USER AUTHENTICATION WITH NEURAL NETWORKS

Nicola Capuano[1], **Marco Marsella**[1], **Sergio Miranda**[1], **Saverio Salerno**[1,2]

[1] CRMPA – Centro di Ricerca in Matematica Pura ed Applicata
c/o DIIMA – University of Salerno, Italy

[2] DIIMA – Dipartimento di Ingegneria dell'Informazione e Matematica Applicata
University of Salerno, Italy

e-mail: niccap@crmpa.unisa.it, marmar@crmpa.unisa.it, miranda@crmpa.unisa.it, salerno@ponza.dia.unisa.it.

Abstract: The purpose of this paper is to introduce Artificial Intelligence in the field of data-security and to propose an easy to implement Neural Networks based method for user authentication. The problem has been faced exploiting an RBF-like Neural Net to recognize the typing style of users asking for connection. The introduction of Neural Nets allows to extract rules directly from row data (users typing times) without any assumption on these. This is the advantage of this approach.

I. INTRODUCTION

An amount bigger and bigger of information is stored, maintained and distributed through computer systems but not all of them must results globally accessible. As a consequence, a primary role is now played by user authentication systems.

Nowadays most computer systems still uses knowledge-based tools of authentication, such as passwords or passphrases. Such methods are economic and easy to implement but they result often ineffective because of the intrinsic facility of knowledge (under form of password) to be transferred. It's easy, for instance, to acquire a user's password through careful observation, and users can disclose their passwords to coworkers and so on, opening up many entry points into a computer system.

A second category of user authentication methods relies on artifacts (keys, magnetic cards, etc.) rather than on the only knowledge. Each access point to the system must be then equipped with specialized hardware, able to refuse access to users without the proper artifact. This approach requires additional hardware costs and suffers the same transferability problems than knowledge-based methods: key, cards and other artifacts can be lost, stolen or duplicated, making useless their protection.

A third category of authentication methods uses, as discriminating factors, some personal characteristics such as fingerprints, voice patterns, signatures, typing styles and so on. Since such personal characteristics are virtually unique of the individual, the *biometrics* is the most promising approach in the field of user identification. Differently from password, keys and cards, personal characteristics are a natural part of the individual so they can't be lost, stolen, forgotten, or transferred.

In this paper we illustrate our method for user authentication which tries to conjugate the easiness of implementation of knowledge-based password approach with the top degree of security offered by the biometric one that appraises a personal and untransferable individual characteristic, such as the typing style. Typing style is an ideal indicator for the authentication because it seems to be unique for many users (also not experienced) and does not involve any additional hardware (except for a keyboard).

The idea of using typing style as a means to identify or authenticate users is not new. Since 1899, Bryan and Harter [2] observed that the telegraph operators had a distinctive keystroking styles called "fists" and that receiving operators often succeeded to identify transmitting operators by listening to the typing rhythms of dot and dashes. Gains et al. in 1980 [3] proposed a solution to the problem of the recognition of individuals from the typing styles using a simple statistic method. The problem has returned in 1993 when Brown and Rogers [1] proposed the first neural approach to the problem. A rather embarrassing limit of their solution is that the recognizing net (a simple MPL with back-propagation) must be trained with correct and wrong samples of password typing. For this reason this method is useless for our purposes because wrong samples should be furnished by other people that, in this way, would know user's password, reducing the system security (knowing other passwords does not allow to enter, but it is a good start).

This problem becomes out-of-date in our project that learns only on patterns furnished by a single user and is able to adapt the own response to contingent changes of his typing style. This results much more useful with inexperienced users that acquire an increasing degree of confidence with the keyboard and tend to increase typing speed.

II. NEURAL TECHNOLOGY USED

A neural network is made of a set of elementary units combined in a layered structure. The elements of the net (neurons) are generally disposed in layers and present a high degree of interconnection with each other. For each connection of the net we associate a weight that represents a sort of inside knowledge. A neural network, by altering the weights of connections across a learning rule, is able to learn a distinctive function from couples of input/output examples (training set), that are repeatedly presented to the model. By the ability of generalizing results of learning on patterns not present in the training set (test set), we deduce the goodness of the trained model.

In our work we use a network structure that is similar to an RBF (Radial Basis Function) net [12] with a learning algorithm that combines characteristics of Martinetz Neural-Gas training [11] with classical Kohonen LVQ (Learning Vector Quantization) [9][10]. Our net is formed by three layers: the first one is composed by a number of input neurons equal to the dimension of the feature space (in our case the feature space is the space of vectors that represent hold and latency times of typing samples); the second one is composed by hidden neurons with radial basis activation function (for example a gaussian); the third one is finally composed by output neurons with "max" activation function. Figure 1 shows the topology of this net.

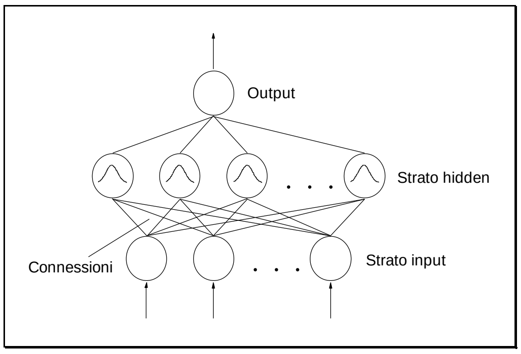

Fig. 1 - Structure of implemented neural net.

If we indicate with *x* an input pattern for the net (a typing sample) and with v_j the weights vector of the *j*-th hidden neuron (composed by weights of all coming-in connections), we could calculate the output of each hidden neuron appraising, in the *x* point, a gaussian centered in v_j with variance s_j as follows:

$$y = \exp\left\{-\frac{x^2}{2s^2}\right\}.$$

Each hidden neuron is then sensitive to the region that locates under the gaussian (receptive field of the neuron) and identifies a cluster or a subcluster in the feature space. The v_j parameter is controlled by the net to move in the space the cluster centroid, while s_j is used to control the spread of the function. Figure 2 shows the region characterized by a function of this kind on a bi-dimensional feature space.

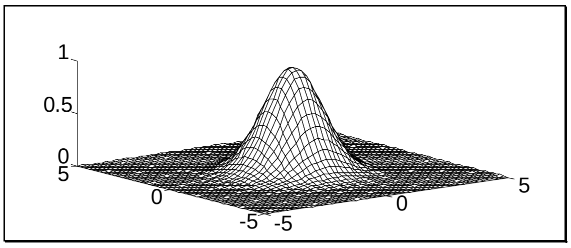

Fig. 2 – An RBF on a bi-dimensional space.

The output neuron has a “max” activation function that forwards as net output the highest activation value among hidden layer that is:

$$z = \max_j y_j$$

The network initially starts with a number of hidden neurons multiple of the number of training set patterns (samples of password typing). Each neuron characterizes a RBF function with spread σ fixed a-priori and centered on the pattern that it represents.

We can’t train this net with classical RBF learning algorithms because we don't have wrong patterns (typing samples of our password furnished by other users are strongly inadvisable) so we decide to use a hybrid LVQ/ Neural-Gas training. Each time that a user tries to enter in the system, the pattern generated from his digitation comes into the net that will answer with a certain output. If this value is over a fixed threshold, the typing style has been recognized like correct and access to the system is then granted. Only in this case the new pattern is used to improve representation that the net has of this user.

Like Neural-Gas algorithm, a classification of activation values of hidden neurons is compiled and then weights update is done as follows:

$$v_{ij}^{(t+1)} = v_{ij}^{(t)} + e^{(t)} \cdot h(rank(j)) \cdot \left(x_i^{(t)} - v_{ij}^{(t)}\right)$$

where $rank(j)$ is the position of *j*-th hidden neuron in the classification, $e^{(t)}$ is the learning rate $\left(0 \le e^{(t)} < 1\right)$ while $h(x)$ is an unimodal function that decreases to the growth of *x*. Using this kind of dynamic learning, neural net is capable to take account of contingent changes of user typing style. This method is preferred to the standard Kohonen training because it converges more quickly and also achieves a lower value of mean distortion.

III. EXPERIMENTS

Necessary training data is collected by capturing press and release times (expressed in milliseconds) of each key in the typing sample. The raw data is then broken into successive keystrokes known as *digraphs*. Latency times are then computed for each digraph by subtracting the release time of the second key from the press time of the first one. The time each key is held down can also be computed by subtracting the release time of a key from its press time. This process is synthesized in table 1.

Press	Release	Time	Hold	Latency
N		0		
	N	109	109	
I		216		107
C		292		-47
	I	339	123	
O		409		-55
	C	464	172	
	O	535	126	
L		561		26
	L	660	99	
A		769		109
	A	851	82	

Tab. 1 – Calculation of hold and latency times.

Times of impact and latency are used directly as input for the early presented neural net without further preprocessing. However a crucial parameter is necessary to be defined: the variance σ for gaussians of hidden neurons. This indicator must be set with accuracy because it governs the recognition severity. A too small σ renders the net very astringent and users can have difficult to be recognized; a too big σ, opposite, renders the net too less astringent and an intrusion can than be easier to happen.

Since radial basis functions are symmetric and no normalization has been made on input patterns, the net could not take account of existing discrepancies among different typing styles: a variance that can results severe for a slow user could results much permitting for a fast one.

A first solution to this problem is to normalize input patterns according to typing duration. Doing this we will preserve the user typing rhythm but we would lose an important evaluation indicator: speed. A second but more legitimate solution is to bind gaussian variance with user speed. Our solution is to associate each hidden *j* to a lot of different variances (building a multivariate gaussian) equal to the number of net inputs in this way:

$$s_{ij} = s + m v_{ij}$$

where s_{ij} is the *i*-th variance of the *j*-th neuron while *m* is an adjustment rate a-priori established. Using this method, a typical hidden neuron activation function changes into the following:

$$y_j = \exp\left\{-\sum_i \frac{(x_i - v_{ij})^2}{2 s_{ij}^2}\right\}.$$

In this way we have a different spread on each axis so we can well manage each different kind of style.

Many experiments have been made to find the best parameters setting. Required values are: the training set cardinality *N* (number of training samples), the hidden layer dimension *M*, the spread indicators σ and *m* and, finally, the learning rate ε for the dynamic adaptation.

The internal net representation of each user improves when *N* is bigger but a too big *N* can slow down algorithm performances. The number *M* of hidden units is a basic parameter, together with learning rate ε, to regulate the dynamic adaptation quality. To avoid initial net training we set *M* as multiple of *N* ($M = nN$) so we can associate each pattern directly to one or more hidden nodes. As bigger is ε is big as more rapid the net is to move toward new configurations to represent changes in the user style. When the net moves toward new configurations, it tends to not recognize the old one anymore (this situation increases when *n* is small). A big *n* gives place to a net with more memory, a small *n*, instead, generates a net that forgets old configurations with the risk that users could access to system with increasing difficulty. The *n* and ε parameters must then be fixed according to the degree of dynamics that we want to allow to the net.

Fixing ε to 0.05 we now show how net memory changes increasing *n* value. To do that, we have trained our net on a training set composed by only one pattern and we have furnished, in the validation phase, a sequence of such samples that each of them is far from the precedent by a casual percent of among 3% and 5% maintaining the same direction. Table 2 shows, for a subset of possible value of *n*, the number of necessary samples (average) to forget (and then not recognize) the initial pattern.

N value	Memory
1	**22**
2	**27**
3	**42**
4	**86**
5	**264**

Tab. 2 – Evaluation of net memory

Now we present some experiments of authentication effected on a set of real users. We have settled down the following parameters configuration: N=5, n=3 (M=nN=15) , σ=0.6, m=20, ε=0.03 and we have trained our net on samples furnished by 10 different users. Results in terms of percent of succeeded authentications and intrusions are showed in table 3.

User	Authentication	Intrusions
1	98 %	5 %
2	99 %	0 %
3	100 %	1 %
4	100 %	2 %
5	78 %	5 %
6	100 %	12 %
7	93 %	0 %
8	89 %	15 %
9	95 %	3 %
10	97 %	0 %

Tab. 3 – Authentication results

Results are more that satisfactory because the percent of authentications always guarantees the access to the owner user (if not at the first attempt, surely at the second one). An extraneous who knows an user password succeeds to enter in the system with very low probability.

IV. CONCLUSIONS

In this paper we have shown our approach to the introduction of Artificial Intelligence (with reference to Neural Nets) in a data-security context. Neural Nets are in fact used to recognize users by their own typing style.

High level of flexibility and ease of implementation was required so, the quantity of training pattern has been reduced to the minimum and the requirement of wrong patterns has been eliminated. An original net model capable to work in these abhorrent conditions has then been proposed and experimental results on real data are more then satisfactory.

However we are still far from the user identification through the only typing style analysis. Positive results in this sense could be obtained only training the system of recognition on long typing sequences.

A possible solution could be obtained by running on the system (in background) a software capable to collect and elaborate, in real time, typing times for each user and train the net continually. If this software would be sufficiently optimized, learning would happen in wholly transparent manner. After few time, a net able to distinguish among all users would be available.

At the moment, the analysis of the typing styles with neural nets can only furnish an additional worth to system reliability supporting already existing authentication methods (as password approach) rather than replacing them completely.

REFERENCES

[1] **M. Brown e J. Rogers** "User Identification Via Keystroke Characteristics of Typed Names Using Neural Networks" in *International Journal of Man-Machine Studies*, no. 39, pp. 999-1014, 1993.

[2] **W. Bryan e N. Harter** "Studies in the Telegraphic Language" in *Psychological Review*, no. 6, pp. 345-375, 1899.

[3] **R. Gaines, W. Lisowsky, S. Press e N. Shapiro** "Authentication by Keystroke Timing: Some Preliminary Results" in *Rand Report R-2526-NSF, Rand, Santa Monica, Calif.*, 1980.

[4] **J. Rogers** "Neural Network User Authentication" in *AI Expert* vol. 10, no. 6, pp. 29-33, 1995.

[5] **J. Legget, G. Williams, M. Usnick e M. Longnecker** "Dynamic Identity Verification Via Keystroke Characteristics" in *International Journal of Man-Machine Studies* no. 35, pp. 859-870, 1991.

[6] **J. Legget e G. Williams** "Verifying Identity Via Keystroke Characteristics" in *International Journal of Man-Machine Studies* no. 28, pp. 67-76, 1988.

[7] **D. Umphress e G. Williams** "Identity Verification Through Keyboard Characteristics" in *International Journal of Man-Machine Studies* no. 23, pp. 263-273, 1985.

[8] **D. Rumelhart, G. Hinton, R.Williams** "Learning Internal Representations by Error Propagation" in *ICS Report 8506 Institute of Cognitive Science University of California*, 1985.

[9] **T. Kohonen** "Self-organized formation of topologically correct feature maps" in *Biological Cybernetics* vol. 43, pp. 59-69, 1982.

[10] **T. Kohonen** "Learning Vector Quantization for Pattern Recognition" in *Technical Report TKK-F-A601, Helsinki University of Technology, Finland*, 1986.

[11] **T. Martinetz, S. Berkovich, K. Schulten** "Neural-Gas Network for Vector Quantization and its Application to Time-Series Prediction" in *IEEE transactions on Neural Networks*, vol. 4, no. 4, pp. 558-568, 1993.

[12] **M. Musavi, W. Ahmed, K. Chan, K. Faris e D. Hummels** "On the Training of Radial Basis Function Classifiers" in *Neural Networks*, vol. 5, pp. 595-603, 1992.