An Intrusion Detection Method based on Stacked Autoencoder and Support Vector Machine

Tan Shuaixin 1*

1 School of Computer Science, South-Central University for Nationalities, Wuhan 430074, China
*Corresponding author’s e-mail: tansx2016@126.com

Abstract. To explore the feasibility of forming a combined classifier for intrusion detection by applying Support Vector Machine (SVM) and Stacked Autoencoder (SAE), an intrusion detection method based on Stacked Autoencoder and Support Vector Machine is proposed. Considering that the Piecewise Radial Basis Function (P-RBF) in the Support Vector Machine can improve the classification performance, the Radial Basis Function (RBF) and P-RBF are selected for the proposed method, and the detection performance of the above two methods are compared. Experiments show that the method based on Stack Autoencoder and Support Vector Machine using P-RBF kernel are superior to the one using RBF kernel in accuracy, detection rate and false alarm rate. The method based on Stack Autoencoder and Support Vector Machine provides an idea for the application in the field of intrusion detection.

1. Introduction

Intrusion detection method based on machine learning have become one of the most popular research methods today. On the one hand, the machine-learning-based intrusion detection method can detect some unknown attack types effectively. On the other hand, people have proposed a large number of machine learning methods. It means whether the intrusion detection method is based on neural network or non-neural network, or shallow learning method or deep learning method, there have been a lot of research findings in terms of single learners. In order to enhance the effect of intrusion detection, some combination learning methods based on these research findings have been proposed, and this produces more findings. In recent years, with the popularity of deep learning, people have not only proposed a large number of deep learning methods and applied them to the field of intrusion detection, but also used deep learning methods to expand the combination learning methods. At present, more tendencies are to use the deep learning method as a feature extractor and the shallow learning method as a classifier, which truly makes the complementary advantages of the two to improve the detection performance.

In 2006, Hinton[1] proposed an autoencoder network based on the Restrict Boltzmann Machine (RBM) to extract the features of handwritten digital pictures in the MNIST dataset. The performance of this method is better than that of the method based on Principal Component Analysis (PCA) and overcomes the vanishing gradient problem of some traditional multilayer neural networks. The following year, Bengio[2] proposed the stacked autoencoder based on the autoencoder network proposed by Hinton by cancelling the Gibbs sampling process involved in each layer of RBM in the neural network. On the other hand, Gao Ni[3] et al. applied the autoencoder network proposed by Hinton et al. to the field of intrusion detection, and the experiment well reflected the effectiveness of the autoencoder network to extract features. Wang[4] and others have introduced Autoencoder into the field of intrusion detection and combined with k-means algorithm for clustering. Experiments show that when
the number of features is reduced to 18 and the number of clusters is 50, this hybrid intrusion detection method based on Autoencoder and k-means can achieve the best results. Despite this, no one has used Stacked Autoencoders for intrusion detection.

Section 2 introduces the basic principles and feature extraction algorithms of the Stacked Autoencoder. Section 3 introduces the basic flow of intrusion detection methods based on Stacked Autoencoder and Support Vector Machine. Section 4 demonstrates the feasibility of the SAE-SVM method in the field of intrusion detection. Section 5 forms a conclusion and summarizes the full text.

2. Stacked Autoencoder

2.1. The structure of Autoencoder

Autoencoder (AE) is a kind of three-layer neural network consisting of an input layer, a hidden layer, and an output layer. Its structure is shown in Figure 1.

![Figure 1. The structure of autoencoder](image)

It can be seen from its structure that the number of nodes of the input layer and that of the output layer in this Autoencoder is the same. In a Stacked Autoencoder, the encoding process directly maps the input data from the input layer to the hidden layer, and the decoding process maps the data hidden layer of the hidden layer to the output layer. Both of these processes are non-linear mapping processes. In fact, when the number of nodes in the hidden layer is less than the number of nodes in the input layer, the encoding process of the Autoencoder represents the compression of the feature, and the output layer reflects the reconstruction of the input layer. In essence, the essence of the training of the Autoencoder is to find the best neural network parameters to achieve the purpose of maximizing the output data to the input data, and to implement feature extraction of the input data. Therefore, the Autoencoder can be used as a feature extractor.

2.2. The structure of Stacked Autoencoder

The Stacked Autoencoder is a deep neural network formed by stacking a plurality of Autoencoders layer by layer, and its structure is shown in Figure 2.
The Stacked Autoencoder is composed of \( n \) autoencoders. It is possible to use the parameters \( W^{(k,1)}, W^{(k,2)}, b^{(k,1)}, b^{(k,2)} \) to represent the parameters \( W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)} \) corresponding to the \( k \)th Autoencoder, and \( \alpha^{(n)} \) is the activation value of the deepest hidden unit. Then the stacked autoencoder neural network coding process satisfies formula (1).

\[
\begin{align*}
\alpha^{(l)} &= f(z^{(l)}) \\
z^{(l+1)} &= W^{(l,2)}\alpha^{(l)} + b^{(l,2)}
\end{align*}
\]

(1)

And the decoding process satisfies formula (2).

\[
\begin{align*}
\alpha^{(n+l)} &= f(z^{(n+l)}) \\
z^{(n+l+1)} &= W^{(n-l,2)}\alpha^{(n+l)} + b^{(n-l,2)}
\end{align*}
\]

(2)

Where \( f(\cdot) \) represents the activation function. It can be known from formula (1) and formula (2) that the encoding process is sequentially performed from the lower layer to the upper layer, and the decoding process is sequentially performed from the upper layer to the lower layer.

Since each Autoencoder consists of an input layer, a hidden layer, and an output layer, when implementing the stacking process of the Stack Autoencoder, the input samples are sent to the input layer of the first layer Autoencoder first, and then these data in the input layer are mapped to the hidden layer. Next, the data in the hidden layer are mapped to the output layer. After that the value of the output layer and the value of the input layer are used to calculate the reconstruction error. The reconstruction error is calculated by the squared error function, and the squared error function is shown in formula (3).

\[
L(x, y) = \|x - y\|^2
\]

(3)
In fact, the Stacked Autoencoder can be applied into two different ways: one is used for feature extraction and the other one is used for classification. For a Stacked Autoencoder used for classification, it is necessary to fine-tune the weights of the layers after pre-training. If a Stacked Autoencoder is used as a feature extractor, it is an unsupervised feature learning and supervised fine-tuning is dispensable.

For the Stacked Autoencoder used as a feature extractor, the following feature extraction algorithms is shown in Table 1.

### Table 1. Feature Extraction Algorithm of Stacked Autoencoder

| Algorithm 1 Stacked Autoencoder Feature Extraction Algorithm |
|---------------------------------------------------------------|
| **Input:** sample data                                        |
| **Output:** features                                         |
| 1 Determine the structure of the Stacked Autoencoder, the number of samples per batch of data, and determine the learning rate of the Autoencoder layer by layer. |
| 2 Let \( W^{(k,1)} = 0, W^{(k,2)} = 0, b^{(k,1)} = 0, b^{(k,2)} = 0 \), and initialize the Stacked Autoencoder. |
| 3 Input the sample data to the input layer of the Stacked Autoencoder. Then, for each layer of the Autoencoder, first encode according to formula (1); then decode according to formula (2) and activate the function to select the sigmoid function; next calculate the reconstruction error according to formula (3). |
| 4 Save the result of the last layer of the Stacked Autoencoder, which is the required feature. |

### 3. The flow of the intrusion detection method based on Stacked Autoencoder and Support Vector Machine

Figure 4 shows the flow of an intrusion detection algorithm based on Stack Autoencoder and Support Vector Machine. Feature normalization refers to converting nominal features into numeric features and normalizing each feature. Using SAE for feature extraction is to extract features greedily layer by layer according to algorithm 1. Finally, classify the features above with SVM and output classification results.

![Figure 4. The flow of the intrusion detection method based on Stacked Autoencoder and Support Vector Machine](image)
4. Experiments

4.1. Dataset
The KDD CUP 99[6] dataset was used in this experiment. There are 5 categories in the dataset which includes normal samples and attack samples. They are Normal, DOS, Probe, U2R and R2L. Considering that the sample size of the Probe, U2R and R2L is relatively fewer than other category in the dataset, experimental data would be randomly extracted from the KDD CUP 99 10% training set and the KDDTest test set. The extracted data distribution is shown in Table 2.

Table 2. Quantities of each type of the data in the sample

| Type of data | Number of Training sample | Number of Test sample |
|--------------|---------------------------|-----------------------|
| Normal       | 20 628                    | 24 576                |
| DOS          | 5 221                     | 4 586                 |
| Probe        | 4 107                     | 4 166                 |
| U2R          | 52                        | 228                   |
| R2L          | 1 126                     | 100                   |
| Total        | 31 134                    | 33 656                |

4.2. Experimental data pre-processing
The experimental data pre-processing was divided into two parts. The first part was to convert the nominal features in the data to the numeric features. Because the extracted training subset and the test subset contain 41 features and among these features, protocol_type, service and flag are nominal features, nominal features are need to be converted into numeric features in discrete way respectively. Here, the above features were directly converted from nominal features to numeric features. The second part was the normalization of the feature. For some features, the range of values inside of each feature are very large, so the value of each feature is normalized in the range of [0,1] after the transformation from the nominal features to the numeric features is implemented.

4.3. Evaluation indicators
The evaluation indicators involved in the experiment are accuracy rate (AC), detection rate (DR) and false alarm rate (FAR), which are expressed by formulas (4)-(6):

\[
AC = \frac{TP+TN}{TP+TN+FP+FN}
\]

(4)

\[
DR = \frac{TP}{TP+FP}
\]

(5)

\[
FAR = \frac{FP}{TN+FP}
\]

(6)

Among these formulas, TP indicates the number of correctly detected as normal data records; FP indicates the number of wrongly detected as normal data records; TN indicates the number of correctly detected errors in the abnormal data records and FN indicates the number of wrongly detected as abnormal data records.

4.4. Experimental analysis

4.4.1. Experiment results. The operating environment of this experiment was a computer equipped with AMD 2.40GHz CPU, 4GB memory and 64-bit Windows 10 system. The software used in the experiment was MATLAB R2010B. The toolbox LIBSVM developed by Professor Lin Chih-Jen[7] was adopted as
an SVM classifier. Deeplearntoolbox[8] developed by Technical University of Denmark was adopted as an SAE feature extractor. The RBF kernel and P-RBF kernel[9] were applied on SVM to classify in the experiment. Both SVM and SAE parameters were selected empirical values. The experimental parameters are shown in Table 3.

Table 3. SAE and SVM parameters

| Parameter                                      | Value |
|-----------------------------------------------|-------|
| Number of nodes in input layer                | 41    |
| Number of nodes in first layer                | 13    |
| Number of nodes in second layer               | 5     |
| Number of nodes in input layer                | 0.01  |
| Pre-training learning rate for each layer     | 1     |
| Whether to use the BP algorithm to fine-tune the weight | No    |
| c                                             | 10    |
| g                                             | 0.0001|

In order to illustrate the effect of the SAE-SVM method better, a comparison between the SAE-SVM-RBF (Stacked Autoencoder Support Vector Machine Based on Radial Basis Function) method and SAE-SVM-P-RBF (Stacked Autoencoder Support Vector Machine Based on Piecewise Radial Basis Function) are set; the comparison between the SVM-RBF (Support Vector Machine Based on Radial Basis Function) method and the SVM-P-RBF (Support Vector Machine Based on Piecewise Radial Basis Function) method are set at another level as well. The experimental results are shown in Table 4.

Table 4. Experimental results of different kernel functions

| Method          | AC/%  | DR/%  | FAR/% |
|-----------------|-------|-------|-------|
| SVM-RBF         | 92.551| 91.843| 23.689|
| SVM-P-RBF       | 94.711| 96.071| 10.704|
| SAE-SVM-RBF     | 93.504| 92.971| 20.165|
| SAE-SVM-P-RBF   | 95.275| 98.350| 4.317 |

It can be seen from Table 4 that the P-RBF kernel function embeds the function of the segmentation transform compared with the traditional RBF kernel function. It is based on whether the sample variance of the feature in the sample is 0. When the sample variance of the feature is 0, let the value of the feature be 0 directly; when the sample variance of the feature is not 0, the value of the transformed feature is the difference between the value of the feature before the transformation and the mean of the feature in the sample divided by the sample variance of the features in the sample. And this achieves the goal of normalization. In this way, not only the phenomenon that the range of feature values is too large is eliminated, but also the detection of normal samples and abnormal samples is improved, thereby improving the detection rate and accuracy, and reducing the false positive rate. For the SAE-SVM method, the accuracy, detection rate and false positive rate of SAE-SVM-P-RBF are better than SAE-SVM-RBF, which fully demonstrates the segmentation normalization function of the piecewise kernel function can play a role in improving detection rate, accuracy and false positive rate. Among all these methods, SAE-SVM-P-RBF achieves the best detection rate, accuracy and false positive rate, which is also attributed to the role of SAE feature extraction.

4.4.2. Comparison with other methods. Here are some other experimental results for shallow-learning-based and deep-learning-based intrusion detection methods, which is shown in Table 5.

Table 5. Experimental results for some shallow-learning-based and deep-learning-based intrusion detection methods

| Method                  | AC/%  | DR/%  | FAR/% |
|-------------------------|-------|-------|-------|
| GRNN[10]                | 87.540| 59.120| 12.460|
| RBNN[10]                | 93.050| 69.830| 6.9500|
| LSTM RNN with SGD[11]   | 96.930| 98.880| 10.040|
| LSTM RNN with GDO[12]   | 97.540| 98.950| 9.9800|
It is shown in Table 4 that among 4 methods, the performance of SAE-SVM-P-RBF is the best. However, when compared to shallow-learning-based methods GRNN and RBNN, the performance of SAE-SVM-P-RBF is better, whereas the false alarm rate of SAE-SVM-P-RBF is superior to those of LSTM RNN with SGD and LSTM RNN with GDO. This also demonstrates that the performance of SAE-SVM-P-RBF needs further improvement in the future.

5. Conclusion
An intrusion detection method based on SAE-SVM is proposed. To form SAE-SVM-RBF method and SAE-SVM-P-RBF method, RBF kernel function and P-RBF kernel function are applied. The performance of both methods is compared as well. Experiments show that the role of feature extraction can also be played by using the Stacked Autoencoder as the feature extractor. For the SAE-SVM method, the function of segmentation transforming features is featured with the P-RBF kernel function in the SAE-SVM-P-RBF method. This function will ensure that for the features extracted by the Stacked Autoencoder, segmentation transformation achieves the purpose of improving accuracy, detection rate and false alarm rate. It provides an idea for improving the classification performance of the SAE-SVM method and applying it to the field of intrusion detection.

References
[1] Hinton, G. E., Salakhutdinov R. R. (2006) Reducing the dimensionality of data with neural networks. Science, 313(28):504-507.
[2] Bengio, Y., Lamblin, P., Popovici, Larochelle, H. (2007) Greedy layer-wise training of deep networks. In: Advances in neural information processing systems, pp. 153-160. NIPS, USA.
[3] Gao, N., Gao L., He, Y. Y., Wang H. (2017) A Lightweight intrusion detection model based on autoencoder network with feature reduction. ACTA ELECTRONICA SINICA, 45(3):730-739. (in Chinese)
[4] Wang, X., Wang, L. (2017) Research on Intrusion Detection Based on Feature Extraction of Autoencoder and the Improved K-Means Algorithm. In: 2017 10th International Symposium on Computational Intelligence and Design (ISCID), pp.352-356. IEEE, Hangzhou.
[5] Nguyen, T. V., Tran, N. T., Le, T. S. (2017) An anomaly-based network intrusion detection system using deep learning. In: 2017 International Conference on System Science and Engineering (ICSSSE), pp. 210-214. IEEE, Ho Chi Minh City.
[6] Stolfo, S. J., Fan, W., Lee, W. K. (1999) KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[7] Chang, C. C., Lin, C. J. (2011) LIBSVM: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST), 2(3): 1-27.
[8] Palm, R B. (2014) Deeplearntoolbox, a matlab toolbox for deep learning. https://github.com/rasmusbergpalm/DeepLearnToolbox.
[9] Tan, S. X. (2019) An SVM-based Intrusion Detection Method Using Piecewise Kernel Function. Software Guide, 18(03):43-46+52. (in Chinese)
[10] Le, T., Kim, J., Kim, H. (2017) An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization, In: 2017 International Conference on Platform Technology and Service (PlatCon), pp. 1-6, IEEE, Busan.
[11] Devaraju, S., & Ramakrishnan, S. (2014) Performance comparison for intrusion detection system using neural network with kdd dataset. Ictact Journal on Soft Computing, 4(3), 743-752.
[12] Kim, J., Kim, J., Thu, H. L. T., Kim, H. (2016) Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection, In: 2016 International Conference on Platform Technology and Service (PlatCon). IEEE, Jeju.