Send Orders for Reprints to reprints@benthamscience.ae

An Anomaly Detection Method Based On Deep Learning

Hong-li Deng*, Tao yang and Jiang-jin Gao

China West Normal University, NanChong Sichuan, China

Abstract: In order to overcome the difficulty of extracting features from data and improve the accuracy of anomaly detection system, this paper proposes a novel anomaly detection method based on deep learning. We build a deep neural network model with multiple hidden layers to automatically learn features of data before detecting anomaly behaviors. The learned features from this network can enhance the discrimination of different behaviors. Moreover, an exactly sparse auto-encoder (ESAE) is proposed to achieve the pre-training of this network. This method does not require manual extraction of features, and is unsupervised, avoiding the difficulty of providing labeled data. Experimental results show that the proposed method could significantly improve the detection accuracy.

Keywords: Deep learning, Anomaly detection, Feature representation, Sparse auto-encoder.

1. INTRODUCTION

Intrusion detection technology [1] is to prevent or reduce the threat of cyber-attacks, under the condition that the network performance is not affected. It can be divided into misuse detection and anomaly detection technology. Compared to misuse detection, anomaly detection can detect unknown attacks, so researches get more attentions on that. At present, most anomaly detection methods can be summarized as follows: all the instances are represented as data points according to their features (attributes), and then the classic pattern classification algorithms such as neural network, decision tree, cluster analysis and Bayesian theory, support vector machine (SVM), K nearest neighbor algorithm (KNN) are used to classify these points [2-4]. We can see that these features are the raw materials of the classification system. Good features play a key role for improving the accuracy of various anomaly detection algorithms.

Geoffrey Hinton, professor of university of Toronto, and a leader in the field of machine learning, has published a paper in science [5] about deep learning, which points out that deep model has outstanding advantages on learning good features. The features learned by this model can represent data with richer information. The basic idea of this model [6] is to stack multiple nonlinear functions, combining low-level features to form more abstract and more useful high-level features. It maps all the samples from their original data space into a new feature space which can facilitate the classification.

Therefore, in order to improve the performance of various anomaly detection algorithms, we proposed to use deep model to learn better and richer features of data before inputting them into anomaly detection system. By building deep neural network model with more hidden layers, the features of original data are extracted automatically from the bottom to the top layer. And then these features are used as the input of any existing anomaly detection algorithms. Good feature representation of original data will help improve the performance of most algorithms. Moreover, these representations are learned completely unsupervised, overcoming the difficulty of lack of labeled data. In order to validate the effectiveness of our method, two anomaly detection algorithms based on this method were given. Experimental results show that the features learned by deep model can significantly improve the detection accuracy compared to original anomaly detection algorithms.

2. ANOMALY DETECTION ARCHITECUTRE BASED ON DEEP LEARNING

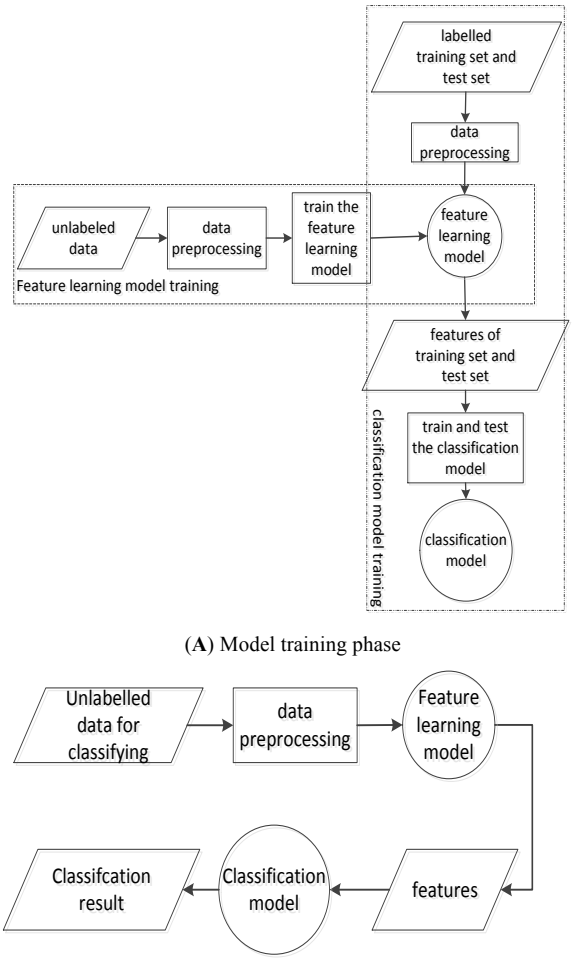
In this paper, we propose to learn the features of data based on deep neural networks. Instead of the attributes designed artificially, these features learned from these networks will be input into the classification model to detect abnormal behaviors.

As shown in Fig. (**1**), this anomaly detection method includes two stages. One is the training phase (Fig. **1A**), which includes the training tasks for feature learning model and classification model. The training of feature learning model is carried out by unsupervised method, while classification model is trained with labeled data. Well trained feature learning model can be directly applied to learn feature of new data. The classification model here can be any anomaly detection algorithm that can be used for the classification, such as KNN, neural network and SVM, and so on. Data preprocessing in this phase is to execute standardization and normalization of input data, as well as data type conversion. Another phase is anomaly detection using the well trained models before. As shown in Fig. (**1B**), when we detect abnormal behaviors, we first preprocess the input data, and then the trained feature learning model in first stage is used to learn the features of data. Finally, these

*Address correspondence to this author at the Shida Road, Nanchong, China. Postcard: 637002; Tel: +86 18090574679; E-mail: 93425538@qq.com

1874-4443/15

features are input to the classifier to predict normal or abnormal information.



(**A**) Model training phase

(**B**) Anomaly detection phase

Fig. (1). Anomaly detection process based on deep learning

3. RESEARCH METHOD

From the section 2, we can see that the main task of our model is how to train a good feature learning model. However, it is well known that, the training of deep neural network has been very difficult before 2006. The objective function, which has many local optimal values [5], is very hard to optimize. Whether or not the network can achieve the optimal solution, the initial parameter has a pivotal role. If the initial value is not good, neural network is easy to fall into local optimum. Until 2006, Geoffrey Hinton put forward that the difficulty of training neural network can be effectively overcome by "layer-wise training" [4]. Initialize network one layer after one layer, and finally form the initial value of the entire network. This initialization will help the gradient descent begin at a better initial search point, so as to converge to better local optima. The process to get this initialization is the pre-training of the network.

Therefore, most important task of our work is to pre-train our deep neural network, so as to get one better initialization. In this section, we will first introduce the whole process of network training. Then we will focus on one frequently-used pre-training method and our improvement on it.

3.1. Training Process

The basic rule for training deep neural network is as follows: first pre-train network by an unsupervised training method (*i.e.*, layer-wise training for initialization); then stack multiple layers that has been initialized to form a deep network; finally fine-tune this pre-trained deep network to get the feature learning model. This paper follows the same principle. The first step, also the most important step, is to use unlabeled data (labeled data can also be used) for pre-training network. This process of pre-training can be described as follows:

Algorithm 1:

Starting from the layer 2 (i = 2)

(1) Use the feature in layer i - 1 (that value of layer 1 is the original data) as the input to train present layer i, so as to learn the encoding parameters at this layer $W^{(i)}$ $b^{(i)}$*, which are applied to get the features in the layer i (*$h^{(i)}$*).*

(2) Input $h^{(i)}$ *to the next layer, followed by the training of the next layer.*

(3) Repeat (1) and (2) to train each layer in the network, until the last layer.

The data of layer 1 is the input of entire deep network, namely the original data. The features obtained for the last layer are the output of deep feature learning model, and the input of the following classifier. $W^{(i)}$and $\mathrm{b}^{(i)}$ represent weight and bias values of layer i respectively.

3.2. Auto-encoder

Existing pre-training methods include auto-encoder (AE), restricted Boltzmann machine (RBM), sparse coding and deep belief network (DBN). In this paper, we use auto-encoder, which is simple but useful for pre-training our deep feature learning model.

Auto-encoder [8-10] usually consists of two parts: encoder and decoder.

Encoder uses nonlinear mapping function f to map input data ($x \in R^n$) into the representation in hidden layer ($h \in R^m$). The mapping is represented as follows:

$$h = f(x) = s_f(Wx + b) \tag{1}$$

The parameters of encoder contain a weight matrix (W) of size $m \times n$, and a bias vector ($b \in R^m$).

Decoder applies nonlinear mapping function g to reconstruct input data from the representation in hidden layer ($h \in R^m$) to form $r \in R^n$.

$$r = g(h) = s_g(W'h + b') \quad (2)$$

W' is the weight matrix of decoder with size $m \times n$, and $b' \in R^n$ is its bias vector. The s_f, s_g are nonlinear activation functions, which usually adopt sigmoid or tanh function.

(1) Network architecture

Suppose that we have an unlabeled training set $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ of N examples, $x^{(k)} \in R^n$ represents a sample. Fig. (**2**) gives the structure of conventional auto-encoder neural network.

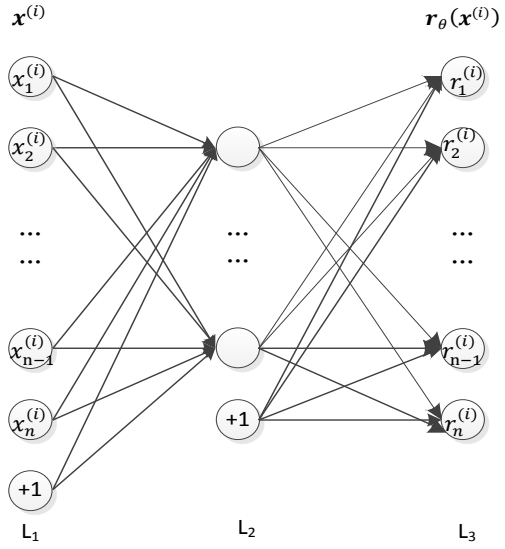


Fig. (2). Network Architecture.

Here, circle represents neurons. The neuron with "+1" is a bias unit. There are three layers in the network: the input layer on the left, the output layer on the right and one hidden layer. Let n_l denotes the number of layers, for example n_l=3, as shown in Fig. (**2**). We label l-th layer as L_l. The parameters of the network are $\theta = (W^2, b^2, W^3, b^3)$, where w_{ij}^l denotes the weight associated with the connection between j-th unit in layer l and i-th unit in layer $l+1$. Also, b_i^l is the bias associated with i-th unit in layer $l+1$.

The notation $r_\theta(x^{(i)})$ is the activation of neurons at the output layer when $x^{(i)}$ is given as input. This network tries to learn an approximation of the identity function, in other words to make $r_\theta(x^{(i)})$ similar to $x^{(i)}$. The mapping from the input of network to the value in hidden layer is to encode input data, namely learning feature of that. And then the reconstruction process from hidden layer to the output is to decode data from the feature. Our goal is to make the output after decoding approach, the original input. The encoded result (activation vector of hidden layer) is the feature expression of the input data.

(2) Cost function

Usually, auto-encoder neural network [8-10] is trained by unsupervised learning algorithm, which uses the back propagation algorithm to make the output (reconstruction of input) approach the input. That is to say the training objective is making $r_\theta(x^{(i)}) \approx x^{(i)}$. The training process is to search the best parameters of network to minimize the reconstruction error on the training set (D_N). Therefore, the cost function can be simply represented as:

$$J_{AE}(\theta) = \left(\sum_{x \in D_N} L(x, g(f(x)))\right) + \frac{\lambda}{2}\sum_{ij}(W_{ij}^l)^2 \quad (3)$$

The first item is the reconstruction error, which usually is squared-error function:

$$L(x, r) = (x - r)^2 \quad (4)$$

or the cross entropy cost function

$$L(x, r) = -\sum_{j=1}^{n} x_j \log(r_j) + (1 - x_j)\log(1 - r_j) \quad (5)$$

In this paper, one-half squared-error function is used for training. In detail, for a training example $x^{(i)}$, we define the cost function with respect to this example to be:

$$L(\theta; x^{(i)}) = \frac{1}{2}\left\| r_\theta(x^i) - x^{(i)} \right\|^2 \quad (6)$$

Given a training set of m examples, the overall cost function is:

$$J(\theta; X) = \frac{1}{N}\sum_{k=1}^{N} L(\theta; x^{(k)}) + \frac{\lambda}{2}\sum_{l=1}^{n_l - 1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l + 1}(W_{ji}^l)^2$$

$$= \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}\left\| r_\theta(x^{(k)}) - x^{(k)} \right\|^2 + \frac{\lambda}{2}\sum_{l=1}^{n_l - 1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l + 1}(W_{ji}^l)^2 \quad (7)$$

The first item is the average squared-errors of all examples. The second item is weight decay to decrease the magnitude of the weights, so as to help prevent over fitting. λ is weight decay parameter used to control the relative importance of the two items. θ is the parameters of network, including the weights and biases in all the layers.

3.3. The Improved Pre-training Method

In this paper, an exactly sparse auto-encoder (ESAE) neural network is proposed to pre-train deep neural networks. ESAE is similar to conventional sparse auto-encoder except that ESAE will force the neurons whose activation approximates zero to exact zero, achieving exact sparsity of learned features.

3.3.1. Exactly Sparse Auto-encoder

Based on conventional sparse auto-encoder, in this paper we try to learn exact sparse feature of data. It undermines that most of the hidden layer neurons are exact zero, which is different from the approximate zero in traditional sparse auto-encoder. That makes some of hidden neurons

are activated, while the activation of other hidden neurons equals zero, so as to make the average activation of hidden neurons in a small range. In this paper, the activation function of neurons is the sigmoid function.

We use $h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)$ to denote the activation of the j-th neuron at hidden layer given the input sample $\boldsymbol{x}^{(i)}$. The average values of that over all the samples in training set is defined to be:

$$\hat{\rho}_j = \frac{1}{N}\sum_{i=1}^{N} h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right) \tag{8}$$

Sparse auto-encoder is to add one sparsity penalty item to the overall cost function to help learn more sparse features. These sparse features can improve the discrimination of different data, so that the accuracy of classifier is increased. This sparsity penalty item is represented as:

$$\sum_{j=1}^{s_2}\left[\rho\log\frac{\rho}{\hat{\rho}_j}+(1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right] \tag{9}$$

Where ρ is a sparsity parameter, usually a small value close to zero? The sparsity penalty item makes $\hat{\rho}_j$ close to ρ, so as to make the average activation of neurons sparse enough.

Then the overall cost function after adding sparsity penalty item is as follows:

$$J_{sparse}(\theta;X) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2}\left\|r_\theta(x^{(k)})-x^{(k)}\right\|^2+\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ji}^l)^2 + \beta\sum_{j=1}^{s_2}\left[\rho\log\frac{\rho}{\hat{\rho}_j}+(1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right] \tag{10}$$

This sparsity penalty item results in the activation of part of neurons in hidden layer approximating zero, but they are still more than zero. In order to achieve exact sparsity of learned feature, we set a threshold s for $h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)$. It forces the activation of neurons in hidden layer, which approximate zero, to equal to zero. We define a threshold function to be:

$$thred\left(h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)\right)=\begin{cases}0, h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)<s\\ 1, h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)\geq s\end{cases} \tag{11}$$

Then, the final activation of hidden neurons is:

$$h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)=h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)\times\text{thred}\left(h_j^{(2)}\left(\boldsymbol{x}^{(i)}\right)\right) \tag{12}$$

(2) Learning algorithm

The learning algorithm is to find the final activation of hidden neurons (one feature expression of input) through searching the minimal $J_{sparse}(\theta;X)$. In this paper, we adopt the batch gradient descent algorithm as follows:

Algorithm 2:

1) Compute the overall cost function $J_{sparse}\left(\theta;X\right)$

Step 1: forward propagation. Compute the activations of every layer

$$z^{(l)} = W^{(l)}a^{(l-1)}+b^l \tag{13}$$

$$a^{(l)} = f\left(z^{(l)}\right) \tag{14}$$

$a^{(l)}$ *denotes the activation of layer* l. $z^{(l)}$ *is the net input of layer* l.

Step 2: compute the average activation of every hidden neuron using formula (8)

Step 3: apply formula (10) to compute the overall cost function $J_{sparse}\left(\theta;X\right)$

2) Compute the gradient of $J_{sparse}\left(\theta;X\right)$ *with respect to every parameter* $\nabla_{W^{(l)}}J_{sparse}\left(\theta;X\right)$ *and* $\nabla_{b^{(l)}}J_{sparse}\left(\theta;X\right)$

3) Update all the parameters

$$W^{(l)} = W^{(l)}-\alpha\nabla_{W^{(l)}}J_{sparse}\left(\theta\right) \tag{15}$$

$$b^{(l)} = b^{(l)}-\alpha\nabla_{b^{(l)}}J_{sparse}\left(\theta\right) \tag{16}$$

Where α *is learning rate.*

4) Repeat 1) - 3) until our cost function $J_{sparse}\left(\theta;X\right)$ *is small enough.*

5) Compute the initial activations of hidden neurons through forward propagation with the trained parameters of network

6) Compute the final activation of hidden neurons according to formula (12)

Using this algorithm, we can obtain one feature expression of input. It becomes the input signal of the second ESAE. Through minimizing the loss function of second ESAE, we can get the second expression of the original input information. Repeating the above steps, we obtain several ESAR networks. Stacking them will form a multilayer network, which is our feature learning model. The parameters of each layer are the feature expression extractors of raw input data at different levels. They can be used to obtain the feature of new data.

4. EXPERIMENTS AND ANALYSES

4.1. Dataset

We use Breast Cancer Wisconsin Data Set (BCW) [11] to verify the effectiveness of this method. It includes 699 instances. Each instance has one of 2 possible classes: benign or malignant, denoted by its class label 2 and 4. There are totally 458 and 241 instances for each class respectively. And each instance is described by 9 different attributes. Therefore, we represent one instance as a data point in 9-dimension space, and all the data points belonging to class benign as self-set, others as nonself set. This paper focuses on improving the detection accuracy for nonself set. So we used all the data in self-set and 100 data points in nonself as the training set, and the remaining 141 samples in nonself as test set.

4.2. Experimental Setup

In our experiments, the feature learning model in this paper contains two feature learning layers. It means that we will learn two feature representations at different levels. So in our experiments two ESAE neural networks were pretrained to initialize parameters of the two feature learning layers. Each ESAE neural network contains three layers: the input layer, hidden layer and output layer. For the first ESAE, the size of input layer (denoted as s_1) is equal to the dimension of the input data, that is $s_1 = 9$. We set its hidden layer size $s_2 = 20$. And the output layer size equals the number of input layer neurons $s_3 = s_1 = 9$, because the representation at output layer is the reconstruction of input data. The input of second ESAE is the feature learned from the first ESAE (namely the representation in hidden layer of the first ESAE). Therefore, the input layer size of the second network equals hidden layer size of first ESAE ($s_1^{'} = s_2 = 20$). The size of its hidden layer and output layer is both set to 20. Other parameters are λ = 0.003, $\beta = 3$, $\rho = 0.5$, and the threshold s=0.001. These parameters are chosen because they worked well in our experiments.

Features learned by well-trained feature learning model are input to other regular anomaly detection algorithms. We have validated the effectiveness of these features on two algorithms: KNN algorithm and multi-layer neural network (MLP) algorithm. Set the nearest neighbor size K from 1 to 200 for KNN algorithm. MLP network includes three layers, the input layer, one hidden layer and the output layer. The size of input and output layer is 9 and 2 respectively. And the hidden layer size changes from 20 to 40.

4.3. The Experimental Results Analyses

All the experimental results are average values of 10 repeated tests. Fig. (**3**) gives the variance of detection rates for KNN and improved KNN based on DL (DL_KNN) with different neighbor sizes. It shows that almost all the detection rates using DL_KNN method are higher than that of KNN algorithm. The best detection rate of KNN method is 90.17 when K equals 16, while that of DL_KNN algorithm is up to 97.16% (K = 10). Moreover, we found that when K approached 200, the detection rate declined quickly. That is because, when K is close to 200, the number of samples belonging to nonself (only 100 instances of this class in training set) is less than or equal to that belonging to self-set in the collection of K neighbors. Therefore, most of the instances in test set were predicted to be self-set. But even in this situation, our method still achieved better results. For example when K equals 200, KNN detection rate is 2.13%, while our detection rate is 13.48%. However, when K is larger than 200, samples of self-set are always more than nonself samples in nearest neighbors. At this time the detection rate of both equals to zero. Fig. (**4**) shows the results of comparison between MLP and DL_MLP method. We can see that the detection rates of DL_MLP method are all higher than that of MLP method.

All the experimental results show that the features learned by deep learning can improve the detection rates of various detection algorithms significantly.

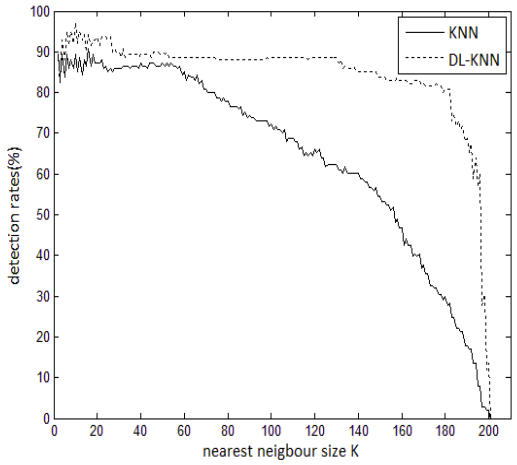


Fig. (3). Comparison between KNN and DL_KNN method.

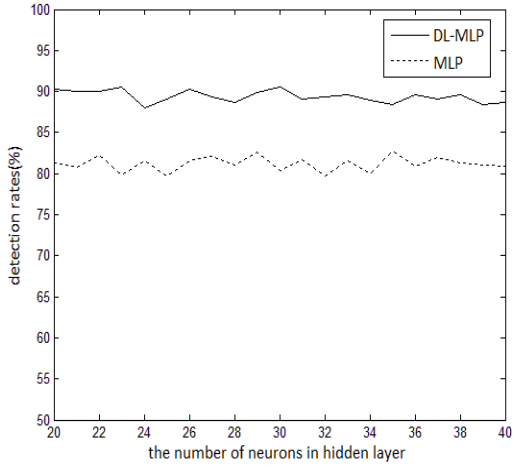


Fig. (4). Comparison between MLP and DL_MLP method.

CONCLUSION

Aiming at overcoming the problems of learning good feature and improving the detection accuracy in anomaly detection system, this paper proposed a new anomaly detection algorithm. This method applied an improved deep feature learning model based on exactly sparse auto-encoder to learn more useful features. These features help the traditional anomaly detection algorithm to obtain better detection accuracy.

To our knowledge, it is the first time to use the deep learning model to extract the features of anomaly detection data. This method proposed in this paper overcomes some limitations of classical anomaly detection methods. The main advantages of this method are: (1) It does not need artificial feature extraction. Artificial neural network with much more hidden layers has outstanding advantages in terms of feature learning. It can automatically learn with full use of big data to get useful features; (2) The deep model transforms data from one expression to another, which implements the mapping of the samples from the original data space to the new feature space. This transformation keeps richer information of the original data, at the same time enhances the ability to distinguish different samples, and accordingly improves the accuracy of anomaly detection; (3) The feature learning model is a completely unsupervised training process, which makes full use of huge amounts of unlabeled data that are easy to access, overcoming the difficulty to obtain labeled data.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This study was supported by the project grant from the Science and Technology Department of SiChuan Province, China. No. 15ZB0147 and No. 15ZB0142.

REFERENCES

[1] Zhang Zhao, Zhang Run-lian and Jiang Xiao-ge, "Anomaly detection method based on feature selection and support vector machine", *Computer engineering and design*, vol. 34 no. 9 pp. 3046-3049, 3162, 2013.

[2] Guo Xiao-fang, Li Feng and Wang Wei-dong, "Local outlier detection algorithm of multivariate time series based on k-nearest neighbor", *Journal of Jiangsu University of Science and Technology*, vol. 26, no. 5, pp. 505-513, 2012.

[3] Hou Di-bo, Chen Yue and Zhao Hai-feng, "Water quality anomaly detection method based on RBF neural network and wavelet analysis", *Transducer and Microsystem Technologies*, vol. 32, no. 2, pp. 138-141, 2013.

[4] Quan Liang-liang and WU Wei-dong, "Anomaly detection model based on support vector machine and Bayesian classification", *Journal of Computer Applications*, vol. 32, no. 6, pp. 1632－1639, 2012.

[5] Hinton G and Salakhutdinov R, "Reducing the dimensionality of data with neural network Science", vol. 313, no. 5786, pp. 504-507, 2006.

[6] Yu Kai, Jia Lei and Chen Yuqiang, "Deep Learning: Yesterday, Today, and Tomorrow, vol. 50, no. 9, pp. 1799-1804, 2013.

[7] Bengio Y. "Learning deep architectures for AI", *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1-127*,* 2009.

[8] Rifai S, Vincent P and Muller X. "Contractive auto-encoders: Explicit invariance during feature extraction". Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011: 833-840.

[9] Bengio Y. "Deep learning of representations: Looking forward", Statistical Language and Speech Processing. Springer Berlin Heidelberg, 2013: pp. 1-37.

[10] Bengio Y, Courville A and Vincent P, "Representation learning: A review and new perspectives", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 35, no. 8, pp. 1798 - 1828, 2013.

[11] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wis consin+%28Original%29

Received: May 26, 2015 Revised: July 14, 2015 Accepted: August 10, 2015