Computational Capabilities of Restricted Two Layered Perceptrons

Avner Priel, Marcelo Blatt, Tal Grossman* and Eytan Domany
Electronics Department, The Weizmann Institute of Science, Rehovot 76100, Israel

Ido Kanter
Department of Physics, Bar Ilan University, 52900 Ramat Gan, Israel
(September 2, 1993)

Abstract

We study the extent to which fixing the second layer weights reduces the capacity and generalization ability of a two-layer perceptron. Architectures with N inputs, K hidden units and a single output are considered, with both overlapping and non-overlapping receptive fields. We obtain from simulations one measure of the strength of a network - its critical capacity, α_c. Using the ansatz $\tau_{med} \propto (\alpha_c - \alpha)^{-2}$ to describe the manner in which the median learning time diverges as α_c is approached, we estimate α_c in a manner that does not depend on arbitrary impatience parameters. The $CHIR$ learning algorithm is used in our simulations. For $K = 3$ and overlapping receptive fields we show that the general machine is equivalent to the Committee with the same architecture. For $K = 5$ and the same connectivity the general machine is the union of four distinct networks with fixed second layer weights, of which the Committee is the one with the highest α_c. Since the capacity of the union of a finite set of machines equals that of the strongest constituent, the capacity of the general machine with $K = 5$ equals that of the Committee. We investigated the internal representations used by different machines, and found that high correlations between the hidden units and the output reduce the capacity. Finally we studied the Boolean functions that can be realized by networks with fixed second layer weights. We discovered that two different machines implement two completely distinct sets of Boolean functions.

Typeset using REVTEX

I. INTRODUCTION

Two - layer perceptrons (2LP) are some of the simplest neural networks [1]. Their operation is purely feed-forward: the "external world" sets the state of N inputs, which then determine the states of K hidden units. In the simplest 2LP the K hidden units dictate the state of a single output. Denoting the inputs by S_j^{in} with $j = 1, 2, ...N$, the states of the hidden units by $S_i, i = 1, 2, ...K$, and that of the output by S^{out}, the operation of the network is summarized by :

$$S^{out} = F\left(\sum_{i=1}^{K} w_i S_i\right) \tag{1}$$

$$S_i = F\left(\sum_{j=1}^{N} W_{ij} S_j^{in}\right) \tag{2}$$

where $F(x)$ is a non-linear function and w_i, W_{ij} are the weights, or connections of the network. Fig. 1,2 depict two basic architectures of such 2LP; those with overlaping receptive fields (ORF) shown in Fig 1, in which each hidden unit may be affected by all inputs, and networks with non-overlaping receptive fields (NRF) that have the tree-like architecture shown in Fig 2. We consider here networks with $K \ll N$. Despite their simplicity, such 2LP's are capable of implementing highly nontrivial classification tasks, since the function $S^{out}(S_1^{in}, S_2^{in}...S_N^{in})$ can be rather complex. In this paper we deal with the case of binary-valued input and output, i.e.

$$S^{out} = \pm 1; \quad S_j^{in} = \pm 1.$$

and use the function

$$F(x) = sign(x). \tag{3}$$

In this case the network implements a Boolean function. It is the *weights* w_i and W_{ij} that determine which Boolean function is implemented by the network described above.

Perhaps the most interesting aspect of neural networks is their capacity for adaptive behavior, or "learning" [1]. In the course of a training session a set of P examples, i.e. inputs for which the desired output is known, is presented to the network. Using its current "guess" for the weights, the network produces an output. Depending on whether this output does or does not agree with the correct one, the weights are modifed. The manner in which the weights change during training is dictated by a *learning algorithm*. Fairly efficient learning algorithms exist for continuous-valued S_i (Back-Prop. [1]) as well as for the binary case (MRII [1], CHIR [2,3]). Nevertheless, no learning algorithm for which a convergence theorem has been proved (for fixed architecture) is known (see however [4]). In general, learning is a computationally intensive task, with non-polynomial scaling with problem size [5]. The main difficulty in training a network is our ignorance of the correct states of the hidden units: that is, we do not know the correct "table" of *internal representations* (IR) associated with the $\mu = 1, 2, ...P$ examples. Had we known these, learning would have become straightforward. The difficulty to guess the IR would be alleviated (at least

partially) if we knew the correct values of the second-layer weights w_i. In the context of learning algorithms that minimize the training error over all weights, the w_i constitute a small subset of K out of the NK unknowns to be determined. On the other hand, when learning is viewed in the context of searching for good IR, the importance of w_i is clearly enhanced; when these K parameters are known, the set of IR that are consistent with the output is significantly reduced (even though the number of correct "tables" of IR is still exponential with P). Hence there is reason to believe that training is simplified considerably if the w_i are fixed, and only the W_{ij} are to be determined. This is expected to be the case for learning algorithms that are suitable for networks with binary-valued units, which are, in fact, based on executing a search in the space of possible IR [3]. Working with fixed w_i amounts to restricting the network so that a predetermined Boolean function is used from the hidden layer to the output.

Networks that use Boolean functions such as parity, majority (or committee), AND, etc. are called by the function used in this manner, e.g. parity machine, committee machine (CM), etc.

The main question addressed in this paper is whether one looses at all by restricting the network in this way, and to quantitatively measure this "loss". In other words - to what extent is a machine with a fixed set of second-layer weights weaker than the most general network with the same architecture ?

One measure of the strength of a network is the number of Boolean functions it is capable to implement. The effect of fixing w_i on this measure will be reported elsewhere (various aspects of the realizable Boolean functions are discussed in sec. 5 - see below). In this paper we use a different measure for the strength of such machines - that of their capacity for storing random input-output associations [6,7]. That is, one generates P random inputs, and assigns $S_\mu^{out} = \pm 1$ at random to each one. For small P a machine will be able to learn the examples, but when P exceeds a critical value,

$$P_c = \alpha_c N_{weights} \tag{4}$$

one expects that no set of weights can be found that implements all members of the training set. Note that $N_{weights}$, the number of adjustable weights, is N for NRF and $K \cdot N$ for ORF. Capacity serves as a measure of the strength of various machines. We have set out to first determine numerically the effect of fixing w_i on the capacity, and to try to understand the reason why some machines have a higher capacity than others.

Some of these issues were addressed previously both analytically and numerically. Mitchison and Durbin [8] used geometrical methods to determine upper bounds for the storage capacity of the parity and committee machines. They have shown that the maximal capacity per synapse (i.e. weight) is bounded for $K \to \infty$ by $\log_2 K$. This bound is violated by replica-symmetric solutions of the critical capacity derived for the parity machine with ORF [9] and NRF. For the latter architecture Barkai et. al [10] were able to show that the replica-symmetric solution (for which one finds $\alpha_c \sim K^2$) becomes unstable, before α_c is reached, against replica symmetry breaking. They also obtained the solution of the mean-field equations with one-step replica symmetry breaking and have shown that the corresponding $\alpha_c = A \log_2 K$, with $A \to 1$ when $K \to \infty$, saturating the Mitchison-Durbin bound. The capacity of the parity machine with NRF and *binary weights* was studied by Barkai and Kanter [11] . The replica-symmetric solution was found to be stable (at $T = 0$),

and the capacity saturates the theoretical bound $\alpha_c = 1$. The situation seems to be different for the AND machine, investigated by Griniasty and Grossman [12]. For this machine, with both ORF and NRF, the replica-symmetric solution seems to agree with numerical simulations and with bounds on the capacity.

Two recent papers [13,14] investigated the capacity of the CM. For the NRF tree architecture and $K = 3$ hidden units the replica-symmetric solution yields $\alpha_c \simeq 4.02$, whereas one-step replica symmetry breaking lowers this to $\alpha_c \simeq 3.0$. In the large K limit the replica-symmetric solution gives $\alpha_c \sim K^{\frac{1}{2}}$, violating the bounds obtained by adapting the arguments of Mitchison and Durbin to this case. Since a replica symmetry breaking instability appears already at finite α, one expects that a solution with broken replica symmetry takes over, and satisfies the (logarithmic) bound.

For the case of ORF not much analytic information has been obtained. In this case one expects breaking of permutation symmetry (i.e. different hidden units playing different roles, even though they are equivalent). This complicates the analysis considerably, so that the only available information regarding α_c is based on numerical simulations.

Turning now to numerical studies we first noted that no reliable results exist in the literature for the capacity of the *general machine* (i.e. non-restricted w_i.) As to machines with various fixed second layer Boolean functions, some studies do exist. The capacity was estimated for the parity machine with NRF [10] and for the $K = 2$ AND machine [12].

Simulations of the CM for various values of K were performed by both Barkai et. al [13] and Engel et. al [14]. For a given architecture $P = \alpha N_{weights}$ random (binary) input patterns are generated, together with their desired (random) outputs. This training set is presented sequentially to the network, whose weights are modified according to the learning algorithm used [8,15]. The network can learn the entire training set with probability $f(\alpha)$. For $N \to \infty$, f is believed to be a step-function, with $f = 1$ for $\alpha < \alpha_c$, and $f = 0$ when α exceeds the critical value. For finite N, $f(\alpha)$ interpolates smoothly between 1 and 0 as α increases. As α_c is approached from below, the learning task becomes either impossible or possible but difficult; in the latter case the volume of solutions in the space of weights decreases towards zero, and hence the learning time needed for any learning algorithm to find a solution becomes exceedingly large. Therefore in the interesting regime, of $\alpha \approx \alpha_c$, determination of f is a computationally intensive task. Obtaining f is severely hindered by the fact that for multilayered networks there is no algorithm that either converges when a solution exists or signals when the problem is insoluble. Hence in order to obtain f from simulations, both groups use some arbitrary impatience parameter to decide when the current task is unlearnable. Barkai et. al [13] stop training and declare the task unsolved when the number of learning steps exceeds a preset value. Engel et. al [14] stop when the change observed in the weights, accumulated over an (arbitraryly chosen) number of learning steps, does not exceed some small preset parameter. The estimates for $f(\alpha)$ obtained in this manner clearly depend on the learning algorithm used and on the arbitrarily chosen halting criteria. Both groups find α_c by estimating the value of α at which $f = \frac{1}{2}$. Therefore these estimates of α_c may be strongly dependent on the details mentioned above.

Comparison of numerical results to analytically obtained values of α_c is complicated further by the need to extrapolate from finite N (used in numerics) to $N \to \infty$ (in which limit the analytic results are obtained). In the case where analytic results are available, the numerical estimates for the capacity obtained by both groups are significantly below the

analytic α_c. We demonstrated that by using a more efficient learning algorithm [3] one can obtain higher $f(\alpha)$ (using the same halting criteria). This indicates that the apparent lack of agreement between simulations and theory may have been due to getting impatient too early, and not to problems of extrapolation to large N.

One of the aims of this work is to determine α_c reliably, in a universal manner, independent of the particular algorithm and halting criteria used. We combine two methods that were used previously. The first is that of measuring for every task τ_{med}, the *median learning time* [2,3]. This is the learning time needed to solve the task in half of the attempted cases. The second aspect relies on extrapolating this quantity [10] from the low-α regime towards α_c. Details of how our method works and comparison of its results to previous studies are presented in Sec 2. Our numerical results are in good agreement with analytic work.

Our numerical results for general networks with $K = 3, 5$ are presented in Sec. 3, where capacities of various Boolean machines are also reported and discussed. For example, we found that the *general* $K = 5$ machine is equivalent to the union of *four* machines with fixed second layer weights. Capacity measurement reveals that not all four machines are of the same strength; we also show that the strongest of the four (the CM) determines, in the $N \to \infty$ limit, the capacity of their union (and hence - of the general $K = 5$ machine). We believe that this is a general result (that has been discussed previously for NRF architecture only [14]); i.e. the capacity of a general 2LP equals that of the CM of the same architecture.

In Sec. 4 we discuss factors that seem to determine the capacity of different Boolean machines. We found that higher correlation between states of the hidden units and the output give rise to lower capacity. Analytic and numerical estimates of these correlations are presented. We also study the manner in which various machines select, during learning, internal representations in a manner that increase their capacity.

In sec. 5 we turn to study the Boolean functions that a machine with fixed second-layer weights can realize. Our main (and perhaps surprising) finding is that two sets of Boolean functions that can be implemented by two such machines are either identical or completely distinct (with not even a single common Boolean function).

II. CAPACITY MEASUREMENTS – THE STRATEGY

We now describe in detail our method for measuring the capacity of MLP's. We set as our aim to develop a *universal* method, i.e. one whose results are independent of arbitrary impatience parameters and halting criteria. The first difference between our work and that of [13,14] is the choice of the learning algorithm. Whereas both Barkai et. al [13] and Engel et. al [14] used versions of the least action algorithm [15], we used a more efficient method, *CHIR2*, adapted to the problem of MLP with fixed second-layer weights. Since this algorithm has been presented elsewhere [3], we give here only a brief description for completeness' sake (see Appendix B). We used several versions of the basic algorithm. These versions differ from one another in the manner in which the hidden unit whose state is modified is selected when the network produces a wrong output during training. It is important to emphasize that usage of a superior learning algorithm is *not* the main point of this discussion; our approach can be implemented with any other technique.

Since we did wish to determine the relative efficiency of our algorithm, we repeated the calculations of [13] for the committee machine (CM) with both ORF and NRF architectures.

For $N = 150, 300$ and 450 inputs and $K = 3$ hidden units we set the same upper bound, of 6000 sweeps of the training set, which contained $P = \alpha N_{weights}$ random binary patterns. We determined $f(\alpha)$, the fraction of such learning sessions in which the *complete* training set was learned. The data points presented in Fig. 3 were obtained by performing for each case 300 experiments. This Figure contains also the results of [13] for comparison. Clearly, with the same halting criterion, CHIR succeeds to learn larger training sets than the least action algorithm. Therefore using $f(\alpha_c) = 0.5$ (obtained from the largest size studied) to determine the critical capacity, CHIR provides an estimate $\alpha_c \approx 2.65$ for NRF and $\alpha_c \approx 3.0$ for ORF which is about 10% higher than that of Barkai et. al [13]. The result of Engel et. al [14] , of $\alpha_c \approx 2$ for NRF, is too low to be included in the Figure.

In order to eliminate the dependence of α_c on either the learning algorithm or the halting criterion, we opted for a different estimator of the critical capacity. The idea is to measure the *median learning time* of a sample of training sets. This is done by recording the time (i.e. number of sweeps of the P training patterns) needed to learn the complete training set. Many such training sets are generated and the corresponding learning times are ordered; τ_{med} is the median of this ordered sequence. Clearly those cases in which the network failed to learn the training set (within a preset maximal number of τ_{max} sessions) are placed at the high end of the sequence. We start with a small sample of training sets to get a rough estimate of τ_{med}; choose τ_{max} as twice this estimate, and then try to learn a large sample. Clearly, as long as we have $f(\alpha) \geq 0.5$, this method yields finite estimates of τ_{med}, for whichever learning algorithm is used.

The median learning time was found to be stable statistically, with small errors in its measurement. In this respect working with τ_{med} is better than with τ_{avg} the *average* learning time (averaged over successfull experiments). This statistic is strongly affected by the long-time tail of the distribution of learning times, and therefore has large statistical fluctuations.

The second and crucial ingredient of our method is based on a scaling *ansatz* [10] for the manner in which τ_{med} diverges as α_c is approached from below. For large enough α the fraction of learnable training sets is less than $\frac{1}{2}$, and therefore the median learning time is infinite (even for finite N). We assumed a power-law divergence of the form

$$\tau_{med} \propto (\alpha_c - \alpha)^{-a} \tag{5}$$

and found that the choice $a = 2$, found in a variety of networks, fits our data quite well, allowing a fairly reliable extrapolation of the measured values to

$$\tau_{med}^{-\frac{1}{2}} = 0 \tag{6}$$

Such extrapolations are presented in the next Section.

III. CAPACITIES OF TWO-LAYER PERCEPTRONS: ANALYSIS AND SIMULATIONS

All our analysis and numerical simulations were done on networks with zero valued threshold for all units, with the *sign* function (eq. 3). Say such a unit receives N' *binary* inputs. All the possible inputs can then be represented as the corners of a hypercube in N' dimensions. The weights entering our unit define a hyperplane that passes through the

origin of this space. In particular, the possible IR of our 2LP form the vertices of a K dimensional hypercube. We turn now to our simplest and smallest 2LP, with $K = 3$.

A. Two-layer perceptrons with three hidden units

We start with the ORF CM with $K = 3$. This machine is completely equivalent to the *general* 2LP with $K = 3$. By this we mean that the CM is capable of implementing the same Boolean functions of the inputs as the machine with no restrictions on its second-layer weights. We now prove this equivalence.

Consider the ORF network of Fig. 1, with $K = 3$ hidden units. Any Boolean function implemented by such a network maps the inputs with assignement $S^{out} = +1$ onto one subset of the possible internal representations, and the inputs for which $S^{out} = -1$ onto another subset. Therefore any network with this architecture has to be based on a dichotomy of the space of IR. Since the output unit has zero threshold, there are only two basic planar dichotomies of this space (recall that we use zero threshold units, hence only planes that go through the origin are used). These two dichotomies, D_1 and D_2, are depicted in Fig. 4; full (open) circles represent those IR which are mapped to $S^{out} = +1$ (-1). All other possible dichotomies are equivalent to either D_1 or D_2: they can be mapped to one of these two by permuting the three hidden units, or by redefining $S_i \to -S_i$ for one or more of the three.

Clearly, if the general network uses the dichotomy D_2, it implements precisely the same Boolean function as the CM with identical W_{ij}. If the general network uses D_2', say, we construct a corresponding CM by setting its weights W'_{ij} as follows:

$$W'_{1j} = W_{1j} \quad W'_{2j} = W_{2j} \quad W'_{3j} = -W_{3j}$$

This change of weights induces a reflection $S_3 \to -S_3$, and therefore if the machine with weights W_{ij} maps input space onto IR according to the dichotomy D_2', the machine with weights W'_{ij} uses D_2, and hence the Boolean function implemented by the general network can be realized also by a CM.

The general machine can use also the dichotomy D_1. A dichotomy of this class means that the state of the output unit is determined by *a single* hidden unit; we call such a 2LP a Ruler machine. For example in the case D_1 of Fig. 4 we have $S^{out} = S_3$. In this case we define our CM as follows:

$$W'_{ij} = W_{3j}$$

for $i = 1, 2, 3$. With this choice all inputs that were mapped onto the full circled corners in D_1 get mapped onto $(1, 1, 1)$, whereas the other inputs get mapped onto $(-1, -1, -1)$. This is a particular case of D_1, and hence the CM with weights W'_{ij} implements the same Boolean function as the general machine with W_{ij}. Having exhausted all the possible general networks, we have proved that indeed for $K = 3$ all networks can be replaced by an equivalent CM. Note that this equivalent CM does not have to use the same IR as the general 2LP; the IR space can be remapped in order to realize the same Boolean function.

We performed simulations of the CM with ORF and $K = 3$, with $N = 50, 100, 150$ inputs. The simulations were done using version 1 of the $CHIR$ algorithm described in Appendix B, with $\mu = 0$, $\kappa = 1.0$. In all cases we chose the time limit in a way that ensured

that more than 50% of the cases were solved. 300 training sets were generated randomly for every value of α, and used to determine the median learning time, τ_{med}. In Fig. 5 we present $\tau_{med}^{-\frac{1}{2}}$ vs. α. There is very little difference between the data for $N = 100$ and 150; our results extrapolate to $\alpha_c \approx 3.05$.

The equivalence between a general 2LP and the CM does not hold for the NRF architecture with $K = 3$. The reason is that in this case the three hidden units cannot be permuted. It is easiest to disprove equivalence by assuming that a Ruler machine with $w_1 = 1$ and $w_2 = w_3 = 0$ is equivalent to the CM with $w_i' = 1$, and demonstrate a contradiction.

Denote by $\boldsymbol{\xi}=(\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$ an input vector that gets mapped onto $S^{out} = +1$ by both machines ($\boldsymbol{\xi}_i$ is the input vector to hidden unit i). This means that the CM maps this input onto one of four IR,

$$(S_1, S_2, S_3) = (+,+,+)\ ;\ (+,+,-)\ ;\ (+,-,+)\ ;\ (-,+,+)$$

If the first of these appears, consider the input vector $\boldsymbol{\xi}' = (\boldsymbol{\xi}_1, -\boldsymbol{\xi}_2, -\boldsymbol{\xi}_3)$. Since we did not change $\boldsymbol{\xi}_1$, this input gets mapped by the Ruler machine onto $S^{out} = +1$, but the CM maps it onto the IR $(+,-,-)$, and hence onto $S^{out} = -1$, and therefore the two machines are not equivalent. Similarly, for each of the other three IR of the CM that are listed above we can construct, by an appropriate sign change, an input vector that keeps producing output $+1$ for the Ruler machine and -1 for the CM. The key to this construction was our ability to change the input in a manner that keeps the input seen by one hidden unit the same while switching the sign of the inputs to the other two. This, of course, cannot be done for the ORF architecture, and for this reason the NRF architecture contains *two* types of machines, that are *not* equivalent; the Ruler machine (with its two permutations) and the CM, $w_i = 1$. The most general NRF 2LP with $K = 3$ can be represented as the union of these four "constituent" machines.

We now present a simple argument which shows that the *capacity of a general 2LP equals that of its constituent with maximal capacity.* To prove this statement we make the commonly accepted assumption that in the $N \to \infty$ limit the function $f(\alpha)$ discussed in the introduction is a step-function for all 2LP's. Denote by $\alpha_c^{(i)}$ the capacity of the i'th constituent of our general 2LP, whose capacity is α_c. Clearly $\alpha_c \geq \alpha_c^{(i)}$. Assume that α_c is larger than all $\alpha_c^{(i)}$. This means that the general 2LP can store, with probability 1, αN patterns, while *none* of its constituents can store them. This is clearly a contradiction, since every particular realization of the general 2LP is equivalent to one of the constituent machines. Note that this argument holds as long as we have a finite number of constituent machines.

Using this argument, together with the previously proved identification of the two constituents of the general 2LP with $K = 3$ and NRF allows us to show that the general 2LP has the same capacity as the $K = 3$ NRF CM. All we have to show is that the CM has a larger capacity than the Ruler. The capacity of the Ruler is that of a single unit perceptron; it can store up to $P_c = 2N_1$ patterns, where N_1 is the number of the inputs seen by the ruling unit. Since $N_1 = N/3$ we have $\alpha_c^R = 2/3$, whereas for the CM $\alpha_c \approx 3$. Thus the capacity of the general 2LP with NRF also equals that of the CM of similar architecture.

We performed simulations of the NRF CM with $N = 150, 300, 450$ inputs. The learning algorithm was the same as the one used for ORF, and again the time limit was set to allow determination of τ_{med}. Fig. 6 presents the results for $\tau_{med}^{-\frac{1}{2}}$ vs α. Size dependence of the

results is again very insignificant, which indicates that our estimate of $\alpha_c \approx 2.75$, obtained from extrapolating to $\tau_{med}^{-\frac{1}{2}} = 0$, is probably valid for large N. The same value was obtained in [4] using the back-propagation algorithm. This value exceeds the numerical estimates obtained by [13,14], but is still lower than the capacity as given by their first-step replica symmetry breaking calculation. This discrepancy can be due either to a need for higher replica symmetry breaking, or signal a numerical difficulty in obtaining α_c from the one-step analytic calculation.

B. Two-layer perceptrons with five hidden units

The IR's of such a network form the vertices of a 5-dimensional hypercube. A plane that passes through the origin can generate *seven* different dichotomies (not related by permutations or sign inversions) of these vertices. Since this result was known for quite some time [16,17], we present only a sketch of the derivation in Appendix A. These seven dichotomies give rise to seven 2LP's, whose second-layer weights $w_i, i = 1, 2..5$ are the following:

$$10000 \ , \ 11100 \ , \ 11111 \ , \ 21110 \ , \ 31111 \ , \ 22111 \ , \ 32211$$

Since we are interested in 2LP's, we have some additional freedom (to remap the IR's) which can be used to reduce this set of independent w_i even further. Consider first a 2LP with first-layer weights W_{ij} and second layer weights $(w_i) = (10000)$. This is a Ruler machine which can be realized by a CM; simply set $W'_{ij} = W_{1j}$, i.e. copy the weights of the ruler unit onto all the others. Now we can use $(w'_i) = (11111)$; the resulting CM produces the same Boolean function as our initial Ruler machine.

Next, consider the 2LP with $(w_i) = 11100$, which ignores the states of units 4 and 5. We can again construct a CM by setting $W'_{ij} = W_{ij}$ for $i = 1, 2, 3, 4$ and $W'_{5j} = -W_{4j}$. This choice of weights ensures that the contributions of units 4 and 5 to the field seen by the CM's output precisely cancel one another, leaving the same output as was generated by the originally considered 2LP.

Finally we turn to the 2LP with $(w_i) = (21110)$. In this case the vote of hidden unit 1 gets weight 2, whereas the hidden unit 5 is ignored. Clearly, by setting $W'_{ij} = W_{ij}$ for $i = 1, ..4$ and $W'_{5j} = W_{1j}$ we generate a CM in which the vote of unit 1 is copied to unit 5 and hence, effectively doubled. This CM then implements the same Boolean function as the machine with 21110.

These simple considerations reduce to four the number of independent 2LP's with $K = 5$ and ORF. The union of these four constituent machines, whose second layer weights are

- 11111 - (the CM)
- 22111
- 31111
- 32211

is equivalent to the general 2LP with unrestricted weights. We performed simulations with $N = 25, 50, 100$. These are smaller sizes than those used for $K = 3$; finite size effects give rise to deviation of $f(\alpha)$ from the sharp step function expected for large N. Hence we first determined the capacity of the general machine by learning $P = \alpha N$ random patterns with the four machines *in parallel*. This was done by taking each learning step with every one of the machines, and stopping the process whenever any machine has learned the training set. The time it took the first machine to obtain a solution was tabulated as the learning time of the union. The same method was used to determine the capacity as before: Fig. 7 presents $\tau_{med}^{-\frac{1}{2}}$ vs α for $N = 25, 50$ and 100. Each data point was obtained from a sample of 300 training sets. In this case we do notice finite size effects: linear extrapolation yields $\alpha_c \approx 3.55$ for $N = 25$ and $\alpha_c \approx 3.4$ for $N = 50$; extrapolation for $N = 100$ gives the same value as for $N = 50$ suggesting that this is approximatly the value of α_c. These simulations were done using version 1 of $CHIR$ with $\mu = 0,\ \kappa = 4.0$ (see Appendix B).

Next we studied the four constituent machines separately. The results for $N = 25, 50$ and the versions used are presented in Fig. 8 and the corresponding extrapolations in Table 1 (except $N = 100$ for the union). The CM and the 22111 machine have the same capacities, with the CM possibly slightly higher; the capacity of 32211 is somewhat lower and that of 31111 significantly lower. We tried to understand the differences between the capacities of these machines, and present some of our findings in the next section.

For the sake of completeness we studied the $K = 5$ CM with NRF as well. Results for $\tau_{med}^{-\frac{1}{2}}$ vs. α are presented in Fig. 9 for $N = 125$ and 250. The critical capacity as found from extrapolation is $\alpha_c \approx 3.15$ for $N = 125$ and $\alpha_c \approx 3.08$ for $N = 250$.

IV. PROPERTIES OF IR'S THAT AFFECT THE CAPACITY OF 2LP

In this Section we describe briefly various factors that apparently influence the capacity of a 2LP with fixed second layer weights. A quantity which plays an important role is the average correlation between each hidden unit and the output unit. To see quantitatively how these correlations affect the capacity of the network, consider ,for instance, a CM with three hidden units. The four internal representations which induce an output $S^{out} = +1$ are $(+,+,+)$, $(-,+,+)$, $(+,-,+)$ and $(+,+,-)$. To simplify the following discussion we assume that all internal representations appear with equal probability. Under this assumption, one can easily verify that the network obeys the following average correlations:

$$\ll S^{out} S_i \gg = \frac{1}{2} \qquad i = 1, 2, 3 \tag{7}$$

$$\ll S^{out} S_1 S_2 S_3 \gg = -\frac{1}{2} \tag{8}$$

where S_i is the state of hidden unit i, and the symbol $\ll \quad \gg$ stands for an average over all internal representations with equal probability. Note that eqs. 7 and 8 are not sufficient conditions to identify the network as a CM. Consider now a network, denoted as $\mathcal{C}$, which on the average obeys only constraint (7). Since the average constraints satisfied by $\mathcal{C}$ are weaker than those imposed on the CM, it is clear that the maximal capacity of $\mathcal{C}$ serves as

an upper bound for the maximal capacity of the CM. In the following, this upper bound is calculated explicitly in the large K limit.

In the general case of a CM with K hidden units, the average probability that S_i differs from S^{out}, is given by

$$\ll \Theta(-S^{out}S_i) \gg = \frac{\sum_{l=\frac{K+1}{2}}^{K-1} \binom{K-1}{l}}{\sum_{l=\frac{K+1}{2}}^{K} \binom{K}{l}} \tag{9}$$

where the symbol $\ll \ \gg$ stands again for an average over all internal represenations with equal probability. The denominator is the total number of internal representations with output $+1$, for instance. The numerator has the same meaning as the denominator, but under the constraint that $S_i = -1$. In the large K limit one can verify that

$$\ll \Theta(-S^{out}S_i) \gg = \frac{1}{2} - \frac{1}{\sqrt{2\pi K}} \quad . \tag{10}$$

Each one of the hidden units of $\mathcal{C}$ with either ORF or NRF acts as a perceptron which embeds random input/output patterns with an average error defined in eq. 10. This problem was discussed by Gardner and Derrida [7], and from their results one deduces that the maximal capacity of each hidden unit (perceptron) is given by

$$\alpha_c \ = \ 3\,\sqrt{2\,\pi\,K^3} \quad . \tag{11}$$

Since the total number of weights in the CM is K times the number of weights which affect each hidden unit, the maximal capacity per weight of the CM is bounded by

$$\alpha_c \ = \ 3\,\sqrt{2\,\pi\,K} \quad . \tag{12}$$

Note that this upper bound for the maximal capacity of the CM, $\alpha_c \leq o(\sqrt{K})$, is identical to the maximal capacity obtained by exact calculations within the framework of replica symmetric solutions [13]. Furthermore, one can verify that any finite disturbance in the probability of appearence of each internal representation, does not affect the leading order of the upper bound for the maximal capacity, $o(\sqrt{K})$. The effect of higher order correlations, such as eq. 8, may affect the leading order of the upper bound for the maximal capacity to be within the right order, $\alpha_c \leq o(log(K))$ [8].

Beside the quantitative analytical result discussed above, the main point we can add at this time is that *the capacity of a 2LP is lower if the hidden units are more correlated with the output.* This statement is intuitively evident; i.e. in the extreme limit, of full correlation of the output with all hidden units, the 2LP becomes identical to a simple perceptron. Full correlation is observed also in the case of a single dominant ("Ruler") hidden unit.

Thus we identify two measures that are expected to affect capacity:

- average correlation of all hidden units with the output.
- maximal correlation of a hidden unit with the output.

We define the following quantity to be the correlation of hidden unit i with the output:

$$c_i = \langle S_i \cdot S_{out} \rangle \equiv \frac{1}{P} \sum_{\mu=1}^{P} S_i^\mu \, S_{out}^\mu \tag{13}$$

This quantity is evaluated when a set of P patterns have been learned. S_i^μ is the state of hidden unit i when pattern μ is presented.

Consider the task of random mapping of P random patterns onto $S^{out} = \pm 1$ in networks with ORF architecture. Using simple arguments, we can estimate the value of c_i for each machine. One can divide the IR's into groups according to their correlations with the output unit.

To see how this is done, we treat first the CM. In this case all hidden units are equivalent.

Consider the following vector :

$$\tilde{S} = (S_{out}S_1, S_{out}S_2, \ldots, S_{out}S_k) \tag{14}$$

For $K = 5$ hidden units, there are 16 such vectors, that can be divided into three classes :

$$\tilde{S} = (+\,+\,+\,+\,+), (+\,+\,+\,+\,-) \text{ and } (+\,+\,+\,-\,-). \tag{15}$$

Clearly the second group of $\tilde{S}$ can appear in 5 different permutations, such as $(+\,+\,+\,-\,+)$ etc. The third group can appear in 10 different permutations, such as $(+\,+\,-\,+\,-)$ etc. Let us denote these three classes by $i = 1, 2, 3$.

Calculating c_i, the average correlation of a hidden unit in IR group i, with the output, one gets $c_i = 1, \frac{3}{5}, \frac{1}{5}$ respectively. Table II summarizes this observation. The left column contains the weights of the second layer. The next columns present the classes of $\tilde{S}$, accompanied by c - the average correlation of each vector element in that class; since all units are equivalent, c is the same within the class. The last lines shows the size of each class (or degeneracy) and its label for later reference.

Had all 32 IR's appeared with the same probability (i.e. assigned to the same number of patterns), the average correlation would have been given by:

$$c = \frac{1}{32}\left[2 + 10\frac{3}{5} + 20\frac{1}{5}\right] = 0.375 \tag{16}$$

In fact we observed that even for very low α the learning algorithm induces a bias in the selection of IR; the probability of selecting the more correlated classes $i = 1, 2$ is suppressed, whereas class 3 is preferred. This tendency becomes more pronounced with increasing α, as the committee machine tries to accomodate more patterns by reducing c further. Similar tendency was observed for the AND machine [12] and CM [14] previously.

Simulations show that the correlation with the output (average and maximal correlations are the same for the CM) decreases from about 0.3 to 0.24 as α increases towards α_c (see Fig. 10). Probability of appearance of the fully correlated IR decreases to $p^{(1)} = 0$, whereas the intermediately correlated IR goes to $p^{(2)} \simeq 0.009$, while $p^{(3)}$ increases to $\simeq 0.045$ (see Fig. 11). These values are to be compared with $p^{(0)} = \frac{1}{32} = 0.03125$. We computed also the entropy of the distribution in the space of the possible IR for all machines, given by :

$$S = -\sum_{i=1}^{32} p_i \log p_i \tag{17}$$

This value can give an indication on the *effective* number of IR that participate in the learning process, that is 10^S . As expected, the entropy decreases with increasing α since not all IR appear with the same probability (see Fig. 13); the less correlated IR appear with increasing probability.

The other three machines are less simple since not all hidden units play the same role. In order to simplify the presentation, we treat only the 16 IR that give $S_{out} = +1$. The other case, of $S_{out} = -1$, is obtained by reversing each IR. Like in the case of CM, one can divide the IR's into classes. We define as a class *all the IR that can be achieved by permutation of equivalent hidden units (i.e. with the same w_i)* ; For example, in the machine with $w_i = 3\ 1\ 1\ 1\ 1$, the class $+\ +\ +\ +\ -$ has four members, obtained by permuting the $(-)$ through all four units with $w_i = 1$.

Tables II,III,IV and V,summarizes the classes of IR's for each machine. The structure of the tables is similar to the one prescribed for the CM. As in table II, c denotes correlation of a hidden unit with the output for IR that belong to class i.

The average correlations of the hidden units for machine 22111 are given by:

$$\begin{aligned} c(1) &= \frac{1}{16}\left[1 + 3\cdot\frac{1}{3} + 2\cdot 1 + 3\cdot(-\frac{1}{3}) + 6\cdot\frac{1}{3} - 1\right] = \frac{1}{4} \\ c(2) &= \frac{1}{16}\left[1 + 3\cdot 1 + 2\cdot 0 + 3\cdot 1 + 6\cdot 0 + 1\right] = \frac{1}{2} \end{aligned} \tag{18}$$

where $c(w)$ is the average (over all classes of IR) correlation of the hidden units with weight w to the output.

The average correlations of the hidden units for machine 31111 are given by:

$$\begin{aligned} c(1) &= \frac{1}{16}\left[1 + 4\cdot\frac{1}{2} + 1 + 6\cdot 0 + 4\cdot(-\frac{1}{2})\right] = \frac{1}{8} \\ c(3) &= \frac{1}{16}\left[1 + 4\cdot 1 - 1 + 6\cdot 1 + 1\right] = \frac{11}{16} \end{aligned} \tag{19}$$

The average correlations of the hidden units for machine 32211 are given by:

$$\begin{aligned} c(1) &= \frac{1}{16}\left[1 + 2\cdot 0 + 2\cdot 1 + 4\cdot 0 - 1 + 1 + 1 + 2\cdot 0 + 2\cdot(-1)\right] = \frac{1}{8} \\ c(2) &= \frac{1}{16}\left[1 + 2\cdot 1 + 2\cdot 0 + 4\cdot 0 + 1 - 1 + 1 + 2\cdot 1 + 2\cdot 0\right] = \frac{3}{8} \\ c(3) &= \frac{1}{16}\left[1 + 2\cdot 1 + 2\cdot 1 + 4\cdot 1 + 1 + 1 - 1 + 2\cdot(-1) + 2\cdot 1\right] = \frac{5}{8} \end{aligned} \tag{20}$$

The average correlations of each hidden unit with the output for the various machines are shown in Fig. 10. The behaviour of this quantity in the region where $\alpha \to \alpha_c$ is different for these machines.

In the machine 22111, the average correlation of units with $w = 2$ approaches the value of units with $w = 1$. This value is similar to that of the CM. Fig 11 shows the probability of

classes of IR and we notice that the class with $c_2 = 0$ and low c_1 (labeled as 5) is enhanced; this explains why the average $c(2)$ decreases (Fig 10).

In the machine 32211, the average correlation of the unit with $w = 3$ decreases strongly. This behaviour suggests that the machine tries to "weaken the dominant unit" in order to enlarge the number of IR participating in the learning process. In Fig. 11 one can observe that an IR in which the dominant unit is anticorrelated to the output is enhanced; this behaviour obviously leads to decrease in the entropy since there is no big class with high probability (see Fig. 13).

The machine 31111 tries to perform the same behaviour as machine 32211 in the sense that it also "weakens the dominant unit" (see Fig. 10). As seen in this figure, it fails to lower this correlation below that of the other four units with $w = 1$ and a gap is maintained. This fact manifests itself in Fig. 11 where the class (3), in which the dominant unit is anticorrelated, has large probability but it competes with classes (4) and (5), which together have a larger probability. It is obvious that this machine has low entropy (see Fig. 13) since it enlarges dramatically the probability of class (3) which has only one member.

Correlation of the unit with *maximal correlation* is depicted in Fig. 12. This factor may also has an effect on the capacity of a machine; the low capacity of the 31111 machine may be due to this, in addition to the low entropy of this machine. On the other hand, the 32211 machine has similar maximal correlation as the CM but it's capacity is smaller due to lower entropy.

As we mentioned, the entropy indicates the effective number of IR that participate. This number (10^S) for the four machines is :

- 11111 - ~ 25.1 at $\alpha = 3.2$.
- 22111 - ~ 23 at $\alpha = 3.2$.
- 31111 - ~ 20 at $\alpha = 2.4$.
- 32211 - ~ 19 at $\alpha = 3.2$ (~ 21.4 at $\alpha = 2.4$).

These two parameters, entropy and maximal correlation, competes and in order to increase it's capacity, a machine has to maintain high entropy and to decrease it's maximal correlation.

V. BOOLEAN FUNCTIONS

We turn now to address the issue of comparison between the sets of boolean functions which can be realized by different 2LP's. We compare two 2LP's that have the same architecture (same number of inputs and hidden units and same connectivity - NRF or ORF), but different fixed weights from hidden layer to the output.
Assume that the weights between the input and the hidden units of the "teacher" network are fixed. It is interesting to verify whether the second network ("student") can choose the weights between its input and hidden units so that it realizes the same boolean function as the "teacher". In the event that the second network can imitate the first one only for a subset of all possible input to hidden layer connections, one may wish to estimate the fraction of possibly realizable common boolean functions. Another indicator is the distribution of

the overlaps between pairs of boolean functions belonging to the two networks (such overlap is defined as the fraction of all inputs that give rise to similar outputs).

The fraction of the input space for which teacher and student give different answers is just the generalization error:

$$\epsilon_g \;=\; \Big\langle\Big\langle \Theta\Big(-S^t_{out}\; S^s_{out}\Big)\Big\rangle\Big\rangle$$

where S^t_{out} and S^s_{out} are the output of the teacher and student respectively; $\langle\langle\cdots\rangle\rangle \;=\; \frac{1}{2^N}\sum_{\vec{S}^{in}}\cdots$ indicates the average over the input space.
ϵ_g can be expressed in terms of the IR, $\vec{S}^{\nu}$, as follows:

$$\epsilon_g \;=\; \sum_{\nu,\eta=1}^{2^K} \Theta\Big(-\mathrm{B}^t\Big(\vec{S}^{\nu}\Big)\;\mathrm{B}^s\Big(\vec{S}^{\eta}\Big)\Big)\;\mathrm{P}\Big(\vec{S}^t=\vec{S}^{\nu},\;\vec{S}^s=\vec{S}^{\eta}\Big) \tag{21}$$

where $\Big\{\vec{S}^{\nu}\Big\}_{1\le\nu\le 2^K}$ is the set of all IR, B^t (B^s) is the boolean function implemented from the hidden layer to the output of the teacher (student). $\mathrm{P}\Big(\vec{S}^t=\vec{S}^{\nu},\;\vec{S}^s=\vec{S}^{\eta}\Big)$ is the fraction of input space for which teacher and student get the IR's $\vec{S}^{\nu}$ and $\vec{S}^{\eta}$ simultaneously;

$$\mathrm{P}\Big(\vec{S}^t=\vec{S}^{\nu},\;\vec{S}^s=\vec{S}^{\eta}\Big) \;=\; \Big\langle\Big\langle \prod_{l=1}^{K}\Theta\Big(\frac{\vec{W}_l^{\,t}\cdot\vec{S}^{in}}{\|\vec{W}_l^t\|}\;S_l^{\nu}\Big)\;\Theta\Big(\frac{\vec{W}_l^{\,s}\cdot\vec{S}^{in}}{\|\vec{W}_l^{\,s}\|}\;S_l^{\eta}\Big)\Big\rangle\Big\rangle \tag{22}$$

where $\vec{W}_l^t$ ($\vec{W}_l^s$) denotes the vector of weights connecting the input to the teacher's (student's) hidden unit l.

In the following discussion, networks with ORF and NRF are treated separately, although the conclusions are similar. The advantage in a detailed discussion of networks with NRF is that some ideas and examples can be explained and solved in detail.

A. Networks with NRF architecture

In this case the input seen by each hidden unit of the 2LP is decoupled from the others, hence each hidden unit acts as an independent perceptron. Averaging for a pair of teacher-student hidden units which are affected by the same $\frac{N}{K}$ inputs is done independently of all other pairs, and hence equation (22) becomes:

$$\mathrm{P}\Big(\vec{S}^t=\vec{S}^{\nu},\;\vec{S}^s=\vec{S}^{\eta}\Big) \;=\; \frac{1}{2^{2K}}\prod_{l=1}^{K}\;(1+S_l^{\nu}\;S_l^{\eta}\;(1-2\;\epsilon_l)) \tag{23}$$

where ϵ_l is the fraction of inputs for which a different state appears on the l^{th} unit of student and teacher. ϵ_l is nothing but the generalization error of the l^{th} hidden unit of the student with respect to the l^{th} hidden unit of the teacher. Using the integral form of the Θ-functions and assuming that $\|\vec{W}_l^s\|,\;\|\vec{W}_l^t\| \;\propto\; \sqrt{N}$, ϵ_l can be expressed, at leading order in $\frac{1}{N}$, as:

$$\epsilon_l \;=\; \frac{1}{\pi}\cos^{-1}R_l$$

where R_l is the cosine of the angle between the student and teacher weight vectors incident on the l^{th} unit:

$$R_l = \frac{\vec{W}_l^s \cdot \vec{W}_l^t}{\|\vec{W}_l^s\| \, \|\vec{W}_l^t\|}$$

Equations (21) and (23) provide a constructive method to evaluate the generalization function for any pair of NRF machines with the same number of input and hidden units. All the pairs $\left(\vec{S}^\nu, \vec{S}^\eta\right)$ for which teacher and student yield different answers are tabulated. Evaluation of $\mathrm{P}\left(\vec{S}^t = \vec{S}^\nu,\ \vec{S}^s = \vec{S}^\eta\right)$ is rather simple; each hidden unit contributes either a factor ϵ_l or $(1-\epsilon_l)$ depending whether the state of unit l is the same in the two IR or not. The generalization function is given by the sum of these probabilities.

First we apply this idea to show that if the second layer of the teacher and the student implement different dichotomies, the student can not imitate the teacher. After this we demonstrate by some examples the evaluation of ϵ_g.

Let us consider two 2LP whose second-layer weights, $\vec{w}^t$ and $\vec{w}^s$, are fixed and give rise to different dichotomies. Hence there exists a set of IR, $\left\{\vec{S}_{dif}^{\lambda}\right\}_{1\le\lambda\le n_{dif}}$ $(n_{dif} \ge 2)$ [18], for which the output of teacher and student is different. We analyze particular situations and then we use the obtained results to treat the general case.
The first instance to be considered is when student and teacher have identical weights, $\vec{W}_l^s = \vec{W}_l^t$ for all l, and hence $\epsilon_l = 0$ for $1 \le l \le K$. Then every input produces the same IR in the two networks. From (21) and (23) we get that $\epsilon_g = \frac{n_{dif}}{2^K} > 0$.
In the second case some ϵ_l are equal to zero and the remaining ones equal to one. Without loss of generality consider $\epsilon_l = 0$ for $1 \le l \le l_1$ and $\epsilon_l = 1$ for $l_1 < l \le K$. We can perform a gauge transformation, that gives rise to a different student network, s', defined by $\vec{W}_l^{s'} = \vec{W}_l^s$ and $\vec{w}_l^{s'} = \vec{w}_l^s$ for $1 \le l \le l_1$, whereas $\vec{W}_l^{s'} = -\vec{W}_l^s$ and $\vec{w}_l^{s'} = -\vec{w}_l^{\,s}$ for $l_1 < l \le K$. Since the characteristic vectors (see appendix A) associated with $\vec{w}_l^{s'}$ and $\vec{w}_l^t$ are different (because the absolute values of their components are distinct) they implement different boolean functions from the hidden layer to the output. So we have a new set $\left\{\vec{S}'^{\lambda}_{dif}\right\}_{1\le\lambda\le n'_{dif}}$ $(n'_{dif} \ge 2)$. Clearly the network s' realizes the same boolean function as s, but it has $\epsilon'_l = 0$ for *all* l, and hence this case is reduced to the previous one.
The third situation to be considered is when is $0 < \epsilon_l < 1$ for $1 \le l \le K$. We cannot calculate ϵ_g explicitly because it depends on $\vec{w}^t$ and $\vec{w}^s$, but we can obtain a lower bound. Using (21) and (23) we identify two certain sources of error. First: when a member of the set $\left\{\vec{S}_{dif}^{\lambda}\right\}_{1\le\lambda\le n_{dif}}$ appears on the teacher's hidden layer and is copied exactly by the student. This will happen for a fraction $\frac{n_{dif}}{2^K}\prod_{l=1}^{K}(1-\epsilon_l)$ of the inputs. The second source is when an IR, $\vec{S}$, not from $\left\{\vec{S}_{dif}^{\lambda}\right\}_{1\le\lambda\le n_{dif}}$ appears on the teacher, but the student errs on *all* hidden units, generating $-\vec{S}$ as its IR, and hence the opposite output to the teacher. This occurs with weight $\frac{2^K - n_{dif}}{2^K}\prod_{l=1}^{K}\epsilon_l$. Therefore $\epsilon_g \ge \frac{n_{dif}}{2^K}\prod_{l=1}^{K}(1-\epsilon_l) + \frac{2^K - n_{dif}}{2^K}\prod_{l=1}^{K}\epsilon_l > 0$.
Finally the general case is when $0 < \epsilon_l < 1$ for $1 \le l \le l_1$, $\epsilon_l = 0$ for $l_1 < l \le l_2$ and $\epsilon_l = 1$

for $l_2 < l \leq K$. Making the same gauge transformation as in the second case we obtain a network that performs the same boolean functions as the student. Since the second layer implements a different mapping from the hidden layer to the output, we have a new set $\left\{\vec{S}_{dif}^{\prime\lambda}\right\}_{1\leq\lambda\leq n_{dif}^{\prime}}$, $(n_{dif}^{\prime} \geq 2)$. Now we can calculate a lower bound for ϵ_g as in the previous case: $\epsilon_g \geq \frac{n_{dif}^{\prime}}{2^K}\prod_{l=1}^{l_1}(1-\epsilon_l) > 0$.
We showed that in all cases the fraction of the input space for which the two networks give different answers is greater than zero. This proves that the two networks do not implement the same boolean functions.

This method can be applied not only for 2LP, but also for any mapping from the hidden layer to the output. Consider the case when the hidden layer contains two units, and the mapping from IR to output is a parity machine, for both teacher and student (which have different weights connecting input to hidden layer). The output of the 2-unit PM is the product of the states of the two hidden units. The student errs when one of the hidden units agrees and the other disagrees with the teacher. Hence for this machine the generalization error of the student is given by

$$\epsilon_g = \epsilon_1(1-\epsilon_2) + \epsilon_2(1-\epsilon_1)$$

The generalization error of this example has been calculated previously by Hansel and Mato [19]. With some algebra one can show that the result obtained by these authors is completely equivalent to this expression.

In the case of a PM with K hidden units with $\epsilon_l = \epsilon$ for $1 \leq l \leq K$ we obtain:

$$\epsilon_g(K) = \sum_{l=1,3,\ldots K}\binom{K}{l}\ \epsilon^l\ (1-\epsilon)^{K-l}$$

Let us consider now a CM with K hidden units. Assume for simplicity that all hidden units have the same generalization error, $\epsilon_l = \epsilon$ $(R_l = R)$ for $1 \leq l \leq K$. The generalization error of the CM is given by

$$\epsilon_g(K) = \frac{1}{2^{(K-1)}}\sum_{m=m^*}^{K}\binom{K}{m}\sum_{p=0}^{m}\binom{m}{p}\sum_{q=q^*}^{m}\binom{K-m}{q}\ \epsilon^{(p+q)}\ (1-\epsilon)^{K-(p+q)} \tag{24}$$

with $m^* = \frac{K+1}{2}$ and $q^* = m - p - \frac{K-1}{2}$.
To interpret this result, note that to get $S_{out}^t = 1$, the teacher must have $m \geq \frac{K+1}{2}$ hidden units in the $+1$ state. There are $\binom{K}{m}$ ways of choosing these m units. To get the opposite output, the student errs on p of the m units with $S_l = +1$, and on q of the $K-m$ units with $S_l = -1$. There are $\binom{m}{p}\cdot\binom{K-m}{q}$ ways of choosing the errant units, with each such configuration occurring with probability $\epsilon^p(1-\epsilon)^{m-p}\epsilon^q(1-\epsilon)^{K-m-q} = \epsilon^{p+q}(1-\epsilon)^{K-q-p}$. This explains nearly all factors in eq. (24). In addition it is necessary to ensure that the student's

output indeed is -1; i.e. the number of units with $S_l = -1$ exceeds those with $S_l = +1$; that is, $p + K - m - q > m - p + q$ which implies $q < \frac{K}{2} - (m - p)$.

It is interesting to observe that as K grows this result approaches the replica-symmetric solution obtained by Schwarze and Hertz (see fig. (14)) for large committee machines [20] which can be written as:

$$\epsilon_g = \frac{1}{\pi}\cos(1 - 2\epsilon)$$

Finally we solve one non trivial example. The teacher is a CM with 5 hidden units and the student has the same architecture but the weights connecting the hidden units to the output are (22111). In general the angles between the weights of teacher and student, incoming to each hidden unit, are independent variables. However, to simplify the calculation and the discussion, we assume that there are only two relevant angles: $\epsilon_l = \epsilon_1$ and $\epsilon_l = \epsilon_2$ giving rise to two errors, for the hidden units of the student which are connected to the output by weights of strengths 1 and 2, respectively. We obtain for the generalization error the expression [21]:

$$\begin{aligned}\epsilon_g = \frac{1}{16}(1 + 12\epsilon_1 - 12\epsilon_1^2 + 8\epsilon_1^3 + 12\epsilon_2 - 24\epsilon_1\epsilon_2 + \\ 36\epsilon_1^2\epsilon_2 - 24\epsilon_1^3\epsilon_2 - 6\epsilon_2^2 + 24\epsilon_1\epsilon_2^2 - 36\epsilon_1^2\epsilon_2^2 + 24\epsilon_1^3\epsilon_2^2)\end{aligned} \tag{25}$$

It is clear that for $\epsilon_1 = \epsilon_2 = \frac{1}{2}$, $\epsilon_g = \frac{1}{2}$. One can also verify from eq. (25) that the minimal value of ϵ_g is $\frac{1}{16}$ and it is obtained only when $\epsilon_1 = \epsilon_2 = 0$, i.e. when the weights of the student and that of the teacher are identical.
The fact that the minimal ϵ_g is larger than zero means that for some inputs the two machines produce different outputs, no matter how $\vec{W}_l$ are chosen. Hence the two machines do not implement any common boolean function: the two sets of functions implemented by them have no intersection.
The maximal overlap between a pair of boolean functions taken from the two sets is $1 - \frac{2}{16} = \frac{7}{8}$. The reason that the minimal generalization error is $\frac{1}{16}$ is clear, since there are only two internal representations which affect the output of the teacher and the student differently. Nevertheless, the above calculations proves that there is no way to circumvent this difficulty by a permutation or a mapping among the internal representations of the teacher and that of the student.

B. Networks with ORF architecture

For this architecture all hidden units see the same input; therefore they can not be viewed as K independent perceptrons. The common inputs induce correlations between the incident weight vectors of different hidden units. These correlations complicate matters, and proving that different 2LP's with ORF implement different sets of boolean functions is more difficult.

We limit our discussion to the generic case of a "regular" teacher; that is, when the weight vectors $\vec{W}_l^t$, $l = 1, ..., K$ are linearly independent. This is the generic case since a finite number (K) of vectors with N components will be linearly dependent when $K \ll N$ only in a subset of measure zero of cases [22,23].

We treat separately two situations. First, when our regular teacher is imitated by a regular student, whose weight vectors, $\vec{W}_l^s$, $l = 1, ..., K$ are linearly independent of the set $\vec{W}_l^t$ and also linearly independent (the set $\vec{W}_l^s \bigoplus \vec{W}_m^t = \left\{\vec{W}_l^s\right\}_{1\le l\le K} \bigcup \left\{\vec{W}_m^t\right\}_{1\le m\le K}$ is linearly independent). In this case we are able to present a relatively simple proof that the two machines implement different boolean functions. Furthermore, our proof contains a constructive method to evaluate the generalization error. Next we turn to the second situation, and study whether allowing the student to choose a linear combination of $\vec{W}_l^t$ as its weight vectors may eliminate the error.

1. Regular, linearly independent teacher and student networks.

As mentioned above, for ORF architecture, there exist correlations between the weight vectors of both networks. These correlations are contained in three $K \times K$ correlation matrices: R^{ss}, R^{tt} and R^{st}. The elements of R^{ss} are determined by the correlations among the weight vectors of the student, and those of R^{tt} by the teacher . R^{st} is the student-teacher correlation matrix. The element l, m of the matrix R^{ab} is defined by:

$$R_{lm}^{ab} \;=\; \frac{\vec{W}_l^a \cdot \vec{W}_m^b}{\|\vec{W}_l^a\|\,\|\vec{W}_m^b\|} \qquad \text{for} \quad a, b = s, t \quad \text{and} \quad 1 \le l, m \le K \tag{26}$$

The fraction of the input space for which we obtain a given IR, $\vec{S}$, for network $a(= s, t)$ is given by (see C):

$$\mathrm{P}^a\!\left(\vec{S}\right) \;=\; \frac{1}{\sqrt{\det\left(R^{a\,a}\right)}} \int_0^\infty \prod_{l=1}^{K}\!\left(\frac{\mathrm{d}h_l^a}{\sqrt{2\pi}}\right) \exp\!\left[-\frac{1}{2}\sum_{l,m=1}^{K} h_l^a\, S_l \left[\left(R^{a\,a}\right)^{-1}\right]_{lm} S_m h_m^a\right] \tag{27}$$

For $S_l = 1$, the integrand of (27) is just the joint probability distribution of the local fields on each one of the hidden units;

$$h_l^a \;=\; \frac{\vec{W}_l^a \cdot \vec{S}^{in}}{\|\vec{W}_l^a\|} \tag{28}$$

It is clear that as long as the weight vectors, $\vec{W}_l^a$, are linearly independent, the inverse of the matrix $R^{a\,a}$ exists, and the result of eq. (27) is a positive number for any IR, $\vec{S}$. Therefore each internal representation appears with a non-vanishing probability.

Let us now compare two different networks. Introducing the integral expression of the Θ-function in eq. (22) we obtain, at leading order (see C):

$$\mathrm{P}\Big(\vec{S}^t = \vec{S}^\nu,\ \vec{S}^s = \vec{S}^\eta\Big) \ = \ \frac{1}{\sqrt{\det(A)}} \int_0^\infty \prod_{l=1}^{2K}\Big(\frac{\mathrm{d}h_l}{\sqrt{2\pi}}\Big) \ \exp\Big[-\frac{1}{2}\sum_{m,n=1}^{2K} h_m \widetilde{S}_m \left(A^{-1}\right)_{mn} \widetilde{S}_n h_n\Big] \tag{29}$$

where A is a $2K \times 2K$ symmetric correlation matrix, of the form:

$$A = \begin{pmatrix} R^{ss} & R^{st} \\ R^{ts} & R^{tt} \end{pmatrix}$$

In (29) we denoted by $\widetilde{S}$ the vector whose $2K$ components are given by $\widetilde{S}_l = S_l^\nu$ and $\widetilde{S}_{K+l} = S_l^\eta$ for $1 \leq l \leq K$. The present case, when the set $\vec{W}_l^s \bigoplus \vec{W}_m^t$ are linearly independent, the matrix A is regular ($\det(A) \neq 0$). Therefore there is a non-vanishing probability of simultaneous appearance for any pair of internal representations S^η and S^ν for the two networks;

$$\mathrm{P}\Big(\vec{S}^t = \vec{S}^\nu,\ \vec{S}^s = \vec{S}^\eta\Big) \ > \ 0 \qquad\qquad \forall\ \nu, \eta$$

In particular, pairs of IR that produce different outputs on student and teacher will appear, and hence the two networks implement completely distinct sets of boolean functions.

In principle equations (21) and (29) can be used to calculate the generalization error explicitly.

2. Teacher and student are linearly dependent

We turn now to the case when some weight vectors of the student are a linear combination of those of the teacher;

$$\vec{W}_l^s \ = \ \sum_{m=1}^{K} a_{lm}\, \vec{W}_m^t \tag{30}$$

where a_{lm} are real numbers. In this case (29) can not be used because A is singular [24].

We consider two instances. First, we analyze a particular example which allows us to understand the problem of a regular teacher but singular A. We study the conditions that the coefficients a_{lm} must satisfy and we show that there does not exist a set of a_{lm} for which teacher and student implement the same boolean function. Next we treat the general case, extending the argument developed in the previous situation.

Let us first examine the following example: consider a teacher with second layer weights $w_1^t = w_2^t = 2$, $w_l^t = 1$, $l = 3, ..., K$, whereas the student is a CM.
Concentrate on $\mathcal{S}_K$, a subspace of the input space for which the field generated by the first $K-1$ hidden units of the teacher on the output is zero; *i.e.*, $\sum_{l=1}^{K-1} w_l^t\ \mathrm{sign}(h_l^t) \ = \ 0$, where h_l^t is the local field on the l^{th} hidden unit of the teacher (see eq. (28)). The teacher's output is determinated by its last hidden unit for all inputs from $\mathcal{S}_K$.

We first establish some properties of $\mathcal{S}_K$ to be used later. The subspace $\mathcal{S}_K$ contains order 2^N input vectors, which span an N-dimensional space. The first part of the statement derives from the fact that every IR of the teacher appears with finite probability, (*i.e.* fraction of the input space), as long as K is finite while $N \to \infty$.
To show that these vectors span an N-dimensional space, note that the integrand of eq. (27) is just the fraction of the input space whose embedding fields belong to the interval $x_l \pm \Delta_{x_l}$. Therefore it is possible to find a vector $\vec{\mathcal{S}}_1$ belonging to $\mathcal{S}_K$ such that all its embedding fields are of order 1. Consider $\vec{\mathcal{S}}_2$ which differs from $\vec{\mathcal{S}}_1$ in only one of the input spins. Flipping one of the input spins produces a variation in the embedding fields of order $\frac{1}{\sqrt{N}}$.
Hence all N vectors $\vec{\mathcal{S}}_2^{\gamma}$, $\gamma = 1, ..., N$ (obtained by flipping only one component γ out of the N components of $\vec{\mathcal{S}}_1$) map onto the same IR as does $\vec{\mathcal{S}}_1$, and therefore all $\vec{\mathcal{S}}_2^{\gamma}$ belongs to $\mathcal{S}_K$. Clearly, the N vectors $\vec{\mathcal{S}}_2^{\gamma} - \vec{\mathcal{S}}_1$ are the basis of the N-dimensional space.
As our next step, note that partitioning the space $\mathcal{S}_K$ by o($K!$) hyperplanes generates subspaces with order 2^N input vectors, whose dimension is still N (this holds trivially for large N and finite $K \ll N$).

Now we return to our problem, of linearly dependent $\vec{W}_l^s$. In order to ensure that the student follows the teacher for the set of inputs $\mathcal{S}_K$, it is necessary and sufficient to have at least $\frac{K+1}{2}$ hidden units of the student copy the K^{th} unit of the teacher. Concentrate on a subset $\widetilde{\mathcal{S}}_K$ of $\mathcal{S}_K$, that contains those inputs for which one particular group of $\frac{K+1}{2}$ hidden units of the student copy the K^{th} unit of the teacher. There are $\binom{K}{\frac{K+1}{2}}$ such groups (and subsets of $\mathcal{S}_K$).

Let us try a solution to the problem (of copying the K^{th} hidden unit of the teacher, for all inputs from $\widetilde{\mathcal{S}}_K$, at $\frac{K+1}{2}$ student hidden units), that has a more general form than (30);

$$\vec{W}_l^s \;=\; \sum_{m=1}^{K} a_{lm}\, \vec{W}_m^t + b_l\, \vec{\mathcal{S}}_l^{\perp} \tag{31}$$

where the vector $\vec{\mathcal{S}}^{\perp}$ is orthogonal to the subspace $\widetilde{\mathcal{S}}_K$. It is necessary to consider such an extention of (30), since for all inputs from $\mathcal{S}_K$, we clearly get the same IR when (30) or (31) are used.
However, we show now, using the properties of $\mathcal{S}_K$ established above, no such vectors $\vec{\mathcal{S}}^{\perp}$ exist. To see this, note that following the properties of $\mathcal{S}_K$, our subset $\widetilde{\mathcal{S}_K}$ has o(2^N) input vectors, and it constitutes an N-dimensional space. Therefore there is no vector $\vec{\mathcal{S}}_l^{\perp}$ which is orthogonal to *all* inputs from $\widetilde{\mathcal{S}}_K$, and it is sufficient to consider solutions of the form (30).

Our next step is to show that the only set of a_{lm} for which $\frac{K+1}{2}$ units copy the teacher's K^{th} unit is given by:

$$a_{lm} \;=\; a_l\, \delta_{mK}; \qquad\qquad a_l > 0$$

so that we have, for $\frac{K+1}{2}$ hidden units of the student:

$$\vec{W}_l^s \;=\; a_l\, \vec{W}_K^t \tag{32}$$

This is trivially a sufficient condition. To see that it is also necessary, assume that some $\vec{W}_m^t$ with $m \neq K$ is also mixed in one of the $\vec{W}_l^s$. Then, again due the fact that $\widetilde{\mathcal{S}}_K$ is

an N-dimensional space, we are certain to find inputs $\vec{S}^{in}$ which are much closer to $\vec{W}_m^t$ that to $\vec{W}_K^t$; sufficiently so that $a_{lm}\,\vec{W}_m^t \cdot \vec{S}^{in} > a_{lK}\,\vec{W}_K^t \cdot \vec{S}^{in}$. Hence for these inputs, the m^{th} hidden unit of the teacher is copied by the student. This unit has $\frac{1}{2}$ the times (on the average) opposite sign to the K^{th} and therefore allowing mixing of $\vec{W}_m^t$ into $\vec{W}_l^s$ contradicts our assumption that unit l copied the teacher for all inputs from $\widetilde{\mathcal{S}}_K$.
Thus we established that one must have $\frac{K+1}{2}$ student weights that are copies of $\vec{W}_K^t$.

Consider now the subspace $\mathcal{S}_{K-1}$ generated by the set of input vectors whose output is determinated by the $(K-1)^{th}$ unit. In order to imitate the teacher for *these* inputs at least $\frac{K+1}{2}$ weight vectors of the student must be proportional to $\vec{W}_{K-1}^t$. This condition and (32) can not be fulfilled simultaneously because at least one of the weight vectors of the student must be proportional to *both* $\vec{W}_{K-1}^t$ and $\vec{W}_K^t$ which is not possible when the teacher is regular.
Hence we have shown that a CM can not reproduce a regular teacher with weights $w_1^t = w_2^t = 2$, $w_l^t = 1$, $l = 3, ..., K$, even when the student's weights are allowed to be linear combinations of the teacher's.

The same line of argument can be followed for any pair of networks. We assume that no hidden unit is redundant; for each hidden unit l there exists at least one internal representation, whose corresponding output changes sign when $S_l \to -S_l$. Since any internal representation appears for a finite fraction of the input space, we can find a set of inputs whose corresponding output is determined by the l^{th} unit. The dimensionality of the subspace spanned by such a set is N. Therefore, in all these cases there is a minimum number (greater than one) of weight vectors of the student that should be proportional to each teacher unit if it were to be able to imitate the teacher. These set of conditions can not be fulfilled in general.
We conclude that given a student and teacher whose the second layer weights implement different dichotomies, the assumption that the student can imitate a regular teacher leads to a contradiction.

VI. SUMMARY

In this work we have studied the computational capabilities restricted two layer perceptrons, whose second layer weights are fixed. Such a study is relevant for our understanding of general, non-restricted 2LP's, since a general 2LP with finite number of hidden units is equivalent to the union of networks with distinct second layer weights. We focussed our attention to two measures of the strength of such 2LP: their storage capacity and the boolean functions that they are capable to implement.

We considered networks with two different architectures; in one architecture (NRF) the connections from the input to the hidden layer form non-overlapping receptive fields, whereas in the second one (ORF) the hidden layer is fully connected to the inputs. We found that for ORF machines of $K = 3$ hidden units all networks are equivalent to the committee machine, and for five hidden units the most general 2LP is equivalent to one of four classes of 2LP with fixed second layer weights. An open question is how the number of classes scales with the

number of hidden units. The upper bound is 2^{K^2}, which is given by the number of different dichotomies that can be implemented by a perceptron with K inputs. The number of classes for the non-overlapping case is determined by the number of different dichotomies that can be implemented in a space of K dimensions. For the overlapping case this number is reduced due symmetries between the input units. For instance the seven different dichotomies that can be implemented in 5-dimensions leads to four classes as mentioned above.

We introduced a numerical method to estimate the capacity of a network which depend neither on arbitrary halting parameters nor on the learning algorithm used. We used this technique to determine the capacity of 2LP with three and five hidden units for both architectures. The learning prescription that we employed was the CHIR algorithm. For $K = 3$ our results are in agreement with theoretical ones. For $K = 5$, we found that the committee machine is the strongest machine; therefore it determines the capacity of the general machine, which equals that of its constituent with highest α_c.
The effect of learning on the probability distribution of internal representations was studied. We found a competition between the entropy and the tendency to reduce the maximal correlation between the hidden units and the output. A machine has to maintain high entropy and to decrease it's maximal correlation in order of increase it's capacity.

We derived a constructive method to calculate the generalization error for all pairs of machines with the same architecture. Explicit expressions were obtained for the NRF architecture, and as well for ORF machines, but only for the case when the teacher and student weights are linearly independent. An important issue for future study is to establish the relation of the teacher/student weights that minimizes the generalization error for the two different machines. In particular, one may ask whether the minimal generalization error is achieved when the weights of the student are a copy of those of the teachers. Preliminary results show that this is not the case.

This method and geometrical arguments leads to a surprising finding: the intersection between the set of boolean functions implemented by two different restricted 2LP's is empty, for all K.
We are currently trying to estimate ϵ_g^{min}, the minimal generalization error (for ORF); that is, how different two such boolean functions have to be ? Further - it is of interest to know how does ϵ_g^{min} scale with N and K. These questions will be answered in the future.

ACKNOWLEDGMENTS

Research of I. Kanter was supported in part by a grant from the Israel Academy of Sciences.

APPENDIX A

Following is the reduction of ORF 2LP with $K = 5$ hidden units to a union of 7 restricted machines, based on [16,17] (Further reduction to four 2LP restricted machines, based on

weight vectors transformations, is presented in sec. 3). In these references, one can find tables containing lists of boolean functions of a *single threshold element*, which is analogous to a perceptron. Each table stands for a different number of input units and the maximal number of units reported is 6 + threshold unit. These tables give *all* sets of weights needed to implement any *linearly separable (l.s.)* function of this number of input units - given the freedom to choose the permutation and signs of those units. For five units, the table contains 7 sets of weights with zero threshold. Translation to our problem enables us to use these sets as the weight vectors w_i , $i = 1, 2..5.$ from the hidden layer to the output. If a problem can be solved with $K = 5$ hidden units, the mapping from the IR's to the output is *l.s.* and one of these weight vectors must give the desired output.

Turning to the evaluation of these weight vectors, consider an N input threshold element - a perceptron. The 2^N different input vectors $x_i = \pm 1$, $i = 1..N$ constitute the corners of a hypercube. Define ρ as the label of a corner, $\rho = 1, 2, ..., 2^N$. For some arbitrary function $f(\rho)$, denote by $< f(\rho) >$ the summation over all possible ρ :

$$< f(\rho) > \equiv \sum_{\rho=1}^{2^N} f(\rho). \tag{A1}$$

Obviously, the input variables satisfy the following relations :

$$< x_i(\rho) > = 0 \; ; \qquad < x_i(\rho) x_j(\rho) > = 2^N \delta_{ij} \; ; \qquad x_i \in \{-1, 1\} \tag{A2}$$

If we define the function $F(\rho)$ by

$$\begin{array}{l} F(\rho) = +1 \;\; \text{if} \;\; f(\rho) = \sum_{i=1}^{N} a_i x_i(\rho) + a_0 \geq 0 \\ F(\rho) = -1 \;\; \text{if} \;\; f(\rho) = \sum_{i=1}^{N} a_i x_i(\rho) + a_0 < 0 \end{array} \tag{A3}$$

One can eqivalently define :

$$\begin{array}{l} f(\rho) F(\rho) = |f(\rho)| \; ; \qquad\qquad f(\rho) \neq 0 \;\; \forall \; \rho \\ \text{or} \\ < f(\rho) F(\rho) > = < |f(\rho)| > \end{array} \tag{A4}$$

The last relation is also a sufficient condition for the realization of a boolean function (presented by F) with a single threshold unit. The reason is that the r.h.s. terms are all positive and the l.h.s are the same terms up to signs. The two sums will be equal if and only if the terms are the same. Denoting

$$\begin{array}{l} b_i = < x_i(\rho) F(\rho) > \;\; (i = 1..N). \\ b_0 = < F(\rho) > . \end{array} \tag{A5}$$

and substituting $f(\rho)$ into (A4) leads to the relation

$$\sum_{i=0}^{N} a_i b_i = < |f(\rho)| > ; \tag{A6}$$

The vector b is called the *characteristic vector*. This vector fully describes the boolean function F. Now any boolean function can be reduced to it's characteristic vector. The

weights that realize a given boolean function can be found by means of an exhaustive search in weight space till the condition on $\sum_{i=0}^{N} a_i b_i$ is satisfied. Apparently, this is a hard problem. The above references suggest an approximation to $f(\rho)$,obtained by expanding the function in a power series. This method leads to some relation between a_i and b_i. As the size of N increases, one must use higher powers in the approximation. This relation is used to tabulate the values of the weight vector.

Returning to the case of 5 units, one finds 7 weight vectors, as mentioned. The values of the weights are :

10000 11100 11111 21110 31111 22111 32211

Since these seven weight vectors are sufficient to realize *any* boolean function of 5 input units (that is *l.s.* with one output unit and zero threshold), the set of seven perceptrons that constitute the union of these vectors, is clearly equivalent to the most general continuous vector.

Moreover, any boolean function (of 5 input units) that is *l.s.* by a single threshold element, must be implemented by one of these 7 weight vectors. Since the reduction to the characteristic vector is unique in the sense that similar boolean functions have similar characteristic vectors, each *l.s.* function can be implemented by *one and only one* weight vector . The conclusion is that *different weight vectors are capable of implementing* **different sets of boolean functions** .

APPENDIX B: THE CHIR ALGORITHM

In this appendix we first make a short review of the CHIR algorithm and then describe the versions used in this work. A comprehensive and detailed explanation of this algorithm is available in [2,3].

1. Review of the CHIR Algorithm

The CHIR algorithm has appeared in the literature in several variants. The variant used in our work is *CHIR2* [3] [25]. This variant, as opposed to the original one [2], doesn't require storing the IR of each pattern. The weights are corrected every time a presented pattern gives wrong output.

Consider the architecture of Fig. 1 with N input, K hidden and 1 output units. The units are binary threshold units. The states of these units are determined by the rules (see eq. 1,2)

$$S_i = sign h_i, \quad h_i = \sum_{j=0}^{N} W_{ij}\, S_j^{in} \tag{B1}$$

$$S^{out} = sign h, \quad h = \sum_{i=0}^{K} w_i\, S_i \tag{B2}$$

Here W_{ij} are weights assigned to connections from input to hidden layer, the weight w_i connects hidden unit i to the output. W_{i0} and w_0 are, respectively, biases of the hidden and output units, with $S_0 = S_0^{in} = 1$. For $i > 0$ the variables $S_i^{in}/S_i/S$ denote the states taken by the input/hidden/output units, respectively. During a training session, the input units are set in any one of $\mu = 1 \ldots P$ patterns, i.e., $S_j^{in} = \xi_j^\mu$. In a typical task, such a network has to "learn" to produce P specified answers, $S^\mu = \xi^\mu$, in response to the P input patterns.

The learning process (see Fig. 15 for a flow chart) starts out by setting w_i and W_{ij} randomly. The procedure alternates then between two learning stages: Learn23 and Learn12.

Learn23 The hidden layer serves as source, and the output as the target unit of the perceptron learning rule (PLR), used to learn the w_i. The weights W_{ij} remain fixed during this stage. We present a pattern μ as input (i.e., set $S_j^{in} = \xi_j^\mu$). The resulting state of the hidden layer, as obtained from (B1), is the internal representation of pattern μ. This IR state is used by the PLR to search for appropriate weights w_i, to obtain the desired outputs $S = \xi^\mu$. If the PLR finds such w_i, we stop; the complete learning problem has been solved. Otherwise we stop after I_{23} learning sweeps of the entire training set [26], keep the current weights, and turn to the next stage, *Learn12*.

Learn12 While the current values of w_i remain fixed, apply a learning process to W_{ij}. Using the PLR at this state is problematic, since only the states of the *source* units, i.e., the input of the network, are known. The corresponding target state (of the hidden layer) is not known. The training set is presented sequentially, pattern by pattern. If we get a wrong output, we choose one hidden neuron that gives a wrong contribution to the field acting on the output unit. This neuron is chosen according to a *minimal disturbance principle* [27]. The state (± 1) of the unit (obtained in response to the current input) is flipped by modifying the weights W_{ij} incident on it, using an Abbott–Kepler linear rule [28], which ensures that the internal representation is actually changed following the modification. At each representation of a pattern, only *one* hidden unit can be flipped.

If the network has achieved error-free performance for the entire training set, learning is completed and a solution of the problem has been found. If no solution has been found after I_{12} sweeps of the training set, we abort the learn12 stage, restart the cycle with *learn23*, and so on.

After each learning step, we normalize the weights incident on the chosen neuron to avoid an exponential decay of the weights of some units arising from repeatedly choosing the same neuron (since we choose with higher probability neurons with small fields).

2. Description of the versions used

One can define different ways of choosing the hidden unit to be flipped, or to modify the weight vector entering the unit picked. Another parameter of the algorithm is the inverse temperature - β , that controls the randomness of the flipping mechanism.

Version 1 This version can be summarized as follows:

1. Identify all hidden units whose contribution to the field acting on the output has wrong sign, as candidates to be flipped.
2. define the flip probability for these units :

$$p(i) \propto \exp^{-\beta(|h_i|-m\Delta E_{err})} . \tag{B3}$$

   where h_i is the field entering the hidden unit; m is some constant that controls the influence of ΔE_{err} on $p(i)$; ΔE_{err} is the change in the output error that is induced by flipping the sign of hidden unit i. In the case of one binary output, we can either have $\Delta E_{err} = 0$ (when flipping the sign of hidden unit i does not correct the error) or $\Delta E_{err} = 1$ when the output **is** corrected by the flip.

   β is a temperature like parameter that determines how often we flip a unit with large field [29].
3. choose one of these units randomly according to it's probability.
4. update the chosen unit's weights so that its sign flips.

Note that the term $m\Delta E_{err}$ has no effect when used in a CM since all units have the same influence on the output.

Version 2 Another version which we considered was similar to *Version 1.* The only difference appears in step 1, in that instead of allowing only the "wrong" units, we take *all* units as candidates for a learning step. The probabilities $p(i)$ are defined in the same manner as in *Version 1.* Flipping a unit that pulls the output in the correct direction can not reduce the error, hence for such units $\Delta E_{err} = 0$.

As mentioned above, one can define the parameter β in different ways. If the network converges, one would like this parameter to decrease in order to avoid a cycle occuring near a solution. The original CHIR2 paper suggested choosing $\beta = N_{err}$ (the number of misclassified patterns). We found that in the random input/output problem, this "heating shedule" is too radical for the early stage of learning.

In this work, we used a *new heating shedule.* The variation we used is based on double stage parameter with a predefined crossover point between the two stages.

$$\beta = \begin{cases} N_{err} & if \quad N_{err} < err_1 \\ err_1 + slp * \tanh\left(\frac{N_{err}}{slp}\right) & if \quad err_1 \le N_{err} < err_2 \\ err_1 + slp * \tanh\left(\frac{err_2 - err_1}{slp}\right) + \frac{N_{err}}{\sqrt{N}} & if \quad N_{err} \ge err_2 \end{cases} \tag{B4}$$

Where we take:

$$\begin{aligned} err_1 &= 0.85 \ N_c \\ err_2 &= 1.25 \ N_c \\ N_c &= \kappa \ \sqrt{N} \\ slp &= \frac{err_2 - err_1}{\log\left(\sqrt{N_c} + \sqrt{N_c - 1}\right)} \end{aligned}$$

Eq. B4 defines two slopes as in Fig. 16, and a turning point, N_c. The point N_c, and the slope of β vs. N_{err} depend on N. The crossover is moderated by a smooth function that connects the two slopes at some points near N_c , err_1 and err_2; we used a hyperbolic tangent function whose first derivative matches the two slopes at both ends. κ is some prechosen constant (of order unity) that is adapted to the problem. Performance is not very sensitive to changes in the parameters of the function β, as long as the general dependence on $\sqrt{N}$ is kept.

In all our simulations, we used this version with different values of κ, as reported in each figure.

To conclude, the parameters that changed in the simulations were :

- The version - 1 or 2.
- The parameter m .
- The parameter κ that controls the function β.

APPENDIX C: DISTRIBUTION OF THE INTERNAL REPRESENTATIONS

In this appendix we derive expressions (27) and (29). The starting point is eq. (22). We calculate the fraction of the input space, P, whose corresponding internal representation is $\vec{S}$ for a 2LP with first layer weights $\vec{W}_l$:

$$\mathrm{P}\big(\vec{S}\big) \;=\; \left\langle\!\!\left\langle \prod_{l=1}^{K} \Theta\!\left(\frac{\vec{W}_l\cdot\vec{S}^{in}}{\|\vec{W}_l\|}\,S_l\right)\right\rangle\!\!\right\rangle \tag{C1}$$

The Θ-functions for each hidden unit l can be expressed as:

$$\Theta(x_l) \;=\; \int_0^\infty\!\!\left(\frac{\mathrm{d}h_l}{\sqrt{2\pi}}\right)\,\int_{-\infty}^{\infty}\!\!\left(\frac{\mathrm{d}\widehat{h}_l}{\sqrt{2\pi}}\right)\,\exp\!\left[i\,\widehat{h}_l\,(h_l-x_l)\right] \tag{C2}$$

where $x_l=\frac{\vec{W}_l\cdot\vec{S}^{in}}{\|\vec{W}_l\|}\,S_l$ is the embedded strengths on the l^{th} hidden unit. Introducing eq. (C2) and taking into account that the input components, S_j^{in}, only affects x_l eq. (C1) becomes:

$$\int_0^\infty\prod_{l=1}^{K}\!\!\left(\frac{\mathrm{d}h_l}{\sqrt{2\pi}}\right)\,\int_{-\infty}^{\infty}\prod_{l=1}^{K}\!\!\left(\frac{\mathrm{d}\widehat{h}_l}{\sqrt{2\pi}}\right)\,\exp\!\left(i\sum_{l=1}^{K}\widehat{h}_l\,h_l\right)\,\left\langle\!\!\left\langle\exp\!\left(-i\sum_{l=1}^{K}x_l\,\widehat{h}_l\right)\right\rangle\!\!\right\rangle \tag{C3}$$

The average over the independent random components of the input vectors gives:

$$\exp\left[\sum_{j=1}^{N}\ln\cos\left(\sum_{l=1}^{K}\frac{\widehat{h}_l\,S_l\,W_{lj}}{\|\vec{W}_l\|}\right)\right] \tag{C4}$$

W_{lj} denotes the j^{th} component of the l^{th} weight vector. Assuming that $\lim_{N\to\infty}\|\vec{W}_l\|=\infty$, eq. (C4) becomes, at leading order:

$$\exp\left(-\frac{1}{2}\sum_{l,m=1}^{K}\ \widehat{h}_l\,\widetilde{R}_{lm}\,\widehat{h}_m\right)$$

with

$$\widetilde{R}_{lm} = S_l\,R_{lm}\,S_m \qquad\qquad \text{for} \quad 1 \leq l,m \leq K$$

where R_{lm} is the correlation between $\vec{W}_l$ and $\vec{W}_m$ defined in eq. (26).
Introducing this expression in eq.(C3), integrating on the $\widehat{h}_l$ and using that:

$$\det\left(\widetilde{R}\,\right) \;=\; \det(R\,)$$

and,

$$\left[\,\widetilde{R}^{-1}\right]_{lm} \;=\; S_l\,\left[R^{-1}\right]_{lm}S_m$$

we obtain:

$$\mathrm{P}\left(\vec{S}\right) \;=\; \frac{1}{\sqrt{\det(R)}}\,\int_0^{\infty}\prod_{l=1}^{K}\left(\frac{\mathrm{d}h_l}{\sqrt{2\pi}}\right)\,\exp\left(\frac{-1}{2}\sum_{l,m=1}^{K}\,h_l S_l\left[R^{-1}\right]_{lm}S_m h_m\right)$$

The fraction of the input space that gives a specific IR can be interpreted as the probability of obtaining this IR given a random input. In this framework this result is a manifestation of the multidimensional central limit theorem by observing that the local fields $h_l = \frac{\vec{W}_l\cdot\vec{S}^{in}}{||\vec{W}_l||}$ becomes random variables with: $\langle\!\langle h_l \rangle\!\rangle = 0$, $\langle\!\langle h_l\,h_m \rangle\!\rangle = R_{lm}$ for random binary uncorrelated input components.

REFERENCES

[*] Present address: Complex Systems Group, Los Alamos National Laboratory, Los Alamos NM 87545, U.S.A. .

[1] see : J. Hertz, A. Krogh and R. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Co. (1991).
D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, MIT Press (1986).
R.P. Lippmann, IEEE ASSP 4,4(1987).
B. Widrow and M.A. Lehr, 30 years of adaptive neural networks: perceptron, madaline, and backpropagation, Proc. IEEE, vol. 78(9), 9(1990).
B. Widrow and R. Winter, Computer 21, No. 3, 25 (1988).

[2] T. Grossman, In Adv. in NIPS II, 516-523 (1990).
T. Grossman, R. Meir and E. Domany, Complex Systems 2, 555-575 (1988).

[3] D. Nabutovsky, T. Grossman and E. Domany, Complex Systems **4**, 519 (1990).
T. Grossman, Proc. of 17'th Conv. of IEEE, 195-198 (1991).

[4] E. Eisenstein and I. Kanter, Europhys Lett., **21** (4), pp. 501-506 (1993).

[5] J.S. Judd, in "proc. of the first intl. conf. on neural networks", IEEE, San Diego, CA.

[6] T.M. Cover, IEEE Trans. Elec. Comp. **14**, 326-334 (1965).

[7] E. Gardner, J. Phys A **21**, 257 (1988).
E. Gardner and B. Derrida, J. Phys A **21**, 271 (1988).

[8] G.J. Mitchison and R.M. Durbin, Biol. Cybern. **60**,345 (1989).

[9] M. Mezard and S. Patarnello (unpublished).

[10] E.Barkai, D.Hansel, and I.Kanter, Phys. Rev. Lett. **65**, 18 (1990).

[11] E. Barkai and I. Kanter, Europhys. Lett. **14**, 107 (1991).

[12] M. Griniasty and T. Grossman, Phys. Rev. A, **45**, 8924 (1992).

[13] E.Barkai, D.Hansel, and H.Sompolinsky, Phys. Rev. A, **45**, 4146 (1992).

[14] A.Engel, H.M.Kohler, F.Tschepke, H.Vollmayr, and A.Zippelius, Phys. Rev. A, **45**, 7590 (1992).

[15] N.J. Nilsson, Learning Machines (McGraw-Hill, New York, 1965).

[16] P.M. Lewis II and C.L. Coates, Threshold Logic (John Wiley & Sons, New York, 1967).

[17] M.L. Dertouzos, IEEE TEC (EC-13), 519 (1964).

[18] n_{dif} is at least 2 since if $\vec{S} \in \left\{\vec{S}_{dif}\right\}$ then $-\vec{S}$ also belongs to it.

[19] D.Hansel, G. Mato and C. Meunier, Europhys. Lett. **20**(5), 471-476 (1992).

[20] H. Schwarze and J. Hertz, Europhys. Lett, **20**(4), pp.375-380 (1992).

[21] Calculation of the generalization error is straight forward, using eqs. (21) and (23). All pairs of teacher/student IR are scanned, and assigned the correct probability. This is done using the symbolic computer language *Mathematica*©. It is easy to calculate ϵ_g for the most general $\vec{\epsilon} = (\epsilon_1, ..., \epsilon_K)$ for every NRF teacher-student pair; implementing any possible Boolean function from the hidden units to the output.

[22] M. Blatt and E. Vergini, Phys. Rev. Lett. **66**, 1793-1796 (1991).

[23] I. Kanter and H. Sompolinsky, Phys. Rev. A **35**, 380 (1987).

[24] In the case that none of the student weights are a linear combination of the teacher weights but the student is not regular, the matrix A is also singular.
If $\vec{W}_1^s, ..., \vec{W}_{K'}^s$, $K' < K$, are the linear independent student weight vectors, every set of local fields $\{h_1^t, ..., h_K^t, h_1^s, ..., h_{K'}^s\}$ appears for a finite fraction of the input space.

In addition $h^S_{K'+1}, ..., h^S_K$ are completely determined by the local fields on the first K' hidden units of the student. For this reason there is a finite probability to find a pair of teacher-student IR whose corresponding outputs are different.
The last situation which yields $\det(A)=0$, is the case when some student weight vectors are allowed to be a linear combination of the teacher and other student weight vectors. This situation leads to the same conclusion as the case considered in the text, following the same reasoning.

[25] Although we actually use here only the **Learn12** part of the algorithm since we train only one layer, we explain the whole process for completeness' sake.

[26] The parameter I_{23} is not used in our version since we train only the first layer weights.

[27] Various versions of the algorithm differ in the manner this unit is chosen - see below. In all versions, the *minimal disturbance principle* operates here on the space of IR rather than on the space of weights as in other algorithms that use this principle.

[28] L.F. Abbott, Thomas B. Kepler, J. Phys. A, **22**, L711 (1989).

[29] Note that taking $\beta \to \infty$ in eq. (B3) means that one always picks the unit with highest "energy" (which has probability one, unless a redundancy occurs).

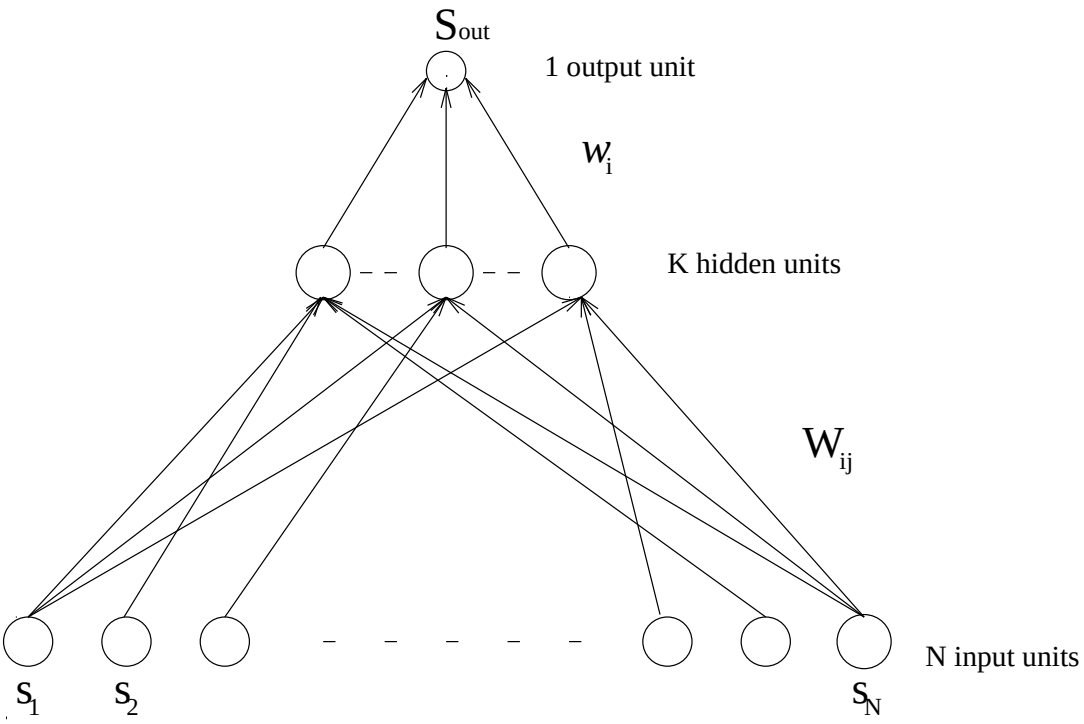


FIG. 1. Overlapping Receptive Field network with N:K:1 architecture.

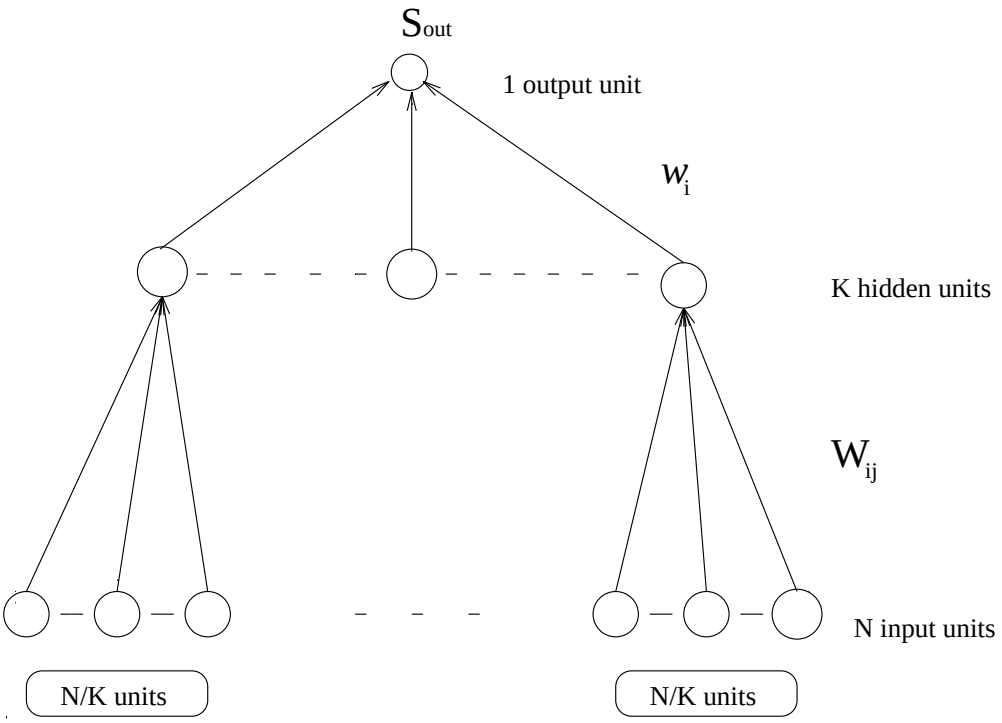


FIG. 2. Non-overlapping Receptive Field network with N:K:1 architecture.

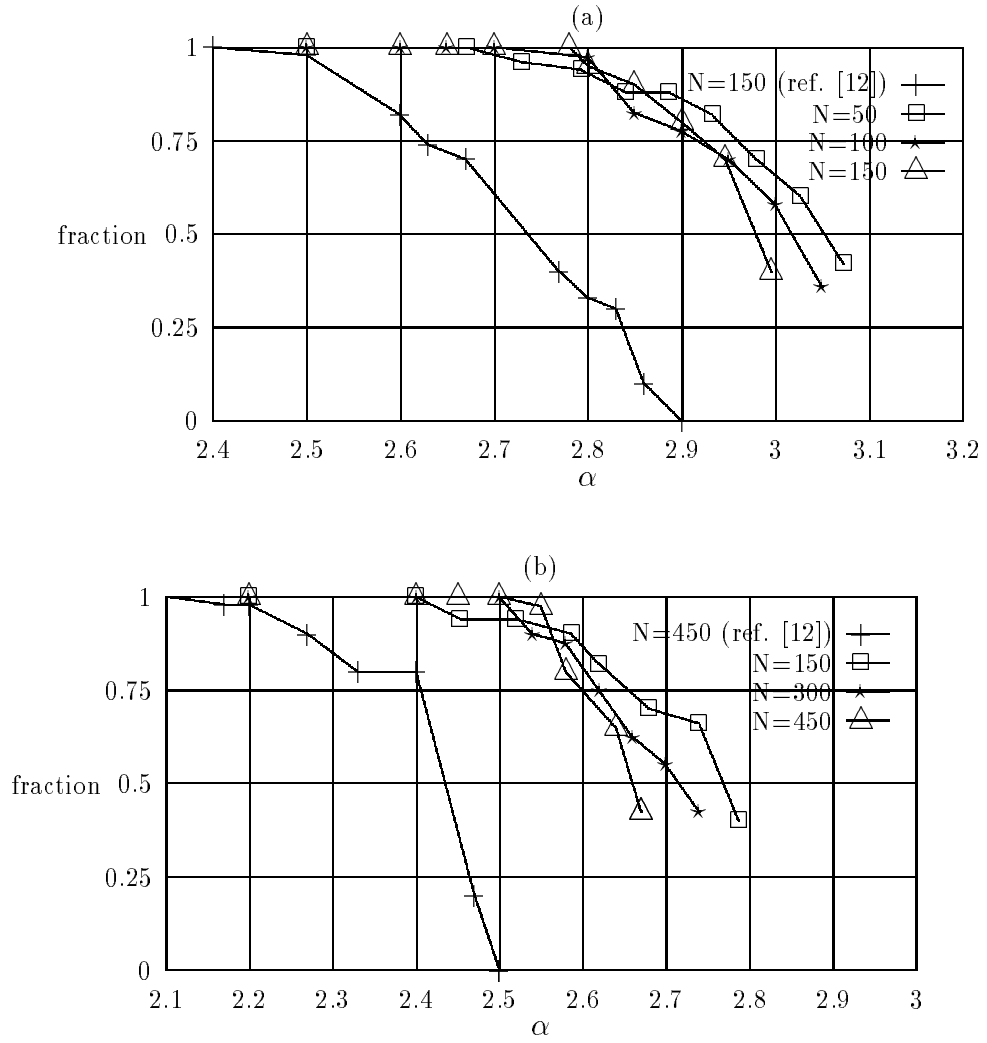


FIG. 3. Fraction of succeses for Committee Machine with K=3 hidden units. The architecture is either ORF or NRF and the input size is $N = 50, 100, 150$ or $N = 150, 300, 450$ respectively. The time limit was up to 6000 sweeps (presentations of the entire training set). The same limit was used by Barkai et al [13] whose results for $N = 150$ (ORF) and $N = 450$ (NRF) are also presented.

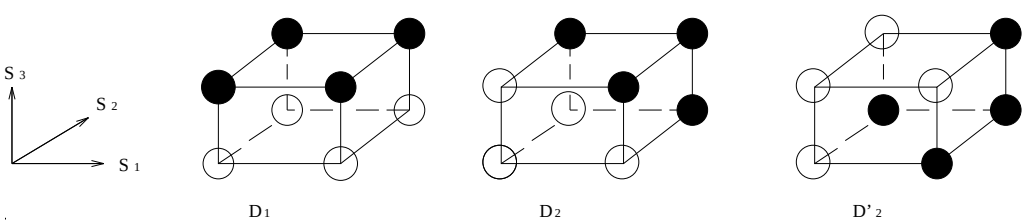


FIG. 4. Two distinct dichotomies D_1, D_2 of the space of internal representations for $K = 3$. D_2' is equivalent to D_2; full (open) circles represent IR's which are mapped to $S^{out} = +1(-1)$.

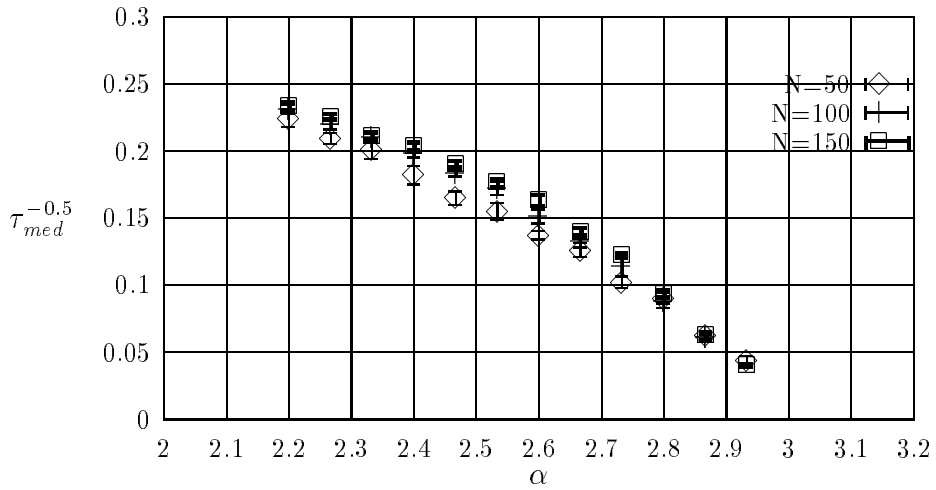


FIG. 5. Scaling of $\tau_{med}^{-0.5}$ for ORF Committee Machine with K=3 hidden units and N=50,100,150 inputs. Version 1 of CHIR, with $\mu = 0, \kappa = 1.0$ was used. Extrapolation yields $\alpha_c \approx 3.05$.

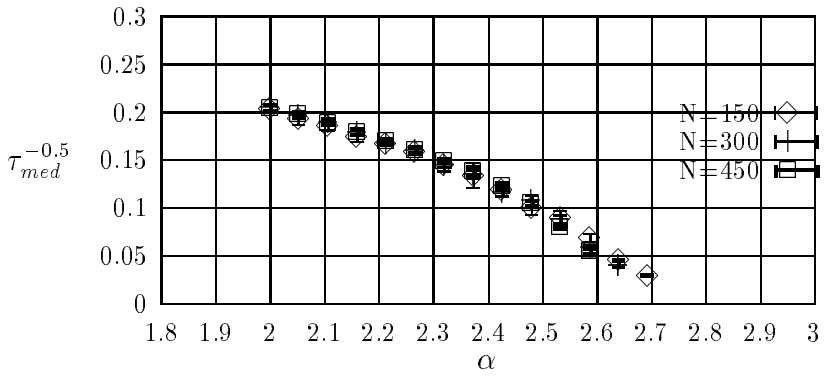


FIG. 6. Scaling of $\tau_{med}^{-0.5}$ for NRF Committee Machine with K=3 hidden units and N=150,300,450 inputs. Version 1 of CHIR, with $\mu = 0, \kappa = 1.0$ was used. Extrapolation yields $\alpha_c \approx 2.75$.

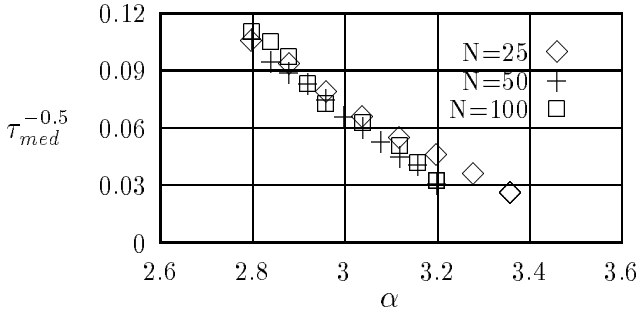


FIG. 7. Scaling of $\tau_{med}^{-0.5}$ for the union of all four machines in **parallel** . The architecture is ORF with K=5 Hidden units and N=25,50,100 inputs. Version 1 of CHIR, with $\mu = 0, \kappa = 4.0$ was used.

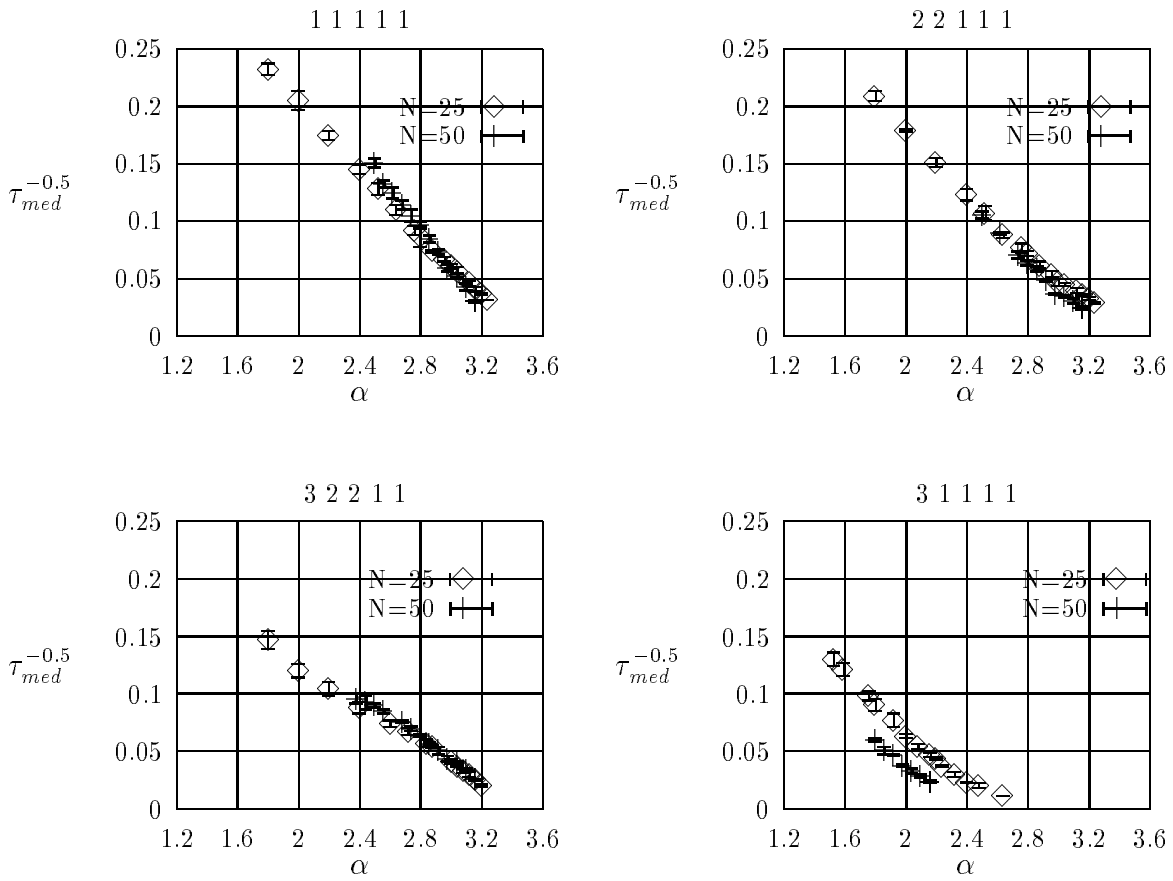


FIG. 8. Scaling of $\tau_{med}^{-0.5}$ for all four machines with K=5 Hidden units and $N = 25, 50$ inputs. We used the following versions of CHIR for each macine: '11111' - version 2, $\mu = 0.7, \kappa = 2.0$. '22111' - version 1, $\mu = 0, \kappa = 2.0$. '32211' - version 2, $\mu = 1.0, \kappa = 2.0$. '31111' - version 1, $\mu = 0, \kappa = 1.0$.

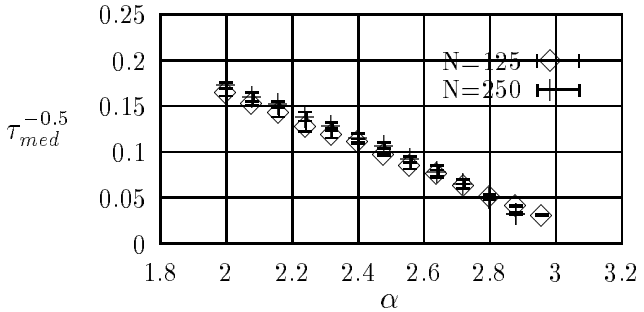


FIG. 9. Scaling of $\tau_{med}^{-0.5}$ for NRF Committee Machine with K=5 hidden units and N=125,250 inputs. Version 1 of CHIR, with $\mu = 0, \kappa = 1.0$ was used.

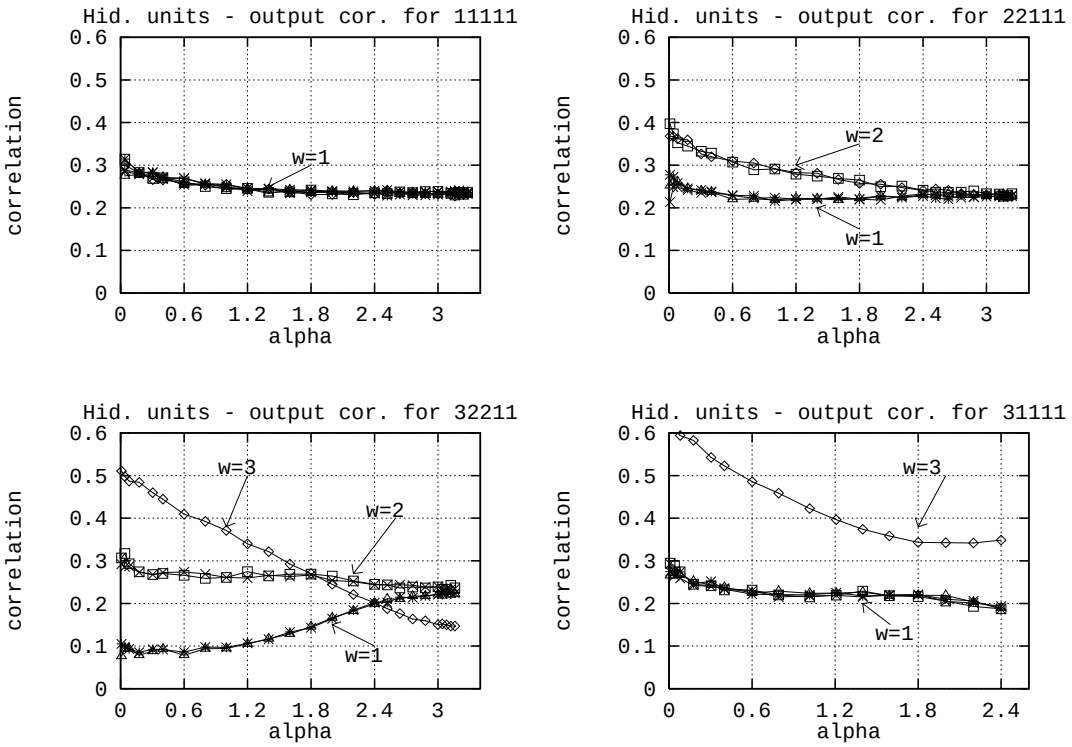
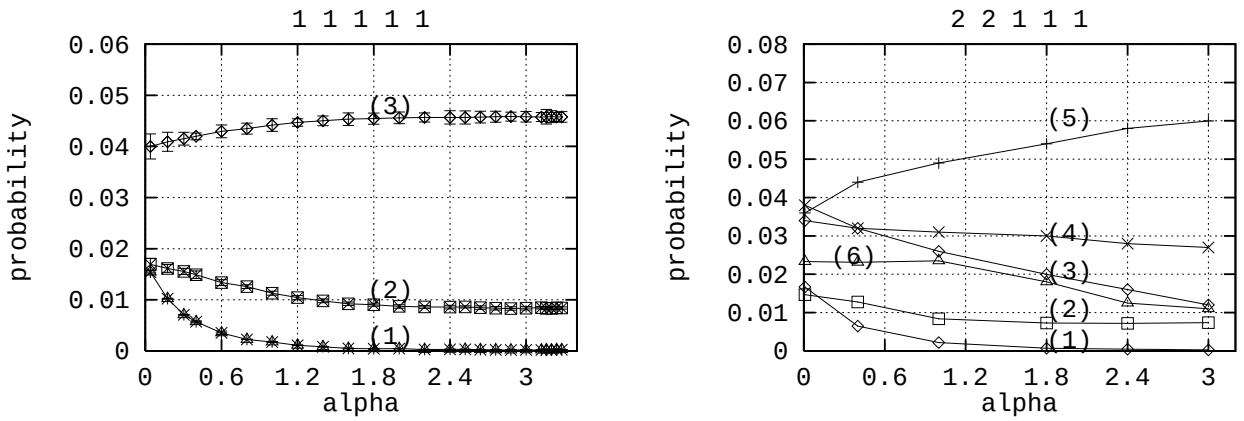


FIG. 10. Correlations between hidden and output units in the four machines. w stands for the units attached to that weight. This results were taken from simulation with $N = 50$.



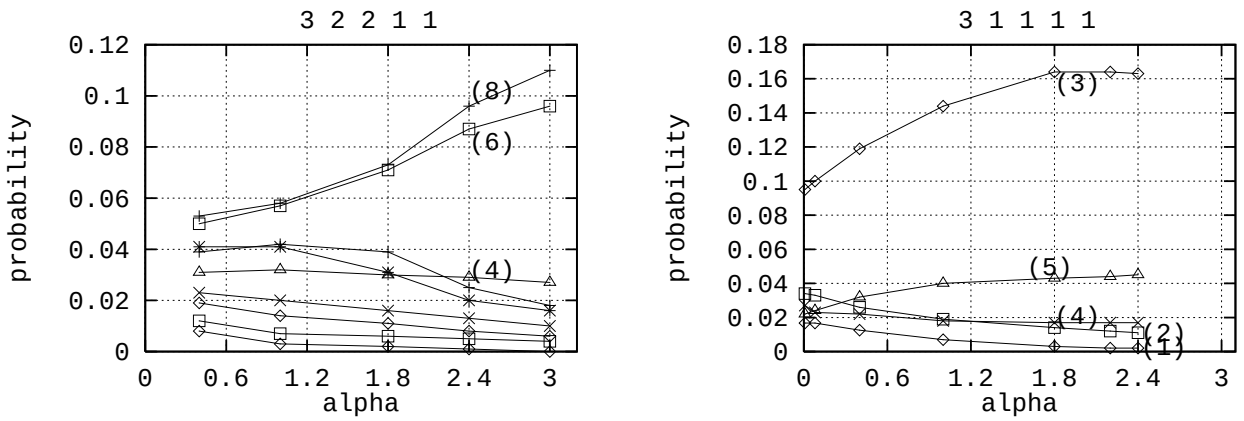


FIG. 11. Probability of classes for all machines. The values relate to IR with $S^{out} = +1$. The numbers in parantheses relate to the groups as labeled in tables II, III, IV and V (the low probability groups in machine '32211' are ignored to prevent confusion).

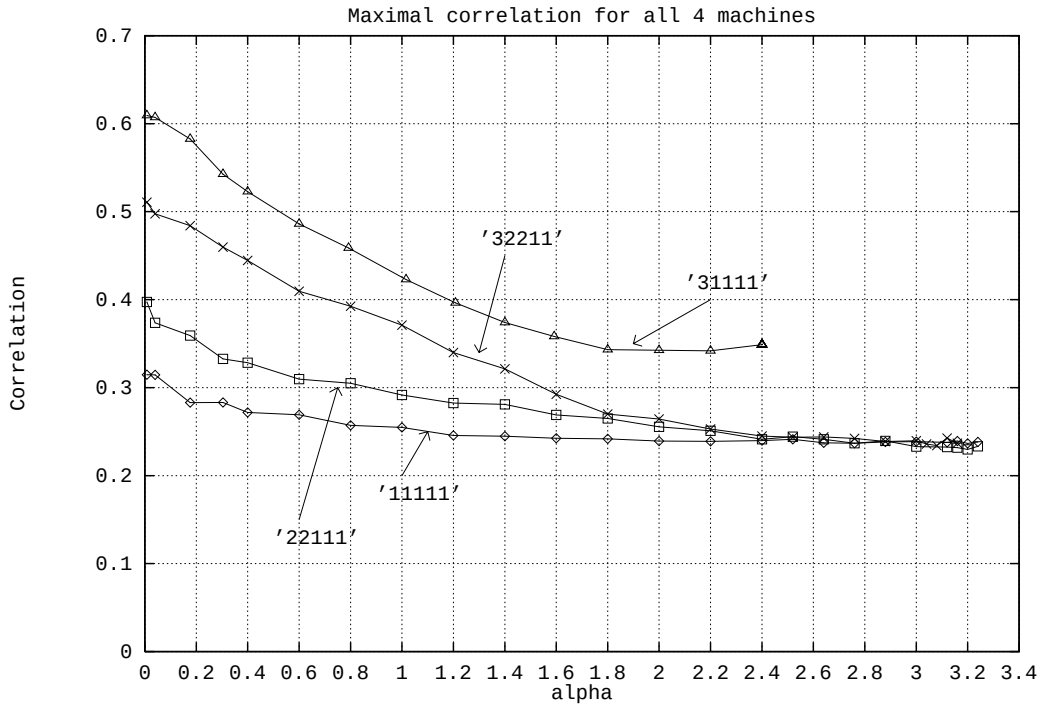


FIG. 12. Maximal correlation (with output) for all four machines. Each point represent the most correlated unit at each machine.

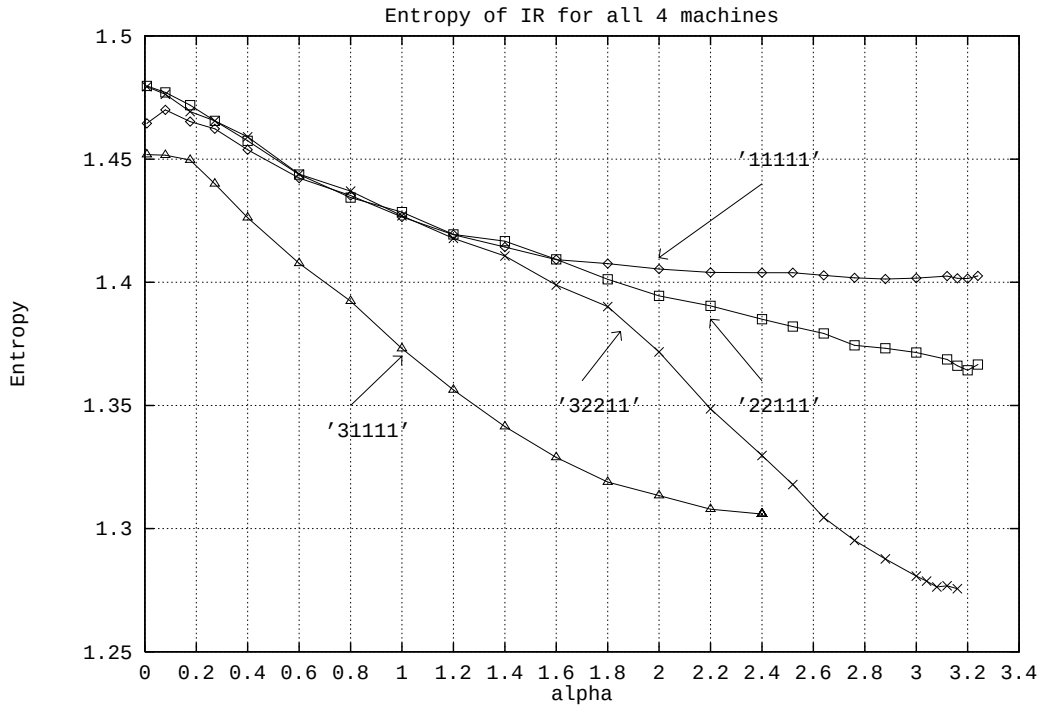


FIG. 13. Entropy of all four machines. The entropy was calculated from all 32 IR by eq. 17 where p_i is the probability of the i'th IR. The values of p_i matches, of course, those of Fig. 11.

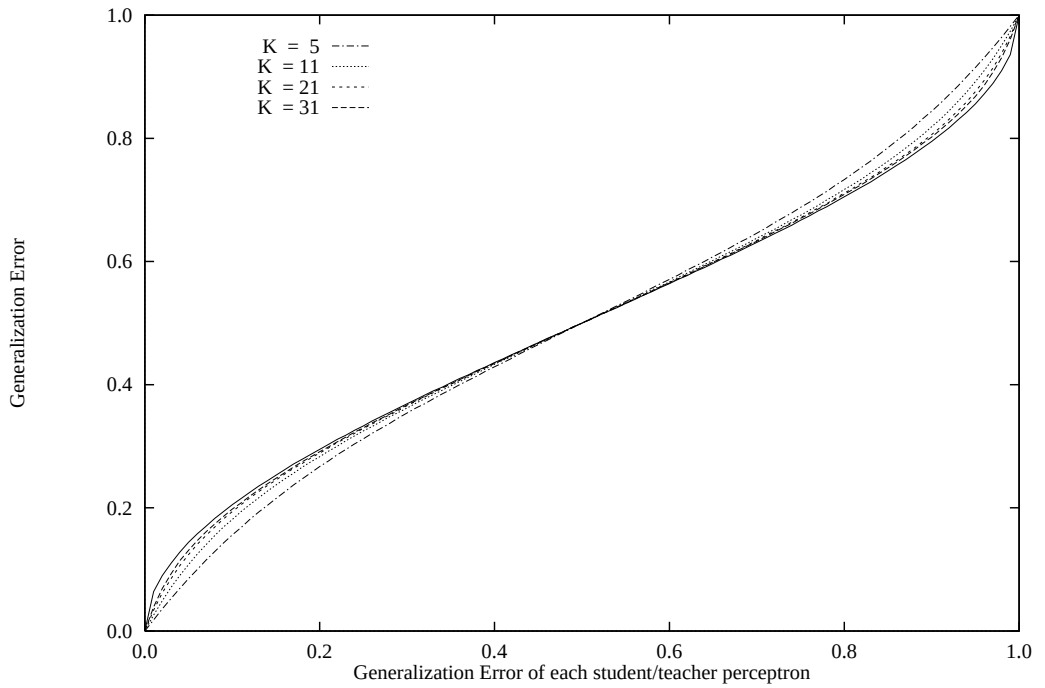


FIG. 14. CM *generalization function* for different values of K as a function of ϵ. The solid line is the replica symmetric result for $1 << K << N$ obtained by Schwarze and Hertz [20]

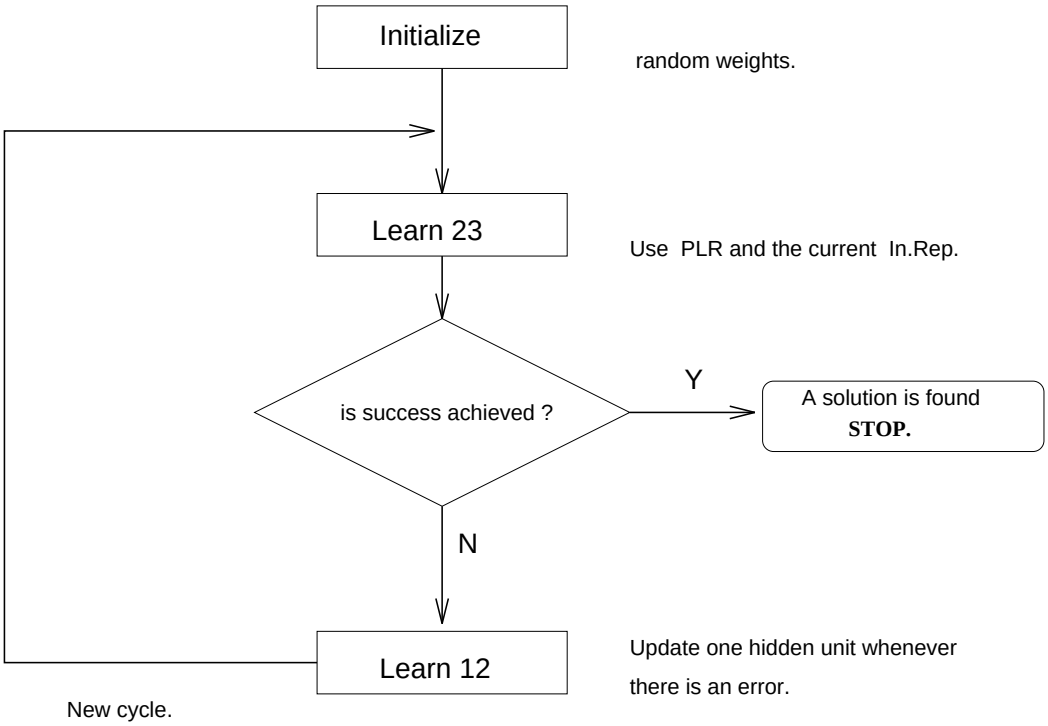


FIG. 15. Flow chart of the CHIR2 algorithm.

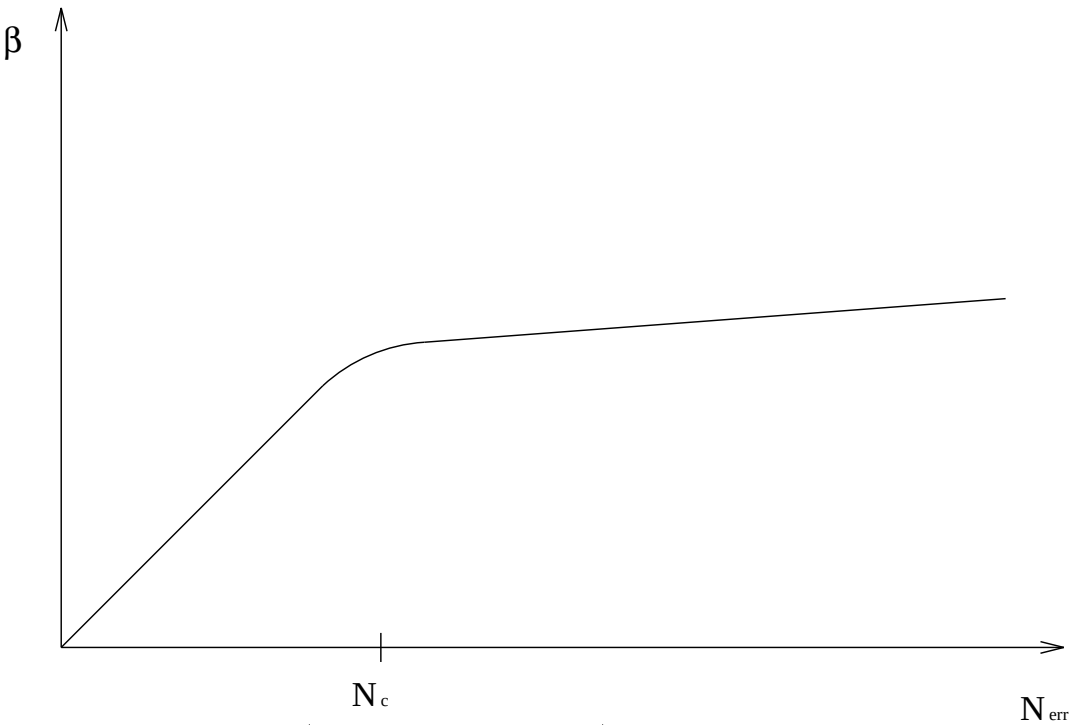


FIG. 16. Typical β (for the heating shedule) as a function of N_{err} - the number of misclassified patterns.

TABLES

TABLE I. Critical capacity of the four constituents machines and their union, the general $K = 5$ 2LP.

	11111	22111	32211	31111	union
$N = 25$	$\alpha_c \simeq 3.55$	$\alpha_c \simeq 3.55$	$\alpha_c \simeq 3.35$	$\alpha_c \simeq 2.8$	$\alpha_c \simeq 3.55$
$N = 50$	$\alpha_c \simeq 3.4$	$\alpha_c \simeq 3.38$	$\alpha_c \simeq 3.33$	$\alpha_c \simeq 2.4$	$\alpha_c \simeq 3.4$

TABLE II. Classes of Internal Representations in Committee Machine .

w_i	$\tilde{S}$	c	$\tilde{S}$	c	$\tilde{S}$	c
1	+	1	+	3/5	+	1/5
1	+	1	+	3/5	+	1/5
1	+	1	+	3/5	+	1/5
1	+	1	+	3/5	−	1/5
1	+	1	−	3/5	−	1/5
degen.	1		5		10	
label	1		2		3	

TABLE III. Classes of Internal Representations in machine 22111.

w_i	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c
2	+	1	+	1	+	0	+	1	+	0	+	1
2	+	1	+	1	−	0	+	1	−	0	+	1
1	+	1	+	1/3	+	1	+	−1/3	+	1/3	−	−1
1	+	1	+	1/3	+	1	−	−1/3	+	1/3	−	−1
1	+	1	−	1/3	+	1	−	−1/3	−	1/3	−	−1
degen.	1		3		2		3		6		1	
label	1		2		3		4		5		6	

TABLE IV. Classes of Internal Representations in machine 31111.

w_i	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c
3	+	1	+	1	−	−1	+	1	+	1
1	+	1	+	1/2	+	1	+	0	+	−1/2
1	+	1	+	1/2	+	1	+	0	−	−1/2
1	+	1	+	1/2	+	1	−	0	−	−1/2
1	+	1	−	1/2	+	1	−	0	−	−1/2
degen.	1		4		1		6		4	
label	1		2		3		4		5	

TABLE V. Classes of Internal Representations in machine 32211.

w_i	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c	S_i	c
3	+	1	+	1	+	1	+	1	+	1	+	1	−	−1	−	−1	+	1
2	+	1	+	1	+	0	+	0	+	1	−	−1	+	1	+	1	+	0
2	+	1	+	1	−	0	−	0	+	1	−	−1	+	1	+	1	−	0
1	+	1	+	0	+	1	+	0	−	−1	+	1	+	1	+	0	−	−1
1	+	1	−	0	+	1	−	0	−	−1	+	1	+	1	−	0	−	−1
degen.	1		2		2		4		1		1		1		2		2	
label	1		2		3		4		5		6		7		8		9	