Correcting Errors in Linear Codes with Neural Network

Maung Maung Htay[*], *S. Sitharama Iyengar*, and *Si-Qing Zheng*

Department of Mathematics and Computer Science.[*]
Virginia Military Institute
Lexington, VA 24450

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

Abstract

In this paper we develop an error-correcting algorithm in the framework of neural networks. We consider the problem of detecting and correcting of a linear code which is assumed to have at most one error.

keyword

Neural networks, error detection, error correction, check matrix, linear code.

1. Introduction

The quest for efficient computational approaches to neurocomputing problems covers a broad spectrum of approaches such as neural connectivity, learning and correcting errors paradigms. The mathematical analog-model of the nervous cell must reflect axiomatically the real neuron features. Thus, the development of better models that simulate the human nervous system is an important research topic. With the advent of massively parallel processing and efficient computational techniques, renewed interest on neural nets has occurred. Neural nets have a vast number of applications. They include visual and audio perception, pattern matching and classification, robotics and sensor processing.

When a digital message is transmitted over a long distance, the received message may not be exactly as it is sent since there may be some interference. In these situations, we should be able to detect and if possible, correct errors. A lot of algorithms has been discovered about error detecting/correcting codes[1,2]. This paper is an initial attempt to study from the neural network perspective the problem of correcting single-bit errors in linear codes.

In this paper, we present the basic concept of neural network, and some definitions and theorem related to linear codes as preliminaries and then the construction of a neural net for error detection and correction with an illustrative example.

2. Preliminaries

In this section, we introduce some basics of neural networks and the theory of error detection and correction in linear codes.

2.1 Neural Network

Neural network is a new information processing technique [3]. It is also a computer-based simulation of living neurons system. Neural net models are composed of many nonlinear computational elements operating in parallel and arranged in patterns reminiscent of biological neural nets. They are called artificial neurons or computational elements or nodes which are connected via variable weights.

McCulloch and Pitts's model computes a weighted sum of its inputs from other units and outputs a one or zero according to whether the sum is above or below a certain threshold. Figure 1 describes a simple model of a neural net. [5]

In Figure 1, the weight w_{ij} represents the strength of the synapse connecting neuron i to neuron j and it can be positive or negative depending on whether synapse is excitatory or inhibitory. If there is

no synapse between neuron i and neuron j, it is zero. μ_i represents the threshold.

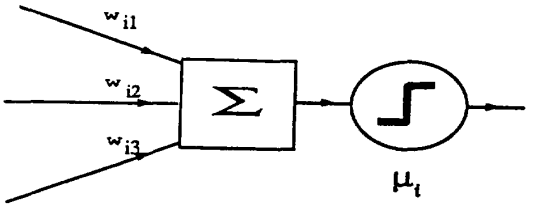

Figure 1. Schematic diagram of a McCulloch-Pitts neuron.

Mathematically [5], we can express the concept of the above model as:

$$\eta_i(t+1)=\Theta(\sum_j w_{ij}\eta_j(t)-\mu_i) \qquad (1)$$

where η_i is either 1 or 0 and represents the state of neuron i as firing or not firing respectively. Time t is taken as discrete, with one time unit elapsing per processing step. Θ (x) is the unit step function. This function is also known as threshold function. All neurons may not have the same fixed delay (t -> t + 1). They are not updated synchronously by a central clock. Thus a simple generalization of the McCulloch-Pitts equation (1), which includes asynchronous updating and some other features such as continuous-valued units is

$$\eta_i=g(\sum_j w_{ij}\eta_j-\mu_i) \qquad (2)$$

where η_i is continuous-valued and is called the state or activation of unit i, g(x) is more general than the threshold function Θ (x) and is called the activation function, gain function, transfer function or squashing function [5].

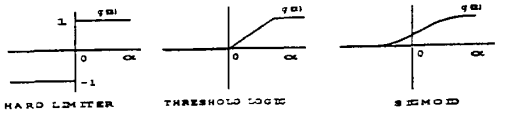

Figure 2. Common Activation Functions

There are three common types of activation functions: hard limiters, threshold logic elements, and sigmoid functions, as shown in Figure 2. [3] There may be other types of activation functions. McCulloch-Pitts neuron is a simple but computationally powerful device. In principle, a synchronous assembly of such neurons is capable of *universal computation* for suitably chosen weight w_{ij}. In other words, it can perform any computation that a digital computer can, though not necessarily as rapid and as convenient as the computer [5].

2.2 Linear Codes

In this section, we describe some definitions and a theorem from [1, 2] as background information concerning our proposed model for detecting and correcting errors in linear codes within the the framework of neural network.

Let V^n be the set of all binary words of length n. A binary *code* of length n is simply a subset C of V^n and the members of C are called *codewords.*

Definition 2.1 : A code C in V^n is linear if whenever a, b $\in$ C then a + b $\in$ C. In other words, C is linear if and only if it is a subgroup of V^n in Z_2, where Z_2 is the set of integers modulo 2. According to Lagrange's theorem, since a linear code is a subgroup of V^n, its size |C| is a divisor of $|V^n| = 2^n$. Hence |C| is an integer of the form 2^k, $0 \le k \le n$ and k is called the dimension of C.

We can also define the linear code in terms of parity-check matrix as follows.

Definition 2.2 : A code C is a linear code if it is defined by C = { x $\in$ V^n | $Hx' = 0'$ } where H is a binary matrix with n columns and known as parity-check matrix (or simply, check matrix), x' denotes the word x in V^n considered as a column vector and $0'$ denotes the all-zero column vector.

For a detailed knowledge of linear codes, the reader may refer to references [1,2]. Here, we will present some definitions and a theorem relevant to our problem.

Theorem 2.1 : If no column of H consists entirely of zeros, and no two columns are the same, then the code C defined by the check matrix H will correct one error.

Proof is given in reference [1].

A conventional algorithm [1] for detecting and correcting a single error in the code C is as follows:

```
begin
  Let z be the received word.
  Compute Hz'  ( z' is transport of z )
  if Hz' = 0' then z is a codeword
  else begin
        Find the column h^(i) of H
        such that Hz' = h^(i)
        ith bit of z is incorrect.
        Complement the ith bit of z
      end
end
```

In this paper, we construct our proposed network model of correcting errors in linear codes assuming that all the necessary conditions prescribed in the above definitions and theorem are satisfied.

3. Neural Network Construction

To solve the given problem, we need to construct a neural net of two phases. In the first phase, the net detects an error position in the codeword. In the second phase, the net corrects the erroneous bit and produces the correct codeword.

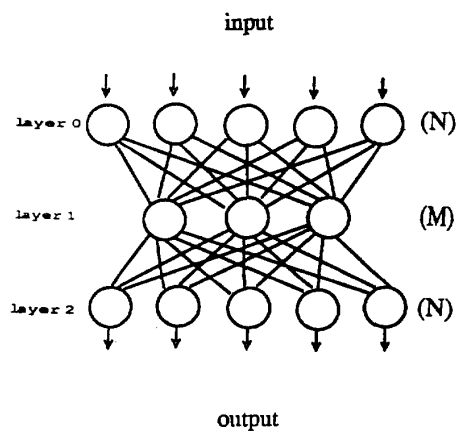

Figure 3. Error detection phase (First phase)

3.1 Error Detection Phase

It is shown in Figure 3. There are N inputs to the network, which consists of associative memories comprising of two layers of neurons. The number N represents the length of a codeword. In the hidden layer (also referred to as layer 1), there are M neurons where M represents the number of rows in the check matrix H which is assumed to be given. Codewords are defined depending on the check matrix. The reader may refer to the references [1,2] for more knowledge of the check matrix. Every input is connected with each neuron of layer 1. Layer 2 has N neurons which determine the position of the error of the received word. Every neuron of layer 1 is connected to each neuron of layer 2. Neurons in every layer are numbered by positive integer in consecutive increasing order starting from 1.

The elements in the given check matrix are used as weights at the connections of the N input with the nodes of the first layer. Let H be the given check matrix and h_{ij} is the element at *ith* row and *jth* column of H.

We denote the weight w_{ij}^{01} of the connection of the *ith* neuron of the input layer with the *jth* neuron of layer 1.

$$w_{ij}^{01}=h_{ji} \ , \ 1 \le i \le \mathrm{N} \text{ and } 1 \le j \le \mathrm{M}$$

We use w_{ij}^{12} to denote the weight of the connection of the *ith* neuron of layer 1 with the *jth* neuron of layer 2. The values of w_{ij}^{12} are also assigned by the elements of the check matrix H, but in the bipolar form (digit 0 is replaced by -1).

i.e. For $1 \le i \le \mathrm{M}$, $1 \le j \le \mathrm{N}$.

$$w_{ij}^{12} = \begin{cases} 1 & \text{if } h_{ij}=1, \\ -1 & \text{if } h_{ij}=0, \end{cases}$$

For each neuron of layer 1, the sgn function [Figure 4.a] of the modulo 2 function is used as the activation function, while the hard limiter activation function [Figure 4.b] used for the neurons of layer 2 [3,4].

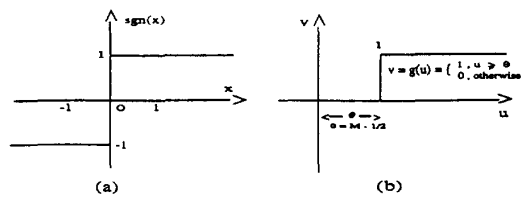

Figure 4. Activation functions used in the proposed network

To detect the error, the received word is passed through the first layer and we allow the network to progress until it falls into a stable situation. In this case, if one neuron of layer 2 produces value '1' while the rest are '0', then it shows the position of the bit which has been transmitted incorrectly. If there is no error in the received word, all neurons of the layer 2 will output 0's.

To demonstrate our solution for detecting error, we introduce the following variables and activation functions.

(1) The initial input v_j^0, $1 \le j \le N$

(2) The output of neuron t in the layer 1 v_t^1, $1 \le t \le M$

(3) The output of neuron i in the layer 2 v_i^2, $1 \le i \le N$

Let g^1 and g^2 be the activation functions for neurons of layer 1 and layer 2 respectively. g^1 is a sgn function of modulo 2 function on the weighted sum of given inputs v_j^0, where $1 \le j \le N$.

Let $u_t^1 = \sum_j^N w_{jt}^{01} v_j^0$, $1 \le t \le M$ and $v_t^1 = g^1(u_t^1) = sgn(u_t^1 modulo\ 2)$,

$$i.e. \qquad v_t^1 = g^1(u_t^1) = \begin{cases} 1, & \text{if } u_t^1 \ mod\ 2 = 1 \\ -1, & otherwise \end{cases}$$

The output values of the neurons of layer 2 are determined by a hard limiter function g^2 [3,4].

Let $u_i^2 = \sum_j^M w_{ji}^{12} v_j^1$, $1 \le i \le N$ and $\theta = M - 1/2$, then we have,

$$i.e. \qquad v_i^2 = g^2(u_i^2) = \begin{cases} 1, & \text{if } u_i^2 \ge \theta \\ 0, & otherwise \end{cases}$$

In other words, since the *ith* neuron of layer 2 accepts as input the value u_i^2, where $u_i^2 = M$ and $u_i^2 > \theta$, the output v_i^2 will be equal to 1. For each neuron $j \ne i$ of layer 2, it holds that $u_j^2 < M - 1$ and $u_j^2 < \theta$. Therefore the output v_j^2 will be equal to 0 [4].

3.2 Error Correction Phase

In this second phase, we use the exclusive or (XOR) network for finding the correct codeword. There are many methods for constructing the XOR network. In this paper we adopted two methods from [5] which are shown in Figures 5a and 5b.

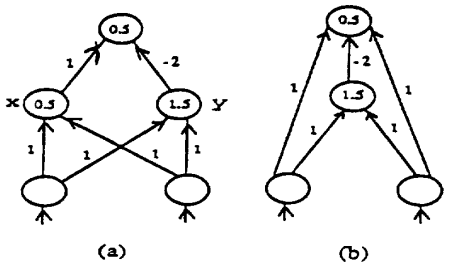

Figure 5. Networks for XOR

These networks use 0/1 threshold units. Each unit is shown with its threshold. In Figure 5a, the two neurons in the hidden layer compute the logical OR (left neuron x) and AND (right neuron y) of the two inputs and the output fires only when the x neuron fires and the y neuron does not fire.

The second method [Figure 5b] needs only two neurons, and one in the hidden layer computes a logical AND to inhibit the output unit when both inputs are on [5].

We use the corresponding pairs of bits from the output of phase 1 and the received word as the input to one of the XOR networks shown in Figure 5. The output of the second phase will be the correct codeword that we have expected.

4. An Illustrative Example

In this section, we use an example to demonstrate how error detection and error correction are worked out using our proposed network.

Let C be the linear code defined by the check matrix

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

If the word 110110 is received and only one error has occurred, we will find the intended codeword by using our proposed network.

In this problem, since the number of row in the check matrix is 3 and the length of the codeword is 6, we have M = 3 and N = 6.

Error Detection

The weights of the synapse connecting between input layer 0 and layer 1 are:

$$\begin{array}{lll} w_{11}^{01}=1, & w_{12}^{01}=1, & w_{13}^{01}=1 \\ w_{21}^{01}=1, & w_{22}^{01}=1, & w_{23}^{01}=0 \\ w_{31}^{01}=0, & w_{32}^{01}=0, & w_{33}^{01}=1 \\ w_{41}^{01}=1, & w_{42}^{01}=0, & w_{43}^{01}=1 \\ w_{51}^{01}=0, & w_{52}^{01}=1, & w_{53}^{01}=0 \\ w_{61}^{01}=1, & w_{62}^{01}=0, & w_{63}^{01}=0 \end{array}$$

Inputs for the layer 1, i.e. bits of the word received, are

$$v_1^0 = 1,\ v_2^0 = 1,\ v_3^0 = 0$$
$$v_4^0 = 1,\ v_5^0 = 1,\ v_6^0 = 0$$

According to the proposed network, we need to find the weighted sum of these inputs as follows:

$$u_1^1 = \sum_j^N w_{j1}^{01} v_j^0 = 1.1+1.1+0.0+1.1+0.1+0.0=3$$

$$u_2^1 = \sum_j^N w_{j2}^{01} v_j^0 = 1.1+1.1+0.0+0.1+1.1+0.0=3$$

$$u_3^1 = \sum_j^N w_{j3}^{01} v_j^0 = 1.1+0.0+1.0+1.1+0.1+0.0=2$$

The outputs of neurons in the layer 1 are :

$$v_1^1 = g^1(u_1^1)=1$$
$$v_2^1 = g^1(u_2^1)=1$$
$$v_3^1 = g^1(u_3^1)=-1$$

The weights of synapse connecting layer 1 and layer 2 are :

$$\begin{array}{llll} w_{11}^{12}=1, & w_{21}^{12}=1, & w_{31}^{12}=1 \\ w_{12}^{12}=1, & w_{22}^{12}=1, & w_{32}^{12}=-1 \\ w_{13}^{12}=-1, & w_{23}^{12}=-1, & w_{33}^{12}=1 \\ w_{14}^{12}=1, & w_{24}^{12}=-1, & w_{34}^{12}=1 \\ w_{15}^{12}=-1, & w_{25}^{12}=1, & w_{35}^{12}=-1 \\ w_{16}^{12}=1, & w_{26}^{12}=-1, & w_{36}^{12}=-1 \end{array}$$

Inputs for neurons at layer 2 are :

$$v_1^1 = 1, \quad v_1^1 = 1, \quad v_3^1 = -1$$

The weighted sum of these inputs are:

$$u_1^2 = \sum_j^M w_{j1}^{12} v_j^1 = 1.1+1.1+1.\ -1 = 1$$

$$u_2^2 = \sum_j^M w_{j2}^{12} v_j^1 = 1.1+1.1+-1.\ -1 = 3$$

$$u_3^2 = \sum_j^M w_{j3}^{12} v_j^1 = -1.1+-1.1+1.\ -1 = -3$$

$$u_4^2 = \sum_j^M w_{j4}^{12} v_j^1 = 1.1+-1.1+1.\ -1 = -1$$

$$u_5^2 = \sum_j^M w_{j5}^{12} v_j^1 = -1.1+1.1+-1.\ -1 = 1$$

$$u_6^2 = \sum_j^M w_{j6}^{12} v_j^1 = 1.1+-1.1+-1.\ -1 = 1$$

Since threshold $\theta = M - 1/2 = 3 - 1/2 = 2.5$, the outputs of neurons in the layer 2 are :

$$v_1^2 = g^2(u_1^2)=0$$
$$v_2^2 = g^2(u_2^2)=1$$
$$v_3^2 = g^2(u_3^2)=0$$
$$v_4^2 = g^2(u_4^2)=0$$
$$v_5^2 = g^2(u_5^2)=0$$
$$v_6^2 = g^2(u_6^2)=0$$

After the first phase, we have detected that second position of the given word is in error.

Error Correction

In the second phase, we use the XOR network with the inputs of the corresponding bit positions of the output of phase 1 (0 1 0 0 0 0) and the received word (1 1 0 1 1 0). Then the XOR network produces the correct codeword (1 0 0 1 1 0).

5. Concluding Remarks

The quest for efficient computational approach to neural network problems has undergone a significant evolution in the last few years. This paper presented an error-correcting algorithm in the framework of neural networks. We considered the problem of detection and correction of a linear code which is assumed to have at most one error. The effort of simulation runs for this network convinced us its correctness, convergence and high speed computation. Follow-up studies are needed to gain better insight into the general problem of error-correcting and neurocomputing formalism for modeling of error correcting paradigms with neural circuits. We may also continue working on the emerging area of dynamic systems in the context of neural network. The advantage of the proposed network may be applied for the real time error detection and correction and parallel computation in the context of information flow in pipelied implementation.

6. Acknowledgement

We would like to thank Dr. Bogdan Oporowski, Department of Mathematics, Louisiana State University, for his useful suggestion during our discussion. We also thank Professor Ray and Mr. Tun Aung Gyaw of University of Illinois, Champaign, for their encouragement and many helpful suggestions.

7. References.

[1]. Biggs, Norman L. "Discrete Mathematics", Revised edition 1989, Oxford University Press.

[2]. J. H. van Litt, "Introduction to Coding", 1982, Springer-Verlag, New York Inc.

[3]. R. P. Lippmann, "Introduction to computing with neural nets."

IEEE Trans. on Acoustics, Speech and Signal Processing (April 1987)4-22.

[4]. D. J. Evans, M. Adamopoulos, S.Kortesis, and K. Tsouros

"Searching sets of properties with neural networks." Parallel Computing 16(1990) 279-285 North-Holland.

[5]. Hertz, Kaogh, and Palmar, "Introduction to the theory of neural computing."

1991, Addison Welsley Publishing Company.

[6]. S. S. Iyengar "Computer Modeling of Biological System", CRC press, 1982.