CIT. Journal of Computing and Information Technology, Vol. 27, No. 2, June 2019, 41–54
doi: 10.20532/cit.2019.1004702

A Phishing Webpage Detection Method Based on Stacked Autoencoder and Correlation Coefficients

Jian Feng[1], Lianyang Zou[1] and Tianzhu Nan[2]

[1]College of Computer Science and Technology, Xi'an University of Science and Technology, Xi'an, China
[2]Xi'an Fenghuo Software Technology Co., Ltd., Xi'an, China

Phishing is a kind of cyber-attack that targets naive online users by tricking them into revealing sensitive information. There are many anti-phishing solutions proposed to date, such as blacklist or whitelist, heuristic-based and machine learning-based methods. However, online users are still being trapped into revealing sensitive information in phishing websites. In this paper, we propose a novel phishing webpage detection model, based on features that are extracted from URL, source codes of HTML, and the third-party services to represent the basic characters of phishing webpages, which uses a deep learning method – Stacked Autoencoder (SAE) to detect phishing webpages. To make features in the same order of magnitude, three kinds of normalization methods are adopted. In particular, a method to calculate correlation coefficients between weight matrixes of SAE is proposed to determine optimal width of hidden layers, which shows high computational efficiency and feasibility. Based on the testing of a set of phishing and benign webpages, the model using SAE achieves the best performance when compared to other algorithms such as Naive Bayes (NB), Support Vector Machine (SVM), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN). It indicates that the proposed detection model is promising and can be applied effectively to phishing detection.

ACM CCS (2012) Classification: Security and privacy → Software and application security → Web application security

Keywords: phishing, deep learning, correlation coefficient

1. Introduction

Phishing refers to a kind of cyber-attack that uses social engineering, technical camouflage and other means of attack methods, by sending fraudulent spam, real-time communication messages, *etc*., to trick users into clicking on fake phishing pages, in order to entice users to disclose sensitive information such as personally identifiable data and financial accounts. The Anti-Phishing Working Group (APWG) reports that the total number of phishing attacks in the first quarter of 2018 is a 46% increase over the last quarter of 2017 [1]. The continued growth of phishing attacks has had a huge negative impact on the healthy development of the Internet and has become one of the most serious security threats on the Internet.

Researchers have proposed a series of detection methods for phishing webpage, including blacklist-based detection methods, heuristic-based detection methods, visual similarity-based detection methods, and machine learning-based detection methods. Among them, machine learning-based detection methods have achieved good detection results. However, with speeding up of phishing webpage update and increasing of the complexity of feature data, traditional phishing detection technology still needs continuous improvement.

On the other hand, deep learning is a promising alternative to traditional methods. In recent years, deep learning has been applied in various fields and has achieved great success. In

order to better detect phishing pages, a novel detection model for phishing webpages based on deep learning is proposed in this paper. After summarizing the existing research results, a detection model for phishing webpages is proposed. The model extracts the significant features of phishing webpages based on analyzing a large number of the latest phishing samples, and proposes a deep learning-based method that combines Stacked Autoencoder (SAE) with Softmax by a combination of unsupervised and supervised learning modes. Hence, in training of model parameters, a method for determining the width of hidden layers based on correlation coefficient is proposed, which effectively improves the training efficiency.

The main contributions in this paper are summarized as follows:

- To characterize phishing pages in all directions and at multiple levels. Based on the analysis of phishing webpages, constructing 52-dimensional feature vectors for phishing webpage detection from structural and lexical features of URL, Whois and DNS information, and source codes of HTML.
- To construct a phishing webpage detection model SSM (SAE-Softmax model), which is based on SAE and uses Softmax regression model to make the classification.
- To determine the width of hidden layers by correlation calculation between weight matrices of SAE, so that the width of hidden layers can be effectively adjusted.

The remaining sections of this paper are organized as follows: Section II shows the related works. Background work for Autoencoder (AE) is shown in Section III. Section IV shows the implementation of SSM. Section V describes the datasets and the comparative experiments. Finally, Section VI draws the conclusion and provides some implications for future work on phishing detection.

2. Related Works

2.1. Traditional Detection Methods

At present, the mainstream phishing webpage detection methods mainly include four categories.

1. **Backlist-based** detection methods simply match information such as URLs, which are easily implemented and have no false positives, but cannot identify phishing pages which are not listed on the blacklist [2].
2. **Heuristic-based** detection methods design and implement heuristic rules based on the similarity between phishing pages. Typical detection systems include CANTINA+ [3], *etc*. These methods can detect most unreported phishing websites in real time, but the premise is that the statistical features of phishing pages are unique and fuzzy matching technologies are adopted, so the false positive rate is high.
3. **Visual similarity-based** detection methods convert the webpages to be detected into pictures and then compare the feature vectors of the tested webpage and the targeting webpage by image processing technologies. A typical method is EMD algorithm proposed in [4]. This type of technology is powerless for phishing webpages which are not visually similar to the targeting webpage.
4. **Machine learning-based** detection methods treat the phishing webpage detection as classification or clustering problem and then use the corresponding machine learning algorithms to construct the detection models. The clustering method first divides the webpage dataset into several clusters, and then distinguishes the phishing webpages and the normal webpages by marking the clusters [5]. The classification method constructs classification rules or classifiers based on the features of the labeled webpages and then maps unknown webpages to one of the given categories [6]. Although machine learning-based methods have good adaptability and extensibility, and the detection accuracy is high, traditional machine learning methods are shallow level algorithms, and the ability to express complex functions is limited in the case of finite samples and computational units. The generalization ability of complex classification problems is limited.

2.2. Deep Learning-Based Methods

In 2006, Hinton *et al.* proposed deep learning theory and then several deep learning models such as Deep Belief Network (DBN), AE, Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) were proposed. It has demonstrated state of the art performance in many applications such as speech recognition, natural language processing, *etc*. In recent years, researchers have applied deep learning to phishing webpage detection. For example, the literature [7-9] mentions applications of deep learning to analyze URLs, and the difference is that they use different methods, namely RNN, Denoising Autoencoder (DAE) and CNN, respectively. Instead of manually extracting the features, all these researches learned representations from URL in different ways. On the other hand, there are some researches which focus on the texts on webpages and try to use new methods to learn new features represents phishing webpages. For example, in [10], features were extracted for phishing and benign webpages and classified using a Deep Belief Network-based detection model, while in [10], a series of semantic features were extracted through word2vec and used to describe the features of phishing webpages. Although all these solutions could classify the phishing websites precisely, they fail to use traditional phishing characters sufficiently. Following these successes, we propose a new model based on deep learning to solve the problem of webpage phishing detection by SAE and manually extracted statistical features, such as known stealing information and third-party services.

3. Background

3.1. AE

AE is an unsupervised deep learning method. The basic framework of AE comprises an input layer, an output layer, and a hidden layer. Therein, the input layer and the output layer have the same structure, and when the input is equal to the output, the hidden layer represents potential structure and characteristics of the input. The aim of AE is to transform inputs into outputs with the least possible amount of deviation.

Considering a dataset X with n samples and m features, the output of encoder Y represents the reduced representation of X and the decoder is tuned to reconstruct the original X to Z from the encoder's representation Y by minimizing the difference between X and Z, as illustrated in Figure 1. Y is the real outcome, which represents potential structure and characteristics of X. Specifically, the encoder is a function f that maps X to its hidden representation Y.

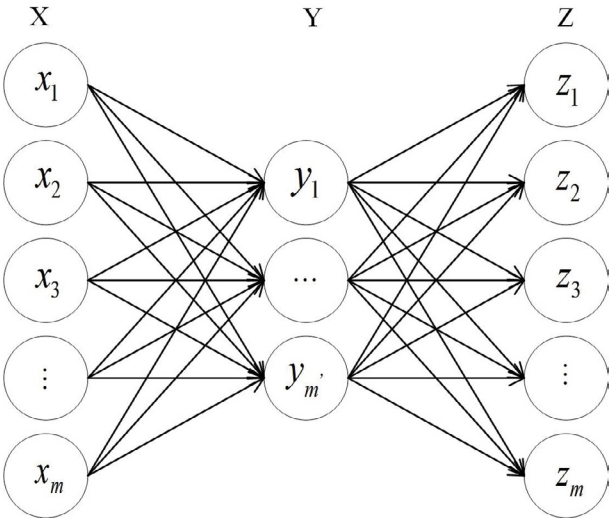

Figure 1. Diagram of the autoencoder.

The process is formulated as:

$$Y = f(X) = S_f(WX + b_X) \tag{1}$$

where S_f is a nonlinear activation function and if it is an identity function, AE will do linear projection. The encoder is parameterized by a weight matrix W and a bias vector $b_X \in R^n$.

The decoder function g maps hidden representation Y back to a reconstruction Z:

$$Z = g(Y) = S_g(W'X + b_Y) \tag{2}$$

where S_g is activation function of the decoder, typically either the identity (yielding linear reconstruction) or a sigmoid. The decoder's parameters are biased vector $\boldsymbol{b_Y}$ and weight matrix $\boldsymbol{W'}$, where $\boldsymbol{W'}$ is the inverse matrix of W. In this paper, we only explore the case of the tied weights when $\boldsymbol{W'} = \boldsymbol{W}^{\mathrm{T}}$.

Training an AE involves finding parameters $\theta = (W, b_X, b_Y)$ that minimize the reconstruc-

tion loss on X and Z, the objective function is given as:

$$\theta = \min_{\theta} L(X, Z) = \min_{\theta} L(X, g(f(X))). \quad (3)$$

For linear reconstruction, the reconstruction loss (L_1) is generally calculated by the squared error:

$$L_1(\theta) = \sum_{i=1}^{n} \|x_i - z_i\|^2 = \sum_{i=1}^{n} \|x_i - g(f(x_i))\|^2 \quad (4)$$

For nonlinear reconstruction, the reconstruction loss (L_2) is generally from cross-entropy:

$$L_2(\theta) = -\sum_{i=1}^{n} [x_i \log(y_i) + (1 - x_i) \log(1 - y_i)] \quad (5)$$

where $x_i \in X$, $z_i \in Z$ and $y_i \in Y$.

In SAE, the weight matrix W and the offset b are adjusted layer-by-layer by stochastic gradient descent:

$$W \leftarrow W - \eta \frac{\partial L(X, Z)}{\partial W} \quad (6)$$

$$b \leftarrow b - \eta \frac{\partial L(X, Z)}{\partial b} \quad (7)$$

where η is the learning rate.

3.2. SAE

AE contains only one hidden layer, which is a shallow learning model. The limitation of the shallow learning model is that the ability to represent complex functions is limited in the case of finite samples and computational units, and its generalization ability is constrained for complex problems. A major advantage of AE is that it is easy to be stacked for generating different levels of new features to represent original ones by adding hidden layers. After training to get the first AE, take its hidden layer as input, use the same method to train the second AE, and then train to get multiple AEs. Multiple AEs are stacked together to form an SAE model. The training process of SAE is shown in Figure 2.

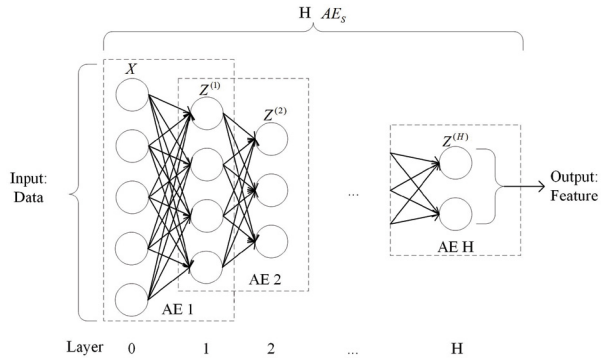

Figure 2. The schematic structure of SAE.

For an SAE with totally H hidden layers, the process of encoding is:

$$Y = f_H(\cdots f_i(\cdots f_1(X))) \quad (8)$$

where f_i is the encoding function of layer i. The corresponding decoding function is:

$$Z = g_H(\cdots g_i(\cdots g_1(Y))) \quad (9)$$

where g_i is the decoding function of layer i and the SAE can be trained by a greedy layer-wise feed-forward approach.

Finally, SAE is combined with Softmax to construct SSM model. The features learned by SAE are used as input to the Softmax classifier to obtain the final classification result.

4. Proposed Methodology

4.1. Problem Setting and Systematic Design

Our goal is to classify a given webpage as malicious or not. We do this by formulating the problem as a binary classification task. Consider a set of N webpages, $\{(P_1, L_1), ..., (P_i, L_i), ..., (P_N, L_N)\}$, where $i = 1, ..., N$, P_i represents a webpage, and $C_i \in \{-1, +1\}$ denotes the label of the webpage, with $C_i = +1$ being a malicious webpage, and $C_i = -1$ being a benign webpage. The first step in the classification procedure is to obtain a feature representation $P_i \rightarrow x_i$ where $x_i \in R^m$ is the m-dimensional feature vector representing webpage P_N. The next step is to learn a prediction function $f: R^m \rightarrow R$ to predict the score of the class assignment for a webpage instance. The prediction made by f is denoted as $\hat{Z} = sign(f(p))$. The aim is to learn a function that can minimize the total number of mistakes in the entire dataset. This is often achieved by minimizing a loss function. In our methodology, the function f is represented as an SAE network, and the cross-entropy loss function is adopted.

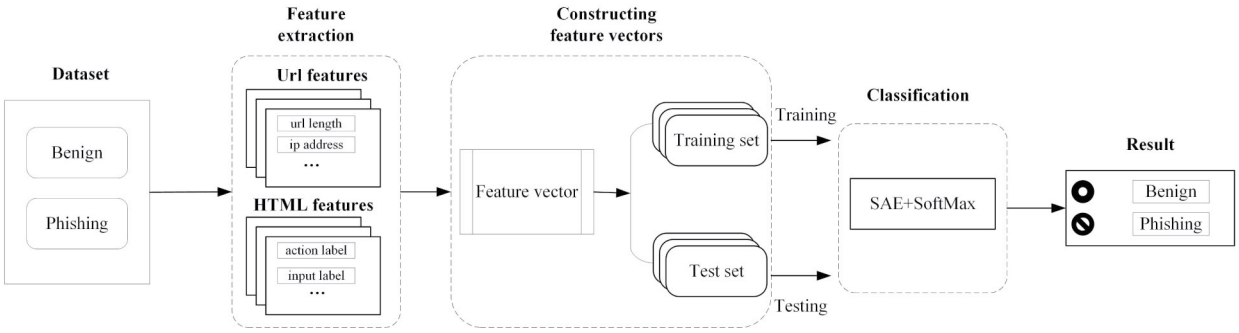

Figure 3. The architecture of SSM.

The systematic design of the SSM is illustrated in Figure 3. The input comprises a set of benign webpages and phishing webpages. Selection of features and normalization processes are carried out in the feature extraction phase. Then the SAE network is used to implement data reconstruction, and Softmax is added as a supervised classifier to assist in adjusting the network, and then binary classification of benign and phishing webpages is carried out.

4.2. Feature Extraction

The first step of obtaining feature representation deals with obtaining useful information from URL and webpage that can be stored in a vector so that the methods based on machine learning can be applied to it. Various types of features have been considered. We have classified the extracted features into two categories including URL related features and HTML based features. There is a total of 52 features which form a vector of 52 dimensions for a webpage P_i, namely $x_i = \left\langle x_i^0, x_i^1, ..., x_i^{51} \right\rangle$.

4.2.1. URL Related Features

URL related features are divided into URL-based features and third-party service features, and the third-party service features further include DNS-based features, Whois-based features, and ranking features.

- URL-based features

Table 1 lists some of the typical features extracted from URLs.

Table 1. URL-based features.

Feature	Type	Description
Length	Numeric	Length of a URL
IP address	Bool	Whether there is IP address in URL
Depth	Numeric	Number of '/' in URL
@	Bool	Whether there is @ in URL

- DNS-based features

Table 2 lists some of the typical features extracted from DNS information corresponding to the domain in URLs.

Table 2. DNS-based features.

Feature	Type	Description
Missing information	Bool	Whether there is information missing in DNS
A record	Numeric	Number of A records in DNS
NS record	Numeric	Number of NS records in DNS

- Whois-based features

Whois describes the details of the DNS, including whether the domain is registered, by whom it was registered, registrar, registration time, updated time, destruction time, and so on. Typical features are shown in Table 3.

Table 3. Whois-based features.

Feature	Type	Description
Missing information	Bool	Whether there is information missing in Whois
End time	Numeric	The difference between current time and termination time of the domain
Survival time	Numeric	The difference between termination time and creation time of the domain

- Ranking features

The ranking features, shown in Table 4, are mainly based on the Alexa Rank, which is also known as the page level.

Table 4. Ranking features.

Feature	Type	Description
Ranking	Numeric	The comprehensive rank of webpages on Alexa Rank
Visiting traffic ranking	Numeric	Visiting rank on Alexa Rank

4.2.2. HTML-Based Features

This paper divides the HTML-based features into three categories. In order to calculate values of the first two types of features, here are a few symbol definitions: L represents the number of links in the attribute of current tag. LE represents the number of empty links in the attribute of current tag. LC represents the number of links pointing to the current domain in the attribute of current tag.

- Feature of empty links

To count the empty links in the attribute of tags, calculate ME:

$$ME = \begin{cases} 0 & \text{if } LE = 0 \\ LE / L & \text{if } LE > 0 \\ -1 & \text{Tag does not exist} \end{cases} \tag{10}$$

- Feature of pointing to the current domain

To count the links to the current domain in the attribute of tags, calculate MC:

$$MC = \begin{cases} 0 & \text{if } LC = 0 \\ LC / L & \text{if } LC > 0 \\ -1 & \text{Tag does not exist} \end{cases} \tag{11}$$

- The third type of features, including the title attribute and the keyword attribute, is all Bool.

Table 5 lists some of the typical features extracted from HTML.

Table 5. HTML-based features.

Feature	Type	Description
Input	Numeric	Count value L of the input tag, and count the number of links with sensitive words after the input tag, recording it as LM, and calculate LM/L
Link_ empty	Numeric	ME of link tag
Link	Numeric	MC of link tag
Title	Bool	Whether the attribute of the title tag contains the domain of the current page

4.3. Normalization

To make features in the same order of magnitude, we made the value of the features normalized to [0, 1]. Assuming that the j-th feature $x_{i,j}$ of the sample x_i is numeric, three kinds of normalization algorithms are adopted, namely Minimum_Maximum normalization, Statistical normalization, and Decimal normalization. Their respective formula is as follows:

- Minimum_Maximum normalization

$$f_1(x_{i,j}) = \frac{x_{i,j} - \min(x_{,j})}{\max(x_{,j}) - \min(x_{,j})} \tag{12}$$

where $\min(x_{,j})$ and $\max(x_{,j})$ are minimum and maximum values of the j-th attribute of all samples, respectively.

- Statistical normalization

$$f_2(x_{i,j}) = (x_{i,j} - \mu) / \sigma \tag{13}$$

where μ represents the average of the j-th attribute of all samples, and σ is the standard deviation.

- Decimal normalization

$$f_3(x_{i,j}) = x_{i,j} / 10^q \tag{14}$$

where q is the smallest integer to enable the maximum of the j-th attribute is in the interval [0, 1].

4.4. Parameter Optimization of SSM

Experience has shown that the performance of deep learning models depends mainly on the network structure, especially the width of hidden layers. However, the selection of the number of hidden layer nodes is a very complicated problem. For the classification problem, if the width of hidden layers is too small, the training and testing accuracy will be relatively poor, and it is difficult to deal with complicated problems; if the width is too large, the training will take too long, and the classification performance will decrease, resulting in over-fitting. The methods for determining the width are mainly trial and error method [12], empirical formula method [13], growth method [14], pruning method [15], adaptive increase and decrease algorithm [16], genetic algorithm [17], *etc*. However, these methods have their limitations. The trial and error method is a kind of blindly searching algorithm with a large computational overhead. The empirical formula method is completely based on experience, as it lacks the corresponding theoretical basis. It is effective for specific samples but lacks a universal formula. Growth and pruning are the most studied methods at present. The growth method starts with the fewest number of nodes and then gradually adds new nodes until the network structure is optimal, and the pruning rule does the same in reverse. However, when to terminate is a problem for both methods, so the calculation costs of both algorithms are huge. Genetic algorithm is generally used in conjunction with the growth method or the pruning method. The main problem is that the convergence speed is slow and prone to oscillation.

In order to determine the suitable width of hidden layers quickly, this paper proposes an adaptive optimization algorithm based on weight correlation.

4.4.1. Basic Idea

According to the definition of the correlation coefficient, the absolute value of the correlation coefficient R between the two variables is between 0 and 1. The closer $|R|$ is to 0, the smaller the correlation between the two variables. According to the theory of detection, R between the samples has a significant influence on the error rate of the classification, and the best effect of the classification is optimal when R is 0. SAE is a multi-layered network. If taking weights of the nodes in the hidden layer as samples, then the closer the correlation coefficient between these weights is to 0, the larger the difference between the nodes, and perhaps the better the classification effect will be. Based on the above analysis, our goal is to find the network structure that maximizes the gap among the hidden layer nodes and select the width of hidden layers in this case.

This paper completes this process in two steps. The first is to initialize the network structure by setting the widths of the input layer and the first hidden layer. The number of input neurons equals the feature dimension of the input data, and the number of nodes in the first hidden layer is set artificially. Because we have not determined the explicit relationship between the input dimension and the width in the first hidden layer, the width in the first hidden layer cannot be adjusted automatically, and therefore is only set to be less than the input neurons. The second step is to determine the width of other hidden layers. A method to calculate correlation coefficient between weight matrices is proposed here, and it is used to determine the width of the current layer.

4.4.2. Determination of the Width by Correlation Coefficient

First, we clarify the concept of weight matrix. Weight matrix is a matrix formed by the connection weights among all nodes of the current layer and the previous layer.

For example, Figure 4 shows weights between two hidden layers. And its weights matrix is shown as $\boldsymbol{W}$.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix}$$

The elements in $\boldsymbol{W}$, while $i = 1, 2, ..., m$, and $j = 1, 2, ..., n$, represent weight of the connection between the i-th neuron of the current layer and

the j-th neuron of the previous layer where m is the number of neurons in the current layer and n is the number of neurons in the previous layer.

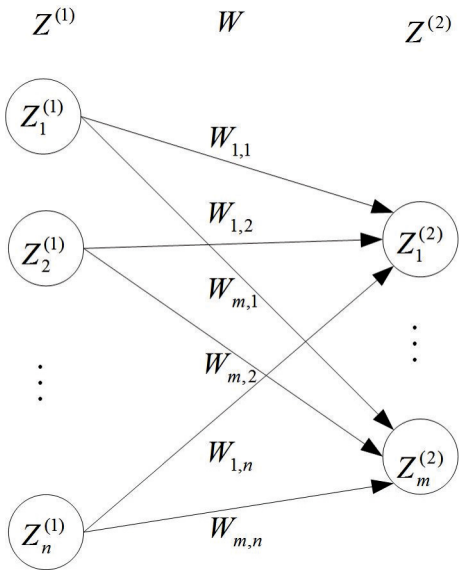

Figure 4. Two hidden layer and weights.

When a hidden layer adopts different widths, the network structure will change, so the weight matrix $\boldsymbol{W}$ will change accordingly. We want to determine the optimal width by learning the correlation of these weight matrices.

Firstly, we need to get the weight matrices under different network structures. To do this, on the basis of the initial width, we increase the width of the current hidden layer by the fixed nodes (for example, t) each round, until the total width is greater than or equal to the width of the previous layer. A weight matrix between the current hidden layer and the previously hidden layer will be got each round.

For example, assuming the width of the previous layer is o, the width of the previous round of the current layer is s, and the weight matrix is denoted as $\boldsymbol{A}$; if the width of the current round of the current layer is $s + t$ ($s + t \le o$), then the weight matrix is denoted as $\boldsymbol{B}$.

$$A = \begin{bmatrix} wa_{1,1} & wa_{1,2} & \cdots & wa_{1,o} \\ wa_{2,1} & wa_{2,2} & \cdots & wa_{2,o} \\ \vdots & \vdots & \ddots & \vdots \\ wa_{s,1} & wa_{s,2} & \cdots & wa_{s,o} \end{bmatrix}$$

$$B = \begin{bmatrix} wb_{1,1} & wb_{1,2} & \cdots & wb_{1,o} \\ wb_{2,1} & wb_{2,2} & \cdots & wb_{2,o} \\ \vdots & \vdots & \ddots & \vdots \\ wb_{s+t,1} & wb_{s+t,2} & \cdots & wb_{s+t,o} \end{bmatrix}$$

And then we try to perform correlation analysis on these matrices. Here a method for calculating correlation coefficient between neighboring matrices is needed. Take correlation coefficient between $\boldsymbol{A}$ and $\boldsymbol{B}$ as example. Since $s \neq s + t$, the correlation coefficient between $\boldsymbol{A}$ and $\boldsymbol{B}$ cannot be calculated directly through traditional method. We first set the average of the values of the same column of each matrix to get two row vectors $\boldsymbol{A'}$ and $\boldsymbol{B'}$. Where:

$$A' = \left[\overline{wa_1}, \overline{wa_2}, ..., \overline{wa_o}\right]$$

$$B' = \left[\overline{wb_1}, \overline{wb_2}, ..., \overline{wb_o}\right],$$

while:

$$\overline{wa_j} = \frac{wa_{1,j} + wa_{2,j} + ... + wa_{s,j}}{s},$$

$$\overline{wb_j} = \frac{wb_{1,j} + wb_{2,j} + ... + wb_{s+t,j}}{s+t}, \; j = 1, 2, ..., o.$$

Then average all the elements of the two generated row vectors to get two mean values:

$$\overline{wa} = \frac{wa_1 + wa_2 + ... + wa_o}{o},$$

$$\overline{wb} = \frac{wb_1 + wb_2 + ... + wb_o}{o}.$$

Bring these values into classical correlation coefficient equation to get the correlation coefficient between the two matrices, see [15]:

$$R = \frac{\sum_{j=1}^{o}\left(\overline{wa_j} - \overline{wa}\right)\left(\overline{wb_j} - \overline{wb}\right)}{\sqrt{\sum_{j=1}^{o}\left(\overline{wa_j} - \overline{wa}\right)^2 \sum_{j=1}^{o}\left(\overline{wb_j} - \overline{wb}\right)^2}} \quad (15)$$

where R is correlation coefficient between $\boldsymbol{A}$ and $\boldsymbol{B}$. After all correlation coefficients for neighboring rounds are calculated, we set the width to the number of nodes whose absolute value of correlation coefficient is nearest to 0.

5. Experimental Results and Analysis

In order to verify the feasibility and effectiveness of SSM, we designed three sets of experiments.

5.1. Experimental Preparation

5.1.1. Experimental Environment

The experimental development environment is shown in Table 6. The fixed hyperparameters used in SSM are as follows: learning rate equals to 0.1, weight regularization is 0.001, times of iterations is 1000, activation function is ReLu, and loss function is cross-entropy. The hyperparameters adjusted in experiments are the number of hidden layers and the width of the hidden layer.

Table 6. Development environment.

Operating system	CPU	RAM	Development environment
Windows 10	IntelCore i5-7200U CPU @2.5GHz	4GB	Matlab2016R

5.1.2. Basic Dataset

The basic dataset used in the experiments is obtained from the real network environment. The legal webpages come from Alexa. Alexa is a dedicated website managed by Amazon to publish an authoritative ranking of websites, so it has a large number of URLs and detailed ranking information. After filtering out some invalid, erroneous and duplicate pages, we collected 8,848 benign webpages from Alexa.

The phishing webpages are from PhishTank.com, which is an internationally renowned website that collects a timely and authoritative list of phishing webpages. We collected 11,321 phishing webpages listed on PhishTank from February 2016 to April 2016. In addition, webpages that do not conform to grammar rules and benign webpages mixed in phishing datasets are processed.

We collected and saved URL, HTML source file, and a screenshot of each collected page.

5.1.3. Evaluating Indicators

The various evaluating indicators in literature were summarized, the most commonly used are Accuracy, Recall, True Positive Rate (TPR), False Positive Rate (FPR), True Negative Rate (TNR) and False Negative Rate (FNR), shown in Table 7.

Table 7. Evaluating indicators.

Evaluating indicators	Formula
Accuracy	(TP+TN) / (TP+TN+FP+FN)
TPR (Recall)	TP / (TP + FN)
FPR	FP / (TN + FP)
TNR	TN / (TN + FP)
FNR	FN / (TP + FN)

In Table 7, TP (True Positive) denotes the number of benign webpages correctly classified as benign webpages, FP (False Positive) denotes the number of phishing webpages classified as benign webpages, TN (True Negative) denotes the number of phishing webpages classified as phishing webpages, and FN (False Negative) denotes the number of benign webpages classified as phishing webpages.

5.1.4. Baselines

In order to see how well the proposed SSM models perform with respect to the existing methods for phishing webpage detection, we compare SSM with four baseline models: Support Vector Machines (SVM), Naive Bayes (NB), CNN and RNN.

5.2. Result Evaluation

5.2.1. Experiment 1: Determining the Number of Hidden Layers

This experiment aims to determine the optimal number of hidden layers in SAE. In the begin-

ning, it sets the number of hidden layers to 2, the width of the input and output layer is both 52 and the width of each layer is 50 and 40, respectively. It uses this structure to conduct an experiment. Then it adjusts the network structure in every subsequent experiment by adding one hidden layer and reducing the width of hidden layers by 10 each time, and completes 4 sets of experiments. The results are shown in Table 8. It should be noticed that in the fifth experiment, the width of the last hidden layer is set to 5.

It should be known that the best classification results are achieved when the number of hidden layers is 2, and the increase of the number of hidden layers could not lead to better results. Especially, when there are more than 4 layers, both phishing and benign webpages are classified as benign. According to the results, in the subsequent experiments, we fixed the number of hidden layers to 2.

Table 8. Experimental results.

Network structure	Accuracy	FPR	FNR	TPR	TNR
50-40	0.9989	0.0007	0.0014	0.9985	0.9992
50-40-30	0.9977	0.0016	0.0026	0.9973	0.9983
50-40-30-20	0.9544	0.0347	0.0150	0.9849	0.9652
50-40-30-20-10	0.5744	1.0	0.0	1.0	0.0
50-40-30-20-10-5	0.5744	1.0	0.0	1.0	0.0

5.2.2. Experiment 2: Determining the Width of Hidden Layers

- Experiment on the basic dataset

The width of the first hidden layer is set to 50, and the width of the second hidden layer is calculated by Equation 15. At first, the initial width of the second hidden layer is set to 5, and then 5 nodes are added each round, keeping other hyperparameters unchanged. Figure 5 shows the correlation coefficients among different widths of the second hidden layer, where the correlation coefficients are the absolute values.

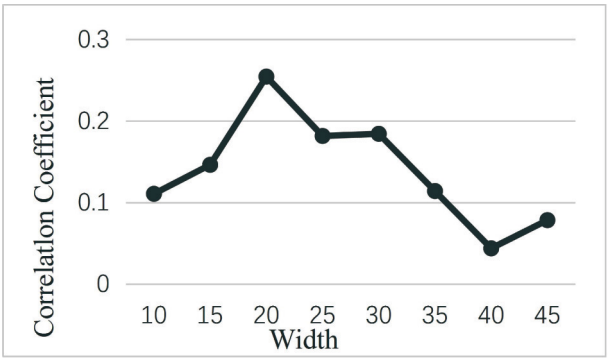

Figure 5. Change of correlation coefficient in the second hidden layer on the basic dataset.

Figure 5 shows that the width is 40 when the correlation coefficient is minimum. According to our hypothesis, the classification effect should be the best when the absolute value of the correlation coefficient is the closest to 0. To verify this, we calculated the performance of SSM under different network structures, as shown in Table 9. The two values in the network structure are the width of the first and the second hidden layer, respectively.

Table 9. Experimental results on the basic dataset.

Network structure	Accuracy	FPR	FNR	TPR	TNR
50-10	0.9978	0.0015	0.0009	0.9990	0.9984
50-15	0.9983	0.0012	0.0009	0.9990	0.9987
50-20	0.9983	0.0012	0.0009	0.9990	0.9987
50-25	0.9980	0.0014	0.0009	0.9990	0.9985
50-30	0.9976	0.0017	0.0004	0.9995	0.9982
50-35	0.9978	0.0015	0.0009	0.9990	0.9987
50-40	0.9995	0.0012	0.0004	0.9995	0.9987
50-45	0.9971	0.0021	0.0002	0.9997	0.9978

From Table 9, we notice that SSM obtains the best classification results when the width of the second hidden layer is 40, and this result is in line with our expectations.

- Experiment on a combined dataset

In order to avoid the contingency and to ensure the credibility of the above experiments, we repeated the above experiments on two other datasets. Firstly, we expand the original dataset to form a combined dataset. The dataset is ob-

tained by copying the original data three times, and then the training set and the testing set are reconstructed for verification by random extraction. The experiment setup is the same as in the previous experiments. Figure 6 shows correlation coefficients among different widths of the second hidden layer.

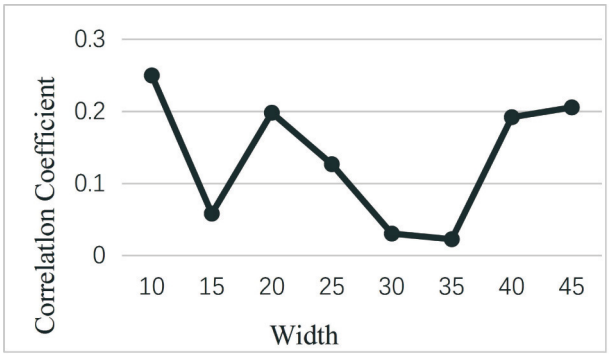

Figure 6. Change of correlation coefficient in the second hidden layer on the combined dataset.

It can be seen that the width of the second hidden layer is 35 when the correlation coefficient is minimum, and we calculated the performance of SSM under different network structures to verify this, as shown in Table 10.

Table 10. Experimental results on the combined dataset.

Network structure	Accuracy	FPR	FNR	TPR	TNR
50-10	0.9997	0.0001	0	1	0.9998
50-15	0.9997	0.0001	0	1	0.9998
50-20	0.9995	0.0003	0	1	0.9996
50-25	0.9997	0.0001	0	1	0.9998
50-30	0.9997	0.0001	0	1	0.9998
50-35	0.9998	0.0001	0	1	0.9999
50-40	0.9995	0.0003	0	1	0.9996
50-45	0.9997	0.0001	0	1	0.9998

As can be seen from Table 10, the best result is when the width is 35. Although in this experiment the width is different from the optimal width on the basic dataset, the result still supports our hypothesis.

- Experiment on a classic dataset

In order to avoid the contingency caused by the particularity of the dataset that we collected, a classic dataset for binary classification problem, GermanCredit [18], was used to check our theory. Each sample of GermanCredit has 24-dimensional features, so the width of the first hidden layer is manually set to 20. Because the dataset is different from the above two, we re-do Experiment 1 to determine the optimal number of hidden layers. The experimental results are shown in Table 11.

Table 11. Determining the number of hidden layers.

Network structure	Accuracy	FPR	FNR	TPR	TNR
20-15	0.5111	0.1885	0.5430	0.4569	0.8114
20-15-12	0.5486	0.2371	0.4503	0.5486	0.7613
20-15-12-9	0.4940	0.4635	0.5707	0.4292	0.5364
20-15-12-9-6	0.4560	1.0	0.0	1.0	0.0
20-15-12-9-6-3	0.4560	1.0	0.0	1.0	0.0

The results show that 3 hidden layers will perform the best. So, firstly, we calculate the correlation coefficients between the first and the second hidden layers and get Figure 7.

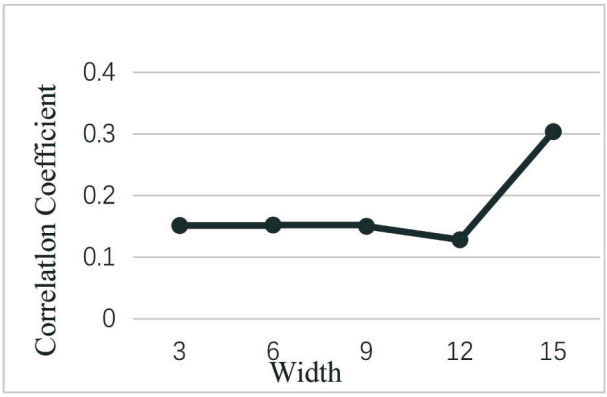

Figure 7. Change of correlation coefficient in the second hidden layer on GermanCredit.

Figure 7 shows that the width is 12 when the correlation coefficient is minimum. Since it is not known at present what width is appropriate for the third layer, the experiment below just considers two hidden layers. The classification result is shown in Table 12.

It can be seen that the performance is optimal when the width of the second hidden layer is 12. We then carry out the same experiment on

the third hidden layer. The result is shown in Figure 8.

According to Figure 8 and our theory, the width of the third hidden layer should be set to 4. To verify this, we use SSM on GermanCredit to get classification results, as shown in Table 13.

Table 12. Experimental results on GermanCredit (two hidden layers).

Network structure	Accuracy	FPR	FNR	TPR	TNR
20-3	0.5118	0.4189	0.5695	0.4304	0.8228
20-6	0.4569	0.2000	0.5430	0.4569	0.8000
20-9	0.5267	0.1771	0.5430	0.4569	0.8228
20-12	0.5419	0.1714	0.5298	0.4701	0.8285
20-15	0.5111	0.1885	0.5430	0.4569	0.8114

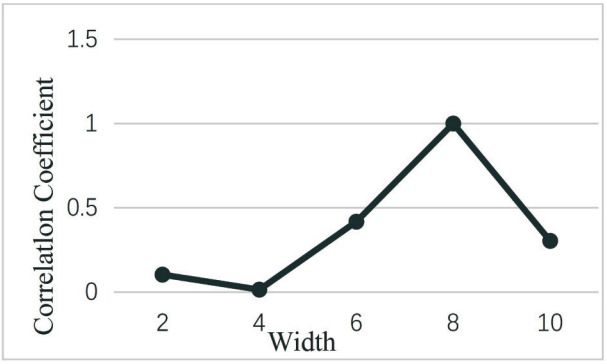

Figure 8. Change of correlation coefficient in the third hidden layer on GermanCredit.

Table 13. Experimental results on GermanCredit (three hidden layers).

Network structure	Accuracy	FPR	FNR	TPR	TNR
20-12-2	0.4685	0.2171	0.5562	0.4437	0.7828
20-12-4	0.5684	0.1800	0.4503	0.5496	0.8200
20-12-6	0.4580	0.2400	0.5298	0.4701	0.2285
20-12-8	0.5099	0.2514	0.4900	0.5099	0.7485
20-12-10	0.5496	0.2371	0.4503	0.5496	0.7628

Integrating all indicators, it is shown that the performance is best when the network structure is 20-12-4.

The experiments on three datasets have proved that the width of hidden layer can be set when the correlation coefficient is the closest to 0.

5.2.3. Experiment 3: Compared with Existing Phishing Webpage Detection Methods

Here we compare the classification results of SSM (with 2 hidden layers and 50-40 network structure) with that of an existing phishing webpage detection methods. Among them, the main hyperparameters of CNN and RNN are the same as the SSM model. CNN convolution kernel is 64, height and width of convolution window are both 3, height and width of max-pooling window are both 2, RNN units are 50. In addition, the CNN and RNN models incorporate dropout (0.5) to prevent overfitting. The experimental results are shown in Table 14. Time of computation for one iteration is shown in the last column.

Table 14. Comparison with existing detection methods.

Algorithm	Accuracy	FPR	FNR	TPR	TNR	Time (s)
SSM	0.9995	0.0012	0.0004	0.9995	0.9987	0.08
SVM	0.9112	0.0552	0.0887	0.9242	0.9112	3.60
NB	0.9441	0.0584	0.0858	0.9205	0.9441	0.45
CNN	0.9952	0.0028	0.0074	0.9925	0.9971	2.12
RNN	0.9962	0.0031	0.0045	0.9954	0.9968	0.49

As can be seen from Table 14, the SSM shows a significant improvement in performance compared to the traditional machine learning methods SVM and NB. Compared to CNN and RNN, SSM also achieved the best performance in the above configuration. Since we can't rule out all the hyperparameters, we don't rule out that SSM is not optimal in some cases. In addition, the SSM model takes the shortest amount of time in computing. This may be because we only use two hidden layers, but we can still see that the SSM algorithm has achieved a certain degree of improvement in evaluation indicators and computational efficiency. The practice has proved that SSM can effectively classify phishing and benign webpages.

6. Conclusion

This paper proposes an SAE-based model for phishing webpage detection. SAE-Softmax model abstracts a variety of features from URL, DNS, Whois, and HTML, and uses SAE-Softmax to construct the classifier model. In particular, a method for determining the width of the hidden layer based on correlation coefficient is proposed. Experiments show that SAE-Softmax model has achieved good performance.

In SAE-Softmax model, the features of the phishing webpages are still manually extracted. However, with the escalation of offensive and defensive competition, some typical features of phishing pages are gradually disappearing, new features are constantly appearing, and methods for discovering new features are time-consuming and laborious. In recent years, researches on using deep learning to extract latent features automatically have emerged. This is also our next research focus.

Acknowledgement

This work is supported by Shaanxi Provincial Natural Science Foundation Project (No. 2017JQ6053 and No. 2018JQ5095).

References

[1] "Phishing Activity Trends Report, 1st Quarter 2018", Anti-Phishing Working Group, Inc., Tech. Rep., July 2018.

[2] P. Prakash *et al.*, "PhishNet: Predictive Blacklisting to Detect Phishing Attacks", in *Proc. 29th IEEE Int. Conf. Comput. Commun.*, 2010, pp. 1–5. http://dx.doi.org/10.1109/INFCOM.2010.5462216

[3] G. Xiang *et al.*, "CANTINA+: A Feature-rich Machine Learning Framework for Detecting Phishing Web Sites", *ACM Trans. Inform. and System Security*, vol. 14, no. 2, pp. 1–28, 2011. http://dx.doi.org/10.1145/2019599.2019606

[4] A. Y. Fu *et al.*, "Detecting Phishing Web Pages with Visual Similarity Assessment Based on Earth Mover's Distance (EMD)", *IEEE Trans. Dependable and Secure Computing*, vol. 3, no. 4, pp. 301–311, 2006. https://doi.org/10.1109/TDSC.2006.50

[5] G. Liu *et al.*, "Automatic Detection of Phishing Target from Phishing Webpage", in *Proc. 20th Int. Conf. Pattern Recognition*, Istanbul, Turkey, 2010, pp. 4153–4156. https://doi.org/10.1109/ICPR.2010.1010

[6] S. Lee and J. Kim, "WarningBird: A Near Real-Time Detection System for Suspicious URLs in Twitter Stream", *IEEE Trans. Dependable and Secure Computing (TDSC)*, vol. 10, no. 3, pp. 183–195, 2013. https://doi.org/10.1109/TDSC.2013.3

[7] A. C. Bahnsen *et al.*, "Classifying Phishing URLs Using Recurrent Neural Networks", in *2017 APWG Symp. on Electron. Crime Research (eCrime). IEEE*, Scottsdale, AZ, USA, 2017, pp. 1–8. http://dx.doi.org/10.1109/ECRIME.2017.7945048

[8] S. Douzi *et al.*, "Advanced Phishing Filter Using Autoencoder and Denoising Autoencoder", in *Proc. Int. Conf. Big Data and Internet of Thing. ACM*, London, United Kingdom, 2017, pp. 125–129. https://doi.org/10.1145/3175684.3175690

[9] R. Vinayakumar *et al.*, "Evaluating Deep Learning Approaches to Characterize and Classify Malicious URL's", *J. of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1333–1343, 2018. https://doi.org/10.3233/JIFS-169429

[10] P. Yi *et al.*, "Web Phishing Detection Using a Deep Learning Framework", *Wireless Commun. and Mobile Computing*, vol. 2018, pp. 9. https://doi.org/10.1155/2018/4678746

[11] X. Zhang *et al.*, "Boosting the Phishing Detection Performance by Semantic Analysis", in *Big Data, 2017 IEEE Int. Conf. IEEE*, Boston, MA, USA, 2017, pp. 1063–1070. http://dx.doi.org/10.1109/BigData.2017.8258030

[12] P. G. Benardos and G. C. Vosniakos, "Optimizing Feedforward Artificial neural Network Architecture", *Eng. Applicat. of Artificial Intell.*, vol. 20, no. 3, pp. 365–382, 2007. http://dx.doi.org/10.1016/j.engappai.2006.06.005

[13] D. Gao and S. Wu, "An Optimization Method for the Topological Structures of Feed-forward Multi-layer Neural Networks", *Pattern Recognition*, vol. 31, no. 9, pp. 1337–1342, 1998. https://doi.org/10.1016/S0031-3203(97)00160-X

[14] M. Islam *et al.*, "A Constructive Algorithm for Training Cooperative Neural Network Ensembles", *IEEE Trans. Neural Networks*, vol. 14, no. 4, pp. 820–834, 2003. http://dx.doi.org/10.1109/TNN.2003.813832

[15] J. F. Qiao *et al.*, "A Fast Pruning Algorithm for Neural Network", *Acta Electron. Sinica*, vol. 38, no. 4, pp. 830–834, 2010. (in Chinese)

[16] M. M. Islam *et al.*, "A New Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks", *IEEE Trans. on Systems, Man, and*

Cybernetics, Part B (Cybernetics), vol. 39, no. 3, pp. 705–722, 2009.
http://dx.doi.org/10.1109/TSMCB.2008.2008724

[17] W. Y. Deng *et al.*, "Research on Extreme Learning of Neural Networks", *Chinese J. of Comput.*, vol. 33, no. 2, 2010. (in Chinese)
http://dx.doi.org/10.3724/SP.J.1016.2010.00279

[18] D. Dua *et al.*, "German Credit Data", UCI Machine Learning Repository, 1994.
http://archive.ics.uci.edu/ml

Received: March 2019
Revised: July 2019
Accepted: July 2019

Contact addresses:
Jian Feng
College of Computer Science and Technology
Xi'an University of Science and Technology
Xi'an
China
e-mail: fengjian@xust.edu.cn

Lianyang Zou
College of Computer Science and Technology
Xi'an University of Science and Technology
Xi'an
China
e-mail: 951625257@qq.com

Tianzhu Nan
Xi'an Fenghuo Software Technology Co., Ltd.
Xi'an
China
e-mail: 453629859@qq.com

Jian Feng was born in Xi'an, Shaanxi, China, in 1973. She received her BSc degree in printing technology from Beijing Institute Of Graphic Communication, Beijing, China, in 1994, and the MSc degree in printing engineering from Xi'an University of Technology, Xi'an, China, in 2001, and the doctoral degree in Computer Software and Theory from Northwest University, Xi'an, China, in 2008. In 2008, she joined the Department of Network Engineering in College of Computer Science & Technology, Xi'an University of Science and Technology, as a Lecturer, and in 2010, she became an Assistant Professor. She is the author of two books, and more than 30 articles. Her research interests include computer networks and commmunications, network security, and distributed computing. She holds three patents.

Lianyang Zou was born in Hunan, China, in 1994. He received the BSc degree in computer science and technology from Xinxiang University in 2017. He is currently pursuing the MSc degree in computer science and technology, Xi'an University of Science and Technology, China. Beginning in 2017, he was engaged in the research of phishing webpages detection. His research areas include network security and web mining. Mr. Zou's awards and honors include the Academy Scholarship, the National Inspirational Scholarship, and the Second Prize for the China Graduate Electronic Design Competition (Business Plan Special Competition).

Tianzhu Nan was born in Shaanxi, China, in 1991. He received the BSc degree in software engineering from Xi'an University of Science and Technology, Xi'an, Shaanxi, in 2015 and the MSc degree in computer application technology from Xi'an University of Science and Technology, Xi'an, Shaanxi, in 2018. In 2008, he joined Xi'an Fenghuo Software Technology Co., Ltd. and became a software development engineer for wireless network systems. His research areas include network security and web mining. He is the author of two articles. Mr. Nan's awards and honors include the Academy Scholarships.