with a range of 2 to 20, with accuracy shown in [Figure 4: see original paper]-10.

As this parameter increases, the accuracy shows a downward trend, so this parameter is ultimately set to 2.

The confusion matrix comparison is shown in [Figure 4: see original paper]-11. The comparison of accuracy, precision, recall, and F1-Score for each model is shown in [Figure 4: see original paper]-12. The ROC curve and P_R curve comparison is shown in [Figure 4: see original paper]-13.

After changing the classification from binary to five-class, the prediction performance of each model differs significantly. Although the ROC curves of the four models intersect, Bayes has the smallest AUC area, while KNN and Decision Tree have the largest areas. In the P_R curves, the intersection point of the BP neural network's diagonal line is the lowest, while KNN's intersection point is the highest. Overall, Bayes performs the worst in terms of accuracy and confusion matrix. The accuracy and other performance metrics of KNN, neural network, and decision tree are similar, but confusion matrices can specifically reflect their differences. Relatively speaking, KNN and Decision Tree classifications are more generalizable. The high performance of the BP neural network is due to dataset imbalance (DoS attack packets occupy a large proportion in the KDD CUP 99 dataset), while its recognition rates for other attacks and normal packets are very low. Under the influence of sample imbalance, the DoS attack anomaly packets are the most numerous, so more features are extracted for them, whereas other attack packets are very few and have fewer extracted features, ultimately leading to reduced convergence during the training phase and decreased generalization during testing. Sample imbalance affects other models as well, but Gaussian Bayes is somewhat more successful in judging minority classes, though its overall effect is worse. Overall, KNN delivers the best performance. In binary classification, all anomaly packets are merged into a single category, so sample imbalance is less pronounced and all four models perform better.

With the rapid development and iteration of deep learning, an increasing number of network models have emerged like bamboo shoots after a spring rain. To pursue faster classification, various fields have begun to use deep learning frameworks. Today, as information security becomes increasingly important, traditional network intrusion detection can no longer achieve real-time and accurate detection, and deep learning-based network intrusion detection has emerged accordingly. This experiment is based on the Sklearn framework and uses the KDD CUP 99 dataset to train network models based on the KNN algorithm for

binary and five-class classification, generating confusion matrices, ROC curves, and P_R curves. By comparing the accuracy of Naive Gaussian Bayes, BP neural network, Decision Tree algorithms, and KNN algorithms with different k values, we compare the advantages and disadvantages of each algorithm and determine the optimal k value. The Pyecharts package is used to achieve data visualization and generate accuracy plots. The prediction effect of this experiment is generally good, meeting previous expectations and offering considerable advantages compared to other algorithms online.

Note: Figure translations are in progress. See original paper for figures.