Journal of Southeast University (English Edition) Vol. 31, No. 3, pp. 339 - 346 Sept. 2015 ISSN 1003—7985

Intrusion detection model based on deep belief nets

Gao Ni[1] Gao Ling[1] He Yiyue[1,2] Gao Quanli[1] Ren Jie[1]

([1]School of Information Science and Technology, Northwest University, Xi'an 710127, China)

([2]School of Economics and Management, Northwest University, Xi'an 710127, China)

Abstract: This paper focuses on the intrusion classification of huge amounts of data in a network intrusion detection system. An intrusion detection model based on deep belief nets (DBN) is proposed to conduct intrusion detection, and the principles regarding DBN are discussed. The DBN is composed of a multiple unsupervised restricted Boltzmann machine (RBM) and a supervised back propagation (BP) network. First, the DBN in the proposed model is pre-trained in a fast and greedy way, and each RBM is trained by the contrastive divergence algorithm. Secondly, the whole network is fine-tuned by the supervised BP algorithm, which is employed for classifying the low-dimensional features of the intrusion data generated by the last RBM layer simultaneously. The experimental results on the KDD CUP 1999 dataset demonstrate that the DBN using the RBM network with three or more layers outperforms the self-organizing maps (SOM) and neural network (NN) in intrusion classification. Therefore, the DBN is an efficient approach for intrusion detection in high-dimensional space.

Key words: intrusion detection; deep belief nets; restricted Boltzmann machine; deep learning

doi: 10.3969/j.issn.1003-7985.2015.03.007

With the growth of network technologies and applications, computer network security has become a crucial issue that needs to be addressed. How to identify network attacks is a key problem. As an important and active security mechanism, intrusion detection (ID) has become a key technology of network security, and has drawn the attention of domestic and foreign scholars world-wide. Intrusion detection based on different machine learning methods is a major research project in network security, which aims at identifying unusual access or attacks on internal networks[1].

In literature, many machine learning methods have made great achievements in IDS, such as the NN[2], support vector machine (SVM)[3] and SOM[4]. These methods have been introduced to the field of intrusion detection by previous researchers. Most of traditional learning machine methods with shallow architectures have one hidden layer (e.g., BP) or no hidden layer (e.g., maximum entropy model).

Owing to limited samples and computing cells, the expressive power of shallow learning methods for complex function is limited, and its generalization ability for complex classification problems is subjected to certain constraints[5]. The continuous collection of traffic data by the network leads to problems concerning the huge amounts of data in network intrusion detection and prediction. Therefore, how to develop an efficient intrusion detection model oriented towards huge amounts of data is a theoretical and practical problem that should be solved urgently.

The deep learning model used in large-scale data analysis has an outstanding performance, which is a promising way of solving intrusion detection problems. The DBN with deep architectures is proposed by Hinton et al.[6], and it uses a learning algorithm which greedily trains one layer at a time with an unsupervised learning algorithm employed for each layer[6]. Bengio et al.[7] followed the research on deep learning, which shows the strong capacity of learning essential characteristics of data sets from a few samples. Although DBN can be trained with unlabeled data, DBN is successfully used to initialize deep feedforward neural networks.

In this paper, an intrusion detection model based on the greedy layer-wise DBN is presented. In order to improve the performance of DBN, its parameters are explored, including the depth of DBN, the number of nodes in the first hidden layer, the number of nodes in the output layer and so on. Finally, the efficiency of DBN is evaluated on the KDD CUP 1999 dataset. The DBN outperforms SOM and NN in detection accuracy and false positive rate.

1 Proposed DBN Model for Intrusion Detection

An overall framework of the network intrusion detection model based on DBN is trained in three stages, as shown in Fig. 1. First, the symbolic attribute features in KDD CUP 1999 dataset[8] are numeralized and subsequently normalized. Then, a DBN is trained on the standardized dataset and the weights of the DBN are used to initialize a neural network. This pre-training procedure consists of learning a stack of RBMs with the unsupervised contrastive divergence algorithm. The learned fea-

Received 2015-02-25.

Biographies: Gao Ni (1982—), female, graduate; Gao Ling (corresponding author), male, doctor, professor, gl@nwu.edu.cn.

Foundation items: The National Key Technology R&D Program during the 12th Five-Year Plan Period (No. 2013BAK01B02), the National Natural Science Foundation of China (No. 61373176), the Scientific Research Projects of Shaanxi Educational Committee (No. 14JK1693).

Citation: Gao Ni, Gao Ling, He Yiyue, et al. Intrusion detection model based on deep belief nets[J]. Journal of Southeast University (English Edition), 2015, 31(3): 339 - 346. [doi: 10.3969/j.issn.1003-7985.2015.03.007]

ture activations of one RBM are used as the data for training the next RBM in the stack. The nonlinear high-dimensional input vector is sampled as its corresponding optimal low-dimensional output vector. Finally, a DBN is constructed by unrolling the RBMs and fine-tuned by using the BP algorithm of error derivatives according to the class labels of the input vectors. The obtained DBN can be used to recognize attacks.

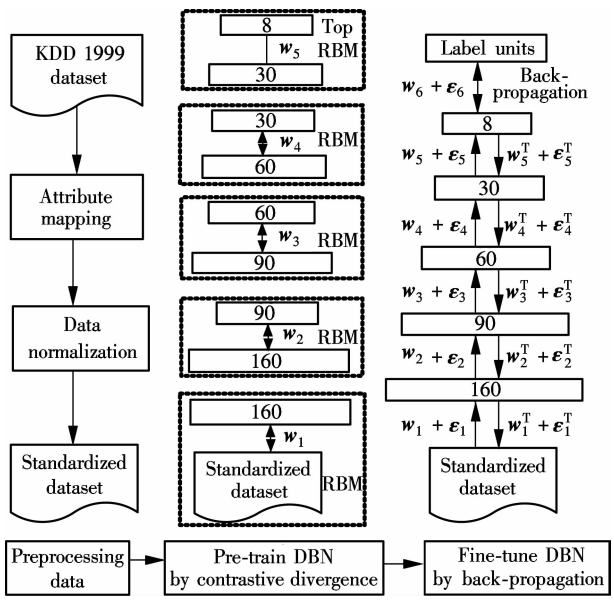

Fig. 1 The pipeline for training IDS

2 Deep Belief Network

The deep learning model attracts more attention from researchers at home and abroad due to its outstanding learning ability for the complex data[6]. A deep learning model containing multilayer hidden units is used to gradually establish the optimal abstract representation of the raw input at the penultimate layer in Fig. 2. The representative deep learning model is the DBN based on the RBM in stacks. Learning DBN is a process of greedy layer-wise training RBM.

The DBN[6], which is a probabilistic generative model, is a deep neural network classifier of a combination of multilayer unsupervised learning networks named RBM and a supervised learning network named BP[9]. In a DBN, the units in each layer are independent given the values of the units in the layer above. Fig. 2 shows a multilayer generative model, in which the top two layers have symmetric undirected connections and the lower layers receive directed top-down connections from the layer above. The bottom layer is observable, and the multiple hidden layers are created by stacking multiple RBMs on top of each other. The upward arrows represent the recognition model, and the downward arrows represent the generative model. In the greedy initial learning, the recognition connections are tied to the generative connections. In the learning process of the generative model, once the weight parameter $\boldsymbol{W}$ is learned, the original data $\boldsymbol{v}$ can be mapped through $\boldsymbol{W}^{\mathrm{T}}$ to infer factorial approximate posterior distributions over the states of the variables in the first hidden layer h_1. A DBN with n layers can be represented as a graphical model. The joint distribution of the visible layer $\boldsymbol{v}$ and the hidden layer h_k, for $k = 1:n$, is defined as

$$p(\boldsymbol{v}, h_1, \ldots, h_n) = p(\boldsymbol{v} \mid h_1) \prod_{k=1:n-2} p(h_k \mid h_{k+1}) p(h_{n-1} \mid h_n) \tag{1}$$

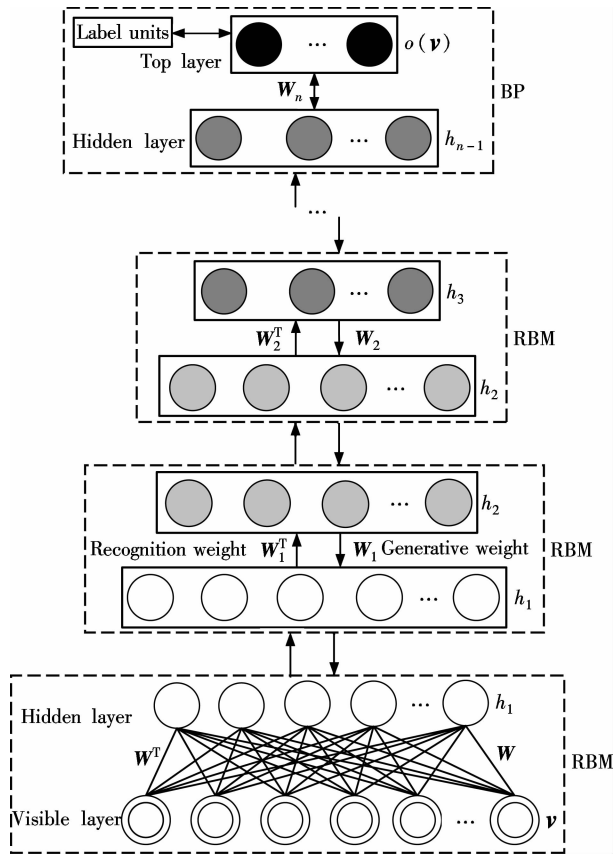

Fig. 2 A graphical representation of a DBN and its parameters

In a bottom-up process, the recognition connections of DBN can be used to infer a factorial representation in one layer from the binary activities in the layer below. In a top-down process, the generative connections of DBN are used to map a state of the associative memory to the raw input. The DBN performs a non-linear transformation on its input vectors and produces low-dimensional output vectors. Using the greedy initial learning algorithm, the original data are perfectly modeled and a better generative model can be acquired.

The procedure of the training DBN consists of two phases. In the first phase, a layer-wise greedy learning algorithm is applied to pre-train a DBN, and the RBM of each layer is trained by the CD algorithm[10]. To obtain an approximate representation of the input vector $\boldsymbol{v}$, the following procedure is used. First, the posterior distribu-

tion $p(h_1 \mid \boldsymbol{v})$ is sampled from the first-level RBM, and the visible variables $\boldsymbol{v}$ are sampled by the posterior distribution $p(\boldsymbol{v} \mid h_1)$. Subsequently, the hidden variables h_1 are sampled repeatedly in the same way. The alternating Gibbs samplings are repeatedly performed k times until an equilibrium distribution is arbitrarily approached. The optimal representation h_1 of the input vector $\boldsymbol{v}$ becomes the input for learning the second-level RBM, and a sample h_2 is computed. The former procedure is repeatedly executed to train each RBM in a bottom-up way until h_{n-1} is computed. A DBN is trained by using the layer-wise greedy learning method which can be recursively repeated, and millions of parameters can be learned efficiently. In practice, applying the CD algorithm can avoid the monstrous overall complexity of training DBN, so this algorithm is efficient.

In the second phase, the parameters of the whole DBN are fine-tuned. The weights on the undirected connections at the top-level RBM are learned by fitting the posterior distribution of the penultimate layer. Using the BP learning algorithm, exact gradient descent on a global supervised cost function between the actual output vector and the desired output vector can be performed in DBN. This phase aims at obtaining the optimal parameters, which corresponds to the minimized difference between the above two vectors.

3 Restricted Boltzmann Machine

3.1 Model structure

The core block networks for the DBN are RBMs, each of which is a two-layer neural network that consists of a visible layer and a hidden layer. Each unit of the hidden layer connects to all the units of the visible layer, and the visible and hidden units form a bipartite graph with no visible-visible or hidden-hidden connections. RBM is an energy-based undirected generative model that uses a layer of binary variables to explain its input data. The visible variables are described as the characteristics of the input data, and the hidden variables automatically generated through machine learning often have no actual meaning. The undirected model is a binary stochastic neuron, meaning that each of them can latch onto one of two states: on or off. This model is called RBM whose probability distribution obeys the Boltzmann distribution.

3.2 Inference of RBM parameters

RBM is an energy-based undirected generative model, which is constructed from a set of visible variables $\boldsymbol{v} = \{v_i\}$ and a set of hidden variables $\boldsymbol{h} = \{h_j\}$, as shown in Fig. 3. Node i is in the visible layer, and node j is in the hidden layer. A property of RBM is the lack of direct connections within nodes of the same layer, while there are connections between the visible layer and the hidden layer. The visible units are conditionally independent given the hidden units states, and vice versa. Therefore, the posterior distributions $P(\boldsymbol{H} \mid \boldsymbol{V})$ and $P(\boldsymbol{V} \mid \boldsymbol{H})$ are sampled and factorized as

$$P(\boldsymbol{H} \mid \boldsymbol{V}) = \prod_j p(h_j \mid \boldsymbol{v}) \tag{1}$$

$$P(\boldsymbol{V} \mid \boldsymbol{H}) = \prod_i p(\boldsymbol{v}_i \mid \boldsymbol{h}) \tag{2}$$

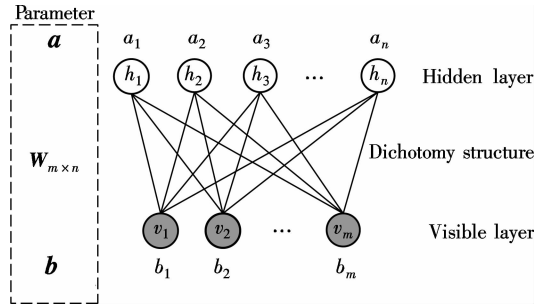

Fig. 3 A graphical representation of RBM and its parameters

Given a visible $\boldsymbol{v}$, a low-dimensional representation $\boldsymbol{h}$ can be sampled by the posterior distributions $p(\boldsymbol{h} \mid \boldsymbol{v})$. Therefore, given a hidden variable $\boldsymbol{h}$, a new representation $\boldsymbol{v}$ can be sampled by the posterior distributions $p(\boldsymbol{v} \mid \boldsymbol{h})$. Since $h_j \in \{0, 1\}$, the binary hidden unit probabilities are given as

$$\left.\begin{aligned} p(h_j = 1 \mid \boldsymbol{v}) &= \sigma\Big(\sum_i w_{ij} v_i + a_i\Big) \\ p(h_j = 0 \mid \boldsymbol{v}) &= 1 - p(h_j = 1 \mid \boldsymbol{v}) \end{aligned}\right\} \tag{3}$$

Since the RBM is a completely symmetric derivation, the binary visible unit probabilities are given as

$$\left.\begin{aligned} p(v_i = 1 \mid \boldsymbol{h}) &= \sigma\Big(\sum_j w_{ij} h_j + b_j\Big) \\ p(v_i = 0 \mid \boldsymbol{h}) &= 1 - p(v_i = 1 \mid \boldsymbol{h}) \end{aligned}\right\} \tag{4}$$

where σ denotes the logistic sigmoid,

$$\sigma(y) = \frac{1}{1 + \mathrm{e}^{-y}} \tag{5}$$

The visible units and the hidden units are assumed to be the binary stochastic units. In an RBM, the energy function with every configuration of visible and hidden variables is defined as

$$E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = -\boldsymbol{v}^{\mathrm{T}} \boldsymbol{W} \boldsymbol{h} - \boldsymbol{b}^{\mathrm{T}} \boldsymbol{v} - \boldsymbol{a}^{\mathrm{T}} \boldsymbol{h} \tag{6}$$

where $\boldsymbol{W}$ is the weight matrix between the visible variable $\boldsymbol{v}$ and the hidden variable $\boldsymbol{h}$; $\boldsymbol{b}$ is a visible variable bias; $\boldsymbol{a}$ is a hidden variable bias; and the parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}\}$ of the energy function are learned. The probability of any particular configuration of the visible and hidden units is denoted by the following energy function:

$$p(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta}) = \mathrm{e}^{-E(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\theta})} \tag{7}$$

4 Learning by Minimizing Contrastive Divergence

The optimal joint probability distribution can be acquired with the Markov chain method on the condition that the number of iterations is close to infinity. However, it is difficult to guarantee fast convergence and determine the step size of the iteration. The maximization process of log likelihood is the same as the minimization process of the Kullback-Leibler (KL) divergence, which is expressed as $\mathrm{KL}(P^0 \parallel P_\theta^\infty)$[10]. Here, P^0 denotes the posterior distribution of the data, and P_θ^∞ denotes the equilibrium distribution defined by RBM. Contrastive divergence proposed by Hinton[10] is a fast RBM training method. Instead of minimizing $\mathrm{KL}(P^0 \parallel P_\theta^\infty)$, the minimized difference between $\mathrm{KL}(P^0 \parallel P_\theta^\infty)$ and $\mathrm{KL}(P^n \parallel P_\theta^\infty)$ is defined as

$$\mathrm{CD} = \mathrm{KL}(P^0 \parallel P_\theta^\infty) - \mathrm{KL}(P_\theta^n \parallel P_\theta^\infty) \tag{8}$$

where P_θ^n denotes the posterior distribution of the reconstructive visible variables sampled by n steps of Gibbs sampling. The surprising empirical result is that even $n = 1$ often gives a good result. $\mathrm{KL}(P^0 \parallel P_\theta^\infty)$ exceeds $\mathrm{KL}(P_\theta^n \parallel P_\theta^\infty)$ unless $P^0 = P_\theta^1$. Since the contrastive divergence is positive, P^0 is equivalent to P_θ^1. The contrastive divergence is equal to zero only when the RBM model is at equilibrium.

The parameters of the model, $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}\}$, can be adjusted in proportion to the approximate derivative of the contrastive divergence:

$$\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n + \varepsilon(\langle v_i^0 h_j^0 \rangle - \langle v_i^n h_j^n \rangle) \tag{9}$$

It works relatively well in practice that the original vector data is reconstructed only by one Gibbs step[10]. The updated parameters are given as

$$\boldsymbol{\theta}^n = \boldsymbol{\theta}^{n-1} + \varepsilon(\langle v^0 h^0 \rangle - \langle v^1 h^1 \rangle) \tag{10}$$

Using an alternative CD learning algorithm, the high-dimensional data is always close to the low-dimensional data. The procedure of the fast CD-k learning algorithm is listed as follows.

Algorithm 1 TrainRBM ($\boldsymbol{V}$, ε, M, N, $\boldsymbol{W}$, $\boldsymbol{a}$, $\boldsymbol{b}$)

Input: $\boldsymbol{V}$ is a sample from the training for the RBM, $\boldsymbol{V} = \{v_1, v_2, \cdots, v_M\}$; ε is the learning rate for the stochastic gradient descent in CD; M is the number of the RBM viable units; N is the number of the RBM hidden units; $\boldsymbol{W}$ is the RBM weight matrix; $\boldsymbol{a}$ is the RBM offset vector for viable units; $\boldsymbol{b}$ is the RBM offset vector for hidden units. Initialize parameter $W_{ij} = a_i = b_j = 0$, $i = 1, 2, \cdots, M, j = 1, 2, \cdots, N$; set the number of iterations T and the number of steps of Gibbs sampling K.

for $t = 0$ to T do
  for each training example v_i, $m = 1$ to M do
    $v^0 = v_i$
    for $k = 0$ to $K - 1$ do
      for each hidden variable h_j, $n = 1$ to N
        sample $h_j^{(k)} \sim p(h_j \mid v^{(k)})$
      end for
      for each viable variable v_i, $m = 1$ to M
        sample $v_i^{(k+1)} \sim p(v_i \mid h^{(k)})$
      end for
      for $i = 1, 2, \cdots, M; j = 1, 2, \cdots, N$ do
        $W_{ij}^{(k+1)} + \varepsilon(p(h_j \mid v^{(0)}); v_i^{(0)} - p(h_j \mid v^{(k)}) v_i^{(k)})$
        $a_i^{(k+1)} = a_i^{(t)} + \varepsilon(v_i^{(0)} - v_i^{(k)})$
        $b_j^{(k+1)} = b_j^{(k)} + \varepsilon(p(h_j \mid v^{(0)}) - p(h_j \mid v^{(k)}))$
      end for
    end for
  end for
end for

5 Fine-Tuning All Layers of DBN by Back-Propagation Algorithm

The weight matrices, which are pre-trained at each layer by contrastive divergence learning, are efficient but not optimal. Therefore, the unsupervised layer-by-layer training algorithm is performed for each RBM network, and the final supervised fine-tuning learning is used to adjust all the parameters simultaneously. The BP algorithm for feed-forward multi-layered neural network plays a key role in fine-tuning the weights of the connections in the DBN[9]. Fine-tuning a DBN based on the BP algorithm consists of two phases. In the first phase, in order to obtain better initialization parameters, the feed-forward DBN is trained by the RBM learning algorithm based on k-step contrastive divergence. In the second phase, a measure of the difference between the actual output vector of DBN and the desired output vector is minimized, and the weight matrices are repeatedly adjusted during the down-pass. The down-pass, which propagates derivatives from the top layer back to the bottom layer, employs the top-down generative connections to activate each lower RBM layer in turn. Then the procedure of the fine-tuning learning algorithm based on the BP algorithm is listed as follows.

Algorithm 2 FineTuneDBN (example, l, numhid, $\varepsilon_{\text{fine-tune}}$, $\boldsymbol{W}$, $\boldsymbol{a}$, $\boldsymbol{b}$)

Input: Example is a training set of $\langle v_i, t_i \rangle$ ($i = 1, 2, \cdots, m$); l is the number of RBM layers; numhid is a set of hidden units $\{\text{numhid}_1, \text{numhid}_2, \cdots, \text{numhid}_l\}$ at each RBM layer; $\varepsilon_{\text{fine-tune}}$ is a learning rate for the DBN training; $\boldsymbol{W}^k$ is the weight matrix for RBM at level k, for k from 1 to l; $\boldsymbol{a}^k$ is the offset vector for viable units for RBM at level k, for k from 1 to l; $\boldsymbol{b}^k$ is the offset vector for hidden units for RBM at level k, for k from 1 to l.

% Forward propagation

$\boldsymbol{V} = \boldsymbol{v}, M = m$

for $k = 1$ to l do
  initialize $\boldsymbol{W}^k = \boldsymbol{a}^k = \boldsymbol{b}^k = \boldsymbol{0}, \varepsilon = \varepsilon_0 = 0.5$
  TrainRBM($\boldsymbol{V}, \varepsilon, M, \text{numhid}_k, \boldsymbol{W}^k, \boldsymbol{a}^k, \boldsymbol{b}^k$)
  $M = \text{numhid}_k$
  $\varepsilon = \varepsilon/(1 + \varepsilon)$
end for
for $i = 1, 2, \cdots, m$ do
  compute $o_i(x_i)$
end for
% Backward gradient propagation and parameter update
for $k = l$ to 1
  if $k = l$
    $\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$
  else
    $\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in \text{outputs}} \theta_{kh}\delta_k$
    $\theta_{ji} \leftarrow \theta_{ji} + \Delta\theta_{ji}, \theta_{ji} = \varepsilon_{\text{fine-tune}}\delta_j x_j$
  end if
end for

6 Experimental Results

6.1 Benchmark dataset description

The KDD Cup 1999 dataset[8], which is provided by the Defense Advanced Research Projects Agency (DARPA) and contains the attack data of several weeks, is employed to assess the performance of various IDS.

The KDD Cup 1999 dataset contains 494 021 records in the training data and 11 850 records in testing data. The data distribution of the dataset is shown in Fig. 4.

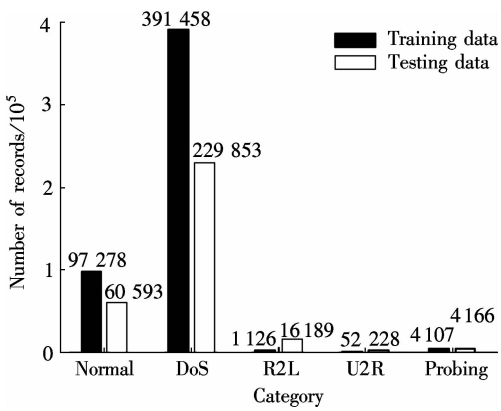

Fig. 4 Attacks distribution in the KDD Cup 1999 dataset

6.2 Data preprocessing

Each record of the KDD Cup 1999 dataset, which is labeled as either normal or one specific kind of attack, is described as a vector with 41 attribute values. Those attributes consist of 38 continuous or discrete numerical attributes and 3 categorical attributes. Deep belief networks require floating point numbers for the input neurons, and the values of floating point numbers range from 0 to 1. Therefore, all features are preprocessed to fulfill this requirement. Preprocessing data consists of two phases as follows:

1) Numeralization of symbolic features. Three symbolic features, including protocol type, service type and flag type, are converted into binary numeric features. For example, protocol type "tcp" is converted into binary numeric features vector $\{1,0,0\}$, "udp" is converted into vector $\{0,1,0\}$, and "icmp" is converted into vector $\{0,0,1\}$. Since the features "service type" can be expanded into 70 binary features and the features "flag type" can be expanded into 11 binary features. Finally, 41 attributes are numeralized as 122 attributes.

2) Normalization of numeric features. Each numerical value obtained in the first phase is normalized to the interval $[0,1]$, according to the following data smoothing method.

$$y = \frac{y - M_{\min}}{M_{\max} - M_{\min}} \tag{11}$$

where y is a numerical value; $M_{\min}$ is the minimum value for the attribute that y belongs to; and $M_{\max}$ is the maximum value for the attribute that y belongs to.

6.3 Evaluation measurement

An IDS requires high accuracy, high detection rate and low false alarm rate. In general, the performance of IDS is evaluated in terms of accuracy A_C, detection rate D_R, and false alarm F_A.

$$A_C = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{12}$$

$$D_R = \frac{T_P}{T_P + F_N} \tag{13}$$

$$F_A = \frac{F_P}{T_N + F_P} \tag{14}$$

where true positive T_P is the number of attack records correctly classified; true negative T_N is the number of normal records correctly classified; false positive F_P is the number of normal records incorrectly classified; false negative F_N is the number of attack records incorrectly classified.

The squared reconstruction error of the raw input is often used to monitor its performance. The squared reconstruction error is defined as

$$e = \frac{\sum_{k=1}^{n}\left(\sum_{i=1}^{122}(v_{ki} - v'_{ki})^2\right)}{n} \tag{15}$$

where v_{ki} is the i-th component belonging to the k-th sample vector; v'_{ki} is the i-th component belonging to the k-th reconstructed sample vector; n is the total number of samples, and the number of attributes after data preprocessing is 122.

6.4 Experimental results and analysis

Experiments are designed and implemented, in which

the KDD Cup 1999 dataset is used to evaluate the performance of the proposed model. All programs are coded in Matlab 7.0 and run in a personal computer with an Intel CPU 1.86 GHz and 2 GB memory.

It is important to determine a proper iteration number. With the increase of iteration numbers, the detection rate increases accordingly, as shown in Fig. 5. DBN can be expressed as DBN_i, where i denotes the number of RBM layers. The deep DBN_4 is used to evaluate the detection rates corresponding to iteration times from 10 to 500. The curve of detection rate shown in Fig. 5 appears to increases and then stabilizes. If the iteration number is greater than 150, the curve will be smoother.

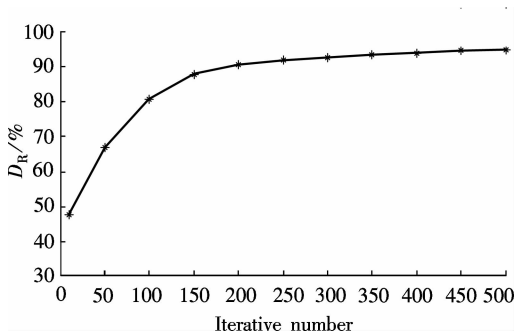

Fig. 5 The iteration number of RBM training

Tab. 1 compares the performances of DBN, SOM and NN on the KDD Cup 1999 dataset. Four different DBNs, including a shallow 122-5 DBN_1, a shallow 122-60-5 DBN_2, a deep 122-80-40-5 DBN_3 and a deep 122-110-90-50-5 DBN_4, are selected. According to the results in Tab. 1, the detection rates of the shallow DBN_1 and DBN_2 are not better than that of SOM, but the detection rate of the deep DBN_3 added one layer RBM is higher than those of SOM and NN. Therefore, the A_C of the deep 122-110-90-50-5 DBN_4 is improved by 2.67% and 6.19%, respectively, compared with SOM and NN. The D_R of the deep DBN_4 is improved by 2.76% and 5.7%, respectively. The F_A of the deep DBN4 is improved by 0.4% and 0.55%, respectively. Therefore, DBN using the RBM network with three or more layers outperforms SOM and NN in A_C, D_R and F_A.

Tab. 1 Performance comparison of the six network structures %

Model	A_C	D_R	F_A
DBN_1	74.22	75.60	3.15
DBN_2	82.65	81.30	2.72
DBN_3	90.97	89.69	1.12
DBN_4	93.49	92.33	0.76
SOM	90.82	89.57	1.16
NN	87.30	86.63	1.31

Larochelle et al. [11] argued that the number of nodes in the first hidden layer has a significant influence on classification performance. Fig. 6 compares the performances of DBN with different number of nodes in the first hidden layer when a deep 122-110-90-50-5 DBN_4 is set. According to the results in Fig. 6, the classified accuracy is the best when the number of nodes in the first hidden layer is set to be 110.

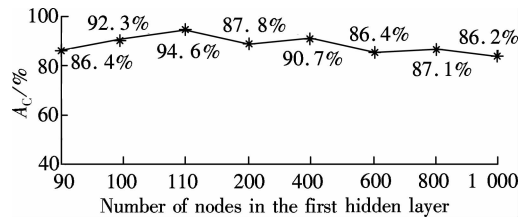

Fig. 6 Performance comparison of DBN with different numbers of nodes in the first hidden layer

Another important exploration is to choose an optimal number of nodes in the output layer to improve the performance of intrusion detection. In Fig. 7, a deep 122-110-90-50-5 DBN_4 is selected, with the different numbers of nodes in the output layer set from 1 to 10. According to the results in Fig. 7, the classification accuracy and detection rate are the optimal when the number of nodes in the output layer is set to be 5.

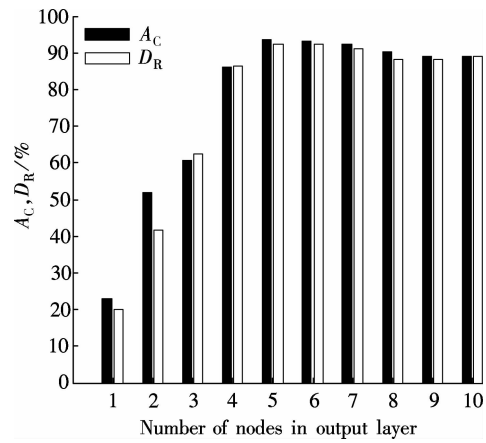

Fig. 7 Performance comparison of DBN with different numbers of nodes in the output layer

The intrusion classification experiment is performed with different types of attacks in the KDD CUP 1999 dataset. Tab. 2 compares the performances of DBN_4, SOM and NN. In Tab. 2, the accuracy rate of the DBN_4 is improved by 2.54% and 5.525% on average, respectively, compared with SOM and NN, and the false alarm rate of the DBN_4 is improved by 0.56% and 0.72%, respectively. The experimental results show that deep DBN_4 can effectively enhance the IDS detection rate and reduce their error rate.

Fig. 8 compares the squared reconstruction errors of pre-trained DBN_4 and randomly initialized DBN_4, both of which have the same parameters. As shown in Fig. 8, the DBN_4 with pre-training can make the fine-tuning faster than that without pre-training. After 100 iterations in the fine-tuning, the average squared reconstruction error per input vector of pre-trained DBN_4 is smaller than that of

Tab. 2 Performance comparison of three classifiers on different types of attacks %

Attack type	NN		SOM		DBN_4	
	A_C	F_A	A_C	F_A	A_C	F_A
DoS	89.76	1.19	93.34	1.12	95.60	0.96
R2L	85.63	3.62	89.57	3.22	93.30	1.24
U2L	86.91	0.92	88.11	0.83	89.69	0.78
Prob	87.52	0.76	90.75	0.69	93.33	0.62

randomly initialized DBN_4, as shown in Fig. 8. If the iteration number is greater than 450, the curve basically maintains stable. The difference of the average squared reconstruction error between the pre-trained and randomly initialized is about 4.36. The experimental results show that using pre-training and fine-tuning can improve the performance in IDS classification.

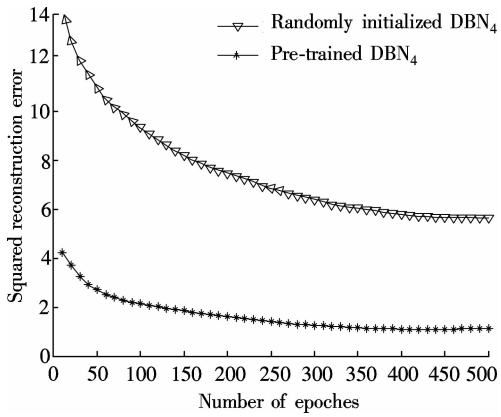

Fig. 8 Comparison of the squared reconstruction error

Another challenge in the classification of huge amounts of data using the proposed model is the real-time analysis and processing of data in a short period of time. Therefore, the assessment of the proposed model is crucial. In order to increase the number of experimental data, duplicate records are randomly added to the KDD CUP 1999 dataset, and the running time of our computer using the DBN_4 model is recorded, as shown in Fig. 9. The experimental results show that the running time approximately increases linearly as the number of records increases.

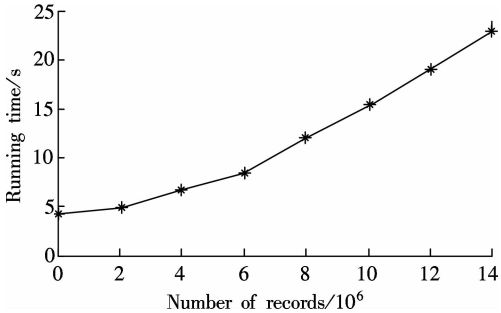

Fig. 9 The scalability of DBN_4

7 Conclusion

This paper aims at demonstrating that DBN can be successfully applied in the field of intrusion detection. DBN can not only extract features from high-dimensional representations but also efficiently perform classification tasks. This paper explores the idea of applying DBN to classify attacks accurately. The performance of DBN is evaluated by experiments, and DBN is compared with other machine learning models, such as SOM and NN. Finally, the experimental results on the KDD CUP 1999 dataset show that a good generative model can be acquired by DBN, which performs well on the intrusion recognition task. To some extent, the DBN, which can replace the traditional shallow machine learning, provides a new design idea and method for future IDS research.

References

[1] Kuang F, Xu W, Zhang S. A novel hybrid KPCA and SVM with GA model for intrusion detection[J]. *Applied Soft Computing*, 2014, **18**(4):178 - 184.

[2] Beqiri E. Neural networks for intrusion detection systems [J]. *Global Security Safety & Sustainability*, 2009, **45**: 156 - 165.

[3] Ahmad I, Abdullah A, Alghamdi A, et al. Optimized intrusion detection mechanism using soft computing techniques[J]. *Telecommunication Systems*, 2013, **52**(4): 2187 - 2195.

[4] Depren O, Topallar M, Anarim E, et al. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks[J]. *Expert Systems with Applications*, 2005, **29**(4):713 - 722.

[5] Bengio Y. Learning deep architectures for AI[J]. *Foundations and Trends in Machine Learning*, 2009, **2**(1):1 - 127.

[6] Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets [J]. *Neural Computation*, 2006, **18**(7):1527 - 1554.

[7] Bengio Y, Lamblin P, Popovici D, et al. Greedy layer-wise training of deep networks[C]//*Advances in Neural Information Processing Systems*. Vancouver, Canada, 2006:153 - 160.

[8] Stolfo S J, Fan W, Lee W K, et al. Cost-based modeling for fraud and intrusion detection: results from the JAM project [EB/OL]. (1999-10-28) [2011-06-27]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[9] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. *Nature*, 1986, **323**(6088):533 - 536.

[10] Hinton G E. Training products of experts by minimizing contrastive divergence[J]. *Neural Computation*, 2002, **14**(8):1771 - 1800

[11] Larochelle H, Bengio Y, Louradour J, et al. Exploring strategies for training deep neural neural networks [J]. *Journal of Machine Learning Research*, 2009, **10**(1):1 - 40.

基于深度信念网络的入侵检测模型

高 妮[1] 高 岭[1] 贺毅岳[1,2] 高全力[1] 任 杰[1]

([1] 西北大学信息科学与技术学院, 西安 710127)
([2] 西北大学经济管理学院, 西安 710127)

摘要:研究了入侵检测系统中海量数据分类的问题. 讨论了深度信念网络(DBN)的原理,提出了基于 DBN 的入侵检测模型. DBN 由多层无监督的限制玻尔兹曼机(RBM)网络和一层有监督的反向传播(BP)网络构成. 该入侵检测模型采用一种快速、贪婪的方法对 DBN 网络进行预训练,利用对比分歧算法逐层训练每一个 RBM 网络;然后,利用有监督的 BP 算法对整个 DBN 网络进行微调,并同时对 RBM 网络输出的低维特征进行入侵数据分类. 基于 KDD CUP 1999 数据集的实验结果表明,使用 3 层以上的 DBN 模型分类效果优于自组织映射和神经网络方法. 因此,DBN 是一种有效且适用于高维特征空间的入侵检测方法.

关键词:入侵检测;深度信念网络;限制玻尔兹曼机;深层学习

中图分类号:TP393. 08