Data encryption Using Backpropagation Neural Network

Raid R. Al-nima[1] , Muhanad L.[1] , Saba Q. Hassan[2]

[1]*Computer Engineering Dept., Technical Colleges, Mosul, Iraq*

[2] *Computer Systems Dept., College of Computer Sciences and Mathematics , University of Mosul , Mosul, Iraq*

(Received 20 / 7 / 2008 , Accepted 12 / 4 / 2009)

Abstract:

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

The encryption data includes three stages: first Stage :- Training a network by using backpropagation to obtain weights. Second Stage:- Encryption data by using the weights obtained from first stage and consider the weights of first layer as a public key. third Stage:- Decryption data by using the weights obtained from the first stage and consider the weights of second layer as a private key. The three stages are attained 100% success for data encryption process and data getting back process.

This technique is similar to coding asymmetric, and have the ability of coding a group of data such as:- text, characters, numbers and waves. This work is executed by computer type P4 with whole equipments and Matlab language version 7.

Keywords: Encryption , Decryption , Backpropagation .

1. 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 it is to eavesdrop on the communications without access to the key.

In 1976, Diflie and Hellmann found a method based on numbers theories for creating a secret key over a public channel accessible to any attacker [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 2003,The thesis of Amera I. has develop a Hebbian network through qualitative primary weight which had a large size for ciphering process [3]. This research was to build a ciphering system by using backpropagation neural network technique. Encryption data is implemented by using the weights which 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.

2. Architecture of standard BPN

A multilayer neural network with one layer of hidden units (the Z units) is shown in Fig. 1 The output units (the Y units) and the hidden units also may have biases. The bias on a typical output unit Y_k is denoted by w_{0k}; the bias on a typical hidden unit Z_j is denoted v_{0j}. These bias terms act like weights on connections from units whose output is always 1. Only the direction of information flow for the feedforward phase of operation is shown. During the backpropagation phase of learning, signals are sent in the reverse direction. The algorithm in the next section is presented for one hidden layer, which is adequate for a large number of applications [4].

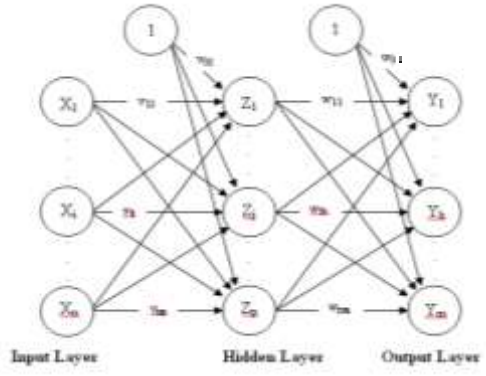

Figure 1: BPN with one hidden layer

3. The training Algorithm of BPN

As mentioned earlier, training a network by backpropagation involves three stages: the feedforward of the input training pattern, the backpropagation of the associated error, and the adjustment of the weights.

During feedforward, each input unit (X_i) receives an input signal and broadcasts this signal to each of the hidden units Z_1 …….. Z_p through the (v) weights. Each hidden unit then computes its activation and results in its signal (z_j) to each output unit through the (w) weights. Each output unit (Y_k) computes its activation (y_k) to form the response of the net for the given input pattern. During training, each output unit compares its computed activation y_k with its target value t_k to determine the associated error for that pattern with that unit. Based on this error, the factor δ_k *(k=1 ……. m)* is computed, δ_k is used to distribute the error at output unit Y_k back to all units in the previous layer (the hidden units that are connected to Y_k). It is also used (later) to update the weights between the output and the hidden layer. In a similar manner, the factor δ_j *(j=1 ……. p)* is computed for each hidden unit Z_j. It is not necessary to propagate the error back to the input layer, but δ_j is used to update the weights between the hidden layer and the input layer. When all the δ factors have been determined, the weights for all layers are adjusted simultaneously. The adjustment to the weight w_{jk} (from hidden unit Z_j to output unit Y_k) is based on the factor δ_k and the activation z_j of the hidden unit Z_j. The adjustment to the weight v_{ij}

from input unit X_i to hidden unit Z_j) is based on the factor δ_j and the activation x_i of the input unit.

An epoch is one cycle through the entire set of training vectors. Typically, many epochs are required for training a BPN. The foregoing algorithm updates the weights after each training pattern is presented. A common variation is a updating batch, in which weight updates are accumulated over an entire epoch before being applied [4].

3.1 The general training algorithm for BPN is as follows:

Step 0. Initialize weights.
(Set to small random values).

Step 1. While stopping, condition is false(output meet the gaol), do Steps 2-9.

Step 2. For each training pair, do Steps 3-8.

Feedforward:

Step 3. Each input unit (X_i, $i = 1……n$) receives input signal x_i and broadcasts this signal to all units in the layer above (the hidden units).

Step 4. Each hidden unit (Z_j, $j = 1……p$) sums its

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

applies its activation function to compute its output signal.

$$Z_j = f(z\_in_j) \quad \text{…….. (2)}$$

and sends this signal to all units in the layer above (output units).

Step 5. Each output unit (Y_k, $k = 1 …… m$) sums its weighted input signals.

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad \text{…….. (3)}$$

and applies its activation function to compute its output signal.

$$y_k = f(y\_in_k) \quad \text{…….. (4)}$$

Backpropagation of error:

Step 6. Each output unit (Y_k, $k = 1……m$) receives a target pattern corresponding to the input training pattern, computes its error information term.

$$\delta_k = (t_k - y_k)\, f'(y\_in_k) \quad \text{…….. (5)}$$

calculates its weight correction term (used to update w_{jk} later).

$$\Delta w_{ik} = \alpha \delta_k z_j \quad \text{…….. (6)}$$

calculates its bias correction term (used to update w_{0k} later).

$$\Delta w_{0k} = \alpha \delta_k \quad \text{…….. (7)}$$

and sends δ_k to units in the layer below.

Step 7. Each hidden unit (Z_j, $j = 1……p$) sums its delta inputs (from units in the layer above).

$$\delta\_in_j = w_{0k} + \sum_{k=1}^{m} \delta_k w_{jk} \quad \text{…….. (8)}$$

multiplies by the derivative of its activation function to calculate its error information term.

$$\delta_j = \delta\_in_j \; f'(z\_in_j) \quad \text{…….. (9)}$$

calculates its weight correction term (used to update v_{ij} later).

$$\Delta v_{ij} = \alpha \delta_j x_i \quad \text{…..……..…….. (10)}$$

and calculates its bias correction term (used to update v_{0j} later).

$$\Delta v_{0j} = \alpha \delta_j \quad \text{……..…….. (11)}$$

Update weights and biases:

Step 8. Each output unit (Y_k, $k = 1…….m$) updates its bias and weights
($j = 0…….p$):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} \quad \text{…….. (12)}$$

Each hidden unit (Z_j, $j = 1…….p$) updates its bias and weights
($i = 0……n$):

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij} \quad \text{…….. (13)}$$

Step 9. Test stopping condition. [4][5]

The learning rate(α) is taken equal to 0.01, and the momentum constant is taken equal to 0.9 [6]. Momentum is a standard technique that is used to speed up convergence and maintain generalization performance [7].

4. Encryption and decryption algorithm

The idea of encryption and decryption processes are by splitting the testing algorithm used in NN into two parts. The first part is used for encrypting data, by sending the input vector to the hidden layer. The second part was used for decrypting data by sending the encrypted vector to the output layer. The algorithm of encryption and decryption are:

Step 0. Initialize weights (from 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 $j = 1……p$:

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{…….. (14)}$$

$$z_j = \frac{2}{1 + \exp(-2 * z\_in_j)} - 1 \quad \text{…….. (15)}$$

Step 4. For $k = 1……m$:

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad \text{…….. (16)}$$

$$y_k = y\_in_k \quad \text{…...…… (17) [4].}$$

The encryption process include steps from 0 to 3, and the decryption process include step 4 only.

The final weights must be known after training. Then it will be divided into two portions. The first portion is from input layer to hidden layer for encryption side, then the encrypted data are obtained from the output of the hidden layer after launching the hidden layer with the input data vector. The second portion is from hidden layer to output layer for decryption side, then the decrypted data are obtained from the output layer after launching the output layer with the hidden data vector.

5. The practical implementation

The suggested network for training is a feedforward backpropagation and the training will be implemented

for data that requires encryption by using a technique in backpropagation that is a number of input units (data before encryption) which is equal the number of hidden layer and equal to the number of data output (data after encrypted). The network was trained by using supervised target that is a trained network on input and target (original input) and this may be entered by user and software system.

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

6. The Results

A ciphering system obtained after training a backpropagation neural network having the same nodes for inputs, hidden and output layers. The output target is equal to the input data in size and values. The hidden layer region gives the encryption data, see fig. 2.

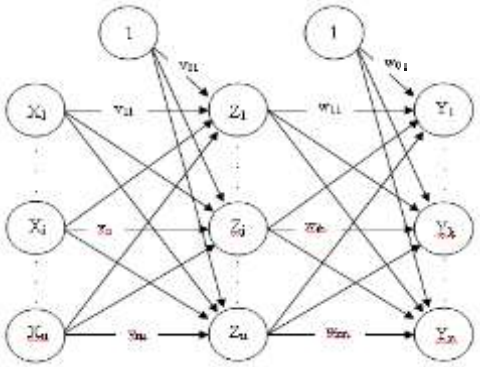

Figure 2: Encryption backpropagation neural network architecture

In this system, retraining the network with the same data produce different encryption data value every time, because of the initialization of backpropagation algorithm with random values of weights. See fig. 3 and fig. 4 which illustrates how obtaining different encryption image data:

Figure 3: First attempt image encryption

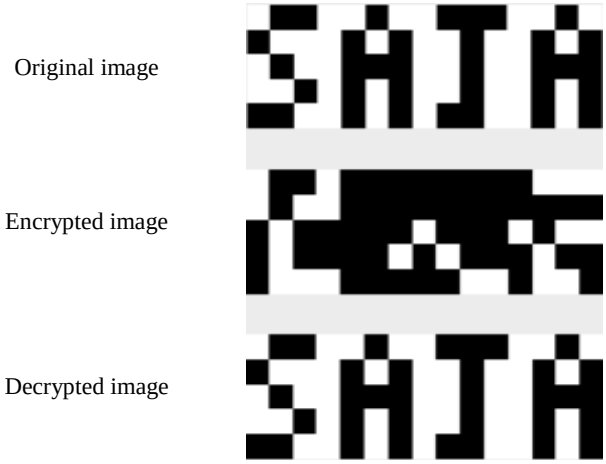

Figure 4: Second attempt image encryption

The suggested encryption system can efficiently ciphering different types of data (binary images, signals type wav and texts). See Fig. 5, Fig. 6 and Fig. 7 for signal before and after encryption process and signal after decryption process. Text encryption can encrypt any type of string symbol such as: characters, numbers, arithmetic symbols, ... etc. table 1 shows samples of characters processed by the system then produced extremely in different characters.

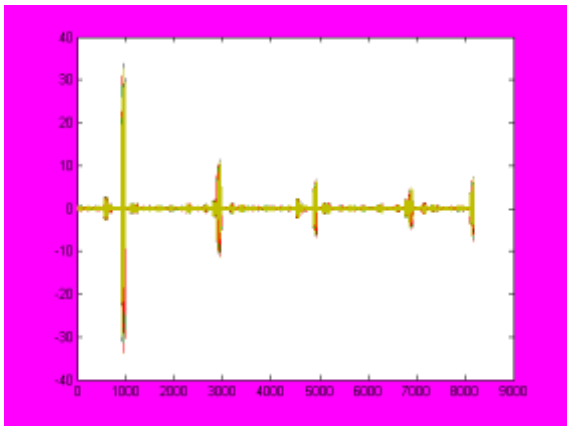

Figure 5 Before encryption process

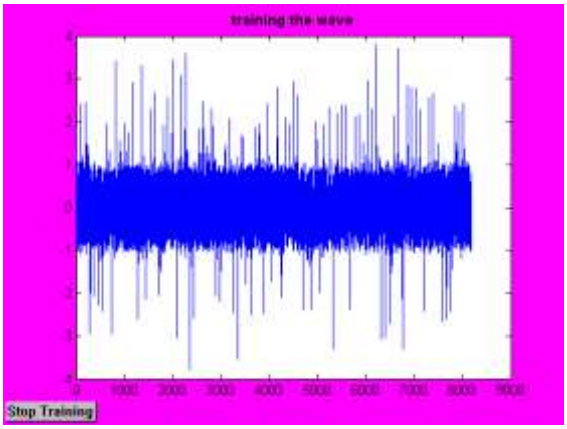

Figure 6: After encryption process

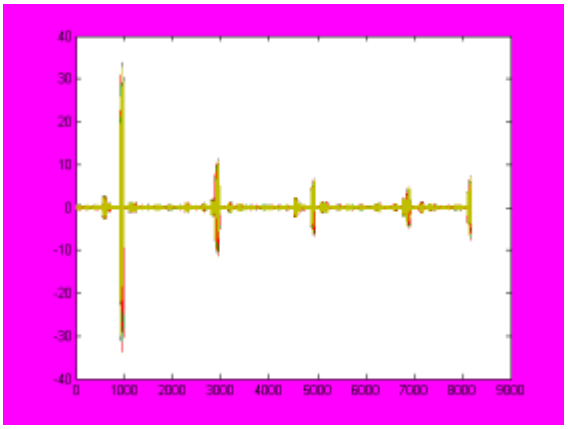

Figure 7: After decryption process

Table 1: Text encryption

Original text	After encryption	Before encryption
Abc	LZ	abc
Xyz	:hH	xyz

From the previous results, it is obvious the precision of the suggested encryption/decryption method. Moreover the accuracy can be tested by using equation 18:

$$e = \sum_{i=1}^{n} \left| (x_i - y_i) \right| \quad \text{.....} \quad (18)$$

Where

e: is the minimum error.
x_i: is the input data before encryption process.
y_i: is the output data after encryption process.
n: is the size of data.

The previous equation lead to the minimum error which compared with tolerance equal to (0.0001).

7. Conclusions

The suggested ciphering system obtains dynamic encrypted data, this make it more powerful against unauthorized access. This system used the backpropagation neural network having 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 values. Then ciphering process consisted of three stages:

1- Training a network by using backpropagation network and consider the target is equal to the input.
2- Encryption data by using the weights which obtained from input layer to hidden layer and consider this weights as a public key.
3- Decryption data by using the weights which obtained from hidden layer to output layer and consider this weights as a private key.

This technique succeeded to encrypt different types of data (texts, signals type wav and binary images) and gave another advantage, when re-encrypting the same input data sample we obtained different encrypted data.

References

[1] D. R. Stinson, "Cryptography: Theory and Practice", CRC press, 1995.
[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”, Thesis in computer and mathematical sciences university of Mosul/Iraq, 2003.
[4] L. Fausett, "Fundamental of Neural Networks, Architectures, Algorithms and Applications", Printice Hall Int. Snc., 1994.
[5] S. Suresh, S.N. Omkar, and V. Mani, "Parallel Implementation of Back-Propagation Algorithm in Networks of Workstations", IEEE Transactions On Parallel And Distributed Systems, Vol. 16, No. 1, January 2005.
[6] The Math Works Inc., "Neural Network Toolbox, For Use with MATLAB", Ver. 6.5, 1998, MA, USA.
[7] E. Istook, T. Martinez, "Improved backpropagation learning in neural networks with windowed momentum", International journal of neural systems, vol. 12, no. 3&4, pp. 303-318, 2002.
http://synapse.cs.byu.edu/detailed_publications.php

تشفير البيانات باستخدام الشبكة العصبية ذات الإنتشار الخلفي

رائد رافع النعمة[1] ، مهند لقمان[1] ، صبا قاسم[2]

[1] *قسم هندسة تقنيات الحاسبات ، الكلية التقنية ،الموصل ، العراق*

[2] *قسم علوم الحاسبات ، كلية علوم الحاسبات والرياضيات ، جامعة الموصل ، الموصل ، العراق*

(تاريخ الاستلام: ٢٠ / ٧ /٢٠٠٨ ، تاريخ القبول: ١٢ / ٤ / ٢٠٠٩)

الملخص:

يهدف البحث إلى بناء تقنية تشفير باستخدام الشبكة العصبية ذات الإنتشار الخلفي لغرض حماية البيانات من هجمات المتطفلين.

تقنية التشفير المطروحة تتضمن ثلاثة مراحل: المرحلة الأولى:- تدريب البيانات باستخدام الشبكة العصبية ذات الإنتشار الخلفي للحصول على الأوزان. المرحلة الثانية:- تشفير البيانات عن طريق استخدام الأوزان المستخلصة من القسم الأول للشبكة العصبية (الأوزان الواقعة بين طبقة الإدخال والطبقة المخفية)، واعتبار هذه الأوزان المفتاح العام. المرحلة الثالثة:- فك تشفير البيانات باستخدام الأوزان المستخلصة من القسم الثاني للشبكة العصبية (الأوزان الواقعة بين الطبقة المخفية وطبقة الإخراج)، واعتبار هذه الأوزان المفتاح الخاص. حققت هذه التقنية نسبة نجاح وصلت إلى ١٠٠% عن تشفير البيانات وإعادتها إلى صيغتها الأصلية.

تعتمد التقنية أسلوب التشفير غير التناظري، وهي لها القابلية على تشفير مجموعة من البيانات مثل: نص، حروف، أرقام و موجات. تم تنفيذ هذا العمل بواسطة حاسبة نوع بنتيوم ٤ كاملة المواصفات والبرمجة كانت تحت نظام الماتلاب الإصدار ٧.