Error Correcting Neural Networks for Channels with Gaussian Noise

I. ORTUÑO[1], M. ORTUÑO[2] AND J.A. DELGADO[3]

[1] *Departament d'Informàtica, Universitat Autònoma de Barcelona 08193 Barcelona, Spain.*

[2] *Departamento de Física, Universidad de Murcia, 30.071 Murcia, Spain.*

[3] *Departament de Teoria del senyal i Comunicació Universitat Politècnica de Catalunya, 08034 Barcelona, Spain.*

Abstract

The adaptability and parallel computing capabilities of neural networks make them specially adequate for error correcting tasks. Feed forward neural networks for soft–decision decoding of block codes in channels with additive white gaussian noise are presented. When the noise is not white, we deduce the optimal set of weights for the connections of the network. These weights are also approximately obtain by an error back propagation algorithm. A practical realization for a BCH (7,4) code is presented, and exhaustive numerical simulations are performed on it.

1. Introduction

A very promising field of application of neural networks is the area of error correction in digital communications, where many parallel calculations have to be performed and high computational speeds are required [1]. At the same time, the flexibility and learning capabilities of neural networks allows them to efficiently operate in real complex situations, where some of the simplifying assumptions of standard decoding techniques are not fulfilled.

Neural network have already been used as decoders of block codes [2,3]. Their adaptability has been employed to correct an imperfect functioning of the syncronization mechanism of the demodulator [4].

In the next section, we will review the main features of error correcting codes and standard decoding techniques. In the third section, we will describe how to construct a neural network for decoding in additive white gaussian channels and show its performance for the BCH (7,4) code. In the fourth section, we will explain how to modify the network to operate in a correlated gaussian channel and present its error bit probability as a function of the correlations.

2. Background on error correcting systems

The aim of error-correcting digital systems is to reliably transmit information over a channel, which is always corrupted by noise. At the origin, an encoder introduces a certain amount of redundancy which makes possible to retrieve the original information up to a given degree of distortion of the message, that depends on the code used.

In a binary (n, k) block code, n bits are sent for each k bits of information. There are 2^k possible messages per block, each of them corresponding to a specific combination of n bits, called a codeword. This correspondence depends on the particular code used, but not on previous messages.

Neural networks can be applied to basically all types of codes. Here we will use them with the BCH (7,4) code [5]. With this code, an information message of 4 bits (a_1, a_2, a_3, a_4) is sent as the following 7 bits sequence $(a_1, a_2, a_1 \oplus a_3, a_1 \oplus a_2 \oplus a_4, a_2 \oplus a_3, a_3 \oplus a_4, a_4)$, where $\oplus$ indicates modulo-2 addition. There are $2^k = 16$ codewords.

For the sake of simplicity, we assume a binary source that is sending a codified set of signals $s_i^{(j)}$ taking values $+\frac{1}{2}$ and $-\frac{1}{2}$ (corresponding to 0 and 1). The index j denotes which of the possible 2^k codewords has been sent, and the index i refers to the position of the bit within a word. For the BCH (7,4) code, j will run from 1 to 16, and i from 1 to 7. We consider that the distortion in the channel can be adequately simulated by an additive gaussian noise n_i, with zero mean and variance σ^2. The demodulator, in the so-called soft-decision techniques, passes to the decoder the unquantized variables $z_i = s_i^{(j)} + n_i$.

If the gaussian noise is white, the different n_i are uncorrelated. Nonetheless, the different bits of each block are implicitly correlated through the codification process. One has to decide block by block which message is more likely to have been sent.

The probability that the j-th codeword was emitted, given the received set of signals z_i, is proportional to the product [5]:

$$\prod_{i=1}^{n} P(z_i|s_i^{(j)}) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp\left[-\sum_{i=1}^{n} \frac{(z_i - s_i^{(j)})^2}{2\sigma^2}\right] \tag{1}$$

This is true provided that all the *a priori* probabilities that a codeword is sent are the same.

Instead of maximizing the previous product, we can maximize the exponent. Taking into account that $|s_i^{(j)}|^2$ $(= \frac{1}{4})$ and $|n_i|^2$ are independent of j, we conclude that the most likely j is the one that maximizes the function:

$$L_j = \sum_{i=1}^{n} z_i \, s_i^{(j)} \tag{2}$$

The decoding technique based on this maximization procedure is called *maximum likelihood decoding* (MLD). In the next section, we will see how to implement this procedure with neural networks.

3. Neural networks as decoders

The parallel computing capabilities of neural networks makes them specially suitable for decoding. If the length of the message k is not too long, the neural network can directly perform the calculations involved in MLD, equation (2).

In figure 1 we show a feed forward neural network that implements the MLD algorithm for the BCH (7,4) code. The seven (n, in general) z_i values of each word are fed into the seven input nodes of the network. There are 2^k ($2^4 = 16$, in our case) intermediate neurons, each of them corresponding to a codeword. The final 4 neurons represent the k information bits of the block considered.

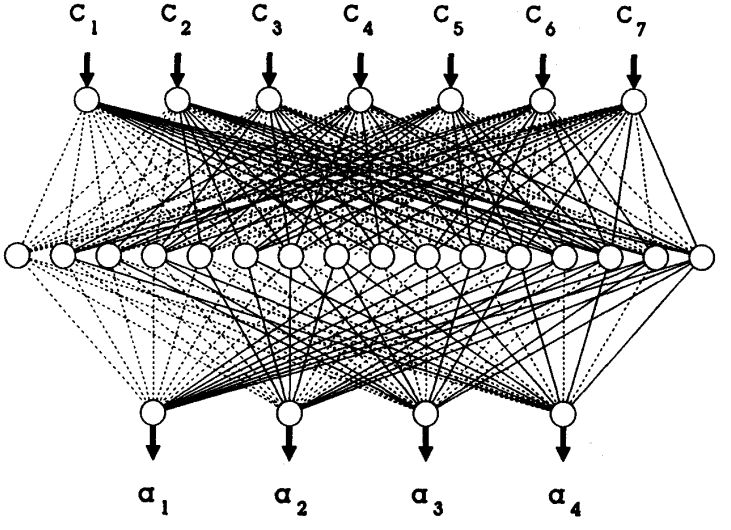

Figure 1: Error correcting neural network for the BCH (7,4) code

The idea is that if there is no error in the received message, or if this is small enough to be corrected by the code, the intermediate neuron corresponding to the right j will be the only activated one in the intermediate layer. This neuron will in turn activates the right bits at the output.

The weights of the connections between the initial and the intermediate layers have to be chosen in such a way that the input values of the neurons in the intermediate layer are proportional to the L_j's, given by equation (2). As the $s_i^{(j)}$ are equal to either $+\frac{1}{2}$ or $-\frac{1}{2}$, we choose:

$$w_{i,j} = 2\, s_i^{(j)} \tag{3}$$

Namely, the weight between the i input neuron and the j intermediate one is either $+1$ or -1, according to whether the i bit of the j codeword is 1 or 0. In figure 1 we represent the weights $+1$ with solid lines and the -1 with dashed lines. The 5th codeword, for example, is equal to 0101100.

If the set of intermediates neurons plays a winner-takes-all strategy, the network is strictly equivalent to an MLD device. Thus, the network decodifies as best as possible, for white gaussian noise. In figure 2 we show (solid line) the bit error probability, P_b, as

a function of the signal to noise level, E/N, measured in decibels. This level is equal to $N = -10\log(8\sigma^2)$, for the signals used.

We have also used 'ordinary' neurons, for the intermediate layer, with a *sigmoidal* response function [7]

$$h(z) = \frac{1}{1+e^{-\alpha z}} \tag{4}$$

with exponent $\alpha = 4$. We adjust the values of the thresholds Θ with an error back propagation learning algorithm, obtaining $\Theta = 5.3$. The performance of this network is quite remarkable. The results achieved are basically indistinguishable from MLD, in the scale of figure 2. For example, for a signal to noise level of 3 dB, we obtain a bit error probability of 0.00086 with MLD and of 0.00087 with this neural network.

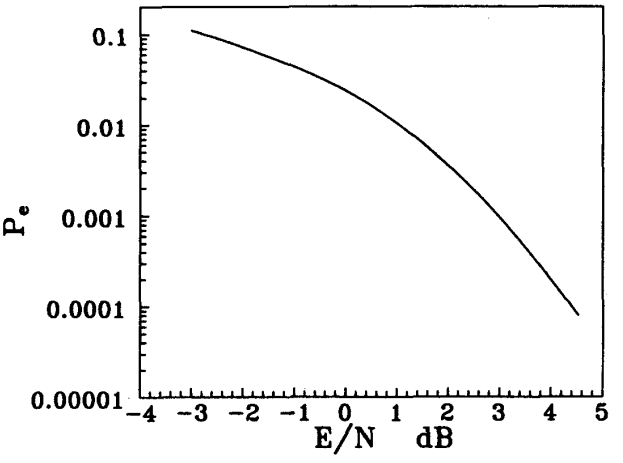

Figure 2: Bit error probability as a function of the signal to noise level

The topology of the network and the values of the weights are given by the structure of the code employed [8]. No learning is required for decoding under the same simplifying assumptions as standard error correcting techniques. Although for real channels, with other types of noise or intersymbol interference, for example, we can improve the performance of the network through learning.

4. Gaussian noise with memory

The use of neural networks for error correction is really advantageous in the case of complex situations, where some of the simplifying assumptions of standard decoding techniques are not fulfilled. In these situations, it is often difficult, or even unfeasible, to analyze and/or to implement the modifications needed to handle the mentioned complexities. However, neural networks can adjust by themselves to such situations due to their flexibility and learning capabilities.

In order to show this, we have applied our previous neural network to a cannel with additive gaussian noise with a non-uniform power spectrum. In this case, the values of the noise in different time intervals, n_i, are correlated random variables. Their joint probability distribution function is the *jointly gaussian* distribution. Assuming correlations between

nearest neighbor time intervals only, this is proportional to [6]

$$\exp\left[-\frac{n_1^2 - 2\rho n_1 n_2 + (1+\rho^2)n_2^2 - \cdots + (1+\rho^2)n_{N-1}^2 - 2\rho n_{N-1} n_N + n_N^2}{2\sigma^2(1-\rho^2)}\right] \quad (5)$$

where N is the total number of bits sent, σ^2 is the variance of the n_i, and ρ is the covariance between successive signals.

We will neglect correlations between the noise corresponding to different blocks. Expression (5) will refer then to each block separately and N will be substituted by n, the length of the block. We have to substitute $n_i = z_i - s_i^{(j)}$ in equation (5) and maximize with respect to j. As in the previous section, we maximize the exponent of equation (5), instead of the whole expression, and through away the terms independent of j. The final quantity to be maximize is:

$$s_1^2 - 2z_1 s_1 + \sum_{i=2}^{n-1}(1+\rho^2)(s_i^2 - 2z_i s_i) + s_n^2 - 2z_n s_n - 2\rho\sum_{i=1}^{n-1}(s_i s_{i+1} - z_i s_{i+1} - z_{i+1} s_i) \quad (6)$$

To implement the previous maximization procedure, we change the weights of the connections between the initial and the intermediate layers according to equation (6): the weight between input neuron i and intermediate neuron j is the factor multiplying z_i in this equation, which is a function of j. An extra initial neuron (with a fixed unitary input) is introduced to take into account the terms not containing any z_i.

We simulate a channel numerically, adding to the sequence of bits sent a correlated gaussian random variable. The correlation coefficient ρ is varied between 0 and 0.7, keeping σ^2 fixed. In figure 3 we plot the error bit probability versus ρ (for $\sigma = 0.25$) for both the original network, with no modifications to take into account correlations, (upper curve) and the new network with weights according to equation (6) (lower curve). The runs are over 1000000 information bits.

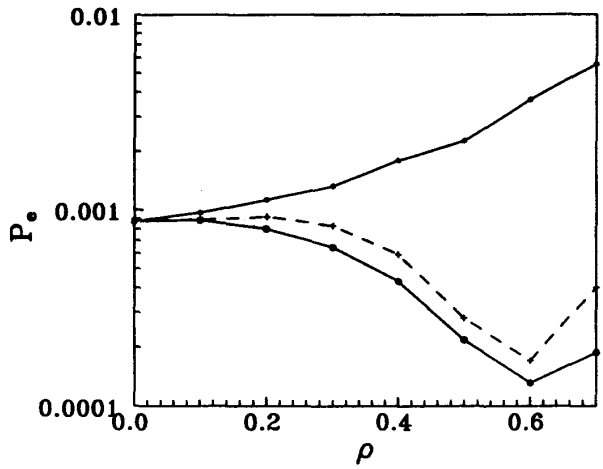

Figure 3: Bit error probability versus the noise correlation coefficient

We can notice the drastic deterioration of the error correcting capabilities of standard decoding techniques (equivalent to our original network) in the presence of noise correlations and the huge improvements obtained with our modified network. The results of the

latter are even better than in the absence of correlations. This is not contradictory since we have some extra knowledge of the noise in the presence of correlations.

We have also used an standard error back propagation learning algorithm to adjust the network to the presence of noise correlations. Starting from the original network, we change the weights of the connections between the input and intermediate layers. The network learns for 10000000 information bits and then its performance is evaluated with a new sequence of 1000000 bits. The results for the bit error probability are shown in figure 3 (dashed line).

In the learning process, the network steadily improves its performance up to the values shown in the dashed curve of the figure. They are reached after the system have learnt for about 5000000 bits. The performance oscillates with further learning and the ideal values of the error probability are never reached. Nevertheless, the difference between the ideal values and those obtained with the learning process is fairly small.

5. Conclusions

Neural networks are very suitable for the field of error correction in digital transmission systems, due to their high computational speeds. Their basic topology and weights, for standard decoding, are given by the structure of the code employed. Their learning capability convert them in the natural choice for decoding in complex situations, such as incoherent demodulation, correlated noise, intersymbol interference, etc...

Acknowledgments

This work was partially supported by the Spanish project DGICYT/UAB, number 113118.

References

[1] E.C. Posner, "Neural Networks in Communication", *IEEE Trans. on Neural Networks*, vol. 1, no. 1, 1990.

[2] Y. Takefuji, P. Hollis, Y. P. Foo and Y. B. Cho, "Error Correcting System Based on Neural Circuits", *Proceedings of International Neural Symposium*, 1987.

[3] M. D. Alston, P. M. Chau, "A Decoder for Block-Coded Forward Error Correcting Systems", *Proc. IJCNN*, Washington, DC, II-302, 1990.

[4] J. F. Yang, C. M. Chen and J. Y. Lee, "Neural Networks for Maximum Likelihood Error Correcting Systems", *Proc. IJCNN*, Washington, DC, I-493, 1990.

[5] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error correcting Codes*, North-Holland, 2nd reprint, 1983.

[6] S. Benedetto, E. Bigliari, and V. Castellani, *Digital Transmission Theory*, Prentice–Hall, Inc. Englewood Cliffs, New Jersey, 1988.

[7] B. Muller and J. Reinhardt, *Neural Networks*, Springer-Verlag, 1990.

[8] J. Bruck and M. Blaum, "Neural Networks, Error-Correcting Codes, and Polynomials over the Binary n-Cube", *IEEE Trans. Inform. Theory*, vol. 35, no. 5, pp. 976-987, 1989.