ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

A New Error-Correcting Syndrome Decoder with *Retransmit* Signal Implemented with an *Hardlimit* Neural Network

José Barahona da Fonseca

Department of Electrical Engineering and Computer Science
Faculty of Sciences and Technology
New University of Lisbon
Monte de Caparica, 2829-516 Caparica - Portugal

Abstract. Still today the problem of counting the errors of a noisy received word is an open problem in literature. This means that when we use an error correcting code we cannot control if the number of errors of the received noisy word is greater than the error correction capability of the code of k errors, k=(d-1)/2, where d is the minimum Hamming distance of the code. The main advantage of our proposal results from the introduction of the *Retransmit* signal when the syndrome decoder detects an *ambiguity situation* and cannot correct the noisy word. These *ambiguity situations* occur when happens one more error than the error correction capability of the error correcting code. This property of the error correcting syndrome scheme allows increasing the error correction capability of an error correcting code by one error at a little increment of bandwidth or delay in the transmission. Although there are some proposals of implementation of error-correcting decoders with neural networks in literature our work is completely different in what concerns three main aspects. First we propose the implementation of the *retransmit* signal based on the detection of ambiguity of the minimum Hamming distance between the received word and each of the codewords, i.e. when there are more than one codeword at the minimum Hamming distance to the too noisy received word. Second we use a constructive approach that does not need training. And finally we use hardlimit neurons that can be implemented in hardware by a single transistor in a high gain setup. We begin with two exhaustive simulation experiments where we introduced *all manners* of occurrence of two errors in all codewords of two codes with minimum Hamming distances 3 and 4, respectively, which only guarantee all possible *one* error *good corrections*, to show how the ambiguities arise in the decoding process. Next we present the building blocks of the error correcting decoder based in hardlimit multilayered perceptrons and then we assembled all them out and show an example for an error correcting decoder for a four codewords error correcting code. Finally we discuss the advantages of our proposal and the consequences of the introduction of the *Retransmit* signal and define possible ways of evolution of our work.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

1 Introduction

Error correcting decoder design is a very active area of research and during 2006 there appeared 26 papers in the IEEE *Xplore* database related with this topic. The use of neural computation in the implementation of error correction decoders permits the *parallel calculation* of the Hamming distance between the received word and each of the words of the code which represents a great speedup for big codes. Nevertheless since 1989 only few experiments of application of neural network to the design of error correcting decoders appeared in the literature, see e.g. [1-2] but *nobody applied multilayered hardlimit perceptrons without training* like our proposal and also nobody implemented the *Retransmit* signal, also known in the literature by the *ARQ* (Automatic Retransmission Request) *scheme*. It seems our work is the first proposal of implementation of error correcting decoders with multilayered hardlimit perceptrons without training and with the implementation of *ARQ* that appeared in the literature. In literature it is normally assumed that the *ARQ feedback signal is error free and zero delay* so it is enough to use one single bit to implement the *Retransmit* signal. We will also assume this simplification. Although for d=2 e - 1, e being the maximum number of errors in the transmission, we found some *Detection Errors*, *detections where the decoded word did not correspond to the original sent word*, for a number of errors of transmission e'=(d-1)/2 + 1, for d=2 e we only found 90 detection ambiguities and 15 good detections and no *detection errors*. By detection ambiguity we mean that the noisy received word is at the same minimum Hamming distance to more than one original codeword. Based on these latter properties we can correct e errors with a code with a lower minimum Hamming distance d=2 e (the Hamming theorem imposes d=2 e + 1), generating a *Retransmit* signal back to the sender. We show that this has great advantages in terms of bandwidth and transmission speed for the cases where the probability of exactly e errors is small. The introduction of the retransmit signal or the *ARQ* scheme can be a great advance in Code Design Theory and Error Correction Algorithms. Surprisingly our empirical studies show a performance increase of our retransmission scheme with d, i.e. when d=2 e increases, the number of ambiguities that arise for e errors decrease in percentage. In section 2 we present the simulation results with two simple error correcting codes to show how the ambiguity in the decoding arises when there happens more errors than the error correcting capability of the code and how to generate the *Retransmit* signal in this situation. In section 3 we present the building blocks of our error correcting decoder implemented with simple hardlimit multilayered perceptrons and in section 4 we assembled them to show how the decoder works and discuss the advantages of our proposal. Finally in section 5 we present the conclusions and possible ways of evolution of our work.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

2 Preliminary simulation experiments

The Hamming theorem guarantees than when the code Hamming distance, *d*, and the number of errors of the received word, *e,* are such that $d \geq 2e+1$ then the error correcting code will be able to correct all *e* errors, i.e. detect correctly the received word with *e* errors. We will show empirically that when $d=2e$ we will have no *bad detections* and only a small percentage of *ambiguities* (the received noisy word being at the same Hamming distance from two or more codewords) and we will show that this percentage reduces with the augment of the code length. This latter property of existence of a small percentage of ambiguities will permit us to implement the *RETRANSMIT* signal with a single hardlimit neuron that identifies this situation as it is described in section 5 and requests the sender to retransmit the too noisy word. In the following table 1 we show an error correcting code with length 6 and *d*=3 and all manners of introducing two errors in each codeword. The cases where there exists ambiguity in the decoding, i.e. there is more than one codeword at a minimum Hamming distance from the received noisy word, are presented in boldface. Since the first code has a minimum Hamming distance 3 and the second 4 (not shown), in both cases the Hamming Theorem only guarantees a *good* correction of *all* possible occurrence of 1 error in the received noisy word. The second code was an optimal code with A(6,4)=4 words and we got 60 ambiguities and no good corrections neither bad detections.

1 0 1 0 1 0	
1 0 0 0 0 1	0 0 0 1 1 1
0 1 1 0 0 1	1 1 1 1 1 1
0 1 1 0 1 0 d=**2 5 2 4 3 A**	1 1 1 0 1 1 d=**2 3 2 4 1 ERR**
0 0 0 0 1 0 d=**2 3 4 2 5 A**	1 0 0 1 1 0 d=**2 3 6 2 3 A**
0 0 1 1 1 0 d=**2 5 4 2 3 A**	1 0 0 0 0 0 d=**2 1 4 4 5 ERR**
0 0 1 0 0 0 d=**2 3 2 4 5 A**	1 0 0 0 1 1 d=**2 1 4 2 3 ERR**
0 0 1 0 1 1 d=**2 3 2 2 3 A**	1 0 1 1 0 0 d=2 3 4 4 3 **OK**
1 1 0 0 1 0 d=2 3 4 4 3 **OK**	1 0 1 1 1 1 d=2 3 4 2 1 **OK**
1 1 1 1 1 0 d=**2 5 4 4 1 ERR**	1 0 1 0 0 1 d=**2 1 2 4 3 ERR**
1 1 1 0 0 0 d=**2 3 2 6 3 A**	

Table 1: Simulation results for a binary 5 words code of length 6 and minimum Hamming distance 3, so with an error correcting capability of 1 error (to correct 2 errors we would need a minimum Hamming distance greater or equal to 5) for *two* introduced errors in the received first word of the code. In the first five lines we present the error correcting code. It is shown that for two errors in the transmission it arises a lot of Bad Detections identified as *ERR* for the transmission of the first word. The ambiguities are identified as *A*.

Increasing the length of the code to 33, *d*=16 and *k*=8 errors and 10 binary words we got 138,404,001 good detections, 437,559 ambiguities and no bad detections. The correctness of the sum of the number of good corrections with the number of ambiguities can be confirmed by the expression $138{,}404{,}001+437{,}559=10\binom{33}{8}$ which is the number of all manners of introducing 8 errors in 10 binary codewords with 33 bits. Note that the 437559 ambiguities are about 1/316 of the number of good corrections. In the situation of ambiguities we use the retransmit signal which would not increase significantly the bandwidth since there are very few ambiguities.

3 Advantages of using the *Retransmit signal* over code length increase

From literature, as a rule of thumb to correct e errors we will need an error correcting code with a length $n \geq 5\ e$ [3] that will guarantee that we will design a code with a minimum Hamming distance $d \geq 2\ e + 1$. So to increase the error correction capability of an error correction code by 1 we must increase its length by at least 5 characters. With the introduction of the *Retransmit* signal we can increase the error correction capability of a code by one without increasing its length but increasing only marginally the bandwidth or the average transmission time. Obviously this latter increment in the bandwidth is proportional to the average number of errors or, by other words, inversely proportional to the signal to noise ratio, *S*/*N*. So with our approach we save a lot of bandwidth at the expense of a little increase in power consumption of the sender to increase *S/N* and so reduce the probability of occurrence of *ambiguity situations*. In this sense our methodology *changes bandwidth by sender power*.

4 Design of the building blocks

In this section we will design the main building blocks of our error correcting decoder with retransmit signal based on hardlimit multilayered perceptrons. We assume that each different character of the received word is translated into a different analogical value such that they can be processed directly by a neural network. To compute the Hamming distance between two words we must have a logical function that is 1 when two characters of each word are different. For binary words that logical function is the *XOR*. It is simple and straightforward to implement a XOR with a two layer hardlimit neurons network. To compute the Hamming distance between to q-ary words we need to implement a generalized *XOR* function that admits as inputs characters different from 0 and 1. The main idea behind the implementation of the *generalized XOR* with hardlimit perceptrons described in figure 2 is that *equality* may be defined as

$$(A \geq B) \text{ AND } (A \leq B) \equiv (A = B) \tag{1}$$

Then we only need one more neuron to *negate* the equality function and thus obtain the function $A \neq B$. In the first layer the two hardlimit neurons implement the two comparisons, in the second layer it is computed the logical *AND* function of these two comparisons and the last hardlimit neuron implements the logical *NOT* function. In the first layer the null biases are omitted.

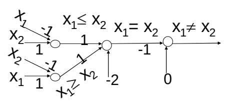

Fig. 1: Hardlimit multilayer perceptron that computes the *generalized XOR* function. Note that the last hardlimit neuron just computes the logical NOT function since *step*(0)=1 and *step*(-1)=0. In the first layer the omitted biases are 0.

Since the Hamming distance between two binary length *L* words *A* and *B* is given by

$$d\_h\_2\_words(A,B) = \sum_{i=1}^{L} XOR(a_i, b_i) \qquad (2)$$

we must add a linear neuron in the last layer of the neural network that computes the Hamming distance between the received noisy word and each one of the words of the code. For *q*-ary codes we must substitute the logical *XOR* function in (2) by the *generalized XOR*. Our approach is a constructive methodology that does not need learning or training. It is based on the idea that if x_i is the maximum of a set of *N* elements then the following logical expression is *true*

$$x_i \;\; is\, Max \equiv \prod_{j=1,\;\; j\neq i}^{N} \left(x_i \geq x_j \right) \qquad (3)$$

5 Assembling the decoder building blocks

In figure 2 we assemble the basic building blocks implemented by hardlimit multilayer perceptrons to implement an error correcting decoder with *Retransmit* signal for a 4 words code with length 6. It is assumed that when the *Sender* receives the *Retransmit* signal with value 0 it is interpreted as the '*the correction is Ok*' and can send the next word. The last hardlimit neuron detects if there exist an *ambiguity*, comparing the number of word detections to 2, generating the *Retransmit* signal when there are more than one input with the value 1, i.e. when there is an ambiguity situation.

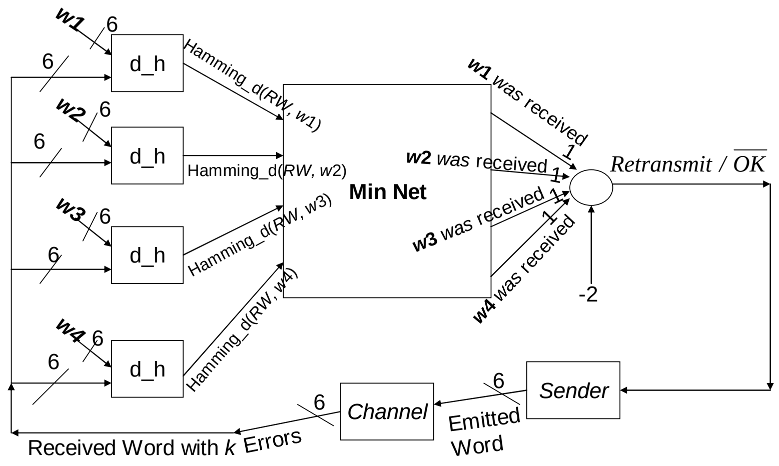

Fig. 2: Hardlimit multilayer perceptron that implements the error correcting decoder for a four words error correcting code.

6 Conclusions and future work

We showed that our error correcting decoder implemented by a hardlimit multilayered perceptron is simpler and more efficient than the previous proposals published in the literature based on neural networks and presents advantages over the classical error correcting decoders especially in terms of the time to decode the received word. It also has the advantage of increasing the error capability of the error correcting code by one error at the expense of a little increase in power consumption. We showed that our methodology has advantages over increasing the code length. In the near future we are planning to build a prototype of our error correcting decoder for a very big error correcting code based on FPGAs due to the simplicity to programme and reprogramme them. Next we plan to develop a VLSI prototype for an even bigger error correcting code.

References

[1] H. Abdelbaki, E. Gelenbe, and S. E. El-Khamy, S.E., Random neural network decoder for error correcting codes. In proceedings of the *10th. international joint conference on neural networks* (IJCNN 1999), pages 3241-3245, IEEE Press, 1999.

[2] Y. Jing and C. S. Chen, Neural net decoders for some block codes, *IEE Proceedings on Communications, Speech and Vision*, 137:309 – 314, IEE Press, 1990.

[3] M. K. Kaikkonen, A new four-error-correcting code of length 20, *IEEE Transactions on Information Theory*, 35: 1344-1345, IEEE Press, 1989.