Random Neural Network Decoder for Error Correcting Codes

Hossam Abdelbaki, Erol Gelenbe
School of Computer Science
University of Central Florida
Orlando, FL 32816
{ahossam, erol}@cs.ucf.edu

Said E. El-Khamy
Department of Electrical Engineering
Alexandria University
Alexandria, Egypt 21544
elkhamy@alex.eun.eg

Abstract

This paper presents a novel Random Neural Network (RNN) based soft decision decoder for block codes. One advantage of the proposed decoder over conventional serial algebraic decoders is that noisy codewords arriving in non-binary form can be corrected without first rounding them to binary form. Another advantage is that the RNN, after being trained, has a simple hardware realization that makes it candidate for implementation as a VLSI chip. This seems to make the neural network decoding inherently more accurate, faster, and more robust than conventional decoding. The proposed decoder is tested on Hamming linear codes and the results are compared with that of the optimum soft decision decoder and the conventional hard decision decoder. Extensive simulations show that the RNN-based decoder reduces the error probability to zero in the range of the error correcting capacity of the used code. On the other hand, it is much better than the hard decision decoder for codewords corrupted with more errors.

1 Introduction

In a digital communication system, significant performance improvement may be obtained if error control coding is properly applied. One type of the error control coding schemes is the forward error control (FEC). In this type, some extra bits are inserted into the symbol stream emitted by the source for purposes of detecting and correcting transmission errors at the receiver. Coding with FEC can be used to lower the signal to noise ratio (SNR) required at the receiver to achieve a given bit error rate (BER). The cost for using FEC coding is the increase in complexity and bandwidth, which may be preferable to the cost of a large final power amplifier or a more sensitive receiver. The two most popular ways of implementing FEC coding are known as block coding and convolutional coding.

Since the decoding process of any code, even in the linear case, can be classified as an NP-Complete problem [1], it is reasonable to consider heuristics including neural networks. Many authors addressed this subject in the literature. For example, in [2] a Hopfield network-based decoder was proposed for binary symmetric channels and in [3, 4] a feed-forward model was used. Hopfield model suffers from its very low storage capacity and the difficulty of choosing between synchronous and asynchronous updating rule, which is difficult to argue in the context of error correcting codes. On the other hand, feed-forward models can produce appropriate mapping between its input and output patterns but it can hardly encode the relation between the elements of each input pattern.

In this paper, we considered the application of the Random Neural Network (RNN) model [5, 6] to the decoding of block coded messages transmitted through Gaussian channel with additive white noise (AWGN channel). The proposed decoder consists of two layers, the input layer or what can be called the association layer and the output layer which acts as a classification layer.

The sections of the paper are organized as follows. Section 2 provides a brief overview of the techniques used for error correction of block codes. It also presents the Hamming codes family which we will use throughout the paper. Section 3 introduces the RNN model. The structure of the proposed RNN based decoder is presented in Section 4. Section 5 presents an experimental evaluation of the proposed decoder as well as the conventional decoders. Finally, Section 6 offers concluding comments and suggestions for further work.
2 Conventional Techniques for Error Correction

In this Section, the basic concepts that describe the forming of block codes are given. Then the techniques used for error correction is reviewed [7, 8] followed by illustrating the specific code family which we use in the simulation.

2.1 Basic Concepts

For the purpose of encoding messages for error protection, the large messages are broken into smaller message blocks consisting of \( k \) bits of information. Redundant bits, normally known as parity bits, are added to these \( k \) bits according to certain rules, and the codewords of length \( n \), inclusive of \((n-k)\) parity bits, are transmitted through the communication channel. Due to the physical nature of channels, quite often they introduce noise and perturbations, spoiling the transmitted data. With \( k \) bits of information per block, there are \( 2^k \) possible distinct messages out of \( 2^n \) codewords that may be generated with \( n \) bits. The set of \( 2^k \) codewords is called a block code. At the receiver, the \( k \) message bits are decoded from the noisy received message blocks using a suitable process. This decoding process can be done in two different ways as will be illustrated in the next subsections.

2.2 Hard Decision Decoding (HDD)

In this technique, a decision is made about each individual bit, the entire codeword is then assembled and this codeword is then compared with each valid codeword. The codeword with minimum Hamming distance to the reconstructed received codeword is then selected. This method for HDD is simple but computationally inefficient. A more efficient technique is to use the parity check matrix to compute the syndrom.

Although the hardware required for HDD is simple, it can only correct \( \frac{d_{\text{min}}-1}{2} \) errors where \( d_{\text{min}} \) is the minimum Hamming distance between the codewords. If the coding scheme can correct \( t \) errors then the probability that an \( n \) bits codeword is in error is the probability that more than \( t \) errors have occurred in \( n \) bits, i.e. \( P_E = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \) where \( p \) is the probability that an individual bit is in error, \( p = \frac{1}{2} \text{erfc} \left( \sqrt{\frac{E_b}{n} \frac{k}{n}} \right) \), \( E_b \) is the energy needed to transmit a bit, which for bipolar mode is \( E_b = 1 \), \( n \) is the noise unilateral spectral density, and \( \text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt \) is the complementary error function.

2.3 Soft Decision Decoding (SDD)

In SDD, the decoding of block codes is carried out by seeking maximum correlation between the incoming corrupted signal and the locally generated codewords. Such a technique does not suggest making decisions in each bit as in HDD but only one final decision. It is complex to be implemented since the received vector is to be correlated with all possible codewords and then the vector yielding the highest correlation (closest codeword to the transmitted one) is selected. When the number of codewords is large, implementing such a receiver is impractical.

2.4 Hamming Code

This is a class of \((n, k)\) single error correcting block codes that has the property \((n, k) = (2^n - 1, 2^{n-1} - r)\) where \( r \) is the number of parity bits. Thus for \( r = 3 \), we have \((7, 4)\) code with the generating matrix \( G \) given by:

\[
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}
\]

Given the four binary information symbols \( b_1, b_2, b_3, \) and \( b_4 \), the corresponding codeword is \( \mathbf{b} = (b_1, b_2, b_3, b_4, b_2 \oplus b_1 \oplus b_3 \oplus b_4, b_1 \oplus b_3 \oplus b_4, b_1 \oplus b_2 \oplus b_4) \). In the \{+1,−1\} representation, this looks like \( \mathbf{x} = (x_1, x_2, x_3, x_4, x_2x_3x_4, x_1x_3x_4, x_1x_2x_4) \) where \( x_j = (-1)^{b_j} - 1 \). The resulting codeword is in systematic form because the information sent appears in the first four positions of each codeword. Since \( d_{\text{min}} = 3 \) for the \((7, 4)\) Hamming code, this code is capable of correcting single error.

3 The Random Neural Network (RNN) Model

In the random neural network model [5] signals in the form of spikes of unit amplitude circulate among the neurons. Positive signals represent excitation and negative signals represent inhibition. Each neuron’s state is a non-negative integer called its potential, which increases when an excitation signal arrives to it, and decreases when an inhibition signal arrives. An excitatory spike is interpreted as a “+1” signal at a receiving neuron, while an inhibitory spike is interpreted as a “−1” signal. A neuron that emits a spike, whether it is an excitation or an inhibition, will lose potential of one unit, going from some state whose value is \( k \) to the state of value \( k - 1 \). The state of the \( n \)-neuron network at time \( t \) is represented by the vector of non-negative integers.
\( k(t) = (k_1(t), \ldots, k_n(t)) \), where \( k_i(t) \) is the potential or integer state of neuron \( i \). We will denote by \( k \) and \( k_i \) arbitrary values of the state vector and of the \( i \)-th neuron’s state. Neuron \( i \) will “fire” (i.e., become excited) if its potential is positive. The spikes will then be sent out at a rate \( r_i \), with independent, identically and exponentially distributed inter-spike intervals. Spikes will go out to some neuron \( j \) with probability \( p_{ij}^+ \), as excitatory signals, or with probability \( p_{ij}^- \) as inhibitory signals. A neuron may also send signals out of the network with probability \( d_i \), and \( d_i + \sum_{j=1}^n \left[ p_{ij}^+ + p_{ij}^- \right] = 1 \). Let \( w_{ij}^+ = r_i \ p_{ij}^+ \), and \( w_{ij}^- = r_i \ p_{ij}^- \). Here the “w’s” play a role similar to that of the synaptic weights in connectionist models, though they specifically represent rates of excitatory and inhibitory spike emission. They are non-negative. Exogenous excitatory and inhibitory signals arrive to neuron \( i \) at rates \( \alpha_i \), \( \lambda_i \), respectively.

This is a “recurrent network” model which may have feedback loops of arbitrary topology. Computations are based on the probability distribution of network state \( p(k,t) = \Pr[k(t) = k] \), or with the marginal probability that neuron \( i \) is excited \( q_i(t) = \Pr[k_i(t) > 0] \).

As a consequence, the time-dependent behavior of the model is described by an infinite system of Chapman-Kolmogorov equations for discrete state-space continuous time Markov systems. Information in this model is carried by the frequency at which spikes travel. Thus, neuron \( j \), if it is excited, will send spikes to neuron \( i \) at a frequency \( w_{ij} = w_{ij}^+ + w_{ij}^- \). These spikes will be emitted at exponentially distributed random intervals. In turn, each neuron behaves as a non-linear frequency demodulator since it transforms the incoming excitatory and inhibitory spike trains’ rates into an “amplitude.” Let \( q_i(t) \) be the probability that neuron \( i \) is excited at time \( t \). The stationary probability distribution associated with the model is given by:

\[
p(k) = \lim_{t \to \infty} p(k,t), \quad q_i = \lim_{t \to \infty} q_i(t), \quad i = 1, \ldots, n. \tag{1}\]

and

\[
\lambda_i^+ = \sum_j q_j w_{ji}^+ + \Lambda_i, \quad \lambda_i^- = \sum_j q_j w_{ji}^- + \lambda_i
\]

\[
q_i = \frac{\lambda_i^+}{r_i + \lambda_i^-} \quad \text{if} \quad q_i < 1. \tag{2}\]

4 RNN Based Decoder (RNND)

The model presented here is designed to decode the (7,4) Hamming code discussed in Section 2. The RNND, as shown in Figure 1, consists of two layers, the input layer, which has 7 neurons, is a fully interconnected layer. This layer is connected to the output layer, which consists of 14 neurons, through feedforward connections. Thanks to the general training algorithm for the recurrent RNN [6], we are able to train the network with such general connections between neurons. The input layer accepts the bits of the received perturbed codeword at the end of the transmission channel and acts as an association layer which, through training, can extracts the relation between the different bits of each codeword and encode this relation into the weight matrix. On the other hand, the output layer acts as a classification layer in which the index of the output neuron that produces the minimum value indicates the index of the decoded codeword. We should note here that the two codewords [0000000] and [1111111] are excluded from the training and testing sets applied to this structure because this showed to produce much better results than if they were included.

On the other hand, another similar structure with 7 inputs and 2 outputs has been successfully trained to decode the all 0′s and all 1′s codewords. In the next subsection we introduce the classification technique and the training set used in designing the proposed RNND.

![Figure 1: Structure of the RNND.](image)

4.1 Classification Technique

Let \( T = \{(x,c)\} \) be a training set of \( N \) vectors, where \( x \in \mathcal{R}^n \) represents a codeword of the block code family under consideration and \( c \in I \) is its class label from an index set \( I \). Let \( y \in \mathcal{R}^m \) be the desired output vector associated with the input vector \( x \). In our application, we relate \( c \) to \( y \) via the following relation: \( c = \{ j : y_j < y_k \quad k = 1, 2, \ldots, m \} \) (\( c \) is the index of the output neuron that produces the minimum value).

The RNND decoder acts as a mapping \( \mathcal{R}^n \to \mathcal{R}^m \), which assigns a vector in \( \mathcal{R}^m \) to each vector in \( \mathcal{R}^n \) or equivalently, it can be considered as a mapping \( C : \mathcal{R}^n \to I \), which assigns a class label in \( I \) to each vector in \( \mathcal{R}^n \). The RNND specifies a partitioning of the codewords into regions \( R_j \equiv \{ x \in \mathcal{R}^n : C(x) = j \} \), where \( \bigcup_j R_j = \mathcal{R}^n \) and \( \bigcap_j R_j = \emptyset \). It also induces a partitioning of the training set into sets \( T_j \subset T \) where \( T_j \equiv \{ (x,c) : x \in R_j, (x,c) \in T \} \). A training pair
(x, c) ∈ T is misclassified if C(x) ≠ c. The performance measure of interest is the empirical error fraction E of the classifier, i.e., the fraction of the training set or testing set which is misclassified:

\[ E = \frac{1}{N} \sum_{(x,c) \in T} \delta(c, C(x)) = \frac{1}{N} \sum_{j \in \mathcal{X}} \sum_{(x,c) \in T_j} \delta(c, j) \]

where \( \delta(c, j) = 1 \) if \( c \neq j \) and 0 otherwise.

4.2 Training Patterns

In the training process, all the codewords of the (7, 4) Hamming code are used excluding the all 0's code and the all 1's code as mentioned before. Thus, we have \( n = 7 \) (inputs), \( N = 14 \) (patterns), \( m = 14 \) (outputs) and \( T = \{1, 2, \ldots, 14\} \). As an example, the codeword \( x_4 = (-1, 1, -1, 1, -1, 1, 1) \) will be associated with the target output \( y_4 = (1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1) \) and hence its class label is \( c = 4 \). The training process is carried out using the RNNSIM v1.0 package\(^1\) (Random Neural Network Simulator) after some slight modifications to incorporate the general recurrent training algorithm into the program.

5 Simulation Results

After training the RNN, the simulation is performed for (7, 4) Hamming code transmitted over AWGN channel. The whole communication system has been simulated with the aid of an interactive JAVA program\(^2\). To evaluate the performance of the proposed RNND and compare it to the HDD and SDD, two different simulations have been carried out.

In the first simulation, the probability of word error is recorded versus the SNR in dB as shown in Figure 2. The SNR is related to the channel variance \( \sigma^2 \) through the relation \( \frac{\sigma^2}{\eta} \). The vertical axis represents logarithmically the probability of word error associated with HDD, SDD, and RNND. The RNND shows coding gain relative to the HDD and its performance approaches that of the optimum SDD.

\(^1\)Free GUI based software available via anonymous ftp at ftp://ftp.mathworks.com/pub/contrib/v5/met/rnnsim
\(^2\)http://www.cs.ucf.edu/~ahossam/rnndecoder/HammRndApp.html

![Figure 2: Probability of word error for (7, 4) Hamming code in an AWGN channel.](image)

In the second simulation, three groups of testing codewords, each of them consists of 100000 randomly chosen (7, 4) Hamming codewords, are applied to the HDD, SDD and RNND. AWGN is added to each testing group so as to produce a single error, two errors, and three errors in the codewords of the first, second, and third testing groups respectively. The noisy testing codewords are then decoded using each of the considered techniques and the number of erroneously detected codewords is recorded for each of the testing groups. The results are shown in Tables 1, 2, and 3.

<table>
<thead>
<tr>
<th>SNR (dB)</th>
<th>HDD</th>
<th>RNND</th>
<th>SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0.071</td>
<td>0.057</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.043</td>
<td>0.036</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.027</td>
<td>0.022</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.006</td>
<td>0.004</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Probability of word error for the first group

<table>
<thead>
<tr>
<th>SNR (dB)</th>
<th>HDD</th>
<th>RNND</th>
<th>SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0.409</td>
<td>0.363</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.314</td>
<td>0.297</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.253</td>
<td>0.201</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.217</td>
<td>0.190</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Probability of word error for the second group

<table>
<thead>
<tr>
<th>SNR (dB)</th>
<th>HDD</th>
<th>RNND</th>
<th>SDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0.774</td>
<td>0.772</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.686</td>
<td>0.648</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Probability of word error for the third group
From the previous tables, it is obvious that the probability of word error for RNND approaches that of the SDD. As an example for the decoding process of the three techniques under consideration, consider the codeword \( x_2 = (-1, -1, 1, -1, 1, -1) \) is to be transmitted through an AWGN channel with \( SNR = 0 dB \). The resulted noisy codeword \( (-1.05, 1.24, 1.04, -2.63, 2.47, -0.88, -1.32) \), which has 2 erroneous bits, is then applied to HDD, SDD and RNND. The corresponding outputs of the decoders are as follows:

- **HDD**: \((-3.60, 6.38, -4.68, 5.90, -3.41, 1.28, 5.37, -5.37, -1.28, 3.41, -5.90, 4.68, -6.38, 3.6)\)
- **SDD**: \((1, 0.52, 1, 0.39, 1, 0.88, 0.62, 1, 0.86, 0.73, 1, 0.65, 1, 0.64)\)
- **RNND**: \((5, 2, 4, 2, 4, 3, 1, 6, 4, 3, 5, 3, 5, 2)\)

From this simulation example, we notice that the HDD produces minimum Hamming distance (1) corresponding to \( x_1 \) thus the transmitted code word is decoded incorrectly. On the other hand, the SDD produces the maximum correlation value (6.38) corresponding to \( x_2 \). The RNND gives the minimum output (0.52) indicating that the decoded code word is \( x_2 \). Thus the RNND and SDD decoded the noisy codeword correctly although it has 2 bits in error. The results of the simulations can be summarized as follows:

- The HDD has the best performance in the case of single error. However, it is almost ineffectual for correcting two or three errors. This is because the \((7, 4)\) Hamming code is single error correcting code.
- The optimum SDD performs well for decoding the words containing one and two errors and has a moderate performance for three errors. This is of course on the account of high implementation complexity.
- As for the RNND, its performance is very close to that of the SDD. It should be noted here that the proposed RNND may have a very simple hardware implementation.

### 6 Conclusions

This paper presents a recurrent random neural network based error correcting decoder for linear block codes. The proposed decoder can reduce the error probability to zero in the range of the error correcting capacity of the used code. It has been shown through simulation that the neural based decoder is much better than the hard decision decoder for decoding codewords corrupted with errors more than the correction capability of the used code. Although the optimum soft decision decoding technique has the minimum probability of error, the proposed model can be considered as a near optimum decoder. The neural based model has a simple hardware realization that eliminates the need of the multiplication of the received vector by the parity check matrix or seeking the maximum correlation between the incoming noisy signals and the locally generated codewords. With sufficient noise the neural network or the soft decision decoder will not produce the correct decoding, however, it is shown that for higher signal to noise ratios, the RNND is unlikely to produce an error. One drawback of the proposed model is that the number of output neurons grows exponentially with the number of codewords, which is not a practical solution for long codes. In future, we can consider using the RNN model as a dynamical system in which the codewords are the constant attractors of the system.

### References


