Proceedings of 1993 International Joint Conference on Neural Networks

A Probabilistic Neural Network for Designing Good Codes

G.Phanendra Babu, and M.Narasimha Murty
Department of Computer Science and Automation
Indian Institute of Science, Bangalore - 560012,(India)

April 21 1993

ABSTRACT

Designing good error-correcting codes typically requires searching in search spaces. The vastness of search space precludes the use of brute force techniques such as exhaustive enumeration. The problem of designing codes so that each code repels others(in the sense of hamming distance) fits well in the framework of neural networks. Formulating an energy function to design codes is very difficult and cannot satisfactorily be solved by Hopfield neural network model. To alleviate these problems, a probabilistic neural network model is proposed. The usefulness of the proposed model is investigated with respect to maximal distance codes and constant weight codes. Results of some code parameters that have been designed using the proposed model are presented.

1 INTRODUCTION

Deterministic algorithms to solve some of the problems that are combinatorial in nature consume a large amount of time. But many problems require optimal solution in a reasonable amount of time. Several probabilistic methods such as Simulated annealing[1] and Genetic algorithms[6] have been proposed to find (near) optimal solutions.

In the recent past, neural network methodologies gained prominence by solving a variety of problems spanning different areas. Neural network models operate using localized computation leading to a globally acceptable solution. As these models are inherently parallel, these are quite effectively utilized to solve combinatorial problems. Hopfield network[5] is the first of its kind that has been applied to Combinatorial Optimization(CO) problems. As Hopfield network converges to local minimum, stochastic models such as Boltzman Machine[2], Stochastic Mean Field Annealing [3] have been proposed to find global optimal solution. These stochastic models find optimal solution asymptotically. These models require to formulate objective function in the form of an energy function that is minimized or maximized by co-operative parallel operations of neurons.

Some problems such as design of codes, discussed in next section, cannot be fit within the framework of Hopfield network. New models have to be designed to study this type of problems.

In this paper, we propose a probabilistic neural network model to generate good communication codes. Both Maximal Distance Codes(MDC) and Constant Weight Codes(CWC) are considered in this study. This paper is organized as follows. Section 2 discusses about coding problem and the complexity involved in finding good codes. A probabilistic neural network model is presented in Section 3. Experimental study and results are presented in Section 4.

2 COMMUNICATION CODES

Shannon's paper[9] on information theory has a tremendous impact on coding theory. Coding theory is mostly used in 1) Digital Data Transmission, and 2) Digital Data Storage. Former deals with the transmission of data via communication channel, whereas the later deals with the storage aspects of the digital data. Coding theory helps in fast and reliable data transmission. Data may get corrupted during transmission due to noise, distortion, and interference. Coding theory deals with the design aspect of codes such that the corrupted data can be detected/corrected at the receiver site. Communication mechanism is shown in Fig. 1.

Data from the source is divided into data segments each of size k, i.e., a sequence of k characters. The k character sequence is fed to encoder which in turn encodes it into an $l(l > k)$ character sequence,

called code word, and sends to modulator. Modulator converts this character sequence into a signal and transmits it via channel. The reverse process is undertaken at receiver site. Out of 2^l possible code words, only 2^k will be taken and are associated with 2^k input sequences. The redundant characters, $l-k$ are used to detect/correct errors at the receiver site. The number of errors that can be corrected for a code with minimum hamming distance, d, are $\lfloor (d-1)/2 \rfloor$. Coding theory helps in designing error-correcting codes such that the maximum number of errors can be corrected with minimum communication bandwidth.

Many coding techniques have been developed. Some of them include, Linear block codes, Cyclic codes, Reed-Solomon codes, BCH codes, and Convolution codes[8]. In this paper, we confine our discussion to two linear block codes namely maximal distance codes, and constant weight codes. Before we briefly explain about these two, we discuss some of the parameters used in coding theory. The size of the code is the number of code words in the code C. The number of different characters that form an alphabet set is called the arity of the code. We use arity 2, i.e., binary code words. The hamming distance, d_H between two code words is the total number of positions at which two code words differ. The weight of a code word, w is the number of non-zero bits in it.

Maximal Distance Code: Maximal distance code, $M(l, N, d)$ is a set of N code words, each of length l and d is the minimum of hamming distances between all pairs of different code words in the code, i.e., $d = min_{i,j,i \neq j}\{d_H(c_i, c_j)\}$ where $c_i, c_j \in C = \{c_1, \ldots, c_N\}$.

It is very difficult to find a code for the given values of l, N, and d. The lower and upper bounds of N are available in the literature[8] for the given values of l and d. The number of codes to be checked in order to find good codes is $\binom{2^l}{N}$. For simple M(7,16,3) code, approximately 10^{20} possible combinations have to be checked to find MDC. This clearly indicates the fact that exhaustive enumeration to find good code is not feasible.

Constant Weight Code: This code, $A(l, N, w, d)$ is a set of N code words and weight of each code word is equal to w. Other parameters are same as above. Mathematical analysis provides upper and lower bounds of N for some values of l and d. Readers are referred to a paper by Conway and Sloane[10] for details.

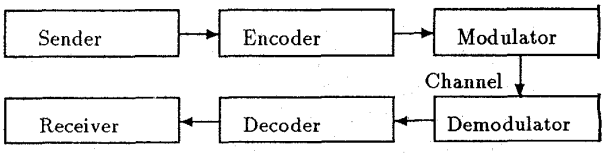

Fig. 1. Communication Mechanism.

The problem of finding good codes has been attempted by using Simulated Annealing[7] and Genetic Algorithms[11]. These techniques solve the problem by formulating it as an optimization problem. In this problem, each code word repels all others by trying to maximize the mutual hamming distance. Such problem fits very well within the neural network framework. In the next section, we present a neural network model in which each node represents a code word.

3 A Probabilistic Neural Network

Coding problem, i.e., generating good codes, cannot be formulated as a quadratic energy function, that is required for Hopfield network type models[5,2]. Subsequently it cannot be solved by using those models. A specialized neural network design is necessary to solve it effectively. We discuss about the network architecture followed by network dynamics. In this model, each node is interconnected to all other nodes and each node represents a code word. The output of neuron is a code word vector. This model assumes that each node possesses some memory and processing capability. Each neuron operates on its own using the outputs of other neurons and tries to repel others, maximizing mutual hamming distance. Network is said to converge to a solution if the minimal hamming distance between any pair of nodes is not less than the specified hamming distance d.

The vectors, a output code vector and a probability vector, reside in each node. Each value in the probability vector denotes the probability of the corresponding bit in the code word being toggled. Double stochasticity is involved in deciding neuronal output vector modification. Each node decides whether to change its output vector or not based on the hamming distances with the outputs of connected nodes and this is performed as follows:

Node i is updated if $\text{rand}(0,1) < exp(-(d - d_i)/T)$,
where $d_i = \sum_{j \neq i} d_{ij}, \forall j$ such that $d_{ij} \leq d$.
Once node decides to update its output, its network

response vector Y is computed and is used to calculate the probability vector. The method of computing the network response vector depends on the problem at hand and is discussed later. Probability vector P_i for node i is computed as follows,

$$P_i(r) = \frac{exp(-\frac{Y_i(r)}{T})}{\sum_q exp(-\frac{Y_i(q)}{T})}$$

In the above equation, the term T is called temperature that controls the stochasticity in deciding which bit to toggle in the output vector. At large values of T, all bits have equal probability of getting selected for toggling. A bit is selected using the probability vector and is toggled.

Now the problem is to compute the network response vector which plays a crucial role in finding good codes. Computation of Y depends on the problem and is different for the MDC and the CWC. In the case of MDC, a difference vector V_{ij} between two nodes i and j is calculated and then Y_i is computed.

$$V_{ij}(r) = |1 - (c_i - c_j)|, i \neq j$$

$$d_{ij} = l - \sum_r V_{ij}(r)$$

$$Y_i(r) = A \sum_{j \neq i} V_{ij}(r) d_{ij}^{-2}$$

Algorithm for the MDC is presented in Fig. 2. For constant weight codes, we have to impose an implicit constraint, that is the weight of the code word should remain w even after updation. This can be guaranteed by toggling one 1 and one 0, so that the weight remains unchanged. Here two probability vectors, P_i^0 and P_i^1 for node i are calculated as given below,

$$P_i^0 = \frac{c_i(r) exp(-\frac{Y_i(r)}{T})}{\sum_q c_i(q) exp(-\frac{Y_i(q)}{T})}$$

$$P_i^1 = \frac{(1 - c_i(r)) exp(-\frac{Y_i(r)}{T})}{\sum_q (1 - c_i(q)) exp(-\frac{Y_i(q)}{T})}$$

Using the two probability vectors, two bit positions are selected and the respective bits are toggled. Algorithm for the CWC is presented in Fig. 3.

All code words are initialized with random bits in the case of the MDC and are initialized with random constant weight code words in the case of the CWC. The initial temperature is set to 10.0 and is decreased by multiplying with a constant factor, $\alpha(0.95)$. The number of inner loop iterations is initially set to 10 and is increased as the temperature decreases, by multiplying with a factor $1/\alpha$. Node updation ceases when network finds a solution satisfying minimal hamming distance.

4 Experimental Study and Results

We carried out our experiments on a PC/AT-386 machine. Parameters of some of the maximal distance codes that were designed are shown in Table I. We found some good constant weight codes than those reported in Reference [7]. Those code parameters are listed in Table II. Detailed listing of these codes are available in the Reference[4]

d = 3 (l,N)	d = 5 (l,N)	d = 7 (l,N)
(5,4)	8,4)	(11,4)
(6,8)	(9,6)	(12,4)
(7,16)	(10,12)	(13,8)
(8,20)	(11,24)	(14,16)

Table I.

l	w	d	N
22	9	10	25
23	9	10	30
23	11	10	41
24	9	10	35

Table II.

Execution time of each run ranged from few seconds to 10 minutes. For small codes, such as M(7,16,3) and M(7,6,3), network converged to final solution within one or two runs. Whereas for codes such as A(23,30,9,10), the network had to be run many times with different temperature schedules to find a solution. The method of computing the network response vector is important in finding good codes.

Our investigation on the use of neural network model to solve coding problem gives an insight into the design aspects of neural network models for some intractable problems.

5 Conclusions

In this paper, a probabilistic neural network model to find good codes has been proposed. In this model, each node assumes some memory and processing capability. Network algorithms for the MDC and the CWC designs have been presented. A code word is represented with a node in the net-

work that repels all other nodes in order to maximize the mutual hamming distances. Experimental results indicate that neural network model is quite effective in finding good codes. Some parameters of the codes that were found with the proposed model have been presented. Parallel implementation of the proposed model is under progress.

REFERENCES

[1] S.Kirkpatrick, C.Gelatt, and M. Vecchi.(1983), "Optimization by simulated annealing," *Science*, Vol.220(4598), pp.671-680.

[2] Ackley, D.H, Geoffrey E. Hinton, and Sejnowski, T.J.(1985)."A learning algorithm for Boltzman Machines." *Cognitive Science*, pp.147-169.

[3] G.P.Babu and M.N.Murty," Optimal Clustering and Neural Networks", Submitted to *IEEE Trans. on Neural Networks.*

[4] G.P.Babu and M.N.Murty,"Design of error-correcting codes using a probabilistic neural network", Tech. Rep. CSA-IISC-1993.c2, IISC.

[5] Hopfield J.J., Tank D. W.(1985)."Neuronal computation of decision in optimization problem." Biol. Cybern. 52:141-152.

[6] Holland John. H.(1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

[7] Gamal, A. A., Hemachandra, L. A., Shperling, I., Wei, V.K.(1987), "Using Simulated Annealing to Design Good Codes", *IEEE Trans. on Information Theory*, IT-33, No 1, pp.116-123.

[8] Hill, R. (1986), *A First Course in Coding Theory*, Oxford University Press, New York.

[9] Shannon, C.E.(1948), A Mathematical Theory of Communication, *Bell Systems Tech. J*, vol 27, (pt I), pp.379-423 (pt II), pp.623-656.

[10] Conway, J. H., and Sloane, N. J. A.,(1986), "Lexicographic Codes: Error-Correcting Codes from Game Theory.", *IEEE Trans. on Information Theory*, IT-32, No 3.

[11] Dontas. K., and De Jong. K.,(1990), "Discovery of Maximal Distance codes using Genetic Algorithm," *Proceedings IEEE 2nd Intl. Conference on Tools for Artificial Intelligence*, pp.805-811.


```
while(T_Max > T_min)
begin
  count = 0,
  while(count < Max_Ite)
  begin
    select a node i randomly,
    compute Y_i, P_i
    j = select_bit(P_i)
    c_i(j) = abs(1 - c_i(j))
    count = count+1
  end
  T_Max = αT_Max
end

select_bit(P_i)
begin
  sum = 0.0; p = random(0.0,1.0);
  for j = 1 to l do
    sum = sum + P_i(j);
    if( sum ≥ p) return(j);
end
```

Fig.2. Algorithm for MDC

```
while(T_Max > T_min)
begin
  count = 0,
  while(count < Max_Ite)
  begin
    select a node i randomly,
    compute Y_i, P_i^0, P_i^1
    j_0 = select_bit(P_i^0)
    j_1 = select_bit(P_i^1)
    c_i(j_0) = 1, c_i(j_1) = 0,
    count = count+1
  end
  T_Max = αT_Max
end
```

Fig.3. Algorithm for CWC.