PRAMANA
— journal of
physics

© Printed in India

Vol. 45, No. 6,
December 1995
pp. 569–582

A comparative study of two learning rules for associative memory

G ATHITHAN
Advanced Numerical Research and Analysis Group, P.O. Kanchanbagh, Hyderabad 500 258, India

MS received 22 May 1995; revised 20 October 1995

Abstract. This paper addresses itself to a practical problem encountered in using iterative learning rules for associative memory models. The performance of a learning rule based on linear programming which overcomes this problem is compared with that of a representative iterative rule by numerical simulation. Results indicate superior performance by the linear programming rule. An algorithm for computing radii of maximal hyperspheres around patterns in the state space of a model is presented. Fractional volumes of basins of attractions are computed for the representative iterative rule as well as the linear programming rule. With the radii of maximal hyperspheres as weight factors for corresponding patterns to be stored, the linear programming rule gives rise to the maximal utilisation of the state space.

Keywords. Associative memory; learning rule; linear programming.

PACS Nos 87·10; 89·10; 89·80

1. Introduction

There has been much interest recently in the McCulloch–Pitts [1] neural networks as models of associative memories. In the Hopfield [2] version of such a network, the main task of a designer is to compute the neural interaction strengths as a function of the patterns to be stored and recalled associatively. Algorithms to compute interaction strengths are generally known as learning rules. The simplest of them is based on an hypothesis by Hebb [3]. Though biologically meaningful, the Hebb rule cannot be used to store more than $0{\cdot}14N$ random uncorrelated patterns where N is the number of neurons in the network [4]. The pseudo-inverse rule [5] can store N linearly independent patterns. Earlier, Willshaw *et al* [6] have proposed a rule for highly correlated patterns which stores patterns of the order of $N^2/(\log N)^2$. In these and other rules [7] the sizes of basins of attraction around the patterns in the state space are not controlled explicitly. Using a number of rules proposed subsequently [8–14], it is not only possible to ensure that patterns are stored in the network but also to provide around each one a basin of attraction whose measure is specified by the designer. Conditions for stability in terms of measures of the sizes of basins of attraction, the patterns and the interaction strengths are formulated first. Beginning with arbitrary values for the interaction strengths, these rules help the designer to move towards the correct values iteratively till these conditions are satisfied. Convergence to the correct values are ensured, provided a solution satisfying the initial conditions exists in the space of interaction strengths. However, in practical situations it will be difficult to know if a solution exists a priori. Therefore, one would have to assume conservative values for

the sizes of basins of attraction. Consequently the basins of attraction may not pack the state space of the network as tightly as possible.

A learning rule based on linear programming [11] overcomes the above mentioned drawback of iterative rules. The linearity of conditions of stability combined with linear normalization of interaction strengths enables one to employ the linear programming approach. Unlike the case of iterative rules where measures of the sizes of basins attraction are expected as external inputs, in the linear programming approach, maximal values for these measures are computed automatically. The interaction strengths are computed as a byproduct. Though this rule was first formulated by Krauth and Mezard, its distinct advantages are not argued or brought out clearly in their paper. Moreover, they do not report extensive numerical studies of this rule's performance in recalling stored patterns. This paper proposes to complement their work and fill out the above mentioned gaps.

While employing the linear programming approach, it is possible to specify different relative weights for different patterns. If these weights are selected in accordance with the inter-pattern distances, then maximal utilization of the state space of the model can be achieved as a result. An algorithm to compute the radii of maximal geometric hyperspheres around each pattern is presented. Use of these radii as relative weights for patterns results in maximal fractional volume of basins of attraction as shown by Monte-Carlo simulations.

The paper is organized as follows. The terms and notations are introduced in § 2 by way of briefly reviewing the iterative learning rules. It is followed by § 3, which describes the rule based on linear programming. Section 4 discusses numerical simulations for comparing the performances of a representative iterative and linear programming rules. The next section presents an algorithm to compute the radii of maximal geometric hyperspheres around each pattern. It also describes a Monte-Carlo method to compute the fractional volume of basins of attraction and presents the results. Discussions and conclusions make up the last section.

2. Iterative learning rules

The standard Hopfield network consists of N Ising spins $\sigma_i = \pm 1$ for $i = 1, \ldots, N$. The interaction strength from spin j to spin i is denoted by J_{ij}. All self-interactions, J_{ii}, are assumed to be zero throughout this paper and the $\{J_{ij}\}$ need not be symmetric. The local field, $h_i(t)$, at site i and at time t is computed by

$$h_i(t) = \sum_{j=1}^{N} J_{ij}\sigma_j(t) \tag{1}$$

where $\sigma_j(t)$ is the Ising spin at site j and at time t. The spins, or equivalently neurons, are updated according to the rule

$$\sigma_i(t + \delta t) = \text{sign}(h_i(t)). \tag{2}$$

A configuration, or equivalently a pattern, $\{\xi_i : i = 1, \ldots, N\}$ is stored, i.e., becomes a fixed point of dynamics (2), if

$$c_i = \xi_i h_i(\{\xi_i\}) \tag{3}$$

is positive for all i. In fact, c_i is a measure of the size of basin of attraction for the pattern $\{\xi_i\}$. Therefore, for effective associative recall, c_i must be greater than a positive constant, say, k. For storing p patterns, biased or random, $\{\xi_i^\mu : i = 1,\ldots,N; \mu = 1,\ldots,p\}$, the fixed point conditions

$$\xi_i^\mu h_i(\{\xi_i^\mu\}) > k \tag{4}$$

must be satisfied at each site i for each pattern μ. The positive constant could be pattern dependent, if required. Without some sort of normalization or bounds for the J_{ij}'s, the inequalities (4) would be devoid of meaning. Two types of normalizations are used in the iterative rules proposed in the literature. In the first type the J_{ij}'s are expected to obey

$$\sum_{j=1}^{N} J_{ij}^2 = N, \tag{5}$$

for each i (in some cases N is replaced by 1). In the second, the bound

$$|J_{ij}| < J_{\max} \tag{6}$$

is imposed on all J_{ij}'s.

In Gardner's version of the iterative approach, the interaction strengths $\{J_{ij}\}$ are computed as follows. The $\{J_{ij}\}$ are assigned any set of values to begin with. A mask ε_i^μ is defined at each site i and for each pattern μ as follows.

$$\varepsilon_i^\mu = \theta\left[k\left(\sum_{j=1}^{N} J_{ij}^2\right)^{1/2} - \sum_{j=1}^{N} J_{ij}\xi_i^\mu \xi_j^\mu\right] \tag{7}$$

where $\theta(x)$ is the Heaviside step function. The iterative corrections for $\{J_{ij}\}$ are computed as

$$\Delta J_{ij} = \varepsilon_i^\mu \xi_i^\mu \xi_j^\mu \tag{8}$$

to update the $\{J_{ij}\}$ till ε_i^μ become zero for all i and μ. The normalization condition (5) is made use of implicitly in computing ε_i^μ using (7). This local, Hebb rule like, algorithm is guaranteed to converge provided a solution of $\{J_{ij}\}$ satisfying the conditions (4) exists in the space of interactions [15]. Existence of a solution is, of course, dependent on the number and nature of patterns and also on the constant k. Larger the value of p and k, the less probable it is for a solution to exist. Gardner's deep analytical results are useful in knowing about the existence of such solutions when one deals with an abstract category of patterns drawn from a known probability distribution. However, in practical situations where one needs to store a specific set of p patterns there is no way of knowing if a solution exists or not. There are no guidelines either for choosing an optimal value for k. Both these problems are easily overcome by adopting the linear programming approach to compute the interaction strengths.

Two other iterative rules [10, 11] make use of the fact that the inequality conditions (4) can be decoupled into N independent sets. The latter rule, called minimum-overlap rule, is a minor variant of the former. Starting with some a priori value of k, the rules iteratively try to satisfy these N sets of conditions independently. Once again, convergence is ensured only if a solution for $\{J_{ij}\}$ exists. Therefore both these rules are subject

to the critical observations made against Gardner's iterative rule. In fact, unless the computation of $\{J_{ij}\}$ is posed as a problem of optimizing k, the problem of finding out a priori if a solution of $\{J_{ij}\}$ exists or not cannot be avoided whatever be the iterative formulation. Griniasty and Gutfreund [16] do point out that learning can be viewed as an optimization problem. However they do not discuss the application of linear programming for learning patterns.

Krauth and Mezard [11] present the linear programming approach in the same paper where they discuss the minimum-overlap rule and compare the two. They compute measures of stabilities or equivalent sizes of basins attraction in case of both the rules. When the normalization condition (6) is used, these measures turn out to be larger for the linear programming rule, as one would expect. If one uses (5) for normalization, then the problem of computing $\{J_{ij}\}$ can no longer be posed as a linear programming problem. However, in their paper a comparison between the two rules with normalization based on (5) is presented which shows the linear programming approach in poor light. It is not clear how they used the simplex algorithm to maximise k subject to conditions (4) and (5) as the latter are quadratic. The point argued here is that when all the conditions involved in the problem are linear the best approach is one based on linear programming. Besides finding out if a suitable solution of $\{J_{ij}\}$ exists or not, this approach provides unique and maximal possible stabilities or sizes of basins attraction for the patterns to be stored.

3. Linear programming rule

Linear programming is concerned with finding optimum values of linear objective functions subject to a set of equality or inequality constraints. The constraints have to be linear and the standard method for solving a linear programming problem (LPP) is known as the simplex algorithm. Starting with a feasible solution which satisfies the given constraints, the algorithm improves the solution from one step to another till the optimum is reached. In case no feasible solution exists, the algorithm is capable of detecting that as well as the possibility of unboundedness of the optimal value. In the canonical formulation of an LPP one has a linear objective function which is to be maximised and a number of linear constraints. The variables are normally assumed to take only positive values.

The problem of computing $\{J_{ij}\}$ can be formulated as an LPP in the following manner. Let the objective function c be specified by

$$c = k. \tag{9}$$

One set of constraints are

$$\xi_i^\mu \sum_{j=1}^{N} J_{ij}\xi_j^\mu - k > 0. \tag{10}$$

The other set of constraints are the normalization bounds specified by (6). The problem is to maximise c subject to (10) and (6) which are a set of $pN + N(N-1)$ linear constraints.

The $\{J_{ij}\}$ can be positive or negative, whereas in the canonical formulation of LPP, negative values for the variables are not permissible. Therefore a new variable

J'_{ij} is defined as

$$J'_{ij} = J_{ij} + J_{\max}. \tag{11}$$

Substituting for J_{ij} in (10) and (6) from (11) one obtains

$$\xi_i^\mu \sum_{j=1}^{N} J'_{ij}\xi_j^\mu - k > \xi_i^\mu \left[\sum_{j \neq i} \xi_j^\mu \right] J_{\max} \tag{12}$$

and

$$J'_{ij} > 0 \tag{13}$$

$$J'_{ij} < 2J_{\max} \tag{14}$$

for all i, j and μ. The diagonal J'_{ii} are presumed to be 0. The standard requirement that the variables must have positive values is met by (13) while (14) provides positive bounds on $\{J'_{ij}\}$. These bounds eliminate the possibility of an unbounded optimum. The final formulation is then, maximise $c = k$ subject to (12) and (14). This formulation is only marginally different from that of Krauth and Mezard [11].

4. Numerical simulation

The principal aim of the numerical simulation reported here is to compare the performances of Gardner's version of iterative rule and the linear programming rule. Throughout the simulations N is assumed to be 100. The performance of a learning rule is evaluated as follows. Firstly a value of p is chosen between 1 and N, and that many unbiased patterns are generated. The interaction strengths are computed next using the learning rule to be evaluated. One of the stored patterns, say $\{\xi_i^\nu : i = 1, \ldots, N\}$, is selected as a test pattern and is persisted with throughout the evaluation. The test pattern is corrupted by selecting IN (denotes Input Noise) sites randomly and inverting the spin values there. Using the result as the input key, the network is used to recall the original test pattern. For a given value of IN, a total of ten recall experiments are carried out. In each case the overlap, m, of the recalled state, $\{\sigma_i\}$, with the original test pattern is computed using

$$m = \frac{1}{N} \sum_{i=1}^{N} \sigma_i \xi_i^\nu,$$

and the average of these ten overlaps is evaluated. By varying $\alpha = p/N$ from $5/N$ to 1 and IN from 0 to 50, a surface of overlaps over the phase plane $\alpha \times$ IN is obtained. The shape of the surface by itself is an indicator of the efficacy of the learning rule being evaluated. Let an overlap value greater than or equal to 0·99 in a recall experiment be termed a success. By sectioning the overlap surface with a $m = 0{\cdot}99$ plane parallel to the phase plane, a contour line of IN as a function of α is obtained. It is easy to see that the value of IN on this line at a given α is an experimental measure of the radius of the basin of attraction, r, around the test pattern. This line separates the phase plane into two regions in one of which the network functions as useful associative memory. Thus, the shape and location of this line on the phase plane is a good index of performance of the learning rule.

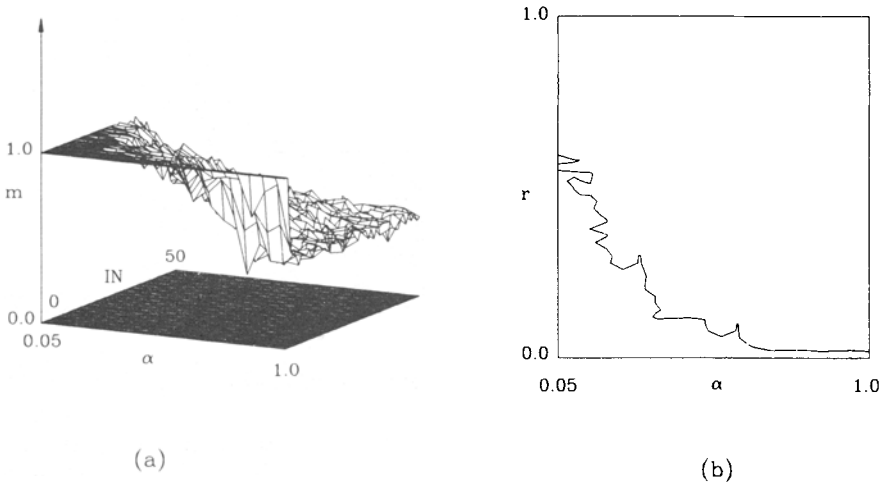

Figure 1. The surface of overlaps (m) between recalled patterns and the original test pattern is shown in (**a**) for the iterative learning rule. The number of bits selected randomly and reversed in the test pattern to generate the input key for recall experiments is represented by IN. The variable α denotes the storage capacity p/N. The contour line on the surface of overlaps at a value of $m = 0{\cdot}99$ is shown in (**b**). This contour line represents the radius of the basin of attraction of the test pattern as a function of α. The value of the parameter k is 10.

The evaluation of Gardner's version of iterative rule has been done taking the value of k to be 10. Also, to make the comparison with the linear programming rule meaningful, instead of using (5), eq. (6) is used for the normalization of $\{J'_{ij}\}$. That is, the mask ε_i^μ is defined as

$$\varepsilon_i^\mu = \theta\left[k - \sum_{j=1}^{N} J_{ij}\xi_i^\mu \xi_j^\mu\right]. \tag{15}$$

While updating $\{J_{ij}\}$, condition (6) is enforced by clipping those values of $|J_{ij}|$ which exceed $J_{\max}$. A value 10 is assumed for $J_{\max}$. With constant magnitude of the correction term ΔJ_{ij} in (8), convergence has been found to be slow and sometimes impossible. Attenuating the correction term progressively as the iterations proceed has helped to speed up the convergence. Assuming a maximum permissible number of iterations, the correction term was linearly attenuated from a maximum value of 1 to a minimum value of 0. The resulting overlap surface and the $m = 0{\cdot}99$ contour line for the iterative rule are shown in figure 1.

Evaluation of the linear programming rule, however, runs into implementational problems as a direct employment of the simplex algorithm would require a huge amount of computer memory. With $N = 100$, the size of the co-efficient matrix representing constraints (12) and (14) would be of the order of 10^8. Such a size cannot be accommodated within the memories of standard workstations which usually have memories of the order of 10^6 to 10^7. Therefore a subdivision of the LPP becomes necessary. In this context it may be noted that the constraints (12) decouple into N independent sets of constraints one set for each i. Optimization of k for each set using

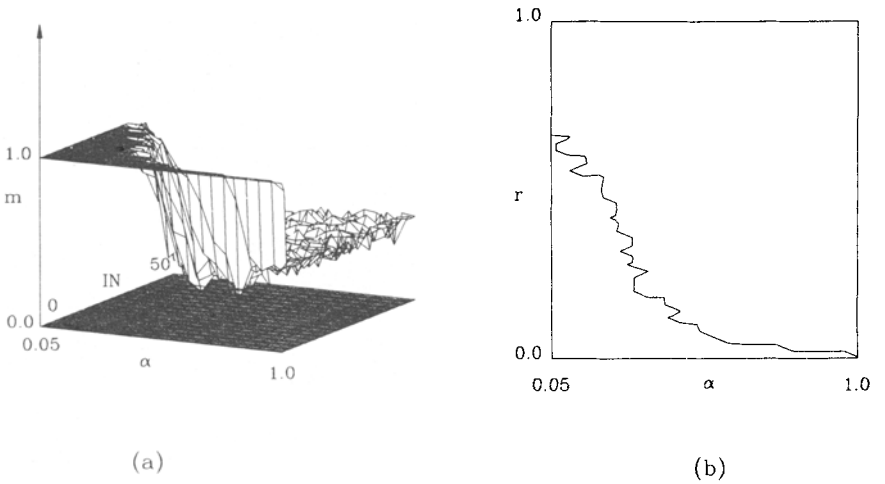

Figure 2. The surface of overlaps (m) and the $m = 0{\cdot}99$ contour line in the case of the linear programming rule. The meaning of the variables and parameters is the same as given for figure 1. The value of J_{max} is 10.

the simplex algorithm will result in the computations of $\{J'_{ij}: j = 1,\ldots,N\}$ for the corresponding i. The entire set of $\{J'_{ij}\}$ can be computed by employing the simplex method for $i = 1,\ldots,N$, individually. This scheme requires memory of the order of 10^4 which is manageable. Keeping the co-efficient matrix small also helps in limiting the numerical round-off and truncation errors. However, a side effect of this scheme of computing $\{J_{ij}\}$ is the dependence of optimal k on i. In the actual simulation, k as a function of i has been found to vary only slightly. In fact, the standard deviation of k's for various i's turns out to be less than 5%. For J_{max} a value of 10 is assumed, as in the case of the iterative rule. The simplex implementation due to Press *et al* [17] has been used in the computations. The overlap surface and the corresponding $m = 0{\cdot}99$ contour line for the linear programming rule are shown in figure 2.

Qualitatively the linear programming rule appears to perform better than the iterative one. The overlap surface bounded between the contour line and the origin is flat with fewer undulations compared to that of the iterative rule implying error-free recall in that region of the phase plane. For a close visual comparison the $m = 0{\cdot}99$ contour lines of both the rules are shown together in figure 3. In view of the fact that only ten recall experiments are carried out at each point to determine the overlap, the error margins on these contour lines are likely to be high. Nevertheless from the general trend of these lines, linear programming rule appears to perform better than the iterative one, though towards the $\alpha = 1{\cdot}0$ limit the contour lines tend to coincide. This is expected, because the optimal values of k around $\alpha = 1{\cdot}0$ as found by the linear programming rule turn out to be close to 10. The optimal values of k as a function of α decrease with increasing α which is again expected. Fewer patterns mean larger basins of attraction for each one of them and vice versa. For a visual comparison of the performances of linear programming and iterative rules with that of Hebb rule, the interested reader may refer to figure 4.

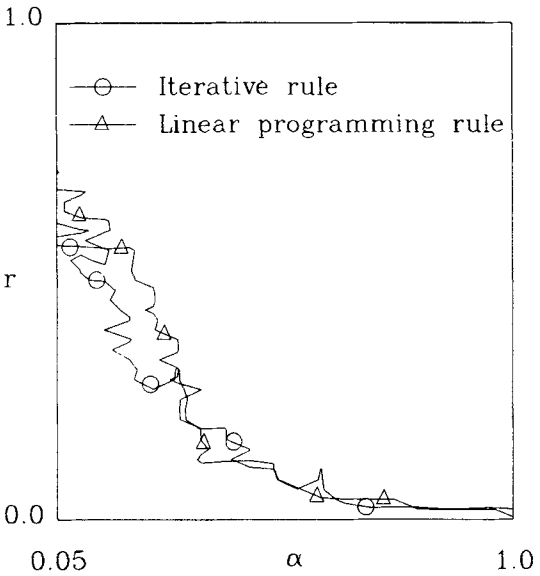

Figure 3. For a close comparison, $m = 0{\cdot}99$ contour lines in figures 1 and 2 are shown together here. The line with circles corresponds to the iterative rule while that with triangles corresponds to the linear programming rule. For a substantial part of the α axis, the linear programming rule has a larger radius of basin of attraction compared to the iterative rule.

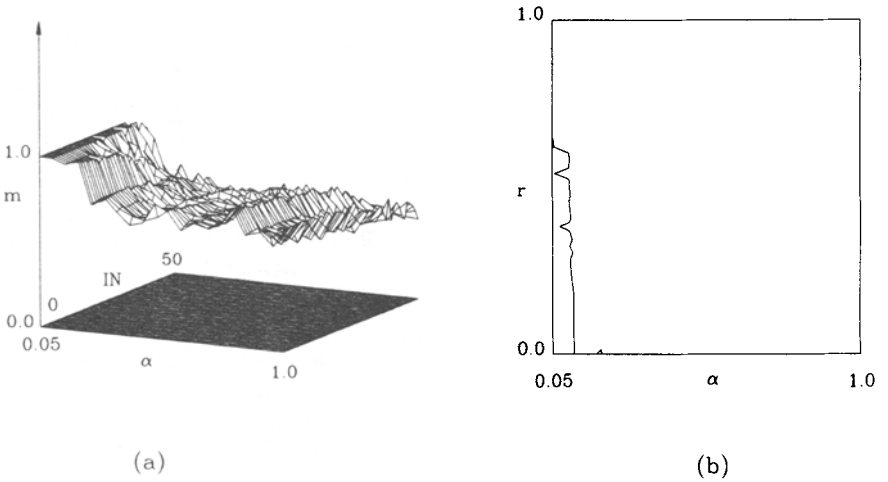

Figure 4. The surface of overlaps (m) and the $m = 0{\cdot}99$ contour line are shown here for the Hebb learning rule. The meaning of the variables and parameters is the same as given for figure 1.

As mentioned earlier, the near perfect flatness on the overlap surface corresponding to the linear programming rule suggests perfect recall in that region of the phase plane. To confirm that, recall experiments are carried out on either side of the $m = 0{\cdot}99$

Table 1. A comparison of the transition from perfect recall performance to no proper recall for two learning rules is presented in this table. In the first column α denotes the storage capacity p/N. While n_u denotes the maximal number of bit changes which ensures perfect recall, n_l denotes the minimal number of bit changes which results in no proper recall. In each case a hundred recall experiments have been carried out for determining these numbers.

	Iterative rule		Linear programming rule	
α	n_u	n_l	n_u	n_l
0·1	18	47	30	48
0·2	1	38	21	42
0·3	1	29	13	32
0·4	1	24	9	21
0·5	1	21	3	10

contour line along the IN axis. A total of 100 experiments are carried out at some discrete intervals of α and at various values of IN. For a given α, a threshold value of IN, n_l, above the contour line is found beyond which none of the 100 experiments results in a perfect recall. Similarly a threshold value of IN, n_u, below the contour line is found so that everyone of the experiments results in a perfect recall. The values of n_u and n_l are presented in table 1 for five different values of α, for both the iterative and the linear programming rules. These data confirm the perfect performance of the linear programming rule over a much larger region of the phase plane compared to the iterative rule. Also the transition from perfect performance to total failure is narrower in the case of the linear programming rule.

5. Maximal basins of attraction

In the evaluation of the linear programming rule, it is presumed that the weights for patterns are equal. As a result the basins of attraction around each one of the patterns are likely to be approximately of the same size. Given a set of patterns, the inter-pattern distances in the state space of the network are bound to have a spread. Therefore some patterns can afford to have larger basins of attraction compared to others so that the state space is more optimally utilised. However, with equal weights for all patterns such an eventuality may not be explicitly realized. Ideally, the basins of attraction for various patterns should close out any gap between them and be tangential to each other. Such an ideal situation may be obtained as follows.

Let the p patterns be conceived as p points in the state space. Then the task is to find maximal hyperspheres centred around each point such that these hyperspheres do not intersect each other but are just mutually tangential. The radii of these hyperspheres can be characterized as follows. Let hyperspheres with zero initial radii around each one of the p points expand at a uniform rate. Once two or more hyperspheres come in contact, they shall stop expanding and their radii reach their final

values. Once all expansion stops, the radii of the frozen hyperspheres become the maximal radii associated with the corresponding points. The assumption of uniform rate of expansion ensures equal rights for all patterns to the space around them. The following procedure which is a heuristic one finds close approximations to these radii using suitable metric.

Firstly, for each point, its nearest neighbour is found. The initial approximation to the radius of the maximal hypersphere at that point is assigned as half the distance to its nearest neighbour. After this step, all the points are sorted with respect to their initial radii in ascending order. For each point i, beginning with the first point in the sorted array, it is checked whether that point and its nearest neighbour are mutual nearest neighbours. If yes, then the initial approximate radii are taken as the final radii for both. If not, maximum permissible radius for the point i is computed as the distance to its neighbour minus the current radius associated with the neighbour. This maximum permissible radius is checked against the current radii of all other points to see if corresponding hyperspheres intersect. During this check, the maximum permissible radius is updated if necessary. At the end of this check, the final value of maximum permissible radius is assigned as the radius of the maximal hypersphere at point i. Once all the points are assigned their radii, the procedure stops. More details of this procedure are given in appendix-1.

To verify the above procedure, maximal circles for several sets of 50 points within a square have been computed. A sample output of the algorithm is illustrated in figure 5. The radii of these maximal circles can, of course, be computed by brute force simulation of the problem scenario assuming a small though finite incremental step for expanding the radii. In practice it has been found that the above heuristic algorithm could compute about 95% of the radii correctly; also, the error in the remaining 5% cases was found to be within 20% of the correct values.

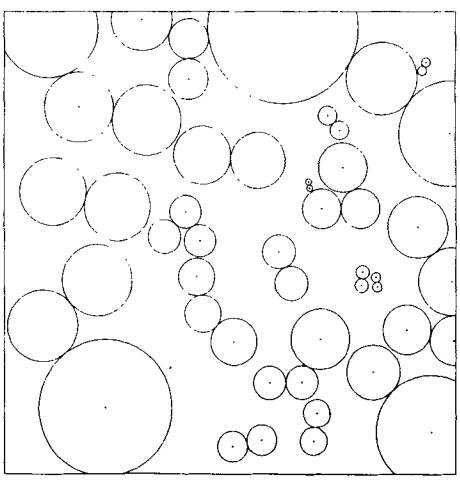

Figure 5. A sample set of fifty random points inside a square on a plane with maximal circles around each one of them. The radii of these circles are computed by an algorithm described in text.

Let the radii for maximal hyperspheres around patterns be denoted by γ^μ. Using these γ^μ's as weight factors for the patterns, condition (10) may be written as

$$\xi_i^\mu \sum_{j=1}^{N} J_{ij}\xi_j^\mu - k\gamma^\mu > 0. \tag{16}$$

The linear programming rule may be employed to maximise k subject to (16) and (6). The outcome is bound to be the maximum possible utilization of the state space to carve out the various basins of attraction. To verify this assertion, the Monte-Carlo method has been employed to compute the fractional volume of basins of attraction. A total of 10000 recall experiments are carried out starting from random input patterns. Each time the recalled state is compared with all the stored patterns and the maximum overlap is computed. If the overlap is above or equal to a preselected threshold the random input pattern is deemed to be inside the basin of attraction; else outside. The fractional volume of basins attraction is then computed as the ratio of the random input patterns that are found to be inside the basins to the total number of experiments.

Employing this method the fractional volumes of basins of attraction have been computed for three cases; firstly for the linear programming rule with radii of maximal hyperspheres around patterns as their weights; secondly for the linear programming rule with equal weights for the patterns; thirdly for the iterative rule. Hamming metric is used while computing the radii of maximal hyperspheres. The fractional volumes are computed as a function of α from $\alpha = 0{\cdot}05$ to $\alpha = 0{\cdot}5$. The value of α is increased by

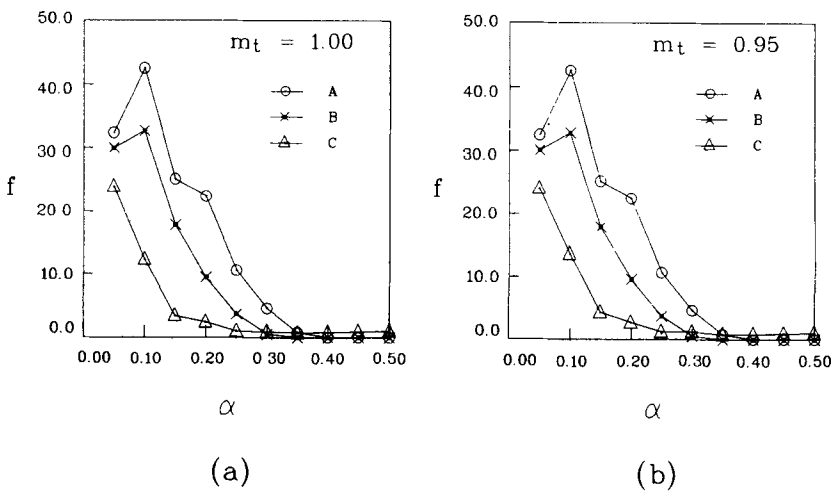

Figure 6. Fractional volume, f, of basins of attraction for stored patterns as a function of storage capacity α is shown here. Line A corresponds to the case of linear programming rule with radii of maximal hyperspheres around stored patterns as weights for these patterns. Line B corresponds to the case of the same rule with equal weights for the patterns. Line C corresponds to the iterative rule. Figure 6a represents the case of basins whose minima have 100% overlap with stored patterns. Figure 6b represents the case of basins with one or more local minima whose overlaps with stored patterns are greater than or equal to 95%.

adding the required number of additional patterns to the set of already existing ones. The results of the Monte-Carlo experiments are presented in figure 6. Figure 6a corresponds to the overlap threshold of value 1·0 while figure 6b corresponds to a value of 0·95. As remarked earlier, the first case utilizes the state space most optimally. Also, the difference between the second and first cases is quite pronounced, implying that the use of maximal radii as weights for the patterns further improves the sizes of the basins of attraction wherever possible.

6. Discussions and conclusions

A striking feature of the LP rule is the sharp transition exhibited by its overlap surface. In contrast, the overlap surface corresponding to the iterative rule shows a gradual downward trend as IN and α increase. Though a steady degradation in performance is desirable, a sharp fall in the recalling ability has its advantage. If an input key does not match with any of the stored patterns then the linear programming rule is capable of reporting a clear failure. Such an ability may be valuable in cognitive modelling. Another important aspect of the linear programming rule is the associated linear normalizing bounds on the $\{J_{ij}\}$. This aspect will be useful in designing VLSI neural chips with finite precision synaptic strengths. Limiting the magnitude of interaction strengths is also biologically meaningful as biological synapses cannot represent strengths of arbitrary precision.

Results of the computer simulations suggest that the basins of attraction in case of linear programming rule have smooth shapes converging to unique local minima. Thus, when an input key falls within a specific basin of attraction it leads to the recalling of the corresponding pattern without any error. Both the flatness on the overlap surface in figure 2 and the high values of n_u in table 1 corroborate this fact. Further evidence is available from data presented in figure 6. The dependence of f on α in cases A and B as shown in the figure has been observed to be invariant with respect to the two values of threshold overlaps implying the presence of unique local minima. With the iterative rule, however, f as a function of α has been found to depend on threshold overlaps.

The radius of the maximal hypersphere around a pattern as defined in the previous section gives an estimate of noise which the pattern can tolerate without losing its uniqueness. In that sense these radii could be useful in the study of associative memory and pattern recognition models. The procedure outlined earlier for computing radii of the maximal hyperspheres requires computing time of the order of $O(p^2)$ where p is the number of patterns. It is not an exact algorithm, though its solutions in hundreds of test cases in two-dimensions have proved to be satisfactory. Developing an exact algorithm based on either an 'incremental' or 'divide and conquer' approach may prove to be a worthwhile exercise.

Maximal possible utilization of the state space of the network is demonstrated employing linear programming rule with differentiated weight factors for the patterns. A trivial approach to find the closest match is, of course, to compute distances between the input and all the patterns to be stored and then select one of the latter that is least distant from the input. It is instructive to know that the basins of attractions in such an approach are nothing but Voronoi polytopes which cover the state space completely. This is a most ideal situation, which may not be achieved by the Hopfield network.

As pointed out earlier, the computational approach adopted here for implementing the linear programming rule makes the sizes of the basins of attraction depend on the neuronal index i. During computer simulations, however, this dependence was found to be small. As the standard Hopfield network is homogeneous, it is necessary that its performance parameters are independent of neuronal indices. In its extensions where homogeneity may not be desirable, making the sizes of basins of attraction depend on neuronal indices may be functionally necessary. The human brain, it may be noted, is highly differentiated even in regions dedicated to specialized tasks and any attempt in understanding it should consider heterogeneous networks sooner or later.

Acknowledgements

The author thanks Prof. Dipan K Ghosh and Dr G Venkataraman for their help and guidance at various stages of the work reported here.

Appendix 1

```
Procedure FindRadiiMaxHyperspheres (p)
    p:Number of points
{
    For each point i
    {
      RadiiAssignedFlags [i] = − 1;
      For each point j ≠ i
        Compute Distance [i,j];
      Compute its nearest neighbour, k;
      NearNeighbour [i] = k;
      Set the initial approximation to the radius of hypersphere at i as
      Radii [i] = Distance [i, k]*0·5;
    }
    Sort the points with their associated radii in ascending order;
    Initialize a pointcounter to 1;
Step L:  Pickup the current point, i, from the sorted points array
    If RadiiAssignedFlags [i] = − 1) then{
      Neighbour = NearNeighbour [i];
      If (i = NearNeighbour [Neighbour]) Then
      {
        RadiiAssignedFlags [i] = 1;
        RadiiAssignedFlags [Neighbour] = 1;
      }
      Else
      {
        MaxRadius = Distance [i, Neighbour] − Radii [Neighbour];
        For each point j ≠ i
          If  (Distance [i,j] < MaxRadius + Radii [j])
              MaxRadius = Distance [i,j] − Radii [j];
```

```
            Radii [i] = MaxRadius;
            RadiiAssignedFlags [i] = 1;
          }
        }
      }
      Increment pointcounter by 1;
      If (pointcounter < = p) Go To Step L;
      Else
        STOP;
}
```

References

[1] W S McCulloch and W A Pitts, *Bull. Math. Biophys.* **5**, 115 (1943)
[2] J J Hopfield, *Proc. Natl. Acad. Sci. USA* **79**, 2554 (1982)
[3] D O Hebb, *Organisation of behaviour* (New York: Wiley, 1949)
[4] D J Amit, H Gutfreund and H Sompolinsky, *Ann. Phys. NY* **173**, 30 (1987)
[5] L Personnaz, L Guyon and G Dreyfus, *J. Phys. Lett.* **46**, L359 (1985)
[6] D J Willshaw, O P Buneman and H C Languet-Higgins, *Nature (London)* **222**, 960 (1969)
[7] E Domany, J L van Hemmen and K Shulten (eds), *Models of neural networks* (Berlin, Heidelberg: Springer Verlag, 1991)
[8] E Gardner, *Europhys. Lett.* **4**, 481 (1987)
[9] G Poeppel and U Krey, *Europhys. Lett.* **4**, 979 (1988)
[10] S Diederich and M Opper, *Phys. Rev. Lett.* **58**, 949 (1987)
[11] W Krauth and M Mezard, *J. Phys.* **A20**, L745 (1987)
[12] B Forrest, *J. Phys.* **A21**, 245 (1988)
[13] L F Abbott and T B Kepler, *J. Phys.* **A22**, L711 (1989)
[14] J K Anlauf and M Biehl, *Europhys. Lett.* **10**, 687 (1989)
[15] E Gardner, *J. Phys.* **A21**, 257 (1988)
[16] M Griniasty and H Gutfreund, *J. Phys.* **A24**, 715 (1991)
[17] W H Press, S A Teukolsky, W T Vellerling and B P Flannery, *Numerical recipes in C* (Cambridge University Press, 1992)