Phishing Detection Using Neural Network

Ningxia Zhang, Yongqing Yuan

Department of Computer Science, Department of Statistics, Stanford University

Abstract

The goal of this project is to apply multilayer feedforward neural networks to phishing email detection and evaluate the effectiveness of this approach. We design the feature set, process the phishing dataset, and implement the neural network (NN) systems. We then use cross validation to evaluate the performance of NNs with different numbers of hidden units and activation functions. We also compare the performance of NNs with other major machine learning algorithms. From the statistical analysis, we conclude that NNs with an appropriate number of hidden units can achieve satisfactory accuracy even when the training examples are scarce. Moreover, our feature selection is effective in capturing the characteristics of phishing emails, as most machine learning algorithms can yield reasonable results with it.

1 Introduction

Recently, a phishing email has been circulating in the Stanford community, aiming to collect SUnetIDs and passwords. As the majority of phishing emails are formatted to appear from a legitimate source, a large percentage of email users are unable to recognize phishing attacks. Moreover, traditional spam email filters are inclined to fail to identify phishing emails since most phishing attacks use more sophisticated techniques and tend to be directed to a more targeted audience. With the increasing severity of this issue, many efforts have been devoted to apply machine learning methods to phishing detection.

One of the most common machine learning techniques for phishing classification is to use a list of key features to represent an email and apply a learning algorithm to classify an email to phishing or ham based on the selected features. Chandrasekaran et al. [4] proposed a novel technique to classify phishing emails based on distinct structural characteristics, such as the structure of the email subject line and some functional words. They used SVM to test their features on 400 emails and obtained a 95% accuracy rate. However, they did not perform different splits between training and test data due to the small sample size. Fette et al. [6] used ten different features specific to the deceptive methods for phishing classification and obtained an F_1-measure of more than 90% using a support vector machine classifier. However they used significantly more ham emails (7000) than phishing emails (860) in their simulation.

In this project, we use approximately 8762 emails out of which 4560 are phishing emails and the rest are ham. We notice that few studies have been done on applications of neural networks (NNs) to phishing email filtering. Although NNs normally require considerable time for parameter training, they usually yield more accurate results than other classifiers [5]. In our project, we try to detect phishing attacks through a feedforward neural network by incorporating some basic features pertaining to the email structure and external links.

2 Methods

2.1 Features

After referring to available literature, we have selected and defined a set of features that capture the characteristics of phishing emails [1, 3, 6].

2.1.1 Structural Features

1. *Total number of body parts*
According to MIME standard, "Content-Type" attribute of one email could be multipart, meaning that this email has multiple body parts. Phishers are likely to utilize this fact to construct phishing emails with sophisticated structures. By counting the number of boundary variables, we obtain the number of body parts in a multipart email. If the "Content-Type" of the email is not multipart, this feature is set to 0, for the purpose of differentiating from multipart emails with only one body part. If one part can be further divided into multiple parts, the number of sub-parts is added to the number of parts of the entire email. For example, if an email has 2 body parts, one of which has 2 sub-parts, the number of body parts is set to 4. However, only 3 parts of the content are scanned in the feature extraction process.
2. *Total number of alternative parts*
The multipart/alternative subtype indicates that each part is an "alternative" version of the same or similar content, each in a different format denoted by its "Content-Type" header [7]. As it is not strictly enforced that each part of the message is the same or similar, phishers often take advantage of this fact to create fraudulent emails.

2.1.2 Link Features

1. *Total number of links*
Phishing emails usually contain multiple links to fake websites for readers to sign in.
2. *Number of IP-based links*
A legitimate website usually has a domain name for identification while phishers typically use multiple zombie systems to host phishing sites. Besides, the use of IP address makes it difficult for readers to know exactly which site they are being directed to when they click on the link. Therefore, the presence of IP-based links can be a good indicator of phishing emails.
3. *Number of deceptive links*
Deceptive links are the ones with visible URLs different from the URLs to which they are pointing. Some phishers use this technique to fool email readers into clicking on the links.
4. *Number of links behind an image*
In order to make the emails look authentic, phishers often place in the emails images or banners linking to a legitimate website. Thus, if URL-based images appear in an email, it is likely to be an phishing email.
5. *Maximum number of dots in a link*
Using sub-domains is another technique phishers often exploit to make links appear legitimate, resulting in a inordinately large number of dots in the URL [3].
6. *A Boolean indicator of whether there is a link that contains one of the following words: click, here, login, update*
To realize the goal of acquiring usernames, passwords, or credit card information from the readers, phishing emails often invite readers to login to the fake websites for reasons such as updating personal information. Therefore, those words appearing in the link text would be a good indicator.

2.1.3 Element Features

1. *A Boolean indicator of whether it is in HTML format*
Phishing emails are mostly in HTML format as plain text does not provide the opportunity to play the tricks of phishing.
2. *A Boolean indicator of whether it contains JavaScript*
JavaScript enables phishers to perform many actions behind the scene, such as creating popup windows and changing the status bar of a web browser [6]. If the email contains strings, "javascript" or "onclick", this feature is set to one.
3. *A Boolean indicator of whether it contains <Form> tag*
HTML forms are one of the techniques used to gather information from readers [3].

2.1.4 Word List Features

1. *Boolean indicators of whether the words or stems listed below appear in the email body: account, update, confirm, verify, secur, notif, log, click, inconvenien*
In typical phishing email examples, these words frequently appear as phishers fabricate stories luring readers to enter their personal information.

2.2 Neural Networks

An artificial neural network, or neural network, is a mathematical model inspired by biological neural networks. In most cases it is an adaptive system that changes its structure during learning [10]. There are many different types of NNs. For the purpose of phishing detection, which is basically a classification problem, we choose multilayer feedforward NN. In a feedforward NN, the connections between neurons do not form a directed cycle. Contrasted with recurrent NNs, which are often used for pattern recognition, feedforward NNs are better at modeling relationships between inputs and outputs. In our experiments, we use the most common structure of multilayer feedforward NN, which consists of one input layer, one hidden layer and one output layer. The number of computational units in the input and output layers corresponds to the number of inputs and outputs. Different numbers of units in the hidden layer are attempted in the following experiments. To fit our dataset, hyperbolic tangent and sigmoid are used as activation functions. A comparison of the two is also conducted. With regard to the training method, we choose resilient propagation training (RPROP), as it is usually the most efficient training algorithm for supervised feedforward NNs [9].

2.3 Other Machine Learning Techniques

To further evaluate the performance of NNs in phishing detection, we compare its performance against that of other major machine learning classifiers – decision tree (DT), K-nearest neighbors, naive Bayes (NB), support vector machine (SVM) and unsupervised K-means clustering. The same dataset and feature set are used in the comparison.

2.4 Cross Validation

Given a training dataset and a proposed classifer, we assess the performance of the classifier by using hold-out cross validation, also known as simple cross validation [8]. The dataset is randomly divided into S_{train} and S_{cv}. The proposed classifier is trained on S_{train} to get parameter estimates and tested on S_{cv}. We then obtain the output which indicates whether each email in S_{cv} is ham or phishing. This procedure is repeated 20 times for different sizes of S_{train} and S_{cv}. The proportions of the dataset used as S_{train} are as follows: 0.1%, 1%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% and 90%.

2.5 Evaluation Metrics

By comparing the classification predictions with the actual categories of the emails, we are able to compute the numbers of true negatives (TN, correctly classified ham email), false negatives (FN, phishing email mistakenly classified as ham), true positives (TP, correctly classified phishing email) and false positives (FP, ham email mistakenly classified as phishing). To evaluate the classifier performance, we compute the accuracy($Accu$) and the weighted accuracy (W_{acc}) by the following formula:

$$Accu = \frac{TN+TP}{TN+FP+TP+FN} \tag{1}$$

$$W_{acc}(\lambda) = \frac{\lambda \cdot TN+TP}{\lambda \cdot (TN+FP)+FN+TP} \tag{2}$$

In phishing email filtering, errors are not of equal importance. A false positive is much more costly than a false negative in the real world [1]. It is thus desirable to have a classfier with a low false positive rate. The "weighted accuracy" measure is proposed by Androutsopoulos et al. [2] to address this issue. Different values of λ can be applied to the formula (1). Notice that when λ is one, the FP and FN are weighed equally. In our simulations, we pick $\lambda = 9$ so that FP are penalized nine times more than FN. In addition, we compute the precision, recall and F_1-score of each classifer as follows:

$$Precision = \frac{TP}{TP+FP} \quad Recall = \frac{TP}{TP+FN} \tag{3}$$

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision+recall} \tag{4}$$

3 Dataset

The dataset comprises of a large number of real world examples of ham and phishing emails, all in standard MIME format. There are a total number of 4202 ham emails and 4560 phishing emails, separated in 7 folders, 3 of which hold ham emails and 4 hold phishing emails. Each text file contains a single MIME email.

4 Implementation and Experiments

4.1 Preprocessing

4.1.1 Feature Extraction

We write a Perl script to extract features from one email example. It reads in the email file, does structural analysis with the help of MIME::Entity and MIME::Parser modules. It summarises link features using HTML::SimpleLinkExtor and HTML::LinkExtractor modules. Other features are obtained by taking advantage of the powerful regular expression manipulation of Perl. Ultimately, it outputs a feature vector together with the ideal value (1 for phishing email and 0 for ham). To process the entire dataset, another Perl script is written to call the feature extracting script and write the obtained feature vectors line by line into one text file.

4.1.2 Normalization

In order to ensure that each feature has an equal impact in the classification process, the vectors should be normalized before applying machine learning algorithms. For each feature, we find the maximum and minimum values, and for each value of this feature, we compute:

$$normalized\_value = \frac{(current\_value - minimum)}{(maximum - minimum)}$$

After normalization, the values of all features fall into the range of 0 to 1 and each feature contributes the same in determining the classification output. The normalized vectors of the whole dataset are stored in another text file.

4.1.3 Training and Test Sets Preparation

To conduct the cross validations described above, we divide the dataset into training and test sets with different proportions. For each proportion, we generate 20 different training and test sets. This is done by Matlab.

4.2 Machine Learning Implementation

The multilayer feedforward NN is implemented in Java with the Encog Java Core package, which provides a powerful framework to conveniently construct NNs and perform training and testing. When implementing other machine learning algorithms, we exploit the corresponding off-the-shelf Matlab packages.

4.3 Data Analysis

Once we obtain the classification predictions, we compute TN, FN, TP, FP, $Accu$, W_{accu}, Precision, Recall and F_1-score as described in the method section. We compare different neural networks by varying the units in the hidden layer as well as the activation function. We also compare the performance of neural networks with that of other machine learning techniques.

5 Results

As mentioned in the previous section, to evaluate each neural network classifier, we calculate the average $Accu$ and W_{accu} ($\lambda = 9$) in 20 cross validation procedures for each training size. As shown in Figure 1 and Figure 2, when the training size is small, more hidden units tend to overfit the data while fewer hidden units tend to underfit. However, when the training set is large enough, the number of hidden units does not greatly affect performance.

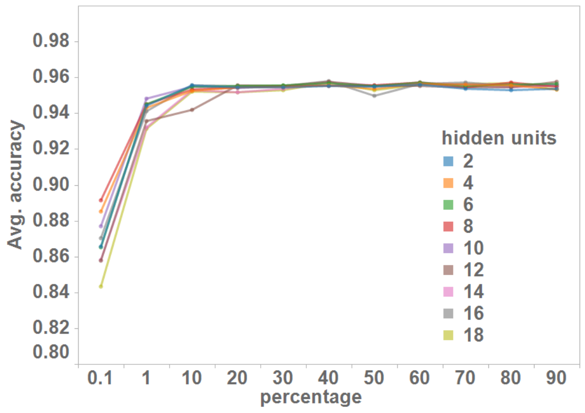

Figure 1: Each curve shows the average Accu for an NN classifier with a specific hidden layer size.

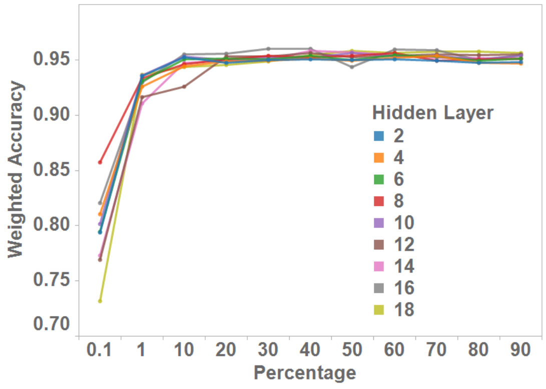

Figure 2: Each curve shows the average W_{accu} for an NN classifier with a specific hidden layer size.

To further demonstrate the overfitting of the dataset with a small training size, we examine the *Accu* and W_{accu} for the 0.1% training set in Figure 3 and Figure 4. We observe that the two curves both peak at 8 hidden units and start to decline as more hidden units are used. It is also worth noting that the W_{accu} generally drops after penalizing FP more than FN.

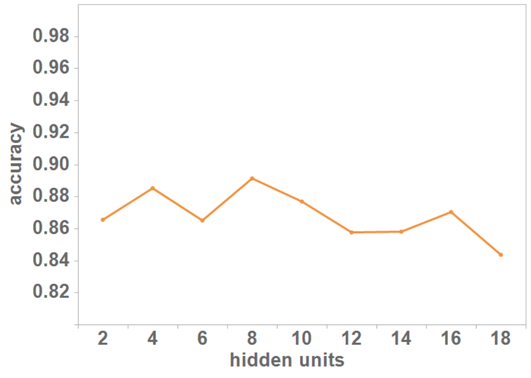

Figure 3: Accu for NN with 0.1% training size

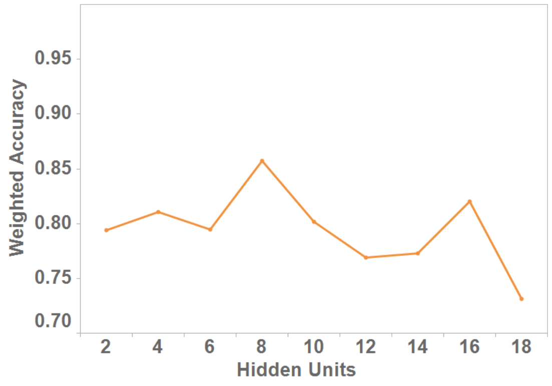

Figure 4: W_{accu} for NN with 0.1% training size

We compare the NN performance using two activation functions: hyperbolic tangent (HT) function and sigmoid function. The results are shown in Figure 5 and Figure 6. It is noticeable that the sigmoid function performs slightly better than the hyperbolic tangent function.

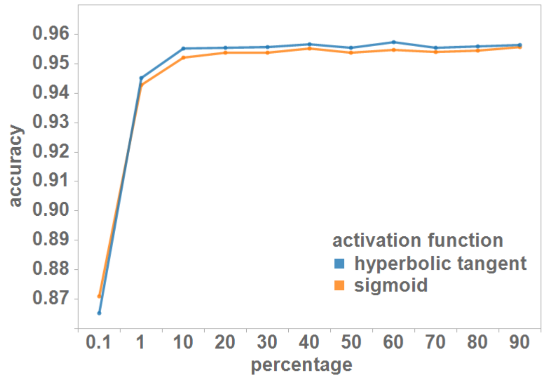

Figure 5: Accu of two NN (8 hidden units) activation functions

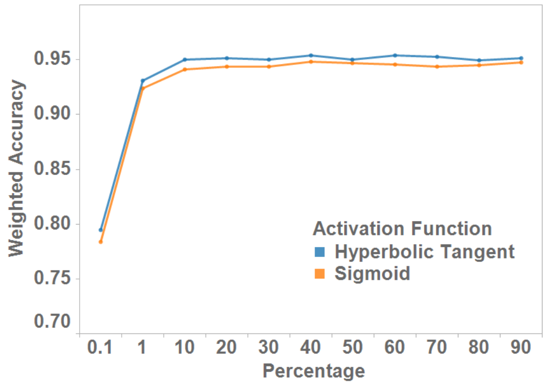

Figure 6: W_{accu} of two NN (8 hidden units) activation functions

We also compare the NN performance with other machine learning techniques. The results are shown in Figure 7 and Figure 8. Decision tree has the best overall performance, while it falls short on small training sets compared to NN and K-nearest. Generally, most algorithms can reach an accuracy of 95%, which suggests that the selected feature set has captured the essential characteristics of phishing emails. When we perform unsupervised 2-means clustering on the entire dataset, we are able to achieve 87% accuracy, which further supports the validity of our feature set.

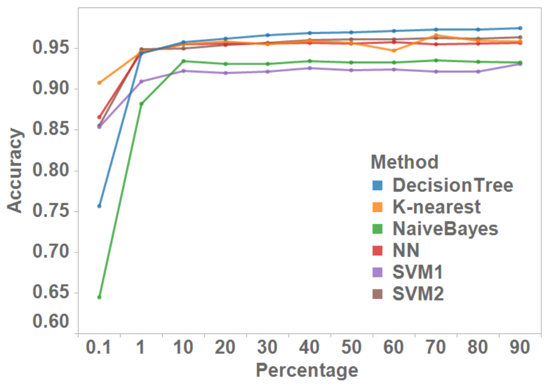

Figure 7: Accu of NN (8 hidden units) and other machine learning techniques

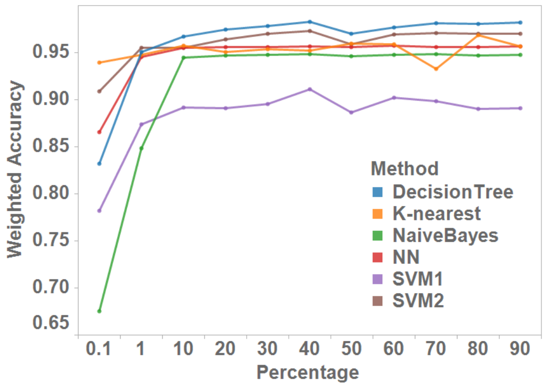

Figure 8: W_{accu} of of NN (8 hidden units) and other machine learning techniques

Table 1: Evaluations of NNs with two activation functions

Activation	*Accu*	W_{accu}	Precision	Recall	F_1
HT	0.9551	0.9494	0.9525	0.9618	0.9571
sigmoid	0.9516	0.9417	0.9450	0.9630	0.9539

Table 2: Evaluations of NNs and other machine learning techniques

Method	*Accu*	W_{accu}	Precision	Recall	F_1
DT	0.9658	0.9742	0.9778	0.9561	0.9668
SVM1	0.9218	0.8929	0.9022	0.9555	0.9275
SVM2	0.9579	0.9654	0.9693	0.9491	0.9591
NB	0.9278	0.9370	0.9460	0.9173	0.9367
K-nearest	0.9558	0.9536	0.9585	0.9583	0.9579
NN	0.9551	0.9494	0.9525	0.9618	0.9571

Table 1 summarizes performance measures for NNs with two activation functions in detail. As seen in the table, HT function performs slightly better in terms of all measures except recall. Notice that the largest difference out of all the measures comes from W_{accu}, which suggests that the HT function is better at avoiding misclassifying ham emails to phishing emails.

Table 2 summarizes the performace measures for NNs and other machine learning techniques. As shown in the table, DT gives the best overall performance. NNs give the highest recall while still mainitaining a >95% precision, suggesting that NNs are excellent at detecting phishing emails while misclassifying only a small portion of ham emails.

References

[1] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the Anti-Phishing Working Group eCrime Researchers Summit*, pages 649–656, 2007.

[2] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, George Paliouras, and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Proceedings of the Workshop on Machine Learning in the New Information Age*, 11th European Conference on Machine Learning, Barcelona, Spain, 2002.

[3] Ram Basnet, Srinivas Mukkamala, and Andrew H. Sung. Detection of phishing attacks: A machine learning approach. *Studies in Fuzziness and Soft Computing*, 226:373–383, 2008.

[4] Madhusudhanan Chandrasekaran, Krishnan Narayanan, and Shambhu Upadhyaya. Phishing e-mail detection based on structural properties. In *Proceedings of the NYS Cyber Security Conference*, 2006.

[5] James Clark, Irena Koprinsk, and Josiah Poon. A neural network based approach to automated e-mail classification. In *Proc. IEEE/WIC International Conference on Web Intelligence (WI)*, pages 702–705, 2003.

[6] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to detect phishing emails. In *Proceedings of the International World Wide Web Conference(WWW)*, 2007.

[7] Network Working Group. Multipurpose internet mail extensions (MIME) part two:media types. `http://tools.ietf.org/html/rfc2046#section-5.1.4`, 1996.

[8] Andrew Ng. CS229 lecture notes. `http://cs229.stanford.edu/notes/cs229-notes5.pdf`, 2012.

[9] Martin Riedmiller and Heinrich Braun. A direct adaptive method for fast backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 5, pages 586–591, 1993.

[10] Wikipedia. Artificial neural network — wikipedia, the free encyclopedia.