Sādhanā, Vol. 21, Part 1, February 1996, pp. 91–99. © Printed in India.

Code design using deterministic annealing

BINOY JOSEPH and ANAMITRA MAKUR

Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560 012, India

Abstract. We address the problem of designing codes for specific applications using deterministic annealing. Designing a block code over any finite dimensional space may be thought of as forming the corresponding number of clusters over the particular dimensional space. We have shown that the total distortion incurred in encoding a training set is related to the probability of correct reception over a symmetric channel. While conventional deterministic annealing make use of the Euclidean squared error distance measure, we have developed an algorithm that can be used for clustering with Hamming distance as the distance measure, which is required in the error correcting scenario.

Keywords. Error correcting code; Hamming distance; deterministic annealing.

1. Introduction

One property of error correcting codes that do not incorporate memory is that it should have the region around each of the codewords as large as possible, so that a larger number of errors can be corrected. Figure 1 shows a geometrical view of some codewords with the corresponding non-overlapping regions, where points denote possible received vectors. At the same time we would also like to have the number of codewords in the codebook as large as possible, so that more input messages can be encoded using the code. The above two requirements are obviously contradictory in nature. The codewords used are vectors of a particular dimension over some alphabet q. If there exists a Galois field over q (i.e. if q is a prime or a prime power) then efficient families of codes are known which make use of polynomial arithmetic, and strict upper bounds to the number of possible codewords exist. For q's where we do not have a Galois field we do not know many codes there are and the maximum number of codewords that we can have is also unknown.

A possible method to form error correcting codes over non-prime q's would be to make use of some sort of clustering algorithms and pick out the codewords as the cluster centres. Efforts to make use of clustering for designing codes can be found in El Gammal *et al* (1987). Figure 1 may also be viewed as the outcome of a clustering algorithm, showing some clusters, cluster centres, and input data points. In the case of designing a code our input data is going to be the set containing all possible $q-$ary $n-$tuples. While designing codes, our objective would be to form homogeneous clusters of the required size over

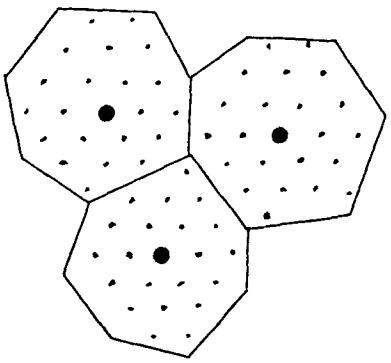

Figure 1. Geometrical view of a code/cluster.

$q-$ary $n-$tuples, such that a code with a given rate performs the best (for some channel). Clustering algorithms usually minimize the *total distortion* (average distortion) incurred if the representative vectors were used to encode the given data. It will be shown in the next section that if the distance measure used is the Hamming distance then the above two objectives are equivalent.

For example, consider the binary [7,4] Hamming code. If we look at it geometrically we see that it consists of 16 codewords which may be considered as the representative vectors of 16 different clusters. Each cluster consists of eight binary 7-tuples which are at a Hamming distance of 0 or 1 from the respective codewords. The clusters are non-overlapping and hence any single error can be corrected at the receiver. The distortion turns out to be a minimum if we use the Hamming code to encode all possible binary 7-tuples. If any other 16 vectors are used for this purpose, it would give a higher total distortion. Thus a direct relationship exists between minimum distortion clustering and code design, and we see that codes may be designed using the same philosophy as used for generating clusters from a given set of input data.

Many of the clustering algorithms use real arithmetic in forming the clusters. If we use algorithms which use real arithmetic for code design, then we will have to translate them back to the finite alphabet. Generally, for non-binary cases this will not be easy as there exists no simple relationship between the distance measure in real arithmetic and that in the finite alphabet. So we need some algorithm which can do clustering in a finite alphabet with Hamming distance as the distance measure.

2. Performance analysis of codes designed using clustering

We show that there exists a direct relationship between the total Hamming distortion incurred if the codewords were used for encoding the particular dimensional space to which they belong and the probability of error while decoding at the receiver. Transmission over a q-ary symmetric channel is assumed. To do so, consider the following example. Let $c_0, c_1, \cdots, c_{N-1}$ be the N codewords in the code and let $s_0, s_1, \cdots, s_{N-1}$ be the corresponding *clusters*. By clusters about a set of points in a space, we mean the partitioning of the space according to the nearest neighbour rule. Now define

$$d(s_i, c_i) = \sum_{x_j \in s_i} d_H(x_j, c_i) \tag{1}$$

where $d_H(x_j, c_i)$ is the Hamming distance between any two vectors x_j and c_i. The set of all possible received vectors consists of all possible $q-$ary $n-$tuples. Let

$$D_{tot} = \sum_{i=0}^{N-1} d(s_i, c_i) \tag{2}$$

be the total distortion incurred if the set of codewords were used for encoding the space of all possible received vectors. Let r be the received vector when c_i is transmitted. If $r \in s_i$ i.e., if r belongs to the cluster around c_i, then there is no error, else an error occurs while decoding. Now,

$$\begin{aligned} P_i(r \text{ is decoded correctly}) &= P(r \in s_i) \\ &= \sum_{r \in s_i} P(r|c_i) \\ &= \sum_{r \in s_i} p_e^{d_H(c_i,r)}(1-p_e)^{n-d_H(c_i,r)} \\ &= (1-p_e)^n \sum_{r \in s_i} \left(\frac{p_e}{1-p_e}\right)^{d_H(c_i,r)} \end{aligned} \tag{3}$$

where p_e is the probability that a bit would be in error for a binary symmetric channel and $P(r|c_i)$ is the conditional probability that r is received given that c_i was transmitted. Now probability of no error is given by

$$\begin{aligned} P_{corr} &= \frac{1}{N} \sum_{i=0}^{N-1} P_i(r \text{ is decoded correctly}) \\ &= \frac{(1-p_e)^n}{N} \sum_{i=0}^{N-1} \sum_{r \in s_i} \left(\frac{p_e}{1-p_e}\right)^{d_H(r,c_i)} \\ &= \frac{(1-p_e)^n}{N} \sum_{j=0}^{n} b_j \left(\frac{p_e}{1-p_e}\right)^j \\ &= \frac{(1-p_e)^n}{N} B\left(\frac{p_e}{1-p_e}\right) \end{aligned} \tag{4}$$

with the assumption that all N codewords are equiprobable, where n is the codelength and

$$B(p) = b_0 p^0 + b_1 p^1 + \cdots + b_n p^n. \tag{5}$$

Here

$$\begin{aligned} b_j &= \sum_{i=0}^{N-1} \sum_{r \in s_i, d_H(r,c_i)=j} 1 \\ &= |\{r : r \in s_i \quad \text{and} \quad d_H(r, c_i) = j\}| \end{aligned} \tag{6}$$

i.e., b_j's are the number of received vectors that are at a distance of j from a particular codeword c_i and fall within the cluster s_i. Hence performance of the code is given exactly by (8) and assuming that $p_e << 1$ it can be approximated as

$$\begin{aligned} P_{corr} &\simeq \frac{1}{N}(b_0 p_e^0 + b_1 p_e^1 + b_2 p_e^2 + \cdots + b_n p_e^n) \\ &= \frac{1}{N} B(p_e). \end{aligned} \tag{7}$$

Now, the total distortion (D_{tot}) incurred if the codewords were to be used for encoding all possible received vectors is given by

$$\begin{aligned} D_{tot} &= 0b_0 + 1b_1 + 2b_2 + \cdots + nb_n \\ &= \frac{d}{dp} B(p)|_{p=1}. \end{aligned} \tag{8}$$

From the above two equations for D_{tot} and P_{corr} it is clear that there exists a relationship between the total distortion and the probability of error.

Consider again the example of the binary [7,4] Hamming code. Here we have $N = 16$, $b_0 = 16$ (16 vectors have a distance 0 from some codeword), $b_1 = 112$ (remaining vectors have a distance 1 from some codeword) and $b_2 = \cdots = b_7 = 0$. Hence from (8) we find that $D_{tot} = 112$. Also $P_{corr} = (1 - p_e)^7 + 7(1 - p_e)^6 p_e$ which is the same as the one that we obtain from (4).

In general the b_j's are very difficult to obtain analytically and hence the analysis of codes designed for specific purposes is not very easy. The maximum value that b_j can have can be calculated as

$$b_{j_{max}} = N^n C_j q^j \tag{9}$$

where N is the number of codewords in the code. If the different b_j's for the code assume the corresponding maximum value or zero then the performance of the code would be the optimum. Examples for such codes are Hamming code, Golay code etc., which are perfect codes.

3. Deterministic annealing for code design

In this paper we use *Deterministic Annealing* (DA) (Rose *et al* 1990b), a stochastic relaxation clustering algorithm, for code design. Attempts to use other clustering algorithms may be found in Joseph (1994). Here we make use of an effective cost function that depends on the control parameter β. This cost function is then deterministically optimized at each β, starting from a low β where β is increased gradually. The probability that a particular input vector x belongs to a cluster C_j is calculated by using the following equation.

$$P(x \in C_j) = e^{-\beta E_x(j)} \Big/ \sum_{k=0}^{N-1} e^{-\beta E_x(k)} \tag{10}$$

where $E_x(j)$ is the cost incurred if vector x is associated with cluster j. The cost function generally used is the Euclidean squared error distance.

3.1 *Annealing schedule*

The performance of the deterministic annealing algorithm has been studied to begin with, in order to gain insight into its working. It is used for obtaining 16 codewords in binary 7 dimensional space with squared error cost function and real arithmetic. While simulating the algorithm we used different numerical representations of the two binary symbols. Table 1 gives the results for the various cases. The table shows that as the numeric value increases the algorithm converges faster. Figure 2 shows the history of a single run of the deterministic annealing algorithm where we have plotted the number of iterations versus the β value. The graph shows regions of low activity and very high activity in terms of

Table 1. Performance of deterministic annealing with different numeric values for the binary symbols.

Binary symbols represented as	β value above which the code is optimum	Initial β
0, 1	4.510	
1, −1	0.796	
2, −2	0.189	10^{-4}
3, −3	0.0911	
4, −4	0.042	

number of iterations, which indirectly describes the stage the algorithm passes through. A large increase in the number of iterations at a particular β stage implies that the codewords are being split. Once this happens it takes some time for them to separate out and hence increased activity is observed immediately after the codewords split. The algorithm was terminated once β becomes greater than 5.

The β schedule suggested by Rose *et al* (1990b) was that of 10% increment between successive β stages. It turns out that if the algorithm were to converge, then the factor by which β is increased should decrease with β. A linear β schedule has this property and it was observed that if we use such a schedule then the algorithm converges better. This may be intuitively explained as follows. To start with, the code consists of only a single distinct codeword. As β increases this codeword splits into two or more codewords at appropriate values of β and they separate out (Rose *et al* 1990a). As the annealing schedule used for

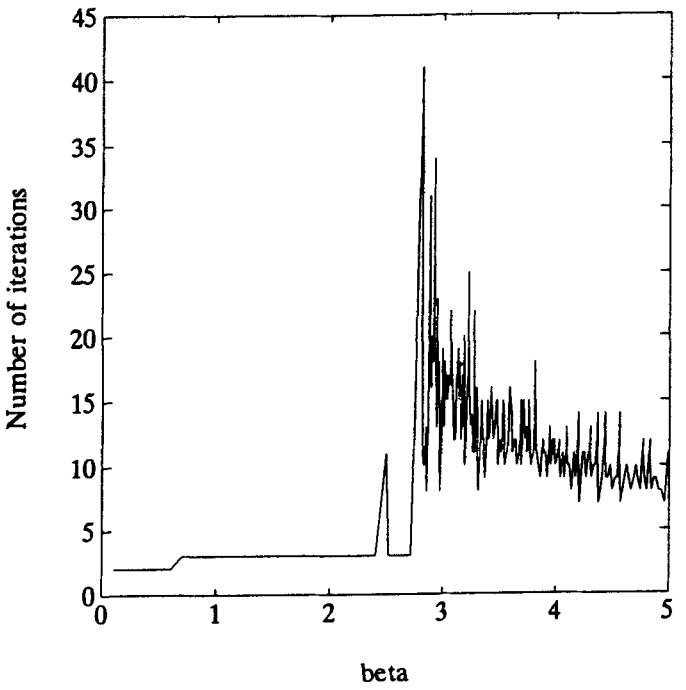

Figure 2. Number of iterations versus β for a single run of DA.

β is increasing in nature, successive splits of the codewords occur at higher and higher β values. Hence if the β schedule proposed by Rose and coworkers were followed, then the difference between successive β stages also would be more as β increases. The philosophy behind the deterministic annealing algorithm is to follow the global minimum from one β stage to the next assuming that the global minimum does not change much from one β stage to the next. If the increase in β from one stage to the next is considerable, then the above assumption may not hold and the algorithm may move away from the global optimum. This is the reason for the improved performance of the algorithm with linear schedule with small step sizes as the annealing schedule.

Also we notice from figure 2 that there is a direct relationship between the number of iterations at a particular β stage and the activity of the algorithm at that stage. If the codewords are being split then the number of iterations will be correspondingly high. This increased number of iterations allows the codewords to separate out once they split. It may be noticed that the splitting and separating out of the codewords are dependent on β. As our aim has been to track the global minimum of the objective function, codewords should be allowed to separate out immediately after the splitting, i.e., before β changes considerably. This then intuitively suggests that in the neighbourhood of β stages where the codewords split, increasing β by smaller amounts would help the codewords to separate out well. Hence, we made the β schedule dependent on the number of iterations at a particular β stage, i.e., we reduced the step size in the neighbourhood of β where the codewords start splitting. It was observed that the above strategy improves the convergence of the algorithm.

3.2 *Incorporating Hamming distance measure*

We propose below an algorithm that can do clustering with integer arithmetic and Hamming distance as the cost function. In the modified algorithm we replace the cluster centre criterion for codebook updating by a method based on probability. Here we calculate the probability, $P(x \in C_i)$ that an input vector x belongs to a particular cluster C_i using (2). Then with this probability we form an array $M_{ij}^{(k)}, 0 \leq i \leq N-1, 0 \leq j \leq n-1, 0 \leq k \leq q-1$, which would then be used for codebook updating. Once we have constructed the array $M_{ij}^{(k)}$, we update each symbol in the different codewords by symbols having the maximum probability. The algorithm is described below.

1. Set $\beta = 0, \beta_{max} = \infty$.
2. Choose an arbitrary set of initial cluster centres.
3. Initialize $M_{ij}^{(k)} = 0, 0 \leq i \leq N-1, 0 \leq j \leq n-1, 0 \leq k \leq q-1$
4. Find the probability $P(x \in C_i)$ that a training vector x belongs to a particular cluster C_i using (2), with Hamming distance as the cost measure.
5. Update the probability arrays $M_{ij}^{(k)}$ corresponding to the different symbols in the training vector (as explained below).
6. Update the codebook with symbols having the highest probability.
7. If $\beta < \beta_{max}$, perturb the codebook, increase β, go to step 3. Else stop.

Consider any particular codeword, say the ith one, in the codebook. We calculate the probability that a training vector belongs to the particular cluster using (2) with Hamming distance as the cost measure. After calculating the probability, we give a weightage equal to

this probability for the q different symbols in the input vector while updating the codebook. We keep a probability array $M_{ij}^{(k)}$ where k corresponds to one of the q different symbols, i denotes the codeword index, and j is the codeword dimension index. Hence $M_{ij}^{(k)}$ has a size of $q \times N \times n$. After calculating the probability that a training vector belongs to a particular cluster, we update the probability array as follows. Taking q symbols to be $0, 1, \cdots, q-1$, and n components of x to be $x_0, x_1, \cdots, x_{n-1}$, we have

$$M_{ij}^{(k)} = \sum_{x:x_j=k} P(x \in C_i). \tag{11}$$

Once we have gone through the entire input sequence, we update the present codebook with symbols with the highest probability in the respective positions,

$$y_{ij} = \max_k{}^{-1} M_{ij}^{(k)} \quad 0 \le i \le N-1, 0 \le j \le n-1. \tag{12}$$

In the case of a tie we resolve it randomly. We have tried different methods while updating the probability array. But none of them stood out as a clearcut winner.

3.3 *Simulation examples*

We used the algorithm for designing codes for some given M over binary 7-tuples and ternary 4-tuples. The motivation for designing codes for some given M comes from the following. As an example consider the case where the input set consists of 10 messages (i.e., $M = 10$) and single bit error correction is desirable. One possible method would be to use 10 codewords from binary [7,4] Hamming code for this purpose so that at the receiver we may correct any single error. Another alternative for this would be to design a code consisting of 10 codewords over binary 7-tuples and use it for the purpose. For values of M where optimum code does not exist/is not known, designing a code using the proposed algorithms might give an improved performance over the second alternative. The improved performance is at the cost of extra computation, since there may not be any easier way of decoding the code than minimum distance encoding at the receiver.

Table 2 shows a code C_1 consisting of 10 codewords designed using the proposed deterministic annealing algorithm for $q = 2, n = 7$. In table 3 we compare this code and the code C_1' with 10 codewords from binary [7,4] Hamming code. It may be noticed that

Table 2. Examples of codes designed for specific applications.

C_1	C_2	C_3
1111101		
1010000	1111110	
0110010	1100101	0001
1011110	0000110	0212
1100100	0110001	1110
0011001	0011100	2121
0101110	1010010	2200
0000111	1011001	1022
0100001	1101011	
1001011		

Table 3. Comparison of codes C_1, C_1', C_2, C_2', C_3 and C_3' designed from corresponding Hamming code and using DA.

b_i	Code with 10 codewords		Code with 8 codewords		Code with 6 codewords	
	From [7,4] Hamming code C_1'	Designed using DA C_1	From [7,4] Hamming code C_2'	Designed using DA C_2	From ternary [4,2] code C_3'	Designed using DA C_3
b_0	10	10	8	8	6	6
b_1	70	70	56	56	48	48
b_2	40	48	48	63	24	27
b_3	8	0	16	1	3	0
Total dist.	174	166	200	185	105	102

C_1 gives less total distortion in terms of Hamming distance, when used for encoding the binary 7-dimensional space when compared to C_1'. Figure 3 gives the relative performance of the two codes in terms of probability of error assuming a binary symmetric channel. From the figure it can be seen that C_1 performs better than C_1'. This implies that we can have improved performance if we design application specific codes.

Also shown in table 2 is a code C_2 consisting of 8 codewords designed using the proposed deterministic annealing algorithm over binary 7-dimensional space. In table 3 we compare this code with a code C_2' consisting of 8 codewords from binary [7,4] Hamming code. Again, figure 3 compares the relative performance of the two codes in terms of probability of error assuming a binary symmetric channel. It can be seen that C_2 performs better than C_2' in terms of probability of error.

Also given in table 2 is a code C_3 consisting of 6 codewords over ternary 4-dimensional space. In table 3 we compare this code with a code C_3' consisting of 6 codewords from

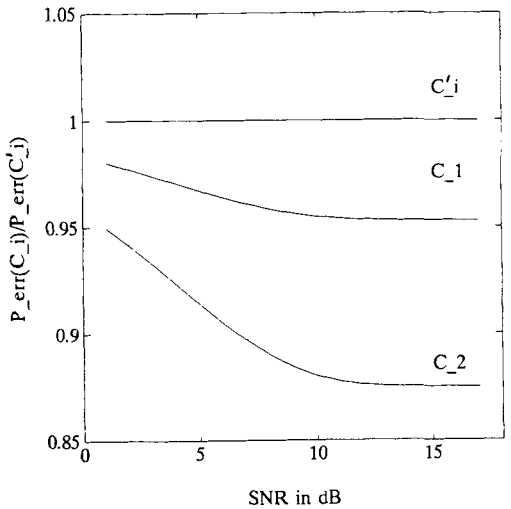

Figure 3. Performance of application specific codes designed using DA.

ternary [4,2] Hamming code. Here also the code obtained from deterministic annealing algorithm gives less distortion when used for encoding the ternary 4-dimensional space. It implies that the code designed using the proposed algorithm performs better in terms of probability of error.

4. Conclusion

In this paper we have developed the concept of the relationship between clustering and code design. We have also seen the need for an algorithm that can do clustering over finite dimensional spaces with Hamming distance as the distance measure. We have chosen deterministic annealing, an efficient clustering algorithm with real arithmetic, for code design. We have proposed a modification to this algorithm which enables it to do clustering using Hamming distance as the distance measure. It was observed that the convergence of the algorithm can be improved by using a linear annealing schedule for β with decreasing step sizes at points where the codewords split. Some examples for codes designed for specific applications have also been produced. The performance of the codes with respect to probability of error has been compared with codes consisting of the corresponding number of codewords from the respective Hamming code. It was observed that the codes designed for specific applications, although having equal minimum distance, perform better than the corresponding ad-hoc codes obtained from Hamming code.

References

El Gammal A A, Hemachandra L A, Shperling I, Wei V K 1987 Using simulated annealing to designing good codes. *IEEE Trans. Inf. Theory* IT-33: 116–123

Joseph B 1994 *Clustering for designing error correcting codes*. MSc (Eng.) thesis, Dept. of Elec. Commun. Eng., Indian Institute of Science, Bangalore

Rose K, Gurewitz E, Fox G C 1990a Statistical mechanics and phase transitions in clustering. *Phys. Rev. Lett.* 65: 945–948

Rose K, Gurewitz E, Fox G C 1990b A deterministic annealing approach to clustering. *Pattern Recog. Lett.* 2: 589–594