Biometric verification of computer users with probabilistic and cascade forward neural networks

S. SHORROCK, A. YANNOPOULOS, S.S. DLAY, D.J. ATKINSON
Department of Electrical & Electronic Engineering
University of Newcastle
Newcastle upon Tyne
UK

Abstract: - The number of people with access to the Internet continues to grow at a phenomenal rate. The potential for e-commerce on the web is huge, but fears about security are holding it back. Authentication of a person's identity is a major concern in the fight against fraud. We present a method of identifying a computer user by using the unique rhythm generated when typing on a keyboard. Three neural networks, the cascade forward neural network (CFNN), probabilistic neural network (PNN) and traditional backpropagation neural network (BPNN) are implemented on two types of data set. Each networks performance and suitability to the task is evaluated. The PNN is seen to outperform the BPNN and a novel implementation based around the CFNN displays a great deal of promise.

Key-Words: - keystroke dynamics, biometrics, verification, security, neural

1 Introduction

In recent years, the numbers of people using the Internet has exploded. The potential for e-commerce on the web has brought with it the need for heightened security. In order to satisfy this need for greater security, much effort has been focused on the use of Biometrics as a suitable solution.

Biometric identification is the use of measurable properties of the human body to verify a person's identity. These properties are unique to each individual and cannot be easily copied, stolen or lost. Biometric features can be divided into two groups, behavioral and physical. Some physical biometric properties such as fingerprints, iris patterns and hand geometry have already been implemented with success. Behavioral biometric properties have received much less attention [2]. They exploit the behavioural characteristics of the human body. The differences exhibited between individual's behavioural traits, are often less apparent than their physical counterparts. This can make them harder for an impostor to replicate.

The focus of this paper is the behavioural biometric properties of keystroke dynamics [1]. This utilizes the unique rhythm generated when a person types on a computer keyboard to verify that individual's identity. The use of keystroke dynamics has several advantages:

- Unlike most physical biometric methods, it requires no hardware other than the computer itself. Therefore cost of implementation is low.
- It is also ideally suited for use during a 'logon' procedure. It can be run in the background during a 'logon' without the knowledge of the user - adding an extra element of security with minimal disruption.
- As most computer users are familiar with entering usernames and password there is no need for re-training personnel.
- Being software based makes it easy to implement across a computer network.

This paper builds on the current work in this field by applying the probabilistic neural network (PNN) as classifiers. The PNN will be trained to classify a previously unseen typing rhythm as belonging to either the correct user or an impostor. Also, a cascade forward neural network (CFNN) will use a novel indirect method, where the network is trained to interpolate single measurements removed from the data and its accuracy differs significantly enough between a correct user and an impostor to allow classification. Both these methods will be compared with the traditional backpropagation approach.

2 Data Capture

Whilst gathering of data is possible on a PC (and essential for the end application), new device drivers need to be written for the target operating system. Operating systems such as Windows™ use a messaging system to convey information such as keypresses to an application. However, the keyboard messages often have a low priority over other operations such as file management. This is evident when typing in a word processor, if there is any disk activity when typing, the text being written is held in a buffer until such time that the processor is free to deal with it (causing the display to pause and then the characters appearing). In order to achieve the high accuracy of timing between keystrokes the keyboard driver and timing device need a high priority so they are not affected by other activity.

The gathering of the typing data has been implemented on a Motorola™ 68000 single board computer, interfaced with a keyboard and a monitor. This platform provides an excellent environment for gathering experimental data as there is no operating system and all code is written in assembly. By using this approach, complete control can be imposed over the collection of data.

Two sets of data were obtained using this method. The first set of data consists of a group of twenty one people, all typing the same phrase: '-u-s-e-r-n-a-m-e-<enter>-p-a-s-s-w-o-r-d-<enter>' where <enter> denotes the enter key being pressed and '-' denotes a time interval between a keypress. In order to emulate a real logon procedure, the characters for username are echoed back to the screen whilst the password characters are replaced with an asterisk. In order to capture a clean sample, the user is not allowed to make any spelling mistakes and any use of the backspace key will result in the sample becoming void. Each user gave twenty samples.

The second set of data consists of a group of five people, all typing their own names and then the names of the others in the group. Again, no spelling mistakes or use of backspace is allowed. Each user gave twenty samples of their own name and ten samples of each of the other names.

The number of samples used were carefully chosen to strike the balance between the maximum samples a user is prepared to submit and the minimum required by the classifier.

The first set of data, where all users type the same phrase, is to ensure any patterns that emerge are purely due to differences and similarities in typing styles. The second set of data is to demonstrate the effect typing a biased phrase has on classification.

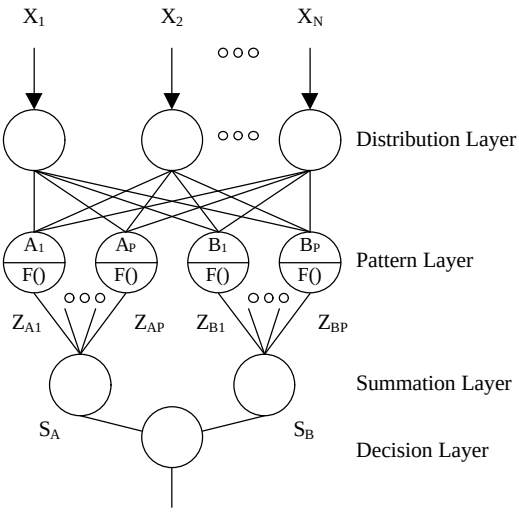


Figure 1 – PNN structure for a 2 class problem

3 Background

3.1 Backpropagation Neural Network

The backpropagation neural network (BPNN) is well established as a benchmark in pattern recognition problems. Its basic structure consists of a feed-forward network with an input layer, an output layer and at least one hidden layer of nodes. The backpropagation gradient descent technique is then applied to train the network's weights in a supervised mode in order to minimise the error with the output nodes target values.

The BPNN has the ability to solve complex problems with a relatively compact network structure. However, its learning process is often unpredictable, time consuming and can become stuck in a solution which is not optimal. Network design is often done on trial-and-error basis, as there are many factors that can be fine-tuned to improve the networks behaviour.

There are many alternatives to the BPNN, some of which have been explored by Obaidat and Sadoun [5]. A promising approach is the probabilistic neural network.

3.2 Probabilistic Neural Network

The PNN classifier has evolved from well-established Bayesian statistical techniques [6]. It is closely related to radial-basis function neural networks. It offers the following advantages when compared to backpropagation:

- Rapid Training: The PNN can train much faster than BPNN. It merely needs to 'read' the data.
- The PNN is guaranteed to converge to an optimal result given sufficient data. This contrasts with the unpredictable nature of the BPNN, which suffers from the problem of local minima.
- Data can be added and deleted without re-training. Useful for adding new users to the system and removing old ones.
- The PNN gives a measure of confidence associated with an output.

The PNN still retains many of the features associated with BPNN such as learning and generalisation. Due to the parallel nature of the algorithms, it is also a candidate for multi-processor systems.

The structure of the PNN for a 2-class problem can be seen in Figure 1. It consists of 4 layers:

1. The Distribution Layer: This performs no computation and just connects the inputs to the next layer. Where X_1 represents the first time interval, X_2 represents the second, etc.
2. The Pattern Layer: This consists of a number of nodes, one node for each training sample. The weights for each node correspond to the time intervals for that sample. For each node, the distance between the weights and the input vector is calculated and the result passed through a gaussian function.
3. The Summation Layer: The outputs of each pattern layer node and each class are summed.
4. The Decision Layer: This picks the largest node in the summation layer, outputting a 1 for $S_a > S_b$ and a 0 otherwise.

Problems with more than 2 classes are easily dealt with by a competitive algorithm in the decision layer to pick the summation node with the highest value.

The outputs of the pattern layer are given by:

$$Z_{ci} = \exp[(X^t X_{Ri} - 1)/s^2] \quad (1)$$

The outputs of the summation layer are given by:

$$S_C = \sum_{i=1} \exp[(X^t X_{Ri} - 1)/s^2] \quad (2)$$

Where c represents nodes belonging to each class and i the samples for each class (σ^2 being the standard deviation). The transpose of the unknown vector to classify is X_t and the weights are X_{Ri} . The network operates by calculating the dot product between the unknown vector and the training vectors. However, this uses normalised training and testing vectors. Normalisation of data for this problem results in loss of information concerning the total typing duration Obaidat and Macchiarolo [4]. In order to overcome the problem with normalisation, the dot product term can be replaced by a distance measure.

3.2.1 Distance Measures

For each node in the PNN the vector distance between the unknown input vector and the weights on the pattern layer needs to be calculated. Two distances which can be applied to this problem are:

3.2.1.1 Euclidean Distance

$$d = \left(\sum_i |x_i - y_i|^2 \right)^{1/2} \quad (3)$$

This is a commonly used function but, while working well for consistent patterns, it can be adversely affected by large outlier values - due to the square term.

3.2.1.2 Manhattan Distance

$$d = \sum_i |x_i - y_i| \quad (4)$$

The Manhattan distance is similar to the Euclidean metric, but eliminates the square term. This makes it less susceptible to spurious values.

3.3 Interpolating with a CFNN to recognise correlation

One property of the data which is not exploited by the standard pattern-recognition of the BPNN and the PNN is that each data vector is in fact a time sequence – and successive measurements can be expected to be highly correlated with each other. Our third approach has therefore been to train a variant of the BPNN, the CFNN[1], to learn these correlations: We slide a window over the input vector, removing the window's central element at each step and using it as the training target for the single network output. This training tunes the network to the specific correlations exhibited by the data generated by the user who is to be recognised. Assuming that new data produced by this user contains similar correlations, the simulation outputs will be close to their targets if training was successful. However, if a different user is the source of the data, its internal correlations will be different and the errors between net simulation outputs and their respective targets will be larger. A collective measure of errors

[1]A Cascade-Forward Neural Network is identical to a feed-forward BPNN except that each layer after the first has weights coming not just from the previous layer but also from the input and all other previous layers. We used the Levenberg-Marquardt [7] training algorithm to accelerate training in this case.

Table 1 – Comparison of network performance

		Data set 1 percentage errors		Data set 2 percentage errors	
Network		**Type 1 error**	**Type 2 error**	**Type 1 error**	**Type 2 error**
BPNN	No data Pre-Processing	41	3	0	7
	With data Pre-Processing	31	7	-	-
PNN	No Pre-Processing & MD	30	1	0	5
	No Pre-Processing & ED	48	2	0	3
	With Pre-Processing & MD	22	9	-	-
	With Pre-Processing & ED	34	8	-	-
CFNN		44	10	12	9

Key: MD = Manhattan Distance, ED = Euclidean Distance.

between outputs and targets for all window positions, for example their mean-square-error, will provide a strong indication of the user's identity. Deciding upon the threshold, beyond which this error indicates an impostor, may not be straightforward. This problem is especially pronounced when the distinctive capability of the network is not very good, whereas achieving a clear distinction would make the thresholding easier. We therefore concentrate on achieving good results assuming ideal thresholding.

4 Pre-processing data

Consistency when typing varies between users. While some individuals may be highly consistent, others are not. This causes problems with the distance measures described due to large spurious outliers.

In order to reduce these problems, the data is pre-processed for the PNN and BPNN. This involves calculating the z-score values for each sample in the training set with respect to its class.

$$z_i = \frac{x_i - \bar{x}}{s^2} \tag{5}$$

The z-score is shown in Equation 5 and is calculated by subtracting the vector from the mean of its class then dividing by the standard deviation of it's class.

Pre-processing is achieved by eliminating values which fall outside a specified range. These values are then replaced by the mean of the remaining components for each feature. A limit is put on the number of outliers that may be removed from each sample in order to preserve the majority of features.

5 Results

The success of the classifier can be measured in two ways:

1. The number of errors occurring where it incorrectly identifies an impostor as the real user.
2. The number of errors occurring where it incorrectly rejects the real user as an impostor.

These are known as type 1 and type 2 errors respectively. A type 1 error occurs when the classifier incorrectly rejects the correct user as an impostor. A type 2 error occurs when the classifier incorrectly accepts an impostor as the correct user.

It is possible to apply a bias to these errors depending on the application. For example, in a high security application if may be preferable to minimise the type 2 error at the cost of a higher type 1 error.

Table 1 shows how the three networks compare. It is divided into the two data sets:

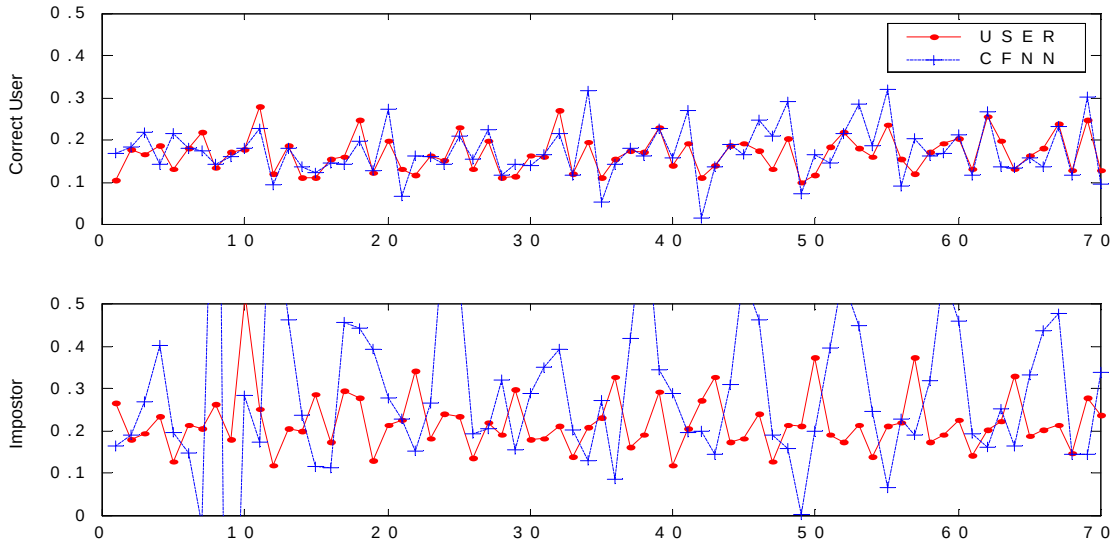
1. Unbiased – Twenty one people, all typing the same, previously unrehearsed, phrase.
2. Biased – Five people, each typing their own names and then trying to type the names of the other four.

The type 1 and type 2 errors for each data set are shown as an overall percentage. This is calculated by summing the errors generated for all users and dividing by the total number of presentations.

The data pre-processing was not required for the biased data set as all users exhibited highly consistent typing styles.

Results for the PNN include both types of distance measure.

Figure 2 – Example of CFNN performance for correct user and impostor



6 Discussion

6.1 Backpropagation Network

6.1.1 Data set 1

The BPNN, while working well for some users, struggles to classify certain individuals. This has an effect on the overall performance. On analysis of the results, the users who return large errors tend to exhibit an inconsistent typing style. The spurious outlier data generated by these inconsistencies may be confusing the network during training.

An improvement is seen in the network's performance for type 1 errors when it is trained with the pre-processed data. This is due to outlier datum caused by the correct user typing inconsistently, being removed from the sample and replaced with a mean value. However, as some outlier data in impostor samples is also replaced (making an impostor sample more like the correct user's), type 2 errors are increased.

6.1.2 Data set 2

These results are much improved, with zero being returned for type 1 errors. This is probably due to the fact users are typing a phrase which is familiar to them (their name) and are therefore more consistent. The level of type 2 errors is comparable with the first data set. Note, that this data set is probably more representative for verification as a user will indeed be biased towards typing their own identifier. We can thus expect the better of our results to be indicative of real-world performance for these methods.

6.2 Probabilistic Network

6.2.1 Data set 1

The PNN gives poor results when the Euclidean distance metric is employed. Some improvement is achieved with the pre-processed data, but it is still worse than the BPNN. Analysis again shows the largest errors occur with the least consistent typists.

The Manhattan distance function seems more suited to this problem. It produces respectable results when applied to the normal data. Better results for type 1 error are achieved when it is applied to the pre-processed data but at the expense of type 2 errors. In both cases it out performs the traditional BPNN, exhibiting a reduction in the type 1 error of approximately 25%.

6.2.2 Data set 2

Improved results for type 1 errors are seen for the biased data with the PNN. However, the type 2 errors remain comparable to data set 1. Again, this could be a result of the increased consistency seen when a person types a familiar phrase.

Figure 3 – Time

User number	1	2	3	4	5
Times the user was recognised (out of 10)	9	9	10	9	7
Accepted impostors at this rate (out of 40)	3	0	7	2	6

6.3 CFNN Neural Network

Figure 2 shows the performance of a trained network attempting to interpolate data produced by the legitimate user and data produced by an impostor. Each user has typed a 14-character id and password 10 times and a window size of 7 was chosen. This gives 13 time-measurements between keystrokes, of which all but the first and last three can be compared to a network output as the window's central element slides over them. The various input vectors and network outputs have been concatenated for compactness: Figure 2 illustrates the network interpolating with much greater accuracy for the legitimate user than for the impostor.

Classification using the mean-square of the network errors for each single id-password combination produces the results in Table 1 (assuming ideal thresholding). Note that each of 5 users has typed an id and password 10 times and 4 impostors have typed the same words 10 times each. The individual results for data set two can be seen in Figure 3.

7 Conclusion

We have shown that there is a distinct pattern in the way people type and that it may be fully exploited to reinforce password security. Our results highlight the PNN and CFNN as strong candidates for this application.

The PNN, by nature, has several advantages inherent in its structure when compared to other networks. Advantages such as data removal/addition without re-training and parallel processing are particularly suited to this application, allowing new users to be added and old ones removed with minimal disruption (a BPNN would require time consuming re-training). The PNN implementation does produce a significantly larger network than the BPNN, with each training sample allocated to a node in the pattern layer. However, the parallel nature of the PNN lends itself easily to implementation with parallel processing.

Using the Manhattan distance measure in place of the Euclidean distance measure has yielded significant improvements in performance for the PNN. We have demonstrated the choice of distance measure to have greater effect when outlier datum is present in the samples.

Problems encountered with outlier data are a major factor affecting the performance of all the neural networks presented. The pre-processing of data using z-scores has been implemented. It offers a method of reducing type one errors, but can have an undesirable effect on the type two errors.

We have demonstrated a novel approach using a CFNN to interpolate typing characteristics. The difference between the CFNN output and the actual sequence can be evaluated, with a large difference indicating an impostor. We have used a very basic interpolation scheme to perform pattern classification with high success rates. The results are good enough to be useful in themselves – with the added benefit that the network is very easy to implement as the windowing is a pre-processing function and, basically, any interpolation method could have been employed – and they also indicate that further work using a more complicated technology such as recurrent networks instead of the CFNN may be very fruitful.

References:

[1] Bleha S., Computer access security systems using keystroke dynamics, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.12, No.12, 1990, pp.1217-22.

[2] Leggett J. and Williams G., Verifying Identity via Keystroke Characteristics, *International Journal of Man-Machine Studies*, Vol.28, No.1, 1988, pp.67-76.

[3] Obaidat M. S. and Macchairolo D., A multilayer neural network system for computer access security, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.24, No.5, 1994, pp.806-13.

[4] Obaidat M. S. and Macchiarolo D., An online neural network system for computer access security, *IEEE Transactions on Industrial Electronics*, Vol.40, No.2, 1993, pp.235-42.

[5] Obaidat M. S. and Sadoun B., Verification of Computer Users Using Keystroke Dynamics, *IEEE Transactions on Systems, Man and Cybernetics*, Vol.27, No.2, 1997, pp.261-69.

[6] Wasserman P. D., *Advanced Methods in Neural Computing*, Van Nostrand Reinhold, 1993.

[7] Hagan M.T. and M. Menhaj, Training Feedforward Networks with the Marquardt Algorithm, *IEEE Transactions on Neural Networks*, Vol.5, No.6, 1994.