Scientific Research Publishing

Social Networking, 2018, 7, 97-109
http://www.scirp.org/journal/sn
ISSN Online: 2169-3323
ISSN Print: 2169-3285

Artificial Neural Network for Websites Classification with Phishing Characteristics

Ricardo Pinto Ferreira, Andréa Martiniano, Domingos Napolitano, Marcio Romero, Dacyr Dante De Oliveira Gatto, Edquel Bueno Prado Farias, Renato José Sassi

Graduate Program in Informaticsand Knowledge Management, Universidade Nove de Julho, São Paulo, Brazil
Email: log.kasparov@gmail.com, andrea.martiniano@gmail.com, domingos.napolitano@gmail.com, mhromero@hotmail.com, dacyrgatto@terra.com.br, edquelfarias@uni9.pro.br, sassi@uni9.pro.br

How to cite this paper: Ferreira, R.P., Martiniano, A., Napolitano, D., Romero, M., De Oliveira Gatto, D.D., Farias, E.B.P. and Sassi, R.J. (2018) Artificial Neural Network for Websites Classification with Phishing Characteristics. *Social Networking*, **7**, 97-109.
https://doi.org/10.4236/sn.2018.72008

Received: March 13, 2018
Accepted: April 24, 2018
Published: April 27, 2018

Abstract

Several threats are propagated by malicious websites largely classified as phishing. Its function is important information for users with the purpose of criminal practice. In summary, phishing is a technique used on the Internet by criminals for online fraud. The Artificial Neural Networks (ANN) are computational models inspired by the structure of the brain and aim to simulate human behavior, such as learning, association, generalization and abstraction when subjected to training. In this paper, an ANN Multilayer Perceptron (MLP) type was applied for websites classification with phishing characteristics. The results obtained encourage the application of an ANN-MLP in the classification of websites with phishing characteristics.

Keywords

Artificial Intelligence, Artificial Neural Network, Pattern Recognition, Phishing Characteristics, Social Engineering

1. Introduction

Phishing is a widely used strategy for spreading malware such as viruses and Trojans [1]. He often uses social engineering tactics to address the victims, causing his social networking accounts to be infected and used to spread the coup [2]. Its most common method of spreading malicious software is through sending spam emails, which direct the user to contaminated sites. Over time, the scams were diversifying and even using real events to take advantage of the curiosity of the unsuspecting Internet users [3].

With the Internet access facility, the number of people and companies that use websites every day increases rapidly. This has attracted criminals to the practice of phishing. In English, it corresponds to fishing. Its function is to obtain important information of users with the intention of the criminal practice, as for example obtaining data of bank accounts, passwords, number of credit cards among others confidential information of individuals or companies that are subsequently used fraudulently [4] [5].

Although state-of-the-art has solutions to detect phishing attacks, there is still a lack of accuracy for detection systems, which is leading to breakthroughs in transactions [3] [6].

Thus, in this context, Artificial Intelligence techniques can be applied to detect phishing attacks [6]-[11].

One of the techniques most used in the detection of phishing attacks is the ANN because of its ability to learn and generalize this learning, fundamental characteristic to detect attacks based on new behavior [12] [13] [14] [15].

In ANNs, learning occurs through a set of simple processing units called artificial neurons. They are particularly efficient for the input/output mapping of nonlinear systems and for performing parallel processing, and besides simulating complex systems, they generalize the results obtained for the previously unknown data. That is, they produce coherent and appropriate responses to patterns or examples that were not used in their training [16].

An important feature of ANNs is their ability to learn from incomplete and subject to data noise, having the ability to learn by example and make interpolations and extrapolations of what they have learned. A well-defined set of procedures to adapt the weights of an ANN so that it can learn a given function is called the training or learning algorithm [17] [18].

The aim of this paper was to apply an ANN-MLP to classify websites with phishing characteristics. The paper is organized after this brief introductory section as follows: in Section 2, the work method is presented theoretical background; Section 3 is the methodology and result of the computational experiments and in Section 4, the work is concluded with the final considerations.

2. Theoretical Background

2.1. Phishing

Phishing is an online fraud technique used by criminals in the computer world to steal bank passwords and other personal information, using them fraudulently [19].

The criminals use this technique to "fish" the data of the victims who "bit the hook" released by the phisher ("fisherman"), name that is given to those who perform a phishing. A phishing attempt can happen through websites or fake emails, which mimic the image of a famous and trusted company to be able to catch the attention of the victims. Typically, website content or phishing emails promise extravagant promotions for the user or ask them to update their bank details, avoiding account cancellation, for example [20].

The most inattentive and uninformed Internet user, when he falls into this trap, is redirected to a web page similar to the original company or bank, where he must inform his personal and banking data. The victim thinks he is only confirming his information with the bank, when in fact he is sending all the data to a criminal [20].

The purpose of phishing is to use the data collected by criminals to make purchases over the internet, bank transfers or even clear the entire bank account of the victim [21].

From the attacker's perspective, the main reasons behind phishing attacks are [20] [22]:

1) Financial gain: Phishers can use stolen bank credentials for their financial benefits.

2) Hidden Identity: Instead of using stolen identities directly, phishers can sell identities to others who may be criminals looking for ways to hide their identities and activities (for example, buying goods).

3) Fame and notoriety: phishers can attack victims because of peer recognition One way to defend yourself is to apply Artificial Intelligence techniques in the detection and classification of phishing attacks [23]. According to [19] [24] [25] these classifiers achieved good precision in this type of application.

2.2. Artificial Neural Networks Maintaining the Integrity of the Specifications

Artificial Neural Networks (ANNs) are models inspired by the structure of the brain to simulate human behavior in processes such as: learning, adaptation, association, fault tolerance, generalization and abstraction when submitted to training [16] [18] [26].

In these networks, learning takes place through a set of simple processing units called artificial neurons. An important feature of ANNs is their ability to learn from incomplete and subject to noise. In a conventional computing system, if a part fails, in general the system as a whole deteriorates, where as in an ANN, fault tolerance is part of the architecture due to its distributed nature of processing. If a neuron fails, its erroneous output is overwritten by the correct outputs of its neighboring elements.

ANNs can be used when there is little knowledge of the relationships between attributes and classes, are suitable for continuous value inputs and outputs, unlike most algorithms, are successful in a wide variety of real world problems, including recognition of manuscript characters, pathologies and medicine. In addition, parallelization techniques can be used to accelerate the computational process, several techniques have been recently developed for the extraction of rules from trained ANNs. These factors contribute to the usefulness of ANNs for numerical classification and prediction in data mining [16] [27] [28].

In ANNs learning occurs through a set of simple processing units called artificial neurons. The representation of the basic elements of an artificial neuron is shown in **Figure 1**. The data (input vectors) of the neuron (x_1, ..., x_n), the

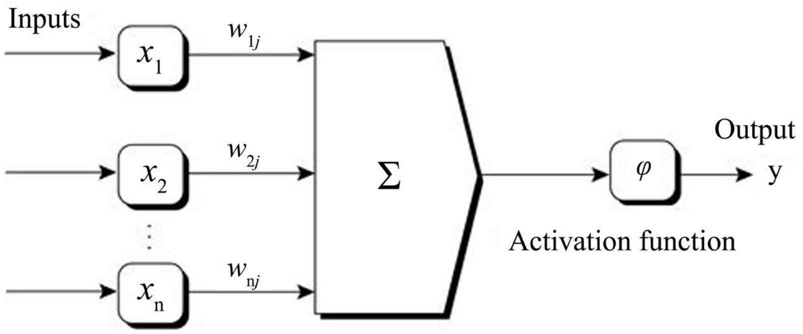

Figure 1. Representation of the basic elements of an artificial neuron. Source: Adapted from [16].

neurons of the input layer (w_{lj}, ..., w_{nj}) with their respective weights are observed, and then the additive junction or sum represented by the letter sigma, then the activation function (φ) and finally the output (y).

Thus, learning (or training) in an ANN is defined as the iterative adjustment of the synaptic weights, in order to minimize errors. Learning is the process by which the parameters of an ANN are adjusted through a continuous form of stimulation by the environment in which the network is operating, and the specific type of learning performed is defined by the particular way in which the adjustments made to the parameters occur [16].

According to [16] learning methods were developed and could be divided into supervised learning and unsupervised learning. In unsupervised learning the values of the desired outputs y_i are not known. Already supervised learning occurs through the identification of input patterns. In supervised learning, there is a prior knowledge about the values of inputs x_i and their outputs and i. This set of ordered pairs (x_i, y_i), which is known a priori, is called the learning database. A widespread training algorithm is error backpropagation used by an ANN MultiLayer Perceptron (ANN-MLP).

The backpropagation error training algorithm works as follows: a pattern is presented to the input layer of the network. This pattern is processed layer by layer until the output delivers the processed response, f_{mlp}, calculated as shown below, in Equation (1).

In which v_l and w_{lj} are synaptic weights; b_{10} and b_0 are the biases; and φ the activation function.

$$f_{\text{mip}}(x) = \varphi\left(\sum_{1}^{N_{on}} v_1 \varphi\left(\sum w_{ij} x_l + b_{l0}\right) + b_0\right) \quad (1)$$

The learning rate parameter has great influence during the MLP training process. A very low learning rate makes ANN learning very slow, while a very high learning rate causes oscillations in training and impedes the convergence of the learning process. Typically, the value ranges from 0.1 to 1.0.

Learning MLP with backpropagation may require many steps in the training set, resulting in a considerably long training time. If a local minimum is found, the error for the training set to decrease and park at a greater than acceptable value. One way to increase the rate of learning without leading to oscillation is to include the term momentum, a constant that determines the effect of the past changes of weights in the current direction of movement in the space of weights. It is recommended that the value of the momentum rate be between 0 and 1 [29].

Figure 2 illustrates the basic structure of an ANN (MLP type). It is possible to observe the input data (data vectors) of the network (x_1, ..., x_n),the neurons of the network input layer (Ne_1, ..., Ne_m) with their respective weights, the neurons that form the middle layer of the network (No_1, ..., No_n) and the output layer (Ns_1), formed by a neuron.

The most commonly used stop criteria are:

1) Number of times (cycles): defines the number of times the training set is presented to the network;

2) Error: consists of closing the training after the average quadratic error falls below a pre-defined value a. This value depends a lot on the problem. One suggestion is to establish a value of 0.01 in the first training and then adjust it in function of the result.

We can find in [16] the deepening of the characteristics of the ANN-MLP and other architectures of ANNs.

3. Methodology

The database used in the experiment was the Phishing Websites Data Set of the University of California's Machine Learning and Intelligent Systems Learning Center [30] available in: https://archive.ics.uci.edu/ml/datasets/phishing+websites. Which contains 11,055 records, 30 attributes and a target (result) for convenience in processing the data were reduced to 3000 records, 2000 records for the training phase and 1000 records for the ANN-MLP test phase. Table 1 shows the list of database attributes used in the experiment.

The parameters used in ANN-MLP were: number of input neurons equal to 30, number of hidden layers equal to 2, number of neurons in hidden layers equal to 18, learning rate equal to 0.7 with decay of 1% every 500 times, equal

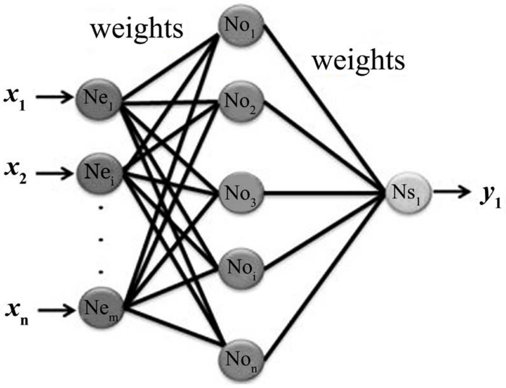

Figure 2. Artificial neural network (ANN-MLP).

Table 1. Attributes of the database phishing websites.

having IP Address	port	Right Click
URL Length	HTTPS token	Pop Up Window
Shortening Service	Request URL	Iframe
having At Symbol	URL of Anchor	age of domain
double slash redirecting	Links in tags	DNS Record
Prefix Suffix	SFH	web traffic
having Sub Domain	Submitting to email	Page Rank
SSL final State	Abnormal URL	Google Index
Domain registration length	Redirect	Links pointing to page
Favicon	On mouseover	Statistical report

moment factor 0.7 with decay of 1% every 300 times.

The stopping criterion was the maximum number of times equal to 10,000, error less than 10 - 2 or training stop if the error began to increase after 200 consecutive times. The output of ANN was websites with and without phishing features. Figure 3 illustrates the ANN-MLP architecture and the method used in the experiment.

The processing time in the training and test phase was 3 minutes and 20 seconds with 4661 times. The code of the program used is in Appendix.

4. Final Considerations

The ANN-MLP correctly classified 87.61% in the training of websites with and without phishing features. The performance of the artificial neural networks was very encouraging considering that the modeled ANN-MLP was able to present a good result, observing the great complexity of the proposed problem. The ANN-MLP presented in the test phase 98.23% of accuracy. A comparison of the accuracy of the ANN-MLP with works that used Artificial Intelligence techniques in the detection of phishing can be observed in Table 2.

Table 2 shows that the accuracy of ANN-MLP was among the best of the studies considered, despite the good results obtained overall. It is noteworthy that an MLP was used. In other words, only a single technique in some cases compared with two related techniques, which indicates the ANN-MLP as a good option to be applied to the problem. It is also worth noting that the comparison was made considering the accuracy of phishing detection, that is, in the application of the problem and not in the database, since the bases used in the works considered are different.

As for future studies, they should change the order of the attributes in order to find better groups to be processed by the ANNs. It is intended to significantly increase the training and testing database with the intention of increasing the generalization capacity of ANN-MLP and consequently to provide better performance in solving the classification problem.

Table 2. Comparative table.

	MLP	Dynamic evolving neural network based on reinforcement learning [13].	Heuristic Approach [31].	Case-Based Reasoning [7].	Fuzzy-rough hybrid system [32].	Detection and Prediction of Phishing Websites using Classification Mining Techniques [11].	New rule-based phishing detection method [4].
Accuracy	98.23%	98.63%	96.57%	95.62%,	88%	Approximately 96% (C4.5 algorithm)	99.14% true positive and only 0.86% false negative

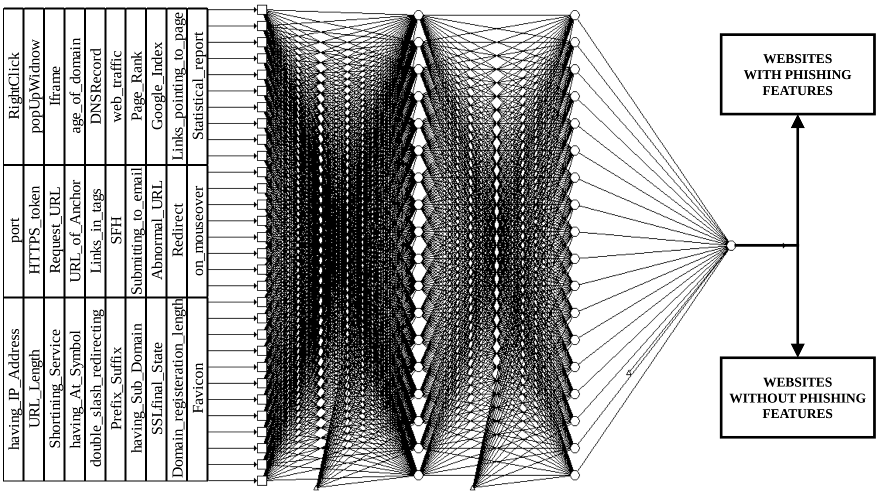

Figure 3. ANN-MLP architecture and the method used in the experiment.

Acknowledgements

We thank the Universidade Nove de Julho for research support, to the Universidade Corporativa dos Correios for all contribution and to the PROSUP/CAPES by grants awarded to studies.

References

[1] Aleroud, A. and Zhou, L. (2017) Phishing Environments, Techniques, and Countermeasures: A Survey. *Computers and Security*, **68**, 160-196. https://doi.org/10.1016/j.cose.2017.04.006

[2] Mouton, F., Leenen, L. and Venter, H.S. (2016) Social Engineering Attack Examples, Templates and Scenarios. *Computers and Security*, **59**, 186-209. https://doi.org/10.1016/j.cose.2016.03.004

[3] Goel, D. and Jain, A.K. (2017) Mobile Phishing Attacks and Defence Mechanisms: State of Art and Open Research Challenges. *Computers and Security*, **73**, 519-544.

[4] Moghimi, M. and Varjani, A.Y. (2016) New Rule-Based Phishing Detection Method. *Expert Systems with Applications*, **53**, 231-242. https://doi.org/10.1016/j.eswa.2016.01.028

[5] Barraclough, P., Hossain, M., Tahir, M., Sexton, G. and Aslam, N. (2013) Intelligent Phishing Detection and Protection Scheme for Online Transactions. *Expert Systems with Applications*, **40**, 4697-4706. https://doi.org/10.1016/j.eswa.2013.02.009

[6] Fernandes, D.A.B., Freire, M.M., Paulo, A., Fazendeiro, A. and Inácio, R. (2017) Applications of Artificial Immune Systems to Computer Security: A Survey. *Journal of Information Security and Applications*, **35**, 138-159.

https://doi.org/10.1016/j.jisa.2017.06.007

[7] Abutair, H.I. and Belghith, A. (2017) Using Case-Based Reasoning for Phishing Detection. *Procedia Computer Science*, **109**, 281-288. https://doi.org/10.1016/j.procs.2017.05.352

[8] Hadi, W., Aburub, F. and Alhawari, S. (2016) A New Fast Associative Classification Algorithm for Detecting Phishing Websites. *Applied Soft Computing*, **48**, 729-734. https://doi.org/10.1016/j.asoc.2016.08.005

[9] Abdelhamid, N., Ayesh, A. and Thabtah, F. (2014) Phishing Detection Based Associative Classification Data Mining. *Expert Systems with Applications*, **41**, 5948-5959.

[10] Lakshmi, S. and Vijaya, M.S. (2012) Efficient Prediction of Phishing Websites Using Supervised Learning Algorithms. *Procedia Engineering*, **30**, 798-805. https://doi.org/10.1016/j.proeng.2012.01.930

[11] Al-Diabat, M. (2016) Detection and Prediction of Phishing Websites Using Classification Mining Techniques. *International Journal of Computer Applications*, **147**, 5-12.

[12] Almomani, A., Wan, T.C., Altaher, A., Manasrah, A., Almomani, E., Anbar, M. and Ra-Madass, S. (2012) Evolving Fuzzy Neural Network for Phishing Emails Detection. *Journal of Computer Science*, **7**, 1099-1107.

[13] Smadi, S., Aslam, N. and Zhang, L. (2018) Detection of Online Phishing Email Using Dynamic Evolving Neural Network Based on Reinforcement Learning. *Decision Support Systems*, **107**, 88-102. https://doi.org/10.1016/j.dss.2018.01.001

[14] Basnet, R.B., Sung, A.H. and Liu, Q. (2012) Feature Selection for Improved Phishing Detection. *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Springer, Berlin, Heidelber, 252-261. https://doi.org/10.1007/978-3-642-31087-4_27

[15] Mohammad, R.M., Thabtah, F. and McCluskey, L. (2014) Predicting Phishing Websites Based on Self-Structuring Neural Network. *Neural Computing and Applications*, **25**, 443-458. https://doi.org/10.1007/s00521-013-1490-z

[16] Haykin, S. (2001) Redes Neurais—Princípios e Práticas. 2nd Edition, Bookman, Porto Alegre.

[17] Bigus, J.P. (1996) Data Mining with Neural Network: Solving Business Problems from Applications Development to Decision Support. McGraw-Hill, New York.

[18] Silva, I.N., Spatti, D.H. and Flauzino, R.A. (2010) Redes Neurais Artificiais para Engenharia e Ciências Aplicadas. Artliber, SP.

[19] Khonji, M., Iraqi, Y. and Jones, A. (2013) Phishing Detection: A Literature Survey. *IEEE Communications Surveys & Tutorials*, **15**, 2091-2121.

[20] Weider, D.Y., Nargundkar, S. and Tiruthani, N. (2008) A Phishing Vulnerability Analysis of Web Based Systems. *IEEE Symposium on Computers and Communications*, Marrakech, 6-9 July 2008, 326-331.

[21] Whittaker, C., Ryner, B. and Nazif, M. (2010) Large-Scale Automatic Classification of Phishing in Pages. *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, 28 February-3 March 2010, 1-14.

[22] Stringhini, G., Kruegel, C. and Vigna, G. (2010) Detecting Spammers on Social Networks. *Proceedings of the* 26*th Annual Computer Security Applications Conference*, Austin, TX, 6-10 December 2010, 1-9. https://doi.org/10.1145/1920261.1920263

[23] Basnet, R., Mukkamala, S. and Sung, A.H. (2008) Detection of Phishing Attacks: A

Machine Learning Approach. In: Prasad, B., Eds., *Soft Computing Applications in Industry*, Springer, Berlin, Heidelberg, 373-383. https://doi.org/10.1007/978-3-540-77465-5_19

[24] Bergholz, A., De Beer, J., Glahn, S., Moens, M.F., Paaß, G. and Strobel, S. (2010) New Filtering Approaches for Phishing Email. *Journal of Computer Security*, **18**, 7-35. https://doi.org/10.3233/JCS-2010-0371

[25] Lalitha, M.P. and Udutha, S. (2013) New Filtering Approaches for Phishing Email. *International Journal of Computer Trends and Technology* (*IJCTT*), **4**, 1733-1736.

[26] Simões, M.G. and Shaw, I.S. (2007) Controle e Modelagem fuzzy. FAPESP, São Paulo.

[27] Han, J., Kamber, M. and Pei, J. (2011) Data Mining: Concepts and Techniques. 3rd Edition, Morgan Kaufmann, Waltham, MA.

[28] Tkác, M. and Verner, R. (2016) Artificial Neural Networks in Business: Two Decades of Research. *Applied Soft Computing*, **38**, 788-804. https://doi.org/10.1016/j.asoc.2015.09.040

[29] Mitchell, T.M. (1997) Machine Learning. McGraw-Hill, New York.

[30] Mohammad, R., McCluskey, T.L. and Thabtah, F.A. (2014) Intelligent Rule Based Phishing Websites Classification. *IET Information Security*, **8**, 153-160.

[31] Rao, R.S. and Ali, S.T. (2015) PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach. *Procedia Computer Science*, **54**, 147-156. https://doi.org/10.1016/j.procs.2015.06.017

[32] Montazer, G.A. and ArabYarmohammadi, S. (2015) Detection of Phishing Attacks in Iranian e-Banking Using a Fuzzy-Rough Hybrid System. *Applied Soft Computing*, **35**, 482-492. https://doi.org/10.1016/j.asoc.2015.05.059

Appendix: Code of the Program

Program Code developed in language C.

```
#include <math. h>
/**
inputs - Array of 30 elements/outputs - Array of 1 element
*/
void Phishing (double * inputs, double * outputs) {
double main Weights[] = {Weights -1 --- 1}
double * mw = main Weights;
double b;
double hidden Layer 1 outputs [18];
double hidden Layer 2 outputs [18];
int c;
hidden Layer 1 outputs [0] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [0] += *mw++ * inputs [c];
hidden Layer 1 outputs [0] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [0]));
hidden Layer 1 outputs [1] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [1] += *mw++ * inputs [c];
hidden Layer 1 outputs [1] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [1]));
hidden Layer 1 outputs [2] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [2] += *mw++ * inputs [c];
hidden Layer 1 outputs [2] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [2]));
hidden Layer 1 outputs [3] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [3] += *mw++ * inputs [c];
hidden Layer 1 outputs [3] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [3]));
hidden Layer 1 outputs [4] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [4] += *mw++ * inputs [c];
hidden Layer 1 outputs [4] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [4]));
hidden Layer 1 outputs [5] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [5] += *mw++ * inputs [c];
hidden Layer 1 outputs [5] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [5]));
hidden Layer 1 outputs [6] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [6] += *mw++ * inputs [c];
hidden Layer 1 outputs [6] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [6]));
hidden Layer 1 outputs [7] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [7] += *mw++ * inputs [c];
hidden Layer 1 outputs [7] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [7]));
hidden Layer 1 outputs [8] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [8] += *mw++ * inputs [c];
hidden Layer 1 outputs [8] = 1.0/(1.0 + exp(-hidden Layer 1 outputs [8]));
hidden Layer 1 outputs [9] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [9] += *mw++ * inputs [c];
hidden Layer 1 outputs [9] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [9]));
hidden Layer 1 outputs [10] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [10] += *mw++ * inputs [c];
hidden Layer 1 outputs [10] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [10]));
hidden Layer 1 outputs [11] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [11] += *mw++ * inputs [c];
hidden Layer 1 outputs [11] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [11]));
hidden Layer 1 outputs [12] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [12] += *mw++ * inputs [c];
hidden Layer 1 outputs [12] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [12]));
hidden Layer 1 outputs [13] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [13] += *mw++ * inputs [c];
hidden Layer 1 outputs [13] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [13]));
hidden Layer 1 outputs [14] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [14] += *mw++ * inputs [c];
hidden Layer 1 outputs [14] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [14]));
hidden Layer 1 outputs [15] = *mw++;
for(c = 0; c < 30; c++) hidden Layer 1 outputs [15] += *mw++ * inputs [c];
hidden Layer 1 outputs [15] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [15]));
hidden Layer 1 outputs [16] = *mw++;
for (c = 0; c < 30; c++) hidden Layer 1 outputs [16] += *mw++ * inputs [c];
hidden Layer 1 outputs [16] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [16]));
hidden Layer 1 outputs [17] = *mw++;
for (c = 0; c < 30; c++) hidden Layer 1 outputs [17] += *mw++ * inputs [c];
hidden Layer 1 outputs [17] = 1.0/(1.0 + exp (-hidden Layer 1 outputs [17]));
hidden Layer 2 outputs [0] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [0] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [0] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [0]));
hidden Layer 2 outputs [1] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [1] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [1] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [1]));
hidden Layer 2 outputs [2] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [2] += *mw++ * hidden Layer 1 outputs [c];
```

```
  hidden Layer 2 outputs [2] = 1.0/(1.0 + exp(-hidden Layer 2 outputs [2]));
  hidden Layer 2 outputs [3] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [3] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [3] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [3]));
  hidden Layer 2 outputs [4] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [4] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [4] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [4]));
  hidden Layer 2 outputs [5] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [5] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [5] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [5]));
  hidden Layer 2 outputs [6] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [6] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [6] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [6]));
  hidden Layer 2 outputs [7] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [7] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [7] = 1.0/(1.0 + exp(-hidden Layer 2 outputs [7]));
  hidden Layer 2 outputs [8] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [8] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [8] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [8]));
  hidden Layer 2 outputs [9] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [9] += *mw++ * hidden Layer
1 outputs [c];
  hidden Layer 2 outputs [9] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [9]));
  hidden Layer 2 outputs [10] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [10] += *mw++ * hidden
Layer 1 outputs [c];
  hidden Layer 2 outputs [10] = 1.0/(1.0 + exp (-hidden Layer 2 outputs
[10]));
  hidden Layer 2 outputs [11] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [11] += *mw++ * hidden
Layer 1 outputs [c];
  hidden Layer 2 outputs [11] = 1.0/(1.0 + exp (-hidden Layer 2 outputs
[11]));
  hidden Layer 2 outputs [12] = *mw++;
  for (c = 0; c < 18; c++) hidden Layer 2 outputs [12] += *mw++ * hidden
Layer 1 outputs [c];
  hidden Layer 2 outputs [12] = 1.0/(1.0 + exp (-hidden Layer 2 outputs
[12]));
```

```
hidden Layer 2 outputs [13] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [13] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [13] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [13]));
hidden Layer 2 outputs [14] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [14] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [14] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [14]));
hidden Layer 2 outputs [15] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [15] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [15] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [15]));
hidden Layer 2 outputs [16] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [16] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [16] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [16]));
hidden Layer 2 outputs [17] = *mw++;
for (c = 0; c < 18; c++) hidden Layer 2 outputs [17] += *mw++ * hidden Layer 1 outputs [c];
hidden Layer 2 outputs [17] = 1.0/(1.0 + exp (-hidden Layer 2 outputs [17]));
outputs [0] = *mw++;
for (c = 0; c < 18; c++) outputs [0] += *mw++ * hidden Layer 2 outputs [c];
outputs [0] = 1.0/(1.0 + exp (-outputs [0]));
}
```