# Using Neural Network in Cryptography

**Eva Volná**

University of Ostrava, 30.dubna-st. 22, 70103 Ostrava 1, Czech Republic,
e-mail: eva.volna@osu.cz

*Abstract: This paper deals with using neural network in cryptography, e.g. designing such neural network, that would be practically used in the area of cryptography both in cryptography and cryptoanalysis. The method should be fully independent on the frequency analyse in the text. Thus, it means: to design the topology of the neural network, to design the method of training algorithm of the neural network, to design the training set for training.*

## 1. Introduction to cryptography

The cryptography deals with building such systems of security of news, that secure any from reading of trespasser. Systems of data privacy are called the cipher systems. The file of rules are made for encryption of every news, it is called the cipher key. Encryption is a process in which we transform the open text, e.g. news to cipher text according to rules. Cryptanalysis of the news is the inverse process, in which the receiver of the cipher transforms it to the original text. The cipher key must have several heavy attributes. The best one is the singularity of encryption and cryptanalysis. The open text is usually composed of international alphabet characters, digits and punctuation marks. The cipher text has the same composition as the open text. Very often we find only characters of international alphabet or only digits. The reason for it is the easier transport per media. The next cipher systems are the matter of the historical sequence: transposition ciphers, substitution ciphers, cipher tables and codes.

Simultaneously with secrecy of information the tendency for reading the cipher news without knowing the cipher key was evolved. Cipher keys were watched very closely. However, it was much easier to set the cipher correspondences, it means to cryptology. Its main goal is to guess the cipher news and to reconstruct the used keys with the help of good analysis of cipher news. It makes use of mathematical statistics, algebra, mathematical linguistics, etc., as well as known mistakes made by ciphers too. The legality of the open text and the applied cipher key are reflected in every cipher system. Improving the cipher key helps to decreases this legality. The safety of the cipher system (key) is in its immunity against the decipher.

P. Sinák et al. (eds.), *The State of the Art in Computational Intelligence*

# 2. Application of evolutionary algorithm to neural network

The paradigm makes the basic attribute for neural network. As paradigm we understand its topology, adaptation and activation. In my paper I work with feedforward neural nets with adapted backpropagation. Optimisation of neural network topology by the help of evolutionary algorithm is used for this type of neural network. The neural network topology must correspond to the complexity of task, e.g. the number of training pattern, its inputs and outputs and the structure of relations, that are described. By the adaptation of a small net (this adaptation) is stopped in local minimum and the network must be enlarged by other units. Conversely, the big network allows to find the global minimum of error function, however, the computing absorption is increased. The found configuration of network usually generalises the training pattern including its inaccuracies to a great extent and for untaught patterns, it gives false results.

Among the stochastic evolutionary algorithms, the genetic algorithms are used for optimisation of the designed neural network topology, the standard training algorithm and genetic algorithms are used for adaptation. These algorithms utilise the information from previous step. We want the sum of square of deviation of output from neural network to be the least. One of the main conditions for representation of variables is the appropriate representation of variables with chain of characters (e.g. bit chain with 0 and 1) and the speed of calculation of fitness function in the given point. The method of calculation [2]: First, we must propose maximal architecture of neural network (e.g. maximum number of hidden layers and maximum number of hidden units) before the main calculation. Every individual in the population is characterised by its representation scheme. To optimise the population is it necessary to solve the defined problems. Thereafter the process of genetic algorithms is applied. The finding of optimal population is ended when the population achieves the maximal generation or when fitness function achieves the maximal defined value. We want to find the best architecture with weights, that are adapted. Next, three digits are generated for every connection coming out from a unit. In case the connection does not exist, three zeros are attached to the given place. To every non zero weight value numerated in the following way is thus assigned:

$$w_{i,j,k,l} = \eta \cdot [e_2 (e_1 \cdot 2^1 + e_0 \cdot 2^0)],$$

where $w_{i,j,k,l} = w(x_{i,j}, x_{k,l})$ is the weight value between the $j$-th unit in the $i$-th layer and the $l$-th unit in the $k$-th layer,

$\eta$ is a parameter of learning (the constant equal 0.1)

$e_i$ ($i = 0, 1$) is a randomly generated digit

$e_2$ is a sign bit: if $e_2$=0 resp. $e_2$=1 the expression is positive resp. negative

Every population is then described by chromosomes of individual. For every scheme the number of connections between units, number of hidden units and number of hidden layers are calculated. Error ($E$) between the desired and the real output is solved in a procedure which implements the forward distribution of signals in multi layer neural network. On the basis of it, these entries for every topology of the appropriate individual its fitness function is calculated as follows:

$$Fitness_i * = k_1 \cdot (E_i)^2 + k_2 \cdot (hidden\_units_i)^2 + k_3 \cdot (hidden\_layers_i)^2$$

The general fitness function is calculated as follows:

$$Fitness_i = \begin{cases} k - \left(Fitness_i^* + k_5\right) & \text{if } E_i > k_4 \\ k - Fitness_i^* & \text{otherwise} \end{cases}$$

for $i = 1, ...,N$,

where $E_i$ is error for the i-th network;

$hidden\_units_i$ are the number of hidden units for the i-th network;

$hidden\_layers_i$ are the number of hidden layers for the i-th network;

$k_1, k_2, k_3, k_4, k_5, k$ are constants;

$N$ is the number of individuals in the population.

Values of constants $k_1, k_2, k_3, k_4, k_5, k$ are needed to be chosen with the dependency of solution tasks, for the fitness function values appropriate neural network to be calculate the best. For each fitness function is calculated the probability of reproduction its existing individual by standard method (see [1]). The main *crossover* runs in two following steps: we choose two suitable parents to crossover randomly. Next we generate two numbers randomly. The first of them is from the above limitation of number of biases (e.g. coded biases values). The other is from the above limitation of number of weight connections (e.g. coded weight values). The chosen individuals exchange their substrings of bias and weight connections from the places defined in substrings. If the input condition of *mutation* is fulfilled, one of the individual is randomly chosen and in its genetic representation is randomly chosen in one place. To optimise the population is it necessary to solve the problems defined in the introduction. Adaptation of the best found network architecture is finished with backpropagation.

# 3. Experiment

Every practical encryption system consists of four fundamental parts:

- The message that you wish to encrypt (called the *plain text*)
- The message after it is encrypted (called the *cipher text*)
- The encryption algorithm
- The key (or keys), which is used by encryption algorithm

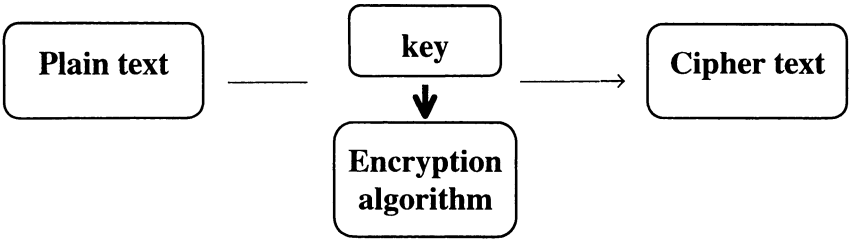


**Fig. 1.** A simple example of encryption

We can successfully use neural networks as an encryption algorithm in cryptography. Parameters of the adapted neural network are then included to key. We will work with a multilayer neural network that is adapted by backpropagation. Topology of this neural network is based on the training set (see table ). Thus, to decrypt the message the neural network will have a topology of 5 - ? - 6, and to encrypt the message it will have a topology of 6 - ? - 5 where symbol „? " is a unknown number of hidden units. The chain of chars of the plain text in a training set is equivalent to a binary value, that is 96 less than its ASCII code. The cipher text is a random chain of bits. The number of units in the hidden layer is suggested using the above described method. The security for all encryption systems is based on a cryptographic key. The simple encryption systems use a single key for both encryption and decryption. The good encryption systems use two keys. A message encrypted with one key can be decrypted only with the other. If we use the neural network as decryption algorithm, the key will have the adapted neural network parameters; it is its topology (architecture) and its configuration (the weight values on connections). Generally, the key is written as follow:

[*Hidden, Input,* $N_1$*....*$N_{hidden}$*, Output, Biases, Weights coming from the input units, Weights coming from the hidden units*]

where *Hidden* is the number of hidden units;

*Input* is the number of input units;

$N_1$*....*$N_{hidden}$ is the number of units in the relevant hidden layer;

*Output* is the number of output units;

*Biases* are the values of biases;

*Weights coming from the input units* are the weight values coming from the input units to hidden units and output units;

*Weights coming from the hidden units* are the weight values coming from the hidden units to next hidden units and output units;

**Table 1.** The training set

| THE PLAIN TEXT | | | THE CIPHER TEXT | THE PLAIN TEXT | | | THE CIPHER TEXT |
|---|---|---|---|---|---|---|---|
| Char | ASCII code (DEC) | The chain of bits | The chain of bits | Char | ASCII code (DEC) | The chain of bits | The chain of bits |
| a | 97 | 00001 | 000010 | n | 110 | 01110 | 011100 |
| b | 982 | 00010 | 100110 | o | 111 | 01111 | 101000 |
| c | 99 | 00011 | 001011 | p | 112 | 10000 | 001010 |
| d | 100 | 00100 | 011010 | q | 113 | 10001 | 010011 |
| e | 101 | 00101 | 100000 | r | 114 | 10010 | 010111 |
| f | 102 | 00110 | 001110 | s | 115 | 10011 | 100111 |
| g | 103 | 00111 | 100101 | t | 116 | 10100 | 001111 |
| h | 104 | 01000 | 010010 | u | 117 | 10101 | 010100 |
| i | 105 | 01001 | 001000 | v | 118 | 10110 | 001100 |
| j | 106 | 01010 | 011110 | w | 119 | 10111 | 100100 |
| k | 107 | 01011 | 001001 | x | 120 | 11000 | 011011 |
| l | 108 | 01100 | 010110 | y | 121 | 11001 | 010001 |
| m | 109 | 01101 | 011000 | z | 122 | 11010 | 001101 |

The figure 2 shows the full-interconnected neural network topology for decryption problem and the corresponding key has the following parameters: [*1, 5, 5, 6, biases, weight values coming from the input units, weight values coming from the hidden units*]

The figure 3 shows the full-interconnected neural network topology for encryption problem and the corresponding key has the follow parameters: [*1, 6, 6, 5, biases, weight values coming from the input units, weight values coming from the hidden units*]

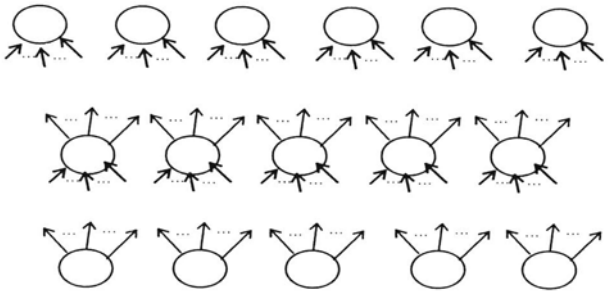

**Fig. 2.** The neural network topology for decryption problem

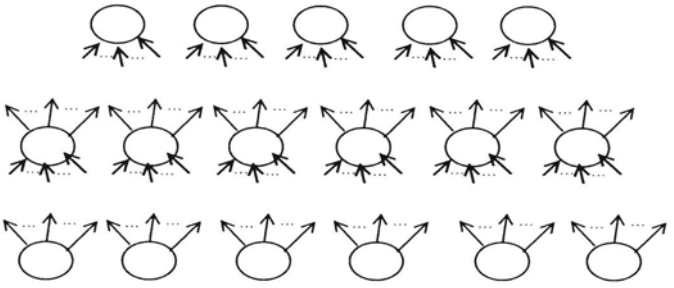

**Fig. 3.** The neural network topology for encryption problem

# 4. Conclusion

The goal of cryptography is to make it impossible to take a cipher and reproduce the original plain text without the corresponding key. With good cryptography, your messages are encrypted in such a way that brute force attacks against the algorithm or the key are all but impossible. Good cryptography gets its security by using incredibly long keys and by using encryption algorithms that are resistant to other form of attack. The neural net application represents a way of the next development in good cryptography.

## References

1. Lawrence, D.: Handbook of genetic algorithms. Van Nostrand Reinhold, New York 1991.
2. Volná, E.: Learning algorithm which learns both architectures and weights of feedforward neural networks. (Neural Network World. Int. Journal on Neural & Mass-Parallel Comp. and Inf. Systems. **8(6)** (1998), 653-664.
3. Garfinger, S.: Šifrování pro každého. PGP: Pretty Good Privanci. Computer Press, Praha 1998.