Complex Systems 4 (1990) 573–586

Training Feed Forward Nets with Binary Weights via a Modified CHIR Algorithm

D. Saad
E. Marom
Faculty of Engineering, Tel Aviv University,
Ramat Aviv 69978, Israel

Abstract. Learning by choice of internal representations (CHIR) is a learning algorithm for a multilayer neural network system, introduced by Grossman et al. [2, 3], based upon determining the internal representations of the system as well as its internal weights. In a former paper [8] we have shown a method for deriving the CHIR algorithm whereby the internal representations (IR) as well as the weights are allowed to be modified via energy minimization consideration. This method is now applied for training a feedforward net with binary weights, supplying a convenient tool for training such a net. Computer simulations show a fast training process for this algorithm in comparison with backpropagation [7] and the CHIR [2, 3] algorithms, both used in conjunction with a feedforward net with continuous weights. These simulations include the restricted cases of parity, symmetry, and parity–symmetry problems.

1. Introduction

Training a feedforward net with binary weights is an issue of significant importance, since it allows implementations with considerably simpler elements. The main problem in training such a net results from the fact that the most common training method, backpropagation (BP) [7], cannot directly handle binary weights. An alternative training method, which can handle perceptrons with binary weights, has been introduced by Venkatesh [10], while other attempts tackled the multilayer net problem differently [3, 1, 5]. These methods are based on random weight flips when erroneous output occur, in a continuously decreasing number, until a solution is found.

Learning by choice of internal representations (CHIR) [2, 3] is a neural network learning algorithm based upon introduction of changes in both the IR of a discrete binary valued multilayer system and in its weights, for an ensemble of learned vectors. The changes are designed to improve the output value of each layer with respect to the current weights and the current IR of the preceding layer, so that a neural network-based recognition or classification system with a faster training convergence rate is obtained. Once the IR are redefined, the perceptron convergence rule [6] is applied for modifying the weights. In a former paper, Saad and Marom [8] studied an energy minimization approach for the CHIR algorithm and showed that generating changes in the IR satisfies an inherently consistent convergence mechanism. The above-mentioned method can also be easily implemented for training a net with binary weights, after several modifications are introduced. In this paper we present a training algorithm for a feedforward net with binary weights based upon the modified CHIR algorithm. The performance of the algorithm was tested on standard computer simulation problems, like those tested by Grossman [2, 3] and Tesauro and Janssen [9].

2. An energy minimization approach for a discrete valued net

We will define an energy function in a similar way as defined in the BP algorithm [7]:

$$E = \sum_{p=1}^{P} E^p = \sum_{p=1}^{P} \sum_{i=1}^{N^H} \left(v_i^{H,p} - \tau_i^p \right)^2 \tag{2.1}$$

where τ^p is the desired output vector related to the p vector out of P training vectors used in the training procedure; $v^{H,p}$ is a discrete value output vector of an H layer system, related to the p training input vector, and N^H is the number of neurons in the output layer.

The output vector $v^{H,p}$ is obtained by the following equation:

$$v_i^{H,p} = f\left\{ \sum_{j=1}^{N^{H-1}} W_{ij}^H v_j^{H-1,p} \right\} = f\left\{ u_i^{H,p} \right\} \tag{2.2}$$

where

$$u_i^{H,p} = \sum_{j=1}^{N^{H-1}} W_{ij}^H v_j^{H-1,p}$$

The operator f, which represents the neural response, is considered to be a nonlinear operator acting on the product of the weight matrix W^H, connecting layers $H-1$ and H, and the IR of the preceding layer $v^{H-1,p}$. N^{H-1} is the number of neurons in the $H-1$ layer.

As in the BP algorithm, we will search for a procedure to minimize the energy E. However, we will allow at this time direct modifications of the IR $v^{H-1,p}$ as well as changes in the weights W^H.

The derivative of the energy function E is of the form

$$\frac{dE}{dt} = \frac{\partial E}{\partial W^H} \frac{dW^H}{dt} + \sum_{p=1}^{P} \frac{\partial E}{\partial v^{H-1,p}} \frac{dv^{H-1,p}}{dt} \tag{2.3}$$

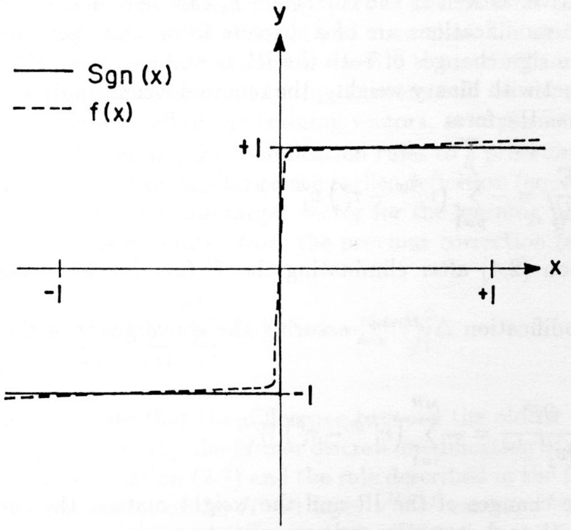

Figure 1: Typical function approximating sgn(x).

where t is the training index and the derivatives are applied to each interconnection weight and each neuron of the IR vector. The following changes in W^H and $v^{H-1,p}$ will ensure a negative contribution to the energy function:

$$\Delta W_{ij}^{H} = -\frac{\partial E}{\partial W_{ij}^{H}} = -\eta \sum_{p=1}^{P} \left(v_i^{H,p} - \tau_i^p\right) f'\left[u_i^{H,p}\right] v_j^{H-1,p} \tag{2.4}$$

$$\Delta v_j^{H-1,p} = -\frac{\partial E}{\partial v_j^{H-1,p}} = -\sum_{i=1}^{N^H} \left(v_i^{H,p} - \tau_i^p\right) f'\left[u_i^{H,p}\right] W_{ij}^{H} \tag{2.5}$$

where η is a convergence coefficient and f' stands for the derivative of f with respect to the argument in the bracket. Equation (2.4) resembles the weight matrix modification obtained by the BP algorithm with one change: the operator f, which represents the neural response, is now defined as the sgn function (note that applying these equations for a continuous valued net retrieves BP equations; see Saad and Marom [8]). Since the sgn(x) function has zero derivative for most of its range, we will approximate it by a function that has a small, almost constant, positive derivative along most of its dynamic range.An example of such a function is shown in figure 1. We will neglect the region near $x = 0$ by defining the widthof this area to be smaller than our resolution. The IR and weight matrix modifications, expressed in equations (2.4) and (2.5) are thus applicable also for the *discrete IR and discrete weight case.* One notes, however, that the derivative function f', which is always positive, as well as the coefficient η, can be omitted in both equations since both modifications are of a discrete form, and therefore we are interested only in sign changes of both the IR as well as the weights.

Therefore, for a net with binary weights, the required weight matrix modification $\Delta W_{ij}^{H-1,p}$ has the form

$$\Delta W_{ij}^{H} = -\frac{\partial E}{\partial W_{ij}^{H}} = -\sum_{p=1}^{P}\left(v_i^{H,p} - \tau_i^p\right) v_j^{H-1,p} \tag{2.6}$$

derived from equation (2.4) after eliminating the f' function as discussed above.

Likewise, the modification $\Delta v^{H-1,p}$, assuring the convergence of the energy function, is

$$\Delta v_j^{H-1,p} = -\frac{\partial E}{\partial v_j^{H-1,p}} = -\sum_{i=1}^{N^H}\left(v_i^{H,p} - \tau_i^p\right) W_{ij}^{H} \tag{2.7}$$

Allowing discrete changes of the IR and the weight matrix, the energy function decreases with each iteration and converges to a minimum value. A flip in the IR or in the weight matrix will be enforced whenever the modifications $\Delta v_j^{H-1,p}$ and ΔW_{ij}^{H} are of opposite sign in relation to the values of $v_j^{H-1,p}$ and W_{ij}^{H} respectively.

The modifications in the IR should be performed for each training vector $v^{H,p}$ taken one at a time, while the weight matrix modification should be performed for all of the vectors in parallel due to the discrete nature of the weights.

To apply these rules we start by modifying the last weight matrix W^H, according to equation (2.6), by summing up all of the contributions from the various input vectors. A bit flip in a weight matrix element is enforced whenever ΔW_{ij}^{H} is of opposite sign in relation to W_{ij}^{H}. Since each weight modification affects the common output in conjunction with other weights ir the same layer, we should not modify all of the weights at once. Therefore, we change part of the weights simultaneously in each iteration, choosing ther randomly from the set of the most contributing weight changes. This als inserts stochasticity to the algorithm. The most contributing weight change are determined by the value of the accumulated weight modification ΔW^H, a will be explained later on. Since all weights are mutually related, we perfor the W^H elements modification several times (1–5 in simple problems, 10–2 in more complicated problems), whereby the random choice of the weight to be modified is different at eachiteration.

Once the W^H weights have stabilized, we perform the IR modificatio according to equation (2.7) in a similar manner, i.e., an IR bit flip will enforced whenever the IR modification $\Delta v_i^{H-1,p}$ is of opposite sign in relatio to $v_i^{H-1,p}$. As in the weight matrix modification case, we cannot modify a of the neurons at once since this might create a too restrictive set of IR that cannot always be implemented via the weight matrix of the former layer W^{H-1}. Therefore, we select only part of the neurons in each IR, the most contributing ones, to be changed and then allow the system to adapt to the new IR via minor modifications in the W^{H-1} matrix (here as well, the criteria for choosing the most contributing neurons will be the accumulated vector modification over all of the training vectors, as explained later on).

For implementing the modification rules to a preceding layer, we define a new energy function similar to the earlier definition (equation 2.1). However, for an internal layer the target vector for the learning procedure will be the modified IR as computed from the previous correction (equation 2.7) :

$$E = \sum_{p=1}^{P} E^p \sum_{p=1}^{P} \sum_{i=1}^{N^{H-1}} \left(v^{H-1,p} - v_{\text{new}}^{H-1,p}\right)_i^2 \tag{2.8}$$

One should note that the difference between the old IR $v^{H-1,p}$ and the new one $v_{\text{new}}^{H-1,p}$ is actually the former discrete modification of the IR, i.e., $\Delta v^{H-1,p}$ obtained by equation (2.7) and the rule described in the following paragraph. The output vector of the $H-1$ layer $v^{H-1,p}$ can now be expressed in terms of the previous weights W^{H-1} and IR $v^{H-2,p}$ as shown below :

$$v_i^{H-1,p} = f\left\{\sum_{j=1}^{N^{H-2}} W_{ij}^{H-1} v_j^{H-2,p}\right\} \tag{2.9}$$

Applying the same procedure for obtaining the required modifications for the weight matrix and the IR in this layer, one gets

$$\Delta W_{ij}^{H-1} = -\frac{\partial E}{\partial W_{ij}^{H-1}} = -\left(v^{H-1,p} - v_{\text{new}}^{H-1,p}\right)_i v_j^{H-2,p} \tag{2.10}$$

$$\Delta v_j^{H-2,p} = -\frac{\partial E}{\partial v_j^{H-2,p}} = -\sum_{i=1}^{N^{H-1}} \left(v^{H-1,p} - v_{\text{new}}^{H-1,p}\right)_i W_{ij}^{H-1} \tag{2.11}$$

We then apply the same procedure for all remaining layers and examine the performance of the system for the exhaustive set of input vectors. If the system does not succeed fully in the classification task, we repeat the training epoch until the training procedure iscompleted.

3. Complete learning algorithm

Up until now we described the main lines of the training algorithm. We will now follow in detail the algorithm, stressing its main computational steps.

3.1 Modifying the weight matrix W^H

Addressing the randomly initialized system with various input vectors, one receives a set of corresponding output vectors as well as a set of internal representations. Based on the desired outputs, by applying equation (2.6) one obtains a set of weight matrix possible modifications ΔW_{ij}^H. In order to determine which weights should be modified, we examine the energy contribution of the possible flips. The energy contribution due to a weight flip is

$$\begin{aligned}\Delta E &= \sum_{p=1}^{P}\sum_{i=1}^{N^H}\left[f\left(\sum_{j=1}^{N^{H-1}}\left(W_{ij}^H+\Delta W_{ij}^H\right)v_j^{H-1,p}\right)-\tau_i^p\right]^2\\ &\quad-\left[f\left(\sum_{j=1}^{N^{H-1}}W_{ij}^H v_j^{H-1,p}\right)-\tau_i^p\right]^2 \qquad (3.1)\end{aligned}$$

Taking in account the fact that the vectors are discrete, one can expand ΔE around $u^{H,p}$ (as defined in equation 2.2) to obtain

$$\Delta E \simeq -\sum_{p=1}^{P}\sum_{i=1}^{N^H}\sum_{j=1}^{N^{H-1}} f'\left\{u_i^{H,p}\right\}\Delta W_{ij}^H v_j^{H-1,p}\tau_i^p \qquad (3.2)$$

Since f' is a positive constant, as indicated earlier, equation (3.2) becomes

$$\Delta E \simeq -\sum_{p=1}^{P}\sum_{i=1}^{N^H}\sum_{j=1}^{N^{H-1}} \Delta W_{ij}^H v_j^{H-1,p}\tau_i^p \qquad (3.3)$$

Thus, the weight matrix flips with maximal contribution will be those for which expression (3.3) is minimal; we shall therefore modify only these neurons, having the maximal $|\Delta W_{ij}^H|$. Note that the decrease of the energy function is guaranteed by the change rule of ΔW^H defined in equation (2.6) and the explanation given following equation (2.7).

In order to insert stochasticity into the algorithm, out of the 1/3 most contributing connection flips we randomly choose one quarter of the weight matrix elements to be modified.

We also found that a certain modification of the change rule (equation 2.6) is useful. In the continuous case it was reasonable to assume that proper output vectors do not contribute to weight modification, as indeed results from equation (2.6). However,in the discrete weight case, a proper output should enhance the current state of the weight and counterbalance the need for a flip as perhaps demanded by other terms. Thus a certain fraction of the current weight polarity (βW_{ij}^H) should be added to the summation (equation 2.6) for each proper output vector. The value of β was 1 in our simulations, although it can be varied.

In order to stabilize the system, one should repeat the weight modification procedure several times. Conforming to Grossman's [2] notation we call I_{23} the number of times the weights are modified before attempting to modify the IR. For some of our simulations we carried this procedure only once while for other cases we used several repetitions, as indicated in the paragraph concerning computer simulations.

3.2 Modifying the IR

Once the weights are determined, one can modify all the IR according to equation (2.7) and the following rule. As with the weight modifications, we do not apply the modification process to all neurons but only to those that will contribute the most. The selection of such neurons is obtained via the energy expression (equation 2.1), which provides an estimate of the contribution resulting from an IR bit flip:

$$\begin{aligned}\Delta E &= \sum_{p=1}^{P}\sum_{i=1}^{N^H}\left[f\left(\sum_{j=1}^{N^{H-1}} W_{ij}^H\left(v_j^{H-1,p}+\Delta v_j^{H-1,p}\right)\right)-\tau_i^p\right]^2 \\ &\quad -\left[f\left(\sum_{j=1}^{N^{H-1}} W_{ij}^H v_j^{H-1,p}\right)-\tau_i^p\right]^2 \qquad (3.4)\end{aligned}$$

Here, too, we use the fact that the vectors are discrete and expand ΔE around $u^{H,p}$ to obtain

$$\Delta E \simeq -\sum_{p=1}^{P}\sum_{i=1}^{N^H}\sum_{j=1}^{N^{H-1}} f'\left\{u_i^{H,p}\right\} W_{ij}^H \Delta v_j^{H-1,p}\tau_i^p \qquad (3.5)$$

Since f' is a positive constant, as indicated earlier, equation (3.5) becomes

$$\Delta E \simeq -\sum_{p=1}^{P}\sum_{i=1}^{N^H}\sum_{j=1}^{N^{H-1}} W_{ij}^H \Delta v_j^{H-1,p}\tau_i^p \qquad (3.6)$$

The decrease of the energy function is guaranteed by the change rule of Δv^{H-1} (equation 2.7 and the following paragraph). The neural flips with maximal contribution will be those for which expression (3.6) is minimal or $|\Delta v|$ is maximal; we will therefore modify only theseneurons.

The number of neurons to be modified in the IR seems to have a minor effect upon the rate of convergence and the percentage of converging cases. The "thumb rule" chosen by us is 1/3 of the number of neurons in the hidden layer.

3.3 Modifying the weight matrix W^{H-1}

Once the IR of the $H-1$ layer have been defined one can modify the weight matrix W^{H-1} according to similar rules as those applied for the W^H matrix.

To perform the changes in any specific layer h, the network is addressed with the same set of learning vectors. If the output vector differs from the desired one, we first update the IR of layer h with the value defined in the former stage and then apply the weight matrix modification ΔW^h. If, on the other hand, the output is correct, we adopt the current IR as the proper one and go on to the other vectors.

The modification of the W^{H-1} matrix will be handled in a similar way to that used for the weight matrix W^H; here too we apply the modification of the change rule (equation 2.6), so that a proper IR contributes to the enhancement of the current weight polarity when evaluating the summation of ΔW^h. The value of this contribution used in our simulations is indicated below as α in the paragraph concerning computer simulations.

Another modification that has been considered is to strengthen the weight matrix modification ΔW^h with a coefficient of the current weight sign whenever a proper *output vector* is obtained. Our simulations show that the system performs better without such a coefficient.

In order to stabilize the system one should repeat this procedure several times (I_{12} per previously mentioned notation). The number of repetitions used for each problem is indicated in the computer simulations section.

It seems that an improvement in the convergence rate can be achieved by optimization of the free parameters described above. However no optimization has been carried out for these parameters.

4. Computer simulations

To compare the performance of this procedure with those provided by BP and the CHIR mechanism (with continuous weights), we examined some of the toy problems also used by Grossman et al. [2, 3] and by Tesauro and Janssen [9] under the same conditions. The problems examined are parity, symmetry, and parity–symmetry.

Before discussing the simulations carried out, we will explain the various methods of representing the results so that consistency with the data presented by Grossman et al. and Tesauro and Janssen is preserved.

The unit used for measuring convergence rates is time steps (or *sweeps*). This unit, used by Grossman et al. [2, 3], measures each internal iteration or IR modification as one time step;therefore each overall cycle equals $I_{12}+I_{23}+1$ time steps (using Grossman's notation, I_{12} for the W^2 matrix and I_{23} for the W^3 matrix). Two other conventions that we use are t_m, which represents the median number of time steps required for convergence, and τ, the average inverse parameter defined as

$$\tau = \left[\frac{1}{n} \sum_{k=1}^{n} \frac{1}{t_k} \right]^{-1} \tag{4.1}$$

where t_k is the number of iterations required for a successful learning procedure with certain initial conditions (nonconverging cases are regarded as if τ_k is infinite).

4.1 Parity

The definition of the parity criterion is to provide an output 1 when the number of +1 bits in the input vector is even and −1 otherwise. In our

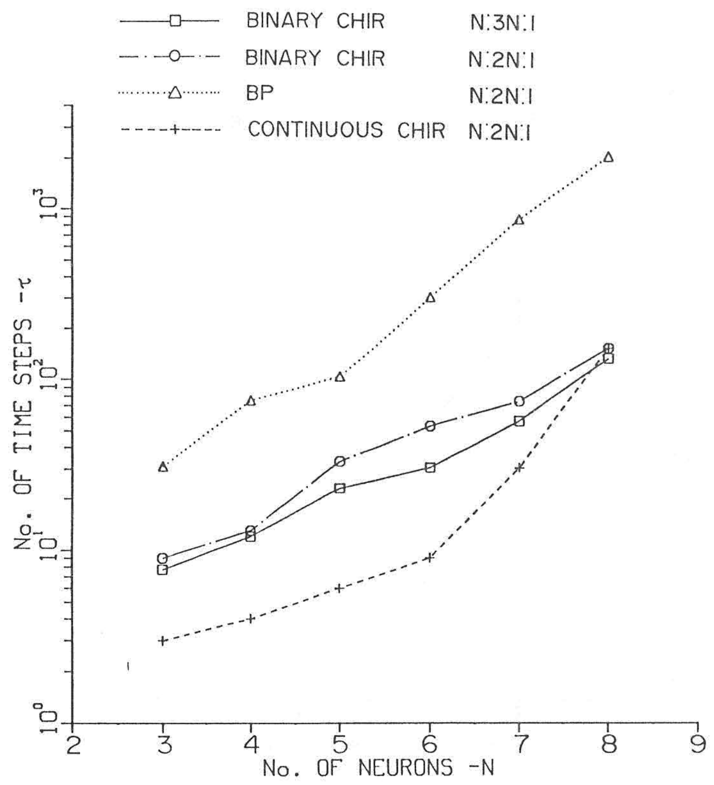

Figure 2: Parity: The average inverse of the number of sweeps required for training a network of the form $N : 2N : 1$ and $N : 3N : 1$ to solve the parity problem (even number of +1 neurons), using an exhaustive training set.

simulations, we used a configuration of $N = 3, 4, 5, 6, 7, 8$ input neurons, a $2N$ and $3N$ neuron hidden layer and a single output neuron. The maximal number of over all sweeps was 300, each iteration carried over the exhaustive ensemble. The weight update repetition parameters are $I_{12} = 1, 1, 3, 3, 3, 3$ and $I_{23} = 1, 1, 1, 3, 3, 3$ respectively for the N values mentioned above. The enhancement factors α that we used for the $N : 3N : 1$ and $N : 2N : 1$ configurations were $\alpha = 0.6, 0.6, 0.8, 1.3, 1.5, 1.8$ and $\alpha = 0.1, 0.1, 0.8, 1.5, 1.5, 0.05$ respectively. A comparison between the performance of our simulations, BP, and continuous CHIR appears in figure 2.

We then used a specific parity problem (with five input neurons) to estimate the effect of changing the number of neurons in the hidden layer upon convergence rates and percentage of converging cases. The results of these

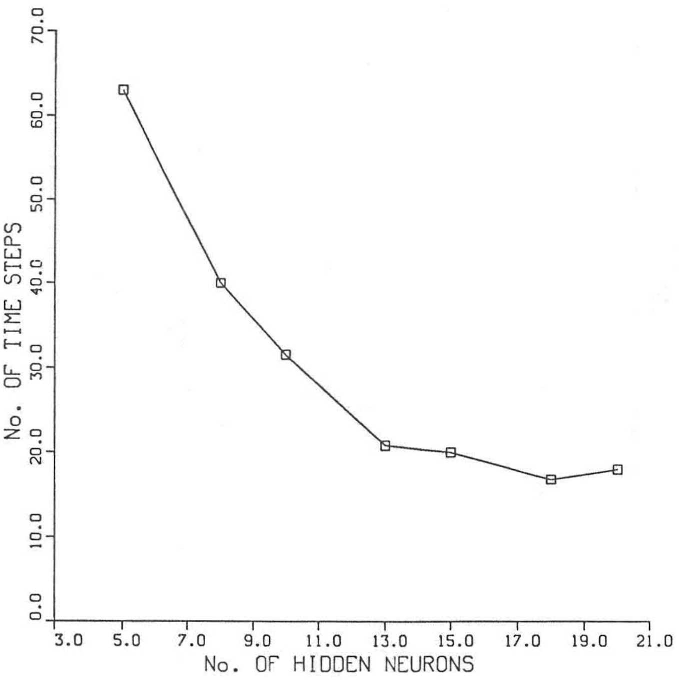

Figure 3: The average inverse of the number of sweeps required for solving the parity-5 problem using various number of hidden neurons.

simulations appear in figures 3 and 4. In these simulations all parameters remain fixed as for the $N = 5$ case mentioned above.

4.2 Symmetry

In this case the desired output is +1 if the input vector is symmetric around its center and −1 otherwise. The learning procedure was tested for $N = 2$, 4, 6, 8, 10 input neurons, N hidden neurons, and a single output neuron. The maximum number of over all iterations was 200, each iteration carried over the exhaustive ensemble. The repetition parameters used for this set are $I_{12} = 3, 3, 5, 5, 5$ and $I_{23} = 1, 1, 3, 3, 3$ and the enhancement factors $\alpha = 0.05, 0.05, 0.08, 0.1, 0.1$.

This problem cannot be solved by a net of the above mentioned configuration with *all* thresholds and weights of a binary form. Therefore, we allowed the threshold of the output neuron to be continuous. Note that this

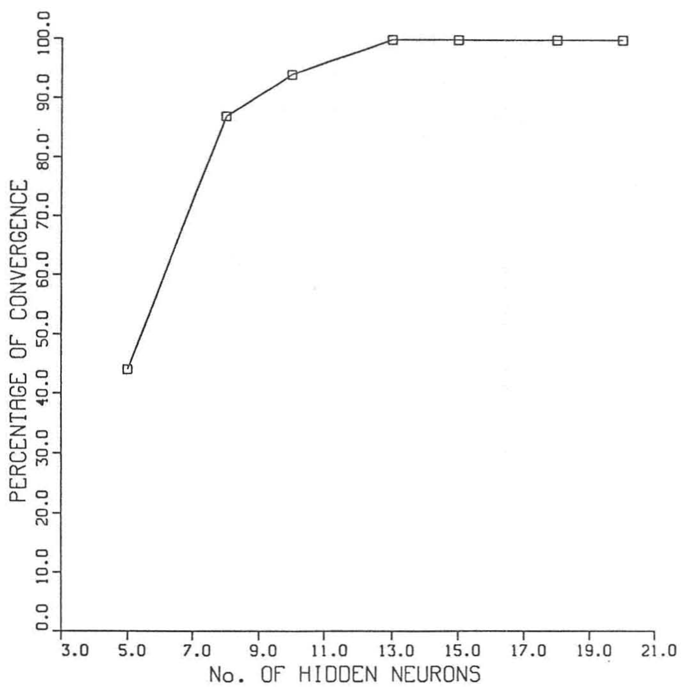

Figure 4: The percentage of converging cases for the parity-5 problem using various number of hidden neurons.

compromise results from the inherent limitations of a net with binary weights and not from limitations of the learningprocess.

Figure 5 compares the median number of pattern representations required for a fully successful learning based on this algorithm versus the results obtained by Grossman et al. [2].

4.3 Parity–Symmetry

The parity–symmetry problem combines the two problems discussed above into one system, providing a two-neuron output: one represents the parity of the input vector and the other its symmetry. The simulations included $N = 4, 5, 6$ input neurons, $3N$ hidden neurons, and two output neurons. The repetition parameters used here are $I_{12} = 7, 11, 30$, $I_{23} = 3, 3, 9$, and $\alpha = 0.1, 0.05, 0.07$.

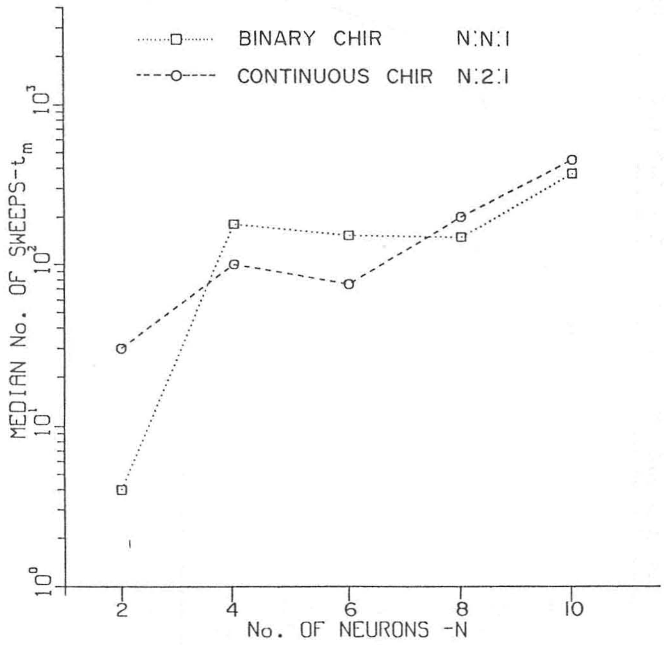

Figure 5: Symmetry: Median number of sweeps required for training a network of the form $N : 3N : 1$ to solve the symmetry problem using an exhaustive training set.

Here, too, we cannot solve the problem using a net with both weights and thresholds of a binary form; we therefore allowed the thresholds to be continuous while the weights remain binary.

Figure 6 compares our results to those presented by Grossman [3] for the continuous value weight CHIR configuration. Our results show poorer learning rate than the continuous CHIR but one should remember that binary networks are much more economical for implementation.

5. Conclusion

We applied in this work a modified version of the CHIR algorithm for training a feedforward neural network consisting of *binary weights*, whereby *both the IR and the interconnection weights* are simultaneously modified. The convergence procedure and the simulation results seem very promising, rais-

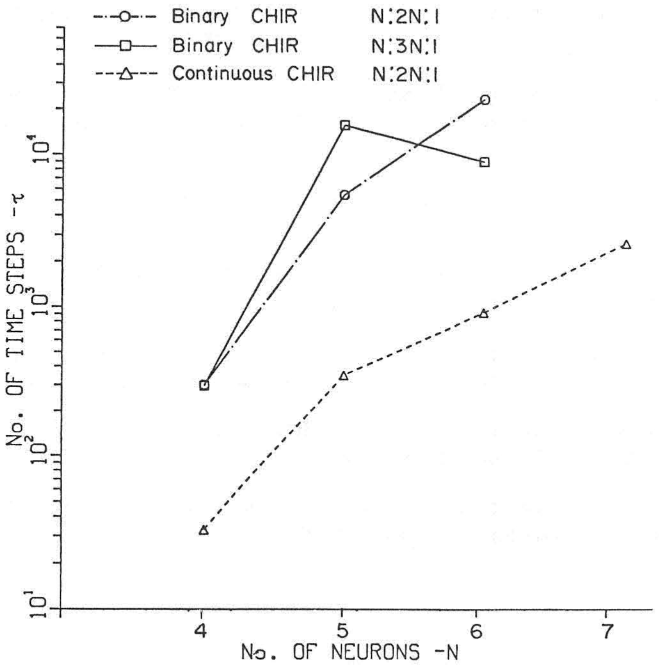

Figure 6: Parity–Symmetry: The average inverse of the number of sweeps required for training a network of the form $N : 3N : 2$ to solve the parity–symmetry problem using an exhaustive training set.

ing the possibility of using feedforward neural networks with binary weights trained by a rapidly converging learning algorithm.

Acknowledgments

The work of one author (D.S.) was supported by a stipend from the Buchmann fund, which is gratefully acknowledged.

References

[1] E. Amaldi and S. Nicolis, "Stability-capacity diagram of a neural network with Ising bonds," *Jour. de Phys.*, **50** (1989) 2333.

[2] T. Grossman, R. Meir, and E. Domany, "Learning by choice of internal representations," *Complex Systems*, **2** (1989) 555.

[3] T. Grossman, "The CHIR algorithm: A generalization for multiple output and multilayered networks," to be published in *Complex Systems*.

[4] T. Grossman, "The CHIR algorithm for feedforward networks with binary weights," preprint.

[5] H. Kohler, S. Diederich, W. Kinzel, and M. Opper, "Learning algorithm for a neural network with binary synapses," *Zeitschrift fur Physik B*, **78** (1989) 333.

[6] M.L. Minsky and S.A. Papert, *Perceptrons*, Third Ed. (MIT Press, Cambridge, MA, 1988).

[7] D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing*, Vol. 1 (MIT Press, Cambridge, MA, 1986).

[8] D. Saad and E. Marom, "Learning by choice of internal representations: An energy minimization approach," to be published in *Complex Systems*.

[9] G. Tesauro and B. Janssen, "Scaling relationships in backpropagation learning," *Complex Systems*, **2** (1988) 39.

[10] S. Venkatesh, "Direct drift: A new linear threshold algorithm for learning binary weights on line," preprint (1989).