The Hamming Code Performance Analysis using RBF Neural Network

Omid Haddadi, Zahra Abbasi, and Hossein TooToonchy, *Member, IAENG*

***Abstract*—In this paper the Hamming decoding model development, and BER curve performance, including Error histogram, target Mean Square Error, Training state, Regression curve and the impact of employing a different number of Neurons in RBF Neural network will be investigated. The Hamming (15,11) will be used to develop the results, and diagrams throughout this article. The results, and simulations in this paper are generated via Matlab Neural Network Toolbox 2013.**

***Index Terms*— BER Performance, Hamming Code, Neural Network, RBFN.**

I. Introduction

WHENEVER a transmitter broadcasts a signal over a long distance, the received message may be different than the original one due to a couple of reasons, including but not limited to noise, fading, and jamming. The impact of such an error could be as small as misunderstanding a word in a telephonic conversation or as big as losing connection to a space station thousands of miles away. Due to possible catastrophic impacts of such errors in communication; the detection, and error correction have always been the centers of interest for scientists, engineers and researchers in the field. One type of error control coding scheme is the linear block coding. In this method, some extra bits are inserted into the symbol stream emitted by the source. This is done to, investigate the error detection process, as well as correcting the transmission errors [1]. By using the channel coding, the probability of bit error (P_B) will be reduced significantly, at the cost of bandwidth, and added network complexity [2].

The very first step in error detection, and correction is the error modeling and simulation. One of the most accurate, and reliable modeling and identification algorithms developed is Artificial Neural Networks (ANN). Artificial neural networks are circuits, computer algorithms, or mathematical representations of the massively connected set of neurons that form artificial biological networks that mimic the neuron behaviors. They have been shown to be useful, as an alternative computing technology, and have proven to be useful in a variety of tasks such as pattern recognition, signal processing, estimation, and control problems. Their Ability to learn from examples has been particularly useful. Among the diverse set of neural network algorithms, the RBF method will be adopted in this paper due to various advantages that will be discussed in the subsequent sections.

In this paper, the Hamming code (15,11) is simulated via RBF neural network, and different outputs, including the BER curve performance, Error histogram, and target MSE are discussed. The different sections of this paper are organized as follows. In sections II, and section III a review and introduction to Hamming code, and Neural Networks will be presented, respectively. In section IV the simulation results are developed and discussed. Finally, in sections V and VI the results and conclusion will be presented respectively.

II. Hamming code

A. A brief Introduction to Hamming Code

In the late 1940's Claude Shannon was developing an information theory, and coding as a mathematical model for communication. At the same time, Richard Hamming, a colleague of Shannon's at Bell Laboratories, found a need for error correction in his work on computers. The parity checking was already being used to detect errors in the calculations of the relay based computers at the time, and Hamming realized that a more sophisticated pattern of parity checking can be used to correct a single error along with the detection of double instances.

In 1950s, Hamming published what is now known as the Hamming code. The single error correcting binary Hamming code, their single error correcting and double error detecting extended version, marked the beginning of coding theory. These codes remain important to this day, for theoretical, practical as well as historical reasons.

Hamming code is a class of block codes characterized by the structure (n,k)= $(2^m-1, 2^m-1-m)$ where m=2,3,…. This is an error detecting or error-correcting binary code, which transmits n, bits for every k source bits. They are capable of correcting all single errors or detecting all combinations of 2 or fewer errors within a block. For performance over a Gaussian channel using coherently demodulated BPSK, the channel symbol probability can be expressed in terms of E_c/N_0 as follows:

Manuscript received July 12, 2014; revised August 9, 2014.

Omid Haddadi, M.Sc. research assistant of Electrical Engineering, Department of California State University, Fullerton (Phone: 310-349-7533; e-mail: Omid.haddadi@csu.fullerton.edu).

Zahra Abbasi, B.Sc. in Radiation Technology, independent research assistant at Saddleback and Irvine Valley Colleges (Phone: 949-616-4551; Email: zabbasi1@saddleback.edu).

Hossein Tootoonchy, M.Sc. research assistant of Electrical Engineering Department at California State University, Fullerton (Phone: 949-616-6249; Email: Tootoonchy@csu.fullerton.edu).

$$P=Q\left(\sqrt{\frac{2E_c}{N_0}}\right) \quad (1)$$

Where is E_c/N_0 a code symbol energy per noise spectral density and Q(x) is called the complementary error function. [2]. In this paper, the m=4, is considered. Thus, the (n,k) Hamming code will be (15,11) . For this Hamming code the generator matrix is:

$$G=\begin{pmatrix}1&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\0&1&1&0&0&1&0&0&0&0&0&0&0&0&0\\0&0&1&1&0&0&1&0&0&0&0&0&0&0&0\\1&1&0&1&0&0&0&1&0&0&0&0&0&0&0\\1&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\0&1&0&1&0&0&0&0&0&1&0&0&0&0&0\\1&1&1&0&0&0&0&0&0&0&1&0&0&0&0\\0&1&1&1&0&0&0&0&0&0&0&1&0&0&0\\1&1&1&1&0&0&0&0&0&0&0&0&1&0&0\\1&0&1&1&0&0&0&0&0&0&0&0&0&1&0\\1&0&0&1&0&0&0&0&0&0&0&0&0&0&1\end{pmatrix}$$

So the parity check matrix is given by:

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

B. Syndrome and Error Detection

Let $V=(V_0,V_1,\ldots,V_{n-1})$ be a code word that was transmitted over a noisy channel and $r=(r_0,r_1,\ldots,r_{n-1})$ be the received vector at the output of the channel. Upon receiving r, the decoder must first determine whether r contains transmission errors. So when r is received, the decoder computes the following (n – k)- tuple:

$$s = r.H^T \quad (2)$$

Which is called the syndrome of r. The s=0 if and only if r is a code word and receiver accepts r as the transmitted code word. The s≠0 if and only if r is not a code word and the presence of errors has been detected. [3]

C. Error correction:

After finding the syndrome s, the coset leader e_j (error pattern) whose syndrome equals $r.H^T$ will be found. Next, the received vector r will be decoded into the code vector v:

$$v = r + e_j \quad (3)$$

III. Neural Network and RBF

A. Introduction to Neural Network

Among the available computational intelligence techniques, the Artificial Neural Networks (ANNs) attempts to mimic the behavior of biological neurons. Among the benefits of ANNs, are the ability to process complex, and interconnected data, arrays, systems' input/output relationships, and nonlinear models, which are notorious for simulation and development. Neural Networks are able to infer, and learn from complex relationships, by generalizing from a limited amount of training data through a process known as training. This concept comes from the fact that animals, and humans are able to learn through observation. Any new situation is training and an experience through which the agent can gain experience that will be used when later is confronted with new, and unpredicted situations. Although, the exact learning mechanism is still unknown, the attempt to mimic the pattern has been successful. Based on scientists' observations, brain consists a large number of interconnected cells called neurons. These cells are known as the critical information-processing units.

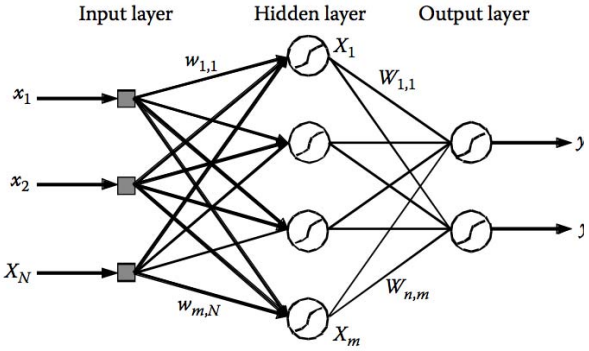

Fig. 1 Neural Network with one Input layer, one hidden and one output layer

Human brain consists of millions of interconnected intelligent agents known as neurons. These neurons will respond to electrical impulses sent from other neurons. In 1943, McCulloch and Pitts developed a simple mathematical model of the neuron that had multiple inputs, and was connected to the output of a neuron via other influencing factors known as weights. Fig. 1 shows the proposed model. It is very interesting to observe that how such a simple model can solve many sophisticated problems.

Later, it was shown that if the perceptrons from different layers were grouped together, also known as multilayer perceptrons. The input layers merely do not perform any computations but distribute the input to the summation through weight factors.

For the neurons in the hidden layer, first the weighted sum of inputs is calculated. Weights play an important role in operation of neural networks. Some inputs are more important than others, and their influence and importance can be highlighted via weights in neural networks.

On the other hand, a nonlinear transfer function, also known as activation function is used so the desired output can be calculated. A sample of such function is shown as below:

$$X_j = f\left(\sum_k W_{jk}x_k\right) \quad (4)$$

ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Transfer functions add the required nonlinearity to the system. Another important feature in ANNs is the ability to learn. Neural networks lack the elements to store data, what they do instead is to utilize the power of weights that shows the importance of each connection. Training of the network means, these weights are selected in such a way that the error between the desire output, and the network output is minimized. There are two steps involved, first is the feed forward calculation of the weights and inputs, and the second is the comparison of the error to plant output. Once the input values to the transfer function were created, the result will be compared to the desired output, the difference, the error then is used to adjust the weights first in the last layer, and then the layer before, etc. This process will continue until the error is minimum.

$$\min E^2 = \frac{1}{2}\sum_n \sum_i \left[y_i(x_n) - Y(x_n) \right]^2 = \frac{1}{2}\sum_n \sum_i \left[y_i(x_n) - f\left(\sum_j w_{ij} . f\left(\sum_k w_{jk}, x_{kn} \right) \right) \right]^2 \quad (5)$$

The gradient descent optimization, and the updated output weights can be found by differentiating the cost function given by equation No. 5. Because these differences are in terms of the other layer outputs, the desired result can be found using the chain rule that the errors are fed backward through the network layer by layer using gradient descent algorithm, and thus this method is called back propagation.

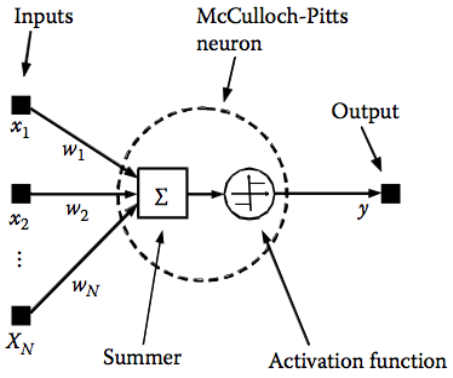

Fig. 2: McCullock-Pitts Neuron Model

During the recent years, the Neural Networks have been the center of attention for researchers, and scientists. New architectures, and learning algorithms are developed all the time. Even though the present neural networks do not achieve human-like performance, they offer interesting means for pattern recognition, including a large collection of very different types of mathematical tools (preprocessing, extraction of features, final recognition). In many cases, it is difficult to say what kind of tool would best fit to a particular problem. Neural networks make it possible to combine these steps, because they are able to extract the features autonomously. They are practical to use, because they are nonparametric. It has also been reported that the accuracy of neural classifiers is better than of traditional counterparts [4][5]. Selection of the proper learning algorithm is vital, because through selecting the right one, it is possible to train those networks that can not be trained with simple algorithms. For example, The error back propagation known as EBP method, is one of the most widely used training algorithms, however, they are more suitable for networks with large number of neurons. On the other hand, the EBP is very efficient in learning, yet to the cost of reduced generalization ability. In other words, the neural network may produce incorrect answers for patterns that were not introduced in the training sets[6][7].

IV. RBF Hamming Code Model

Different techniques have been developed for correction of errors from the received data. Instead of using traditional error correcting techniques, Artificial Neural Networks have been used because of their adaptive learning, self-organization, and real time operation, and to project what will most likely happen on the analogy of the human brain.

Many researchers have contributed much in the field of channel decoding with artificial neural networks (ANN). L. G. Tallini and P. Cull proposed a scheme of using ANN technology to decode Hamming codes and Reed-Muller codes [8]. S.E. El-Khamy used ANN to decode block codes and compared the performance between soft-decision decoding and hard-decision decoding [9]. Due to their parallel processing capability, ANNs are a promising technology for error correction to meet the needs of high data-rate transmission. R. Annauth et al proposed a scheme of using error back propagation (EBP) algorithm to decode Turbo codes [10]. However, they only got rather poor performance results although the decoding complexity was reduced. [11]

Radial Basis Function (RBF) Algorithm for the Artificial Neural Networks has been simulated using Matlab for decoding block codes. The Simulator is trained on all the possible code words to detect/correct the errors.[12]

The model explained here is designed to code, and decode the (15,11) Hamming algorithm, which was presented in section II. For a better result, it is advised that a large number for N is selected. For instance, assume N=10^6 where N is the number of bits which are emitted from the transponder through the channel. The modulation used in this paper is BPSK with the channel noise of Additive White Gaussian Noise (AWGN).

The RBF decoder could be treated as a Single-Input-Single-Output (SISO) model from the viewpoint of code-words, where its input or output is corresponding to 1-codeword information. There are two stages of decoding for Hamming codes with RBF technology, i.e., the first stage of training the RBF network and the second stage of testing, and validating the RBF network. [11]

Training stage:

Based on the principle of Minimum Mean Square Error (MMSE), the known information is used to train the RBF network, and the weights of the NN are accordingly changed during training to obtain the optimal output. The number of samples for training influences the performance of RBF decoder a lot. There should be neither too few nor too many training samples. In the first case of too few samples, the weights of the RBF network would not be correct, which would lead to poor decoding performance. On the other

ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

hand, if too many samples were applied, the performance would not be improved substantially while it would prolong the training time.

Based on default of "nftools" in Matlab codes and our experiments, 70% of data is allocated for training stage Figure 3 illustrates the train data and Regression plot.

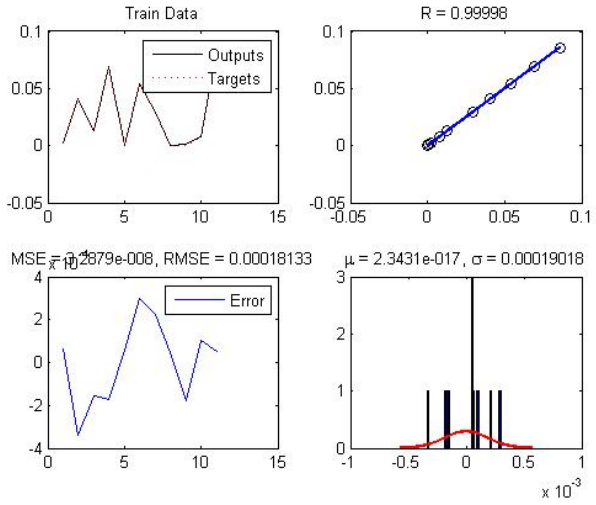

Fig. 3: Train data and Regression

For the test, and validation stage, the time for training and network learning is allocated. Similar to training state, some portions of information should be allocated to testing, and data validation. In this paper, 15% of the information is considered for each of them. Figure 4 shows the testing data, and Regression plot.

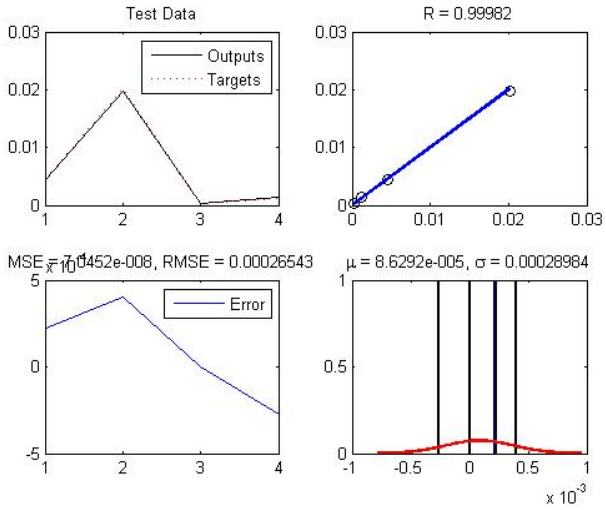

Fig. 4: Test data and Regression

Figure 5, depicts a more comprehensive collection of data sets used to derive the simulation results. For creating, and training the RBF network, the NEWRB Matlab code is employed. According to "newrb" Matlab code, different parameters can be modified to achieve a better result in terms of performance, and error reduction. Table 1, depicts the important parameters that were used in the simulation process.

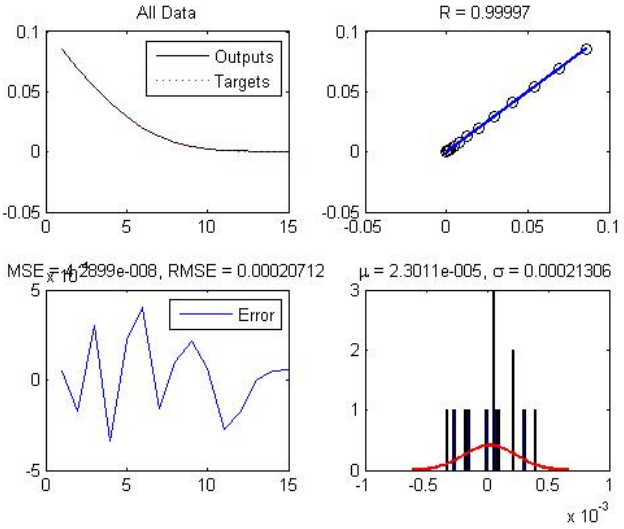

Fig. 5: All data and Regression

Table 1
Parameter used in RBF tool

Parameter	Value
Desired Minimum Error	0
Spread	1
Maximum number of neuron	7
DisplayAt	1
Percentage of training data	70
Percentage of testing data	15
Percentage of validation data	15

Table 2
The value of Mean Square Error (MSE) for using different number of Neurons.

Table 2: MSE for different number of neuron

Number of neuron	MSE
2	6.24836e-005
3	3.03697e-005
4	1.9699e-007
5	1.71009e-007
6	1.43405e-007
7	3.28795e-008

V. Results

In order to simulate the results, a proper training data set was used for a (15,11) Hamming code. This simulation was performed with the presence of an additive white Gaussian noise (AWGN). The whole simulation was carried out with Matlab Neural Network Toolbox 2013. In order to achieve the desired minimum square error, the "newrb" function is employed. This function will increase the number of neurons in the hidden layer of the radial base network (RBN) until the desired MSE is achieved.

After training the RBF network with proper data set, and running the algorithm, the following result was achieved. Fig.6 is known as the Bit Error Rate (BER) curve, which is

ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

an indicative of the channel conversion accuracy in using different SNRs. The simulation consists of four graphs including, theory, symbol error rate, hamming BER, and RBFN Hamming.

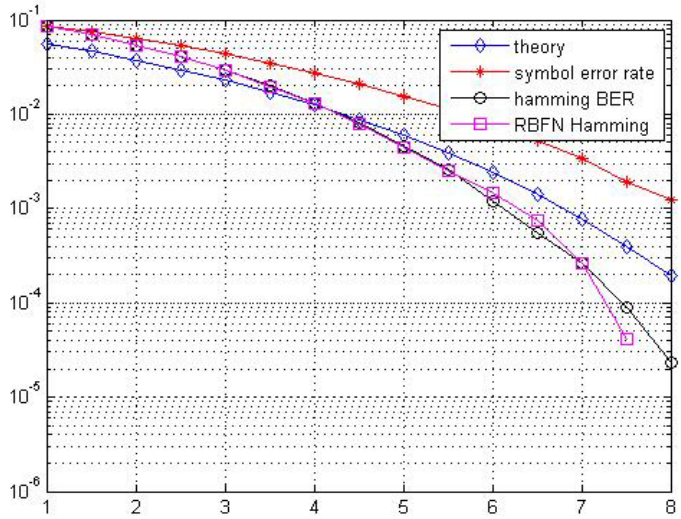

Fig 6: SNR vs BER Gragh for RBFN (15,11) Hamming code

According to simulated results all graphs showed a decreased error rate while increasing SNR. This decrease was not the same for different algorithms. It is shown through numerous papers, and articles that the Hamming code will produce better results than uncoded BER in terms of performance and error reduction. The Hamming code simulation results in this paper, also advocates the same theory.

Increasing new layers, and neurons will add to the complexity of the network but not necessarily improves the results. It is shown that sing optimized neural network algorithm, along with Hamming code could produce much lower BER while maintaining the network simplicity. This feature will make network debugging and troubleshooting much easier than complex networks.

REFERENCES

[1] H. Abdelbaki and E. Gelenbe, "Random Neural Network Decoder for Error Correcting Codes 3 The Random Neural Network."
[2] B. Sklar, Digital communications, vol. 2. Prentice Hall NJ, 2001.
[3] J. Micolau, D. Rodriguez, and J. A. Vidal, "Hamming Block Codes," no. January, 2000.
[4] T. Sorsa, H. H. N. Koivo, and H. Koivisto, "Neural networks in process fault diagnosis," Syst. Man Cybern. IEEE Trans., vol. 21, no. 4, pp. 815–825, 1991.
[5] S. R. Naidu, E. Zafiriou, and T. J. McAvoy, "Use of neural networks for sensor failure detection in a control system," Control Syst. Mag. IEEE, vol. 10, no. 3, pp. 49–55, 1990.
[6] B. WILAMOWSKI, "How Not to Be Frustrated with Neural Networks," eng.auburn.edu, no. December, pp. 56–63, 2009.
[7] B. G. Lipták, Instrument Engineers' Handbook, Volume Two: Process Control and Optimization, vol. 2. CRC press, 2005.
[8] L. G. Tallini and P. Cull, "Neural nets for decoding error-correcting codes," Ital. J. Pure Appl. Math., vol. 10, pp. 91–106, 2001.
[9] S. E. El-Khamy, E. A. Youssef, and H. M. Abdou, "Soft decision decoding of block codes using artificial neural network," Proc. IEEE Symp. Comput. Commun., pp. 234–240, 1995.
[10] R. Annauth and H. C. S. Rughooputh, "Neural network decoding of Turbo codes," Int. Jt. Conf. Neural Networks (IJCNN' 99), vol. 5, pp. 3336–3341, 1999.
[11] X. Liu, Z. Chen, Z. Wang, and P. Cull, "Decoding of Block Turbo Codes with RBF Networks," Int. Conf. Sensing, …, pp. 1986–1990, 2006.
[12] A. Haroon, "Decoding of Error Correcting codes Using Neural Networks," 2012.

ISBN: 978-988-19253-7-4
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)