Tikrit Journal of Eng. Sciences/Vol.17/No.2/June 2010, (51-57)

Multiple Data Type Encryption Using Genetic Neural Network

Raid R. Al-Nima
Assistant Lecturer
Computer Eng. Dept.

Ali N. Hamed
Assistant Lecturer
Power Eng. Dept.

Ridwaan Y. Srdeeq
Assistant Lecturer
Power Eng. Dept.

Technical Colleges Mosul

Abstract

The aim of this research is to build a ciphering system by using genetic neural network technique to protect data against unauthorized access to the data being transferred.

The encryption data includes three stages: first Stage :- Using the genetic algorithm to train backpropagation neural network for obtaining weights. Second Stage:- Encryption data by using the weights obtained from first backpropagation layer and consider its weights as a encrypted key. third Stage:- Decryption data by using the weights obtained from second backpropagation layer and consider its weights as a decrypted key.

This system is similar to coding asymmetric, and have the ability of coding a group of data such as:- pictures, waves and texts.

Keywords: Encryption , Decryption , Genetic , Neural network , Genetic neural network.

تشفير أنواع مختلفة من البيانات باستخدام الشبكة العصبية الجينية

الخلاصة

يهدف البحث لبناء نظام تشفير فعال باستخدام الشبكات العصبية الذكية لغرض حماية البيانات من أي محاولات لكشفها والتلاعب بها.

تم اعتماد ثلاثة مراحل لأجل تشفير أنواع مختلفة من البيانات، المرحلة الأولى تضمنت تدريب الشبكة العصبية ذات الإنتشار الخلفي باستخدام الخوارزمية الجينية لإيجاد الأوزان، المرحلة الثانية (وهي مرحلة تشفير البيانات) إرسال البيانات المطلوب تشفيرها إلى داخل الشبكة العصبية واستخلاص البيانات المشفرة من الطبقة المخفية، المرحلة الثالثة (وهي مرحلة فك تشفير البيانات) ترسل البيانات المشفرة خلال الشبكة العصبية من الطبقة المخفية إلى طبقة الإخراج.

النظام المقترح يستخدم اسلوب التشفير الغير متناظر حيث أن مفتاح التشفير (أوزان الطبقة الأولى) يختلف عن مفتاح فك التشفير (أوزان الطبقة الثانية). وقد نجح هذا النظام بتشفير أنواع مختلفة من البيانات مثل: صور، موجات ونصوص.

Tikrit Journal of Eng. Sciences/Vol.17/No.2/June 2010, (51-57)

Abbreviations

DES: Data Encryption Standard
GA: Genetic Algorithm
NN: Neural Network

Introduction

Encryption is the conversion of data into a form, called a **ciphertext**, that cannot be easily understood by unauthorized people . Decryption is the process of converting encrypted data back into its original form so it can be understood.

In order to recover the contents of an encrypted signal easily, a correct decryption key is required. The key is an algorithm that "undoes" the work of the encryption algorithm. Alternatively, a computer that can be used in an attempt to "break" the cipher. The more complex encryption algorithm, the more difficult to eavesdrop on the communications without access to the key.

In 1996, Menezes et. al. found a method which has commonly used until recently employs named DES [1]. In 2000 Toru Ohira identified the encryption process by a coupling dynamics with nonlinear threshold function and various time delays between different bits, or neurons, in the original data [2]. In 2008, The research of Amera I. has develop a Hebbian network through qualitative primary weight which had a large size for ciphering process[3]. This research aim is to build a ciphering system by using genetic neural network technique. Encryption data is implemented by using the weights which is obtained from hidden layer. Decryption data is implemented by using weights obtained from output layer.

The aim of this research is to build a powerful ciphering system to obtain a dynamic encrypted data and this make it more powerful against unauthorized access.

Evolving Neural Networks

Neural networks are biologically motivated approaches to machine learning, inspired by ideas from neuroscience. Recently some efforts have been made to use genetic algorithms to evolve aspects of neural networks.

In its simplest "feed forward", a neural network is a collection of connected activatable units ("neurons") in which the connections are weighted, usually with real −valued weights. The network is presented with an activation pattern on its input units, such a set of numbers representing features of an image to be classified (e.g., the pixels in an image of a handwritten letter of the alphabet). Activation spreads in a forward direction from the input units through one or more layers of middle ("hidden") units to the output units over the weighted connections. Typically, the activation coming into a unit from other units is multiplied by the weights on the links over which it spreads, and then is added together with other incoming activation.

The result is typically thresholded (i.e., the unit "turns on" if the resulting activation is above that unit's threshold). This process means to roughly mimic the way activation spreads through networks of neurons in the brain. In a feed forward network, activation spreads only in a forward direction, from the input layer through the hidden layers to the output layer.

After activation has spread through a feed forward network, the resulting activation pattern on the output units encodes the network's "answer" to the input (e.g., a classification of the input pattern as the letter A). In most applications, the network learns a correct mapping between input and output patterns via a learning algorithm [4].

Typically the weights are initially set to small random values. Then a set of training inputs is presented sequentially to the network. In the back−propagation learning procedure[5], after each input has propagated through the network and an output has been produced, a "teacher" compares the activation value at each output unit with the correct values, and the weights in the network are adjusted in order to reduce the difference between the network's output and the correct output. Each iteration of this procedure is called a "training cycle," and a complete pass of training cycles through the set of training inputs is called a "training epoch." (Typically many training epochs are needed for a network to learn to successfully classify a given set of training inputs.) This type of procedure is known as "supervised learning," since a teacher supervises the learning by providing correct output values to guide the learning process. In "unsupervised learning" there is no teacher, and the learning system must learn on its own using less detailed (and sometimes less reliable) environmental feedback on its performance [5,6].

To apply GAs to neural networks. Some aspects that can be evolved are the weights in a fixed network, the network architecture (i.e., the number of units and their interconnections can change), and the learning rule used by the network[4]. In this research the weights in a fixed network are trained by GAs.

Suggested Genetic Characteristics

The genetic neural network which is suggested has the following characteristics:

- Genetic neural network has the same number of nodes in the input layer, hidden layer and output layer. The activation functions used are tan-sigmoid activation functions for the neurons on the hidden layer and pure linear activation functions for the neurons on the output layer.
- The fitness which used in the encryption system is shown in equation (1) below:

$$\delta_k = \sum_{k=1}^{m} |t_k - y_k| \quad \text{..... (1)}$$

Where each output unit (y_k, k = 1……..m) receives a target pattern (t_k) corresponding to the input training pattern to computes its error information term (δ_k).

- The population type which specifies the type of the input to the fitness function [7], was used as Bit string.
- The population size which specifies how many individuals there are in each generation[7], was used equal to 30 generations.
- The Rank scaling function used the raw scores based on the rank of each individual, rather than its score. The rank of an individual is its position in the sorted scores. The rank of the fittest individual is 1, the next fittest is 2 and so on. Rank fitness scaling removes the effect of the spread of the raw scores.[4,7]
- Stochastic uniform selection option used to lays out a line in which each parent corresponds to a section of the line of length proportional to its expectation. The algorithm moves along the line in steps of equal size, one step for each parent. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.[7]
- Gaussian mutation function used to add a random number to each vector entry of an individual. This random number is taken from a Gaussian distribution centered on zero. The variance of this distribution can be controlled with two parameters. The Scale parameter which determines

the variance at the first generation and the Shrink parameter which controls how variance shrinks as generations go by[7]. The Scale parameter and the Shrink parameter both were used equal to 1.0.

- Scattered crossover option is used to create a random binary vector. It then selects the genes where the vector is 1 from the first parent, and the genes where the vector is 0 from the second parent, and combines the genes to form the child. For example: [7]

  p1 = [a b c d e f g h]
  p2 = [1 2 3 4 5 6 7 8]
  random crossover vector = [1 1 0 0 1 0 0 0]
  child = [a b 3 4 e 6 7 8]

By this topology the genetic algorithm succeeded in back propagation training stage and then it was able to used in the next stages (encryption and decryption).

Encryption and Decryption Algorithm

We split the testing algorithm used in NN into two parts. the first part is used for encrypting data, the encrypted data are obtained from the output of the hidden layer after launching the hidden layer with the input vector (data). The second part was used for decryption by sending the encrypted vector from the hidden layer to the output layer. The algorithm of encryption and decryption are:

Encryption Algorithm

Step 0. Initialize weights (from GA training algorithm).
Step 1. For each input vector, do step 2-3.
Step 2. For i = 1……n: set activation of input unit x_i
Step 3. For j = 1……p: calculate the activation of hidden unit z_j

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{.....} \quad (2)$$

$$z_j = f(z\_in_j) \quad \text{....} \quad (3)$$

It is clear that the bias on a typical hidden unit z_j is denoted v_{0j}.

Decryption Algorithm

Step 4. For k = 1……m: calculate the activation of output unit y_k

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad \text{...} \quad (4)$$

$$y_k = f(y\_in_k) \quad \text{.....} \quad (5)$$

It is clear that the bias on a typical output unit y_k is denoted by w_{0k}.[8]

Practical Implementation

The suggested network for training is a feed forward back propagation and the training will be implemented by GA for data that requires encryption by using a technique in back propagation that is a number of input units (data before encryption) equal to the number of hidden units and equal to the number of output units (data after decryption). The network was trained by using supervised target which is the original input itself.

We use the single hidden layer and the activation function (tan-sigmoid) type while in the decryption case. It is done by using pure linear activation function. This is a technique in the encryption/decryption process in the back propagation neural network. See Fig.(1).

Results and Discussion

A ciphering system obtained after training a backpropagation neural network by applying genetic algorithm. As mentioned, this network have the same nodes for inputs hidden and output layers. The input data is equal to the output target in size and value. The hidden layer region gives the encryption data and the output layer region gives the decryption data. Fig. (2) shows the best function value in each generation versus iteration number.

In this system, retraining the network with the same data produce different encryption data value every time, because of the randomly genetic algorithm technique for getting the weights. See Fig. (3) and Fig. (4) which illustrate how obtaining different encryption data picture.

The suggested encryption system can efficiently ciphering different types of data (pictures, waves and texts). See Fig. (5), Fig. (6) and Fig. (7) which illustrate a wave before and after encryption process.

Also as mentioned in the picture encryption process, retraining the network with the same wave produce different encryption wave. As show in Fig. 8.

Text encryption can encrypt any type of string symbol such as: characters, numbers, arithmetic symbols, ... etc. table 1 illustrates samples of characters encrypted in extremely different characters.

We can see the strong difference and difficult relationship between the original characters and the encrypted characters after a neural network training process.

Conclusions

The suggested ciphering system obtains dynamic encrypted data, this make it more powerful against unauthorized access. This system used the genetic algorithm to train backpropagation neural network which have number of input nodes equals to the number of hidden nodes and also equal to the number of output nodes. The target which the network used to train is equal to the input data in size and value. Then ciphering process consisted of three stages:

1- Training a network by using genetic algorithm to obtain weights.
2- Encryption data by using the weights which obtained from stage one and consider the weights of first layer is encrypted key.
3- Decryption data by using the weights which obtained from first stage and consider the weights of second layer is decrypted key.

This technique succeeded to encrypt different types of data (pictures, waves and texts) and gave another advantage, when re-training the same input data sample we obtained different encrypted data.

References

1. Menezes, A. J., van Oorschot, P., and Vanstone, S. A. (1996). "Handbook of applied cryptography". Boca Raton: CRC Press.

2. Toru Ohira, "Toward encryption with neural network analog" Bruges (Belgium), 26-28 April 2000, D-Facto public.,ISPN 2-930307-00-5,pp.147-152.

3. Amera I., " Using Hebbian network for cipher", The first scientific conference for information technology SCIT, 2008.

4. M. Melanie, "An Introduction to Genetic Algorithms", A Bradford Book The MIT Press Cambridge, Massachusetts • London, England Fifth printing, 1999

5. Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error

propagation". In D. E. Rumelhart, J. L. McClelland. PDP Research Group, Parallel Distributed Processing, Volume 1: Foundations. MIT Press. 1986.

6. Hertz, J., Krogh, A., and Palmer, R. G., "Introduction to the Theory of Neural Computation", Addison−Wesley, 1991.

7. The Math Works Inc., "Neural Network Toolbox, For Use with MATLAB", Ver. 7.6, 2008, MA, USA.

8. L. Fausett, "Fundamental of Neural Networks, Architectures, Algorithms and Applications", Printice Hall Int. Snc., 1994.

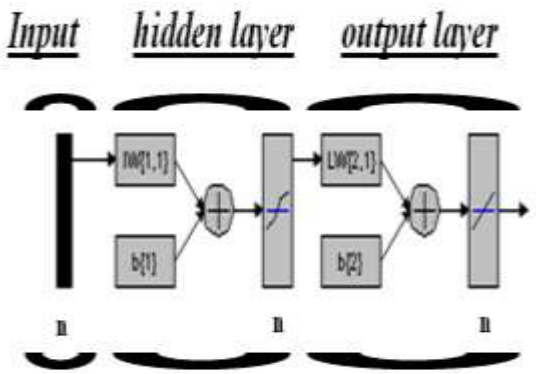

Figure (1): Encryption backpropagation neural network architecture

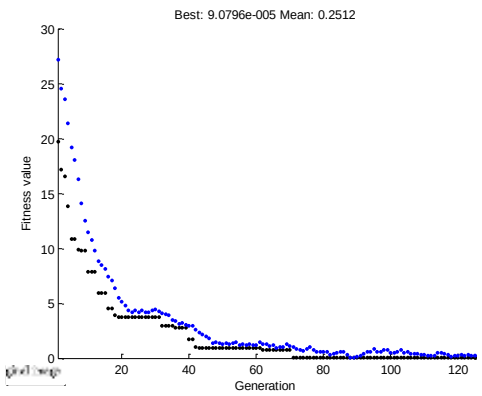

Figure (2): The best function value in each generation versus iteration number

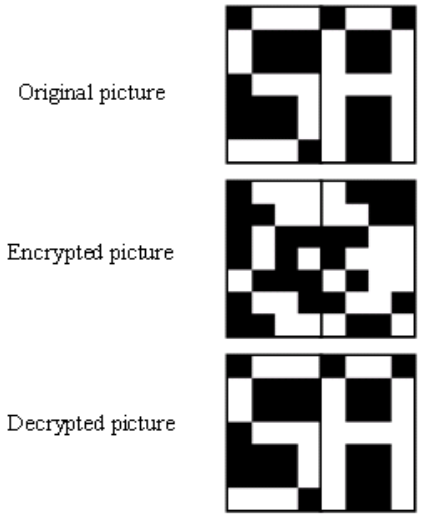

Figure (3): first attempt picture encryption

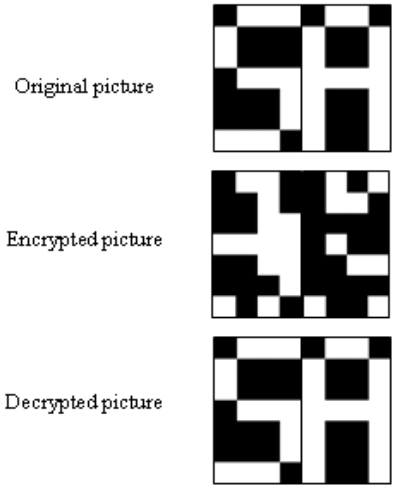

Figure (4): second attempt picture encryption

Tikrit Journal of Eng. Sciences/Vol.17/No.2/June 2010, (51-57)

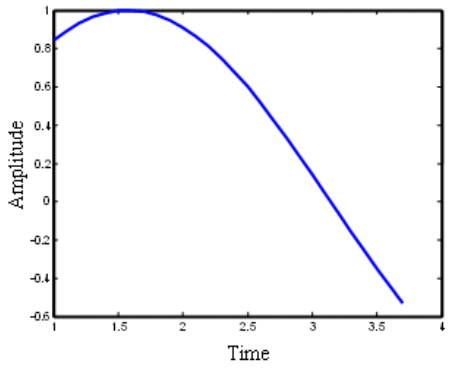

Figure (5) Original wave form before encryption process

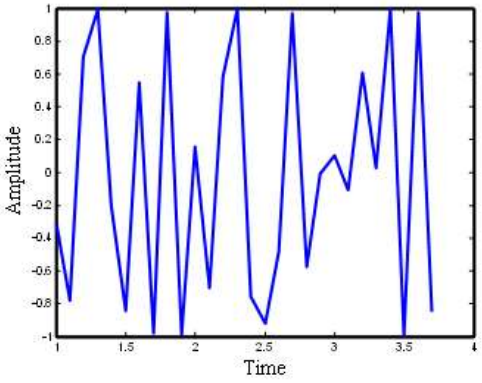

Figure (8): Wave form after second encryption process

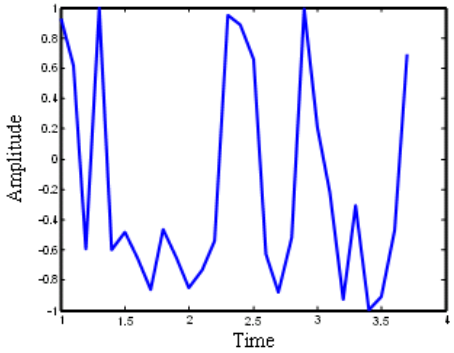

Figure (6) Wave form after encryption process

Table (1): Text encryption

Original text	After encryption	After decryption
abc	vWD	abc
xyz	÷°e	xyz
123	Qc@	123

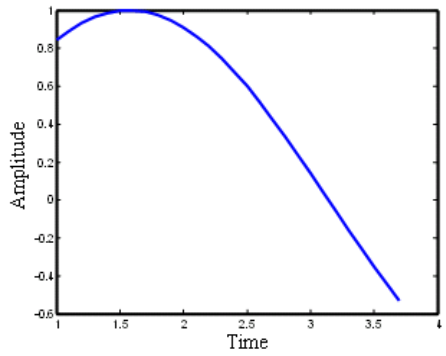

Figure (7) Wave form after decryption process