Complex Systems **4** (1990) 477–490

Benchmark of Some Learning Algorithms for Single Layer and Hopfield Networks

Eddy Mayoraz
Ecole Polytechnique Fédérale de Lausanne, Département de Mathématiques, Chaire de Recherche Opérationnelle, CH-1015 Lausanne Switzerland

Abstract. Many algorithms have been proposed for training a single layer or a Hopfield network with binary activations. The purpose of this work is to compare some of these algorithms experimentally and point out the advantages of each. Experiments are also reported in which the density of the synaptic connections is reduced or in which a few quantization levels are used for the synaptic weights.

1. Introduction

During the last decade, artificial neural networks (ANN) have been proposed for many different applications, mainly for mapping approximation (multilayered networks), optimization problem (Kohonen and Hopfield networks), or pattern recognition and pattern association (Hopfield, Kohonen, and layered networks). In this paper, I will focus on the third type of applications. ANN used as associative memories present three major advantages over the classical methods for recognition and association. First, the sequential research is replaced by a parallel process. Second, the stored data are widely distributed in the network and this brings some robustness with respect to the parameters of the model. Last, a memory based on an ANN model presents some insensitivity to input noise. While the first characteristic depends only on the architecture of the network, the other two can vary with the learning algorithm used.

In what follows, I will describe some widely used learning algorithms, based either on algebra, geometry, or combinatorial optimization. I will investigate the influence of the number of stored patterns and the correlation of these patterns on the quality of error correction, while the memorization is performed by each of these algorithms. The robustness of the memory while using these different algorithms will be pointed out. Finally, I will study the performances of networks when the weights are restricted to take only three values.

2. Network models and learning quality measures

The *single-layer network* (SLN) is composed of m output neurons connected with n input neurons. In a binary model, each neuron is characterized by an *activation* taking the value -1 or $+1$ and denoted a_j for an input neuron j and b_i for an output neuron i. A *synaptic weight* w_{ij} is associated with each edge connecting the input neuron j to the output neuron i. For an input vector $\mathbf{a}$, the output of a cell i is defined by

$$b_i = sgn(v_i) = \begin{cases} -1 & \text{if } v_i \leq 0 \\ 1 & \text{if } v_i > 0 \end{cases}$$

where $v_i = \sum_{j=1}^{n} w_{ij} a_j$ is called the *membrane potential* of the cell i for the input $\mathbf{a}$. If $\mathbf{W}$ is the $m \times n$ matrix of synaptic weights and if $\mathbf{sgn}$ is a vectorial operator applying sgn for each component, we can also write

$$\mathbf{b} = \mathbf{sgn}(\mathbf{Wa}) \tag{2.1}$$

The SLN is suited for the pattern association problem, which can be formulated as follows: for a given learning set of p pairs of input–output vectors $\{(\mathbf{a}^1, \mathbf{b}^1), \ldots, (\mathbf{a}^p, \mathbf{b}^p)\}$, find a synaptic matrix $\mathbf{W}$ such that

$$\mathbf{B} = \mathbf{Sgn}(\mathbf{WA}) \tag{2.2}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the $n \times p$ and $m \times p$ matrices whose columns are the vectors $\mathbf{a}^k$ and $\mathbf{b}^k$.

If we consider a particular SLN with $n = m$ and if we connect each output to one different input such that after the computation of an output $\mathbf{b}$, $\mathbf{b}$ is the new input, we get a recurrent fully connected synchronous network called a *Hopfield network* (HN). So the evolution of an HN is a temporal process and the state at a time t is defined by

$$\mathbf{a}(t) = \mathbf{sgn}[\mathbf{Wa}(t-1)] \tag{2.3}$$

A usual variation of the HN model results from a threshold function sgn^*, which differs from sgn only when the membrane potential is 0:

$$sgn^*(v_i, a_i) = \begin{cases} -1 & \text{if } v_i < 0 \\ a_i & \text{if } v_i = 0 \\ 1 & \text{if } v_i > 0 \end{cases}$$

In general, this difference is not relevant because the probability of v_i to be 0 is small, but with some learning rules it may not be the case. In this HN model, the activation rule sgn^* is symmetric and then $\mathbf{a}' = \mathbf{sgn}^*(\mathbf{Wa})$ iff $-\mathbf{a}' = \mathbf{sgn}^*[\mathbf{W}(-a)]$.

The HN can be used to store a set of vectors $\{\mathbf{a}^1, \ldots, \mathbf{a}^p\}$ as stable points of its dynamic. The learning of such a content-addressable memory (CAM) consists in determining $\mathbf{W}$ such that

$$\mathbf{A} = \mathbf{Sgn}(\mathbf{WA}) \tag{2.4}$$

As mentioned above, ANN working as memories presents error-correcting abilities, that is, if $\mathbf{a}$ is an input vector such that $H(\mathbf{a}, \mathbf{a}^k)$[1] is small, we can hope that the output of the SLN will be $\mathbf{b}^k$ for the input $\mathbf{a}$. Similarly, if we set an HN in an initial state $\mathbf{a}(0) = \mathbf{a}$, it will converge after a finite number of steps t on the state $\mathbf{a}(t) = \mathbf{a}^k$. The *basin of attraction* of a prototype input $\mathbf{a}^k$ is defined as the set of input $\mathbf{a}$ such that for this input, the SLN output is $\mathbf{b}^k$ and the HN stability point is $\mathbf{a}^k$. In the recurrent case, we define the *direct basin* of a prototype input $\mathbf{a}^k$ as the set of inputs $\mathbf{a}$ such that $\mathbf{a}(1) = \mathbf{a}^k$. When studying the direct basin only, an HN can be considered as a particular SLN. From now on, SLN will be considered and m will be taken to be equal to 1 since each output neuron is independent. $\mathbf{w}$ and $\mathbf{b}$ will denote the row vector corresponding respectively to the unique line of $\mathbf{W}$ and $\mathbf{B}$.

Let ρ^k be the maximum radius of a ball centered on $\mathbf{a}^k$ and contained in the basin of attraction of $\mathbf{a}^k$. Then a good measure of the error-correcting capability of a memory is also given by $\rho = \min_k \rho^k$. From the definition of sgn and sgn^*, it is clear that the behavior of the network does not depend on linear normalization of $\mathbf{w}$. The *unsigned membrane potential* u^k defined as $v^k b^k$ for an association k, is an interesting measure of the attraction power of the input $\mathbf{a}^k$. The output is correct for the input $\mathbf{a}^k$ iff $u^k \geq_* 0$. Moreover,[2] if $u^k \geq_* 0$ and if $\mathbf{w}$ is normalized, for instance, $\max_j |w_j| = \bar{w}$, we have[3]

$$\lfloor \frac{u^k}{2\bar{w}} \rfloor_* \leq \rho^k \tag{2.5}$$

For a given association k, define ω_j^k as $w_j a_j^k b^k$; then the tightness of the lower bounds for ρ^k given by (2.5) depends on the number of ω_j^k closed on $\bar{w}$. If we reorder the input units such that $\omega_1^k > \ldots > \omega_n^k$, we can compute ρ^k exactly:

$$\rho^k = \max \left\{ r | u^k - 2 \sum_{j=1}^{r} \omega_j^k \geq_* 0 \right\} \tag{2.6}$$

and if $u^k < 0$ then ρ^k is arbitrarily set to -1.

When the ω_j^k are in increasing order, the same right expression of (2.6) gives the largest radius λ^k of the basin of attraction of the input $\mathbf{a}^k$. $\bar{w}$ and the arithmetic mean of the $|w_j|$ will give an idea of the size of the basins of attraction outside the direct ball of radius $u^k/2\bar{w}$.

The operation performed by each output neuron can be interpreted geometrically as a hyperplane determined by $\mathbf{w}$ and separating the input space $\mathcal{R}^n$ in two parts such that an input $\mathbf{a}$ will give the output $+1$ iff it lies in the first half-space. The aim of each learning rule is to determine a hyperplane that classifies correctly all the input vectors. In the following section, I will briefly describe the learning rules that will be compared in the last section.

[1] $H : \{-1, 1\}^n \times \{-1, 1\}^n \to \{0, \ldots, n\}$ denotes the Hamming distance.

[2] The symbol $\geq_*$ must be interpreted as $>$ if the activation function is sgn and as $\geq$ if the activation function is sgn^*.

[3] The expression $\lfloor x \rfloor_*$ represents the biggest integer i such that $-i \geq_* 0$; in other words, it means either $\lceil x \rceil - 1$ or $\lfloor x \rfloor$ when the activation function is sgn or sgn^* respectively.

Many references can be found in the literature for theoretical properties of these rules [3,6,9,11–14,16].

3. Learning rules

The first rule I will consider is based on the correlations of each input with the output through every prototype association k and can be expressed simply as

$$\mathbf{w} = \mathbf{b}\mathbf{A}^{\top} \tag{3.1}$$

This well-known rule, referred to as the *Hebb rule* [5], is certainly the simplest and the most studied. The Hebb rule is based on the correlations between pairs of input vectors and on the rank of $\mathbf{A}$. In the particular case where all inputs are orthogonal, we have

$$\lfloor \frac{n}{2p} \rfloor_* \leq \rho \tag{3.2}$$

We find in [15] a proof for $\lceil n/2p \rceil - 1 \leq \rho$ for the activation function sgn^*, but the result of (3.2) can be directly deduced from (2.5) if we note that from (3.1) we have $\bar{w} \leq p$ and $u^k = \sum_j w_j a_j^k b^k = \sum_{jl} b^l a_j^l a_j^k b^k = n$, when the $\mathbf{a}^k$ are orthogonal. In the general situation, $n/2r$ — where r is the rank of $\mathbf{A}$ — is a good approximation of a lower bound for ρ.

A generalization of the Hebb rule is the *pseudo-inverse rule* [15]

$$\mathbf{w} = \mathbf{b}\mathbf{A}^{\dagger} \tag{3.3}$$

where $\mathbf{A}^{\dagger}$ is the pseudo-inverse or Moore–Penrose inverse of $\mathbf{A}$ [1]. The matrix $\mathbf{A}^{\dagger}$ can be computed in $O(nmp)$ by the iterative Greville algorithm. The Moore–Penrose inverse $\mathbf{A}^{\dagger}$ is defined such that $\mathbf{x}^{\top} = \mathbf{y}^{\top}\mathbf{A}^{\dagger}$ is the vector that minimizes $\|\mathbf{x}^{\top}\mathbf{A} - \mathbf{y}^{\top}\|$. So $\mathbf{w} = \mathbf{b}\mathbf{A}^{\dagger}$ is the matrix that minimizes the error $\|\mathbf{w}\mathbf{A} - \mathbf{b}\|$ on the output unit. In the case of an HN, $\mathbf{W} = \mathbf{A}\mathbf{A}^{\dagger}$ is the projection matrix on the space generated by the input vectors $\mathbf{a}^k$ and can be computed in $O(n^2p)$ by adding each input vector iteratively, according to

$$\mathbf{W}(k) = \mathbf{W}(k-1) + \frac{\mathbf{d}^k \mathbf{d}^{k^{\top}}}{\|\mathbf{d}^k\|^2} \qquad k = 1, \ldots, p$$

where $\mathbf{W}(0) = 0$ and $\mathbf{d}^k = \mathbf{a}^k - \mathbf{W}(k-1)\mathbf{a}^k$.

The Hebb rule and the pseudo-inverse rule are direct rules since they give an expression for $\mathbf{W}$ independently of the previous state of the synaptic matrix. However, there are iterative rules that define a temporal learning process, such as the famous *perceptron rule* [13]

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t(\mathbf{b} - \mathrm{sgn}[\mathbf{w}(t)\mathbf{A}])\mathbf{A}^{\top} \tag{3.4}$$

or the *adaline rule*, also called *delta rule* or *Widrow–Hoff rule* [18]

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t(\mathbf{b} - \mathbf{w}(t)\mathbf{A})\mathbf{A}^\top \tag{3.5}$$

These formulas correspond to the case where the weights are adjusted after a complete cycle through the learning set. Note that $\mathbf{w}$ can also be updated after presentation of every single association.

The advantage of iterative rules with respect to direct rules is that they can preserve some already memorized informations while learning some new patterns. Nevertheless, I will not present results about these two rules because the adaline rule produces its best solution on the convergent point, which is $\mathbf{w} = \mathbf{b}\mathbf{A}^\dagger$ — with a sequence of learning speed decrease like $\eta_t = 1/t$ — and the perceptron rule is incomparable with the others when the goal is robustness on input errors, because it stops to modify the synaptic weights as soon as the output is correct, even if there is no power of attraction (moreover, it does not converge if the inputs are not separable).

The last rule I will examine in this paper was proposed by Krauth and Mézard in 1987 [10]; it tries to maximize the stability of every association k. Because u^k gives a lower bound for ρ^k (equation 2.5) when $\bar{w}$ is fixed, Krauth and Mézard propose to maximize $u = \min_k u^k$ and choose a linear normalization of the weights that reduce the problem to a linear program. This gives naturally the optimal ρ determined by

$$\lfloor \frac{u}{2\bar{w}} \rfloor_* \leq \rho \tag{3.6}$$

A second interesting advantage of this rule is that when $\alpha = p/n$ is not too large, most of the w_j are in $\{-\bar{w}, 0, \bar{w}\}$. This empirical observation can be explained by the fact that the linear program is composed of n constraints $-\bar{w} \leq w_j \leq \bar{w}$ and p constraints $u^k \geq u$, therefore the optimal solution $(w_1, \ldots, w_n, u)$, which is a vertex of the polyhedron drawn by these $n+p$ constraints, satisfies at least $n+1-p$ equations $w_j = \pm\bar{w}$. This suggests that in some particular cases, with a quantization of the weights, the performances of the network could remain quite interesting [17].

4. Simulation results

In this section, a selection of numerous simulations of learning in neural networks will be presented. I will focus essentially on their sensitivity to input noise when they are used as CAMs or as pattern associators.

The first two figures illustrate the average power of attraction of the prototype inputs by plotting an approximation of the fraction of inputs that are at a given distance of a prototype input and that belong to its basin of attraction. To perform this, for each input $\mathbf{a}^k$ and for each radius r in $\{0, \ldots, 20\}$, 1000 patterns were randomly chosen on the orbit at distance r of $\mathbf{a}^k$ (if there are less than 1000 elements on the orbit, all of them are taken), and tested. Then, for each radius, the mean value, the maximum and minimum values on every prototype, are stored and this operation is

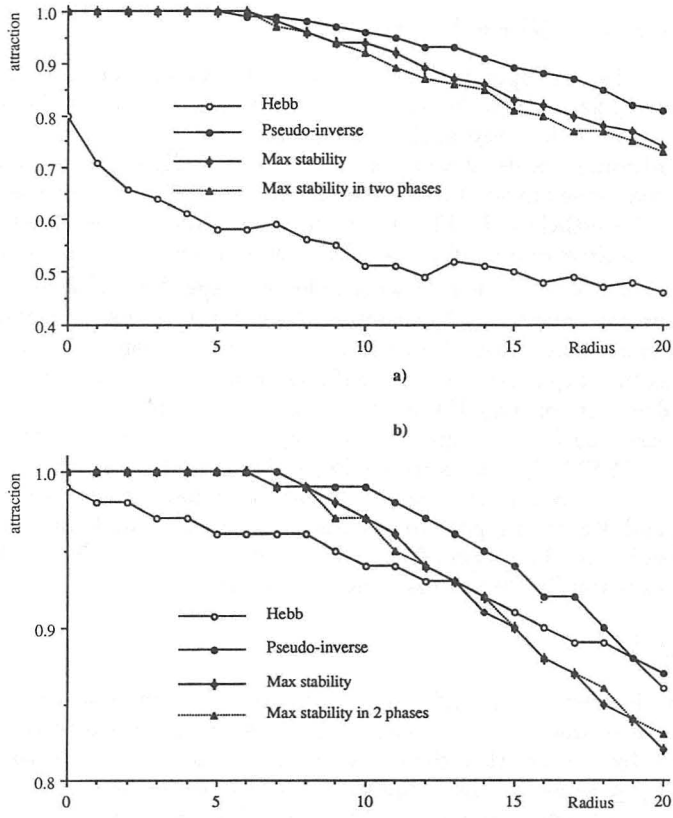

Figure 1: Part of attracted inputs on each orbit around the prototype inputs, with $n = 81$ and $p = 18$. In (a), the minimum value over each prototype is plotted (b) the average value is shown.

repeated 20 times for different random matrices $\mathbf{A}$ in which each a_j^k take the value -1 or $+1$ with equal probability. Figures 1a and 1b show the curves of minimum and mean values respectively for $n = 81$ and $p = 18$.

A first remark is that the curves of the Hebb rule are quite different in the two figures and this suggests that the Hebb rule can privilege some prototypes of a problem and neglect some others. This fact is clear in figures 2a and 2b, which display the minimum and the maximum values respectively for a much harder problem ($\alpha = 35/63 = 0.56$ instead of $18/81 = 0.22$). On one hand, the curve of the Hebb rule is the constant 0; this means that in each of the 20 problems, at least one of the prototype pair was not correctly associated by the Hebb rule. On the other hand, there is always one input $\mathbf{a}^k$ that

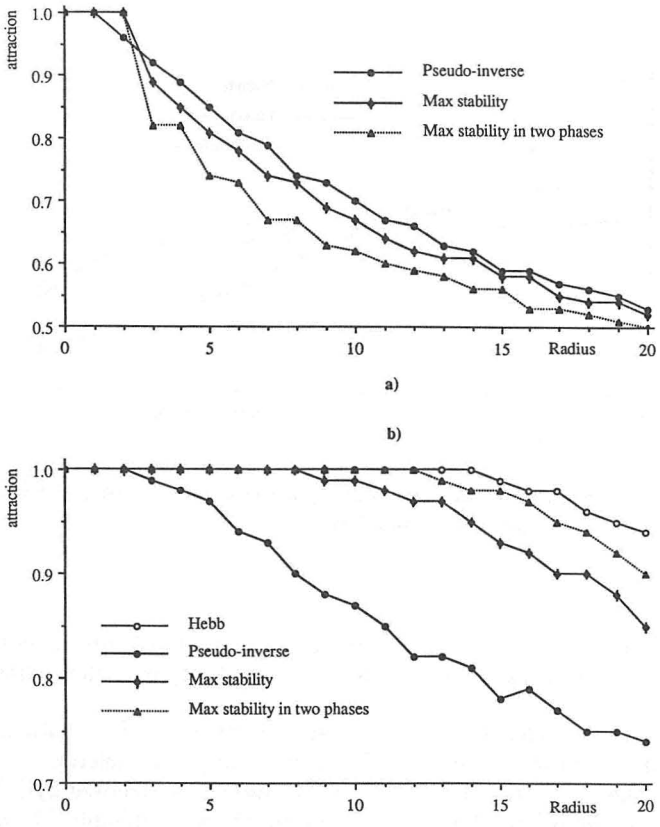

Figure 2: Part of attracted inputs on each orbit around the prototype inputs, with $n = 63$ and $p = 35$. (a) and (b) present respectively the minimum and maximum value over each prototype.

contains a ball of radius 14 in its basin of attraction. On the contrary, with the pseudo-inverse rule, the basins of every prototype are nearly identical.

The curves corresponding to the pseudo-inverse and the maximum-stability rule are very similar. Most of the time, the pseudo-inverse rule produces a ρ as good as the one given by the maximum-stability rule, and usually, further this value ρ, pseudo-inverse is a bit better than maximum-stability. Therefore I experiment with a rule referred to as *maximum-stability in two phases*, which works as follows: It first maximizes u while w_j is in the interval $[-\bar{w}, \bar{w}]$; then, if $\hat{u}$ is the optimum and if it is not a multiple of $2\bar{w}$, it relaxes $\hat{u}$ to $\tilde{u}$, the nearest smaller multiple of $2\bar{w}$, and tries to maximize $\sum_k u^k$ while

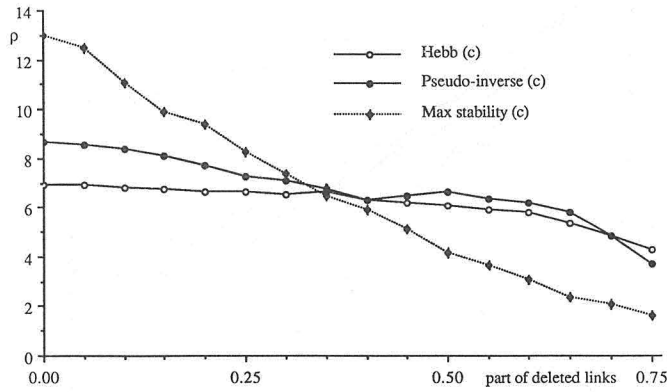

Figure 3: ρ is plotted against the proportion of deleted weights when the smallest weights are chosen first.

keeping each u^k bigger than $\tilde{u}$. In fact, as can be seen in figures 1 and 2, this second phase has no positive effect on the attractivity over the radius.

Let us now consider the network robustness in terms of synaptic destruction. I will present results of two different types of weight deletion: one when weights of the smallest absolute value are deleted first (denoted by (c) in the legends) and the other when the weights are chosen randomly (denoted by (r)). To estimate the evolution of the associative properties of the network during the deletion operation, I will look at the variation of ρ. Figure 3 presents the variations of ρ during deletion of the first type, for a network of 85 neurons an a learning set of 5 patterns. Once more, the tests are repeated 20 times.

Note that the learning rules that take into account the correlation between the input vectors tend to afford higher deletion rate without a sensible memory degradation. Figure 4 shows typical examples of both types of destruction. The first curves (4a) present the average of ρ^k for $n = 85$ and $p = 25$ and the second (4b) show ρ for a more difficult problem $n = 85$ and $p = 45$. It is not surprising that the network capabilities persist longer when the smallest weights are deleted first than when they are randomly chosen.

The main weakness of the Hebb rule lies in the fact that a weight w_{ij} depends only on the activations of the input unit j and the output unit i. This guarantee locality but may lead to undesirable situations. Let consider,

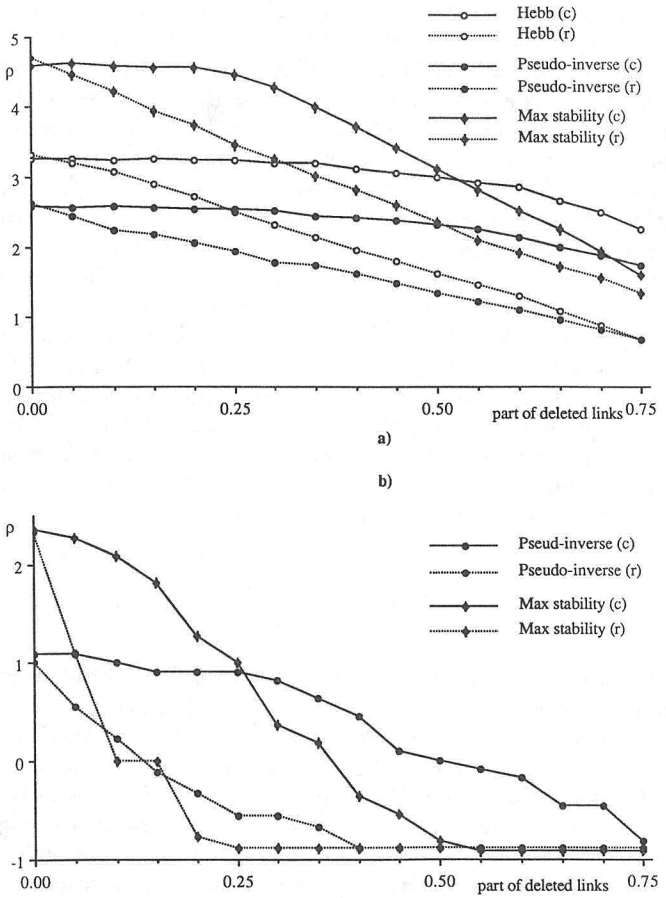

Figure 4: ρ is plotted against the proportion of deleted weights. The continuous lines (c) correspond to a deletion where the smallest weights are chosen first, and the dashed lines (r) to a random deletion. In (a) the average of ρ^k is shown for $n = 85$, $p = 25$ and in (b) ρ is plotted for $n = 85$, $p = 45$.

for example, an HN with $n = 8$ and the two following problems $\mathbf{A}$ and $\mathbf{A}'$

$$\mathbf{A} = \begin{pmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 \\ +1 & -1 & -1 & +1 \end{pmatrix} \qquad \mathbf{A}' = \begin{pmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & -1 \\ +1 & +1 & -1 & +1 \\ +1 & -1 & +1 & +1 \\ +1 & -1 & -1 & -1 \end{pmatrix}$$

The two ranks are 4, and $H(\mathbf{a}^k, \mathbf{a}^l) = H(\mathbf{a}'^s, \mathbf{a}'^t) = 2$ for every $k \neq l$ and $s \neq t$; nevertheless, the Hebb rule stabilizes every vector $\mathbf{a}^k$ and none of the $\mathbf{a}'^s$. However, this characteristic of the Hebb rule has above all the following inconvenience: it cannot take into account a particular architecture. To illustrate adaptability of the pseudo-inverse and the maximization of stability rules, some simulations were carried out where synapses were deleted before learning. Unlike the Hebb rule, these two rules will set the synaptic weights differently and adapt them to the new problem. Figure 5 presents 3 plots of averages over 20 problems with $n = 85$ and respectively $p = 5$, 25, 45. Identically to figures 3 and 4b, each plot shows ρ against the part of deleted links. For each of the two rules, there are two curves (r) and (b) corresponding respectively to a random deletion after and before learning. There is a clear difference between curves (r) and (b), and it increases with the ratio α.

To study how the associative properties of the network depend on the coding of the weights, an algorithm based on tabu search [4] has been developed for maximizing the stability of the input vectors when the weights take one of three values $\{-1, 0, +1\}$. This algorithm starts from the continuous solution obtained when $\bar{w} = 1$, rounded according to

$$w_j = \begin{cases} -1 & \text{if } w_j < -\frac{1}{3} \\ 0 & \text{if } |w_j| \leq \frac{1}{3} \\ 1 & \text{if } w_j > \frac{1}{3} \end{cases}$$

and search a better solution, moving at each step one weight from 0 to ± 1 or from ± 1 to 0, and using a tabu list of moves of length $n/2$.

For the sake of comparison with the case of binary weights $\{-1, +1\}$ studied in [2], experiments were done with same values of n and α. Figure 6a, b, c, and d plots u against α for, respectively, $n = 27$, 45, 63, 81. The first curve corresponds to result of linear program with continuous weights. The second curve shows the best solution obtained with tabu search, after 500 steps without improvement, starting from the rounded continuous solution. The third curve is the result obtained by E. Amaldi and S. Nicolis with the following adequate normalization $u\sqrt{n}$. According to the information theory, the best u obtained with ternary weights is always between the value obtained in the two other cases. Moreover, when n is increasing, the curve for ternary weights comes nearer the curve for binary weights.

5. Conclusion and further research themes

A large panoply of well-known learning algorithms is proposed today to the software designer who decides to use a neural network model for some CAMs. If he is interested in rapidity, in robustness, or in the adaptability to modifications of the problem, he will easily choose in this set the most suitable algorithm for his application. However, the hardware constructor who wants to build a network for some applications faces some complex choices as the number of bits to be used for the representation of the model's parameters. The last figure and related comments suggest that for a low capacity α, a

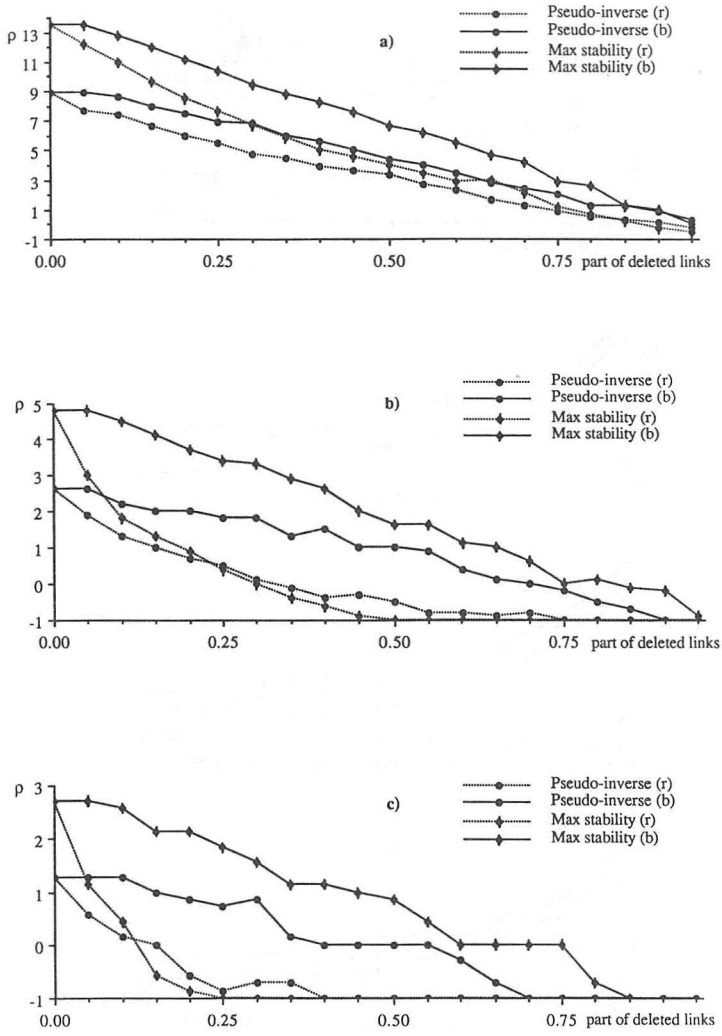

Figure 5: ρ is plotted against the proportion of deleted weights. The dashed lines (r) correspond to a random post-learning deletion, and the continuous lines (b) to a random pre-learning deletion. $n = 85$ and in (a), (b), and (c), $p = 5$, 25, and 45, respectively.

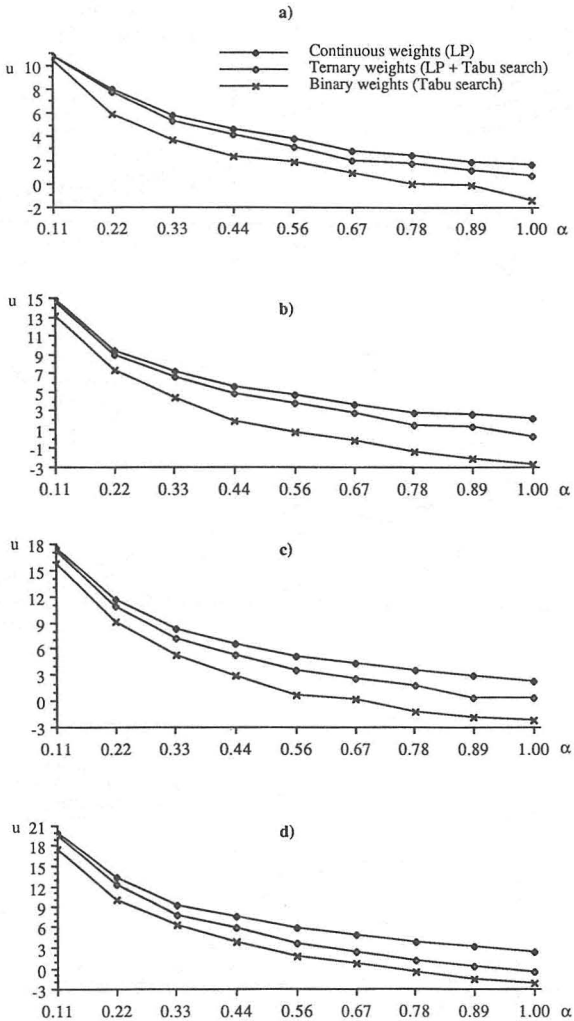

Figure 6: The maximum stability u is plotted against the ratio α. The three curves correspond to the cases of continuous, ternary, and binary synaptic weights. $n = 27$, 45, 63, and 81 in (a), (b), (c), and (d), respectively.

quantization of w_j on a few values does not affect considerably the capabilities of the network. Actual research lacks some theoretical results that could be used as guide lines for these people. As is well known in combinatorial optimization, the quantization of the search space leads usually to harder problems (cf. linear programming). Therefore, some specific algorithms have to be developed for learning in such a network type.

Acknowledgments

Many thanks are due to E. Amaldi for his expert advice and his always constructive remarks and to D. de Werra for many helpful discussions. Supported by Grant 20-5637.88 of the Swiss National Science Foundation.

References

[1] A. Albert, *Regression and the Moore–Penrose Pseudoinverse* (Academic Press, New York, 1972).

[2] E. Amaldi and S. Nicolis, "Stability-capacity diagram of a neural network with Ising bonds," *J. Physique*, **50** (1989) 2333–2345.

[3] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).

[4] F. Glover, "Tabu Search, Part I," *ORSA J. Computing*, **1(3)** (1989) 190–206.

[5] D.O. Hebb, *The Organization of Behavior* (Wiley, New York, 1949).

[6] G.E. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, **40** 185–234.

[7] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc Natl Acad Sci USA*. **79** (1982) 2554–2558.

[8] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci. USA*, **81** (1984) 3088–3092.

[9] T. Kohonen, *Self-organization and Associative Memory*, 2nd Ed. (Springer-Verlag, New York, 1988).

[10] W. Krauth and M. Mézard, "Learning algorithms with optimal stability in neural networks," *J. Phys. A*, **22** (1986) 247–259.

[11] R.P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine* (April 1987).

[12] E. Mayoraz, *Introduction aux Modèles Mathématiques de Réseaux de Neurones Artificiels*, Tech. Rep. ORWP 9/07, Dept. of Mathematics, EPF-Lausanne, Switzerland (June 1989).

[13] M.L. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, 2nd Ed. (MIT Press, Cambridge, MA, 1988).

[14] P. Peretto, "An introduction to the modeling of neural networks," (in press).

[15] L. Personnaz, I. Guyon, and G. Dreyfus, "Collective computational properties of neural networks: New learning mechanisms," *Phys. Rev.* **A34(5)** (1986) 4217–4226.

[16] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, eds., *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Vols. I & II (MIT Press, Cambridge, MA, 1986).

[17] M. Verleysen and P. Jespers, "Implémentations VLSI analogiques de réseaux de neurones," *Proc. Intl. Conf. Artificial Neural Networks*, EPF-Lausanne, Switzerland (Presses Polytechniques Romandes, 1989) 280–289.

[18] B. Widrow and M.E. Hoff, "Adaptative switching circuits," *1960 IRE WESCON Convention Record*, 4 (1960) 96–104.