II. ІНФОРМАТИКА

УДК 681.32:007.58

THE INFLUENCE OF SELF-CONNECTION ON THE PERFORMANCE OF PSEUDO-INVERSE AUTOASSOCIATIVE NETWORKS

Dmitry O.Gorodnichy

The performance of pseudo-inverse autoassociative neural networks proposed by Personnaz et al. [1] and Kanter and Sampolinsky [2] is known to be affected by the value of self-connection of the network. This paper summarizes the results obtained on the phenomenon and describes the desaturation procedure based on partial reduction of self-connection, which is shown to improve the performance of the network. The optimal value of the desaturation coefficient, which is the coefficient self-connection reduction, is obtained.

Как известно, на эффективность псевдо-инверсных автоассоциативных нейронных сетей, предложенных Персонназом, Кантером и Самполинским, воздействует значение самосвязанности сети. В настоящей работе обобщаются результаты, описывающие процедуру разнасыщения, основанную на частичном уменьшении самосвязанности, которое улучшает свойства сети. Получено оптимальное значение коэффициента разнасыщения, являющегося коэффициентом уменьшения самосвязанности.

Як відомо, на ефективність псевдо-інверсних автоасоціативних нейронних мереж, запропанованих Персонназом та ін. [1] та Кантером і Самполінським [2], впливає значення самосполучення мережі. У роботі наведені результати, що описують процедуру рознасичення, яка грунтується на часковому зменшенні самосполучення, яке, як показано, уточнює ефективність мережі. Отримане оптимальне значення коефіцієнта рознасичення, що є коефіцієнтом зменшення самосполучення.

I. INTRODUCTION

An autoassociative memory is a system which, given a noisy version of a prototype (a pattern stored in the memory) as input, retrieves the original prototype as output (Figure 1a).

The idea of using a self-organizing fully connected network of binary neurons for designing an autoassociative memory is mainly attributed to Amary. As early as 1972 he showed [3] that such a network "remembers" some vectors as stable equilibrium states of the network, where the retrieval process consists in converging from an arbitrary state to a stable state. The main questions concerning the network are:

Ql: How to adjust or to calculate the parameters of the network, which are N^2 coefficients of the weight matrix C (N being the number of neurons), so that the desired prototypes $\vec{V}^1, \ldots, \vec{V}^M$ become stable states of the network?

Q2: How many prototypes M can be stored by a network of size N and how good is the retrieval capability of the network?

While the answer to the first question gives a learning rule, the answer to second one describes the efficiency of the rule. By 1977 mainly two learning rules had been considered [4], [5]. The first one is the Hebbian, correlation, or outer product rule. The second rule is closely related to Kohonen's generalized inverse approach [6], [7], which has been intensively used for linear associative memories since 1974. It is called the orthogonal, projection, or pseudoinverse learning rule and is the subject of this paper.

This paper is organized with the above two questions in mind. The formal description of autoassociative neural networks and their properties follows in the next section. In Section III we focus our attention on pseudo-inverse neural network (PINNs), which are neural networks designed with the pseudo-inverse learning rule. In Section IV we theoretically study the effect of self-connection reduction on the perfomance of PINNs, and Section V presents experimental data which show how, by appropriately choosing the coefficient of self-connection reduction, the retrieval capability of PINNs can be considerably enhanced. Conclusions resume the paper.

II. AUTOASSOCIATIVE NEURAL NETWORKS

An autoassociative neural network (AANN) is defined as a self-organizing network of N mutually interconnected two-state neurons, the evolution of which is determined by a synchronous update rule[1]:

$$y_i(t+1) = \operatorname{sign}[s_i(t)] = \begin{cases} 1, & \text{if } s_i(t) > 0, \\ -1, & \text{otherwise,} \end{cases} \qquad (1)$$

where $s_i(t) = \sum_{j=1}^{N} C_{ji} y_j(t)$ or in vector form:

$$\vec{Y}(t+1) = \operatorname{sign}[\vec{S}(t)], \ \vec{S}(t) = C\vec{Y}(t). \qquad (2)$$

1.A more general definition includes a threshold and allows asynchronous dynamics. These are not considered in this paper

Column vector $\vec{Y}(t)$ is referred to as the state of the network at time t and C is an $N \times N$ weight matrix (Figure 1).

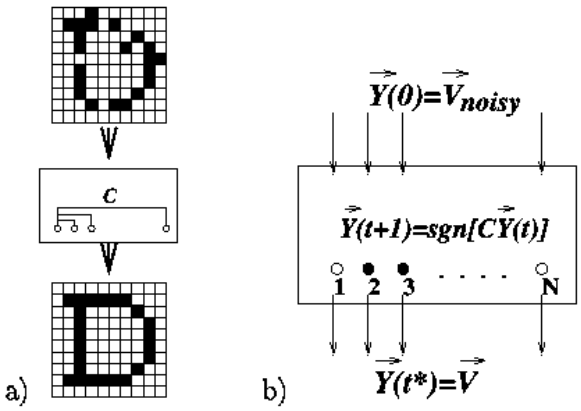

Figure 1 - Retrieval of a pattern $\vec{V}$ by an autoassociative network from an initial noisy pattern $\vec{V}_{noisy}$. The weights of the network determine the quality of retrieval

The self-organizing feature of the network is attributed to its ability to converge from an arbitrary state to a stable state, which is called an attractor. But how can we be sure that the network converges? What is the nature of these attractors? The following proposition gives the answer.

Proposition 1: As a result of free evolution, an AANN with symmetric weights converges to either a single stable state, called a static attractor, or a cycle consisting of exactly two states, called a dynamic attractor.

Proof: The easiest way to show this [8] is to use the "energy" function proposed by Reznik [9]:

$$E_R(\vec{Y}(t)) = -\frac{1}{2}\vec{Y}^T(t)\vec{S}(t-1) = -\frac{1}{2}\sum_{i=1}^{N}\left|s_i(t-1)\right|. \qquad (3)$$

As can be seen, if $C = C^T$

$$\begin{aligned}\vec{Y}^T(t)\vec{S}(t-1) &= \vec{Y}^T(t)C\vec{Y}(t-1) = \\ &= \vec{Y}^T(t-1)C\vec{Y}(t) = \vec{Y}^T(t-1)\vec{S}(t) = \\ &= \vec{Y}^T(t+1)\vec{S}(t) - [\vec{Y}^T(t+1) - \vec{Y}^T(t-1)]\vec{S}(t). \end{aligned} \qquad (4)$$

and

$$E_R(\vec{Y}(t)) = E_R(\vec{Y}(t+1)) + \sum_{h=1}^{H(t+1,\,t-1)}\left|s_h(t)\right|, \qquad (5)$$

where in the last summation h ranges over the neurons for which $y_h(t-1) \neq y_h(t+1)$, and $H(t+1, t-1)$ is the number of such neurons.

Eq. 5 shows that the "energy" function monotonically decreases with t until the second term of the equation becomes zero, which happens when $\vec{Y}(t-1) = \vec{Y}(t+1)$. That is, the network is guaranteed to converge to either a stable state: $\vec{Y}(t-1) = \vec{Y}(t) = \vec{Y}(t+1)$, or a cycle: $\vec{Y}(t-1) \neq \vec{Y}(t)$.

This proof can also be used for neural networks with thresholds (see [8]).

Note, that in the above proof we do not use the conventional Hopfield energy [10], which we will use later, defined as

$$E(\vec{Y}(t)) = -\frac{1}{2}\vec{Y}^T(t)\vec{S}(t) = -\frac{1}{2}\vec{Y}^T(t)C\vec{Y}(t), \qquad (6)$$

because it has been shown in [11] that in the case of synchronous dynamics this energy is not always monotonically decreasing.

The fact that an AANN may have cycles of length 2 is well known [12]. The above proof however differs from the previously obtained one [13], which is based on the deterministic nature of the evolution process.

A. The update flow technique

Proposition 1 gives us an easy way of identifying dynamic attractors - the indices of the updated neurons should be stored until the next update. If at the next update they are the same, the network is trapped in a dynamic attractor.

In [14] the update flow technique for updating the state of an AANN was introduced. It is based on storing only those neurons which have been updated in the last iteration, instead of storing the whole state vector. This technique is proved to be preferable on account of its high evaluation speed and suitability for parallel implementation. This technique also has another advantage in that it does not require extra processing to check for dynamic attractors.

In Section V we use this technique in simulations.

III. PSEUDO-INVERSE NEURAL NETWORKS

The pseudo-inverse learning rule (PILR) is obtained from the condition that all prototypes are stable states of the network:

$$C\vec{V}^m = \lambda_m\vec{V}^m, \ (m = 1, \ldots, M), \ \lambda_m > 0, \qquad (7)$$

which can be rewritten in matrix form for $\lambda_m = 1$ as

$$CV = V, \qquad (8)$$

where V is the matrix made of column prototype vectors.

Resolving this matrix equation for the case of linearly independent prototypes ($M \leq N$) gives us the rule:

$$C = V(V^TV)^{-1}V^T = VV^+, \qquad (9)$$

where $V^+ = (V^TV)^{-1}V^T$ is the pseudoinverse of matrix V,

hence the name of the rule.

The matrix defined by Eq. 9 is the orthogonal projection matrix in the subspace spanned by the prototype vectors $\{\vec{V}^m\}$, which explains the other names of the rule: orthogonal and projection.

Rewriting the PILR (Eq. 9) in scalar form gives

$$C_{ij} = \frac{1}{N} \sum_{m_1, m_2} V_i^{m_1} (C')^{-1}_{m_1 m_2} V_j^{m_2}, \tag{10}$$

$$\text{where } C'_{m_1 m_2} = \frac{1}{N} \sum_{i=1}^{N} V_i^{m_1} V_i^{m_2}. \tag{11}$$

This formula is not convenient for implementation because of the matrix inversion involved. Therefore, in simulations the iterative formula, which can be obtained from the Greville formula [15], is used[1]:

$$\begin{gathered} C^0 = 0, \\ C^m = C^{m-1} + \frac{(\vec{V}^m - C^{m-1}\vec{V}^m)(\vec{V}^m - C^{m-1}\vec{V}^m)^T}{\left\|\vec{V}^m - C^{m-1}\vec{V}^m\right\|^2}, \\ \text{if } \vec{V}^m \neq C^{m-1}\vec{V}^m, \end{gathered} \tag{12}$$

or in scalar form

$$\begin{gathered} C^0_{ij} = 0, \\ C^m_{ij} = C^{m-1}_{ij} + \frac{(v_i^m - s_i^m)(v_j^m - s_j^m)}{E^2}, \\ \text{where } E^2 = N - \sum_{i=1}^{N} v_i^m s_i^m, \quad s_k^m = \sum_{i=1}^{N} C^{m-1}_{ik} v_i^m. \end{gathered} \tag{13}$$

If $\vec{V}^m \neq C^{m-1}\vec{V}^m$, which means that the prototype $\vec{V}^m$ is a linear combination of other already stored prototypes, then the weight matrix remains unchanged.

A. Weights of the network

Assume now that the weights have been calculated according to the PILR (Eq. 9). Is it possible to say how many prototypes have been stored by just examining the weights? The following result obtained in [16], [17] gives the answer.

The weights of a FINN satisfy the following conditions:

$$\langle C_{ii} \rangle = \frac{M}{N}, \quad 0 \le C_{ii} \le 1, \tag{14}$$

$$\langle C_{ij}^2 \rangle = \frac{M(N-M)}{N^3}, \quad i \neq j, \tag{15}$$

$$\lim_{M \to N} |C_{ij}| = \langle |C_{ij}| \rangle \tag{16}$$

where $\langle C_{ij} \rangle$ designates the average (arithmetical mean) of values C_{ij}, and $|C_{ij}|$ is the absolute value of C_{ij}. The relationship between the weights and the number of prototypes M as given by the above equations is illustrated in Figure 2,[2] weight values being approximated as

$$C_{ii} \approx \langle C_{ii} \rangle \text{ and } |C_{ij}| \approx \sqrt{\langle C_{ij}^2 \rangle}. \tag{17}$$

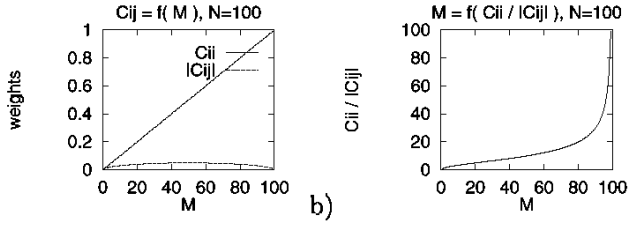

Figure 2 - Relationship between the weights and M. a) Diagonal C_{ii} and non-diagonal $|C_{ij}|$ weights as functions of M; b)The ratio $C_{ii}/|C_{ij}|$ indicates the number M of prototypes stored

This result makes PINNs very transparent (as opposed to the "black box" concept). In particular, it can be seen that as M increases, the diagonal weights start dominating the non-diagonal weights. This domination decreases the flexibility of the network, thereby downgrading its associative properties, as will now be shown using the concept of direct attraction radius.

B. Direct attraction radius

It follows from the derivation of the PILR that a PINN can store up to N prototypes as stable states. Yet for an autoassociative memory this is not enough because each prototype should also have a non-zero attraction basin. Therefore, it is important to know the direct attraction radius (DAR) of the network[3] which is defined as the largest Hamming distance Hattr from within which all vectors are guaranteed to converge to a prototype in a single iteration.

French researchers Personnaz et al. [1] were the first to calculate the DAR of a PINN. They showed that in the case of orthogonal prototypes

$$Hattr = \frac{N}{2M}. \tag{18}$$

Another formula for the DAR which relates the retrieval capabilities of the network to the values of its weights has

1. See [6], [9] for the derivation.
2. We use notation $z = f(x, y)$ in the figures to indicate that value z depends on values x and y.
3. The indirect attraction radius will be considered later.

been obtained in [16]:

$$\langle Hattr \rangle = \frac{\frac{1}{2} - \langle C_{ii} \rangle}{\langle |C_{ij}| \rangle}, \qquad (19)$$

where $\langle Hattr \rangle$ is the average DAR of a PINN.

Substituting the estimates of weights (Eqs. 14-17) into Eq. 19 gives

$$\langle Hattr \rangle = \frac{\frac{1}{2} - \frac{M}{N}}{\sqrt{\frac{M(N-M)}{N^3}}}. \qquad (20)$$

From Eqs. 18 and 20 it follows that a PINN stops exhibiting complete retrieval ($\langle Hattr \rangle < 1$) when $M = N/2$, and Eq. 19 explains this in terms of weights: increasing self-connections reduces the area of attraction. This conclusion will be used in Section IV. Now we outline other properties of PINNs.

C. Energy, cycles, and attractors

It has been shown by Personnaz et al. [1] that the energy of a PINN defined by Eq. 6 is monotonically decreasing during the evolution of the network. This means that, in spite of synchronous dynamics, PINNs do not have dynamic attractors. Rather they evolve until reaching a static attractor corresponding to either a global or a local minimum of the energy in the state space.

Global minima correspond to prototypes and their linear combinations and are given by

$$E(\vec{V}) = -\frac{1}{2}\vec{V}^T C \vec{V} = -\frac{N}{2}. \qquad (21)$$

It might be thought that these linear combinations of prototypes introduce a lot of undesirable stable states. However, as shown in [18], [19], this happens very rarely because of the binary nature of neurons. This means that most spurious attractors $\vec{Y}^*$ of the network correspond to local minima of energy, that is:

$$E(\vec{Y}^*) = -\frac{1}{2}\vec{Y}^{*T} C \vec{Y}^* > -\frac{N}{2}. \qquad (22)$$

This inequality can be used as a mechanism for detecting spurious attractors. Another mechanism for detecting spurious attractors in PINNs can be found in [20].

D. Implementations of the rule

The bottleneck of the PILR is that it is not local, which is crucial for optical and hardware implementations. Therefore, approximations of the PILR [21], [22], [23], [24] are used, and there exist a number of local iterative methods which converge to the PILR [25], [26], [27], [28], [29]. The most popular of these methods are the Widrow-Hoff rule [29] and the Gardner formula [28].

Comparisons of the performance of the PILR with that of other learning rules have been done [24], [30], [31], [32], [33] and the PILR remains among the most efficient rules, even if approximations are used.

For optical and hardware implementations of the PILR see [25], [29], [34].

IV. REDUCTION OF SELF-CONNECTION

The influence of self-connection on the performance of AANNs has been studied intensively for the Hebbian rule [35], [36]. In particular, it has been observed that the greater the self-connection, the less sensitive the network and the higher the number of spurious stable states. Kanter and Sompolinsky [2] also indicated that self-connections severely restrict the size of the attraction basins of PINNs. They compared two PINN models: one which contained self-connections and another which did not, and showed that the latter significantly outperforms the former, though the problem with the occurrence of cycles was observed.

The main objective of this and the next sections is to show that rather then being completely removed, selfconnections should only be partially reduced. This increases the attraction radius, improves the retrieval capability of the network, and does not cause too many cycles.

A. Desaturation

As a natural consequence of Eqs. 14, 15 and 19, a modification of the PILR, consisting of partial reduction of self-connections, has been suggested in [16], [17]. More specifically, the modification is described by the equation:

$$C_{ii}^D = D \cdot C_{ii}, \ (0 < D < 1); \ C_{ij}^D = C_{ij}, \ (i \neq j), \qquad (23)$$

which is applied after all weights C_{ij} have already been calculated according to the PILR (Eq. 9).

This modification is termed desaturation, and the coefficient of self-connection reduction D is termed the desaturating coefficient. The reasoning behind this is that by applying the modification we restore the balance of weights, thereby allowing a saturated network, i.e. the network which has already lost the retrieval capability as a result of storing too many prototypes, to exhibit retrieval again. In other words, we desaturate the network. The learning rule obtained with the modification (Eq.23) can be rewritten in matrix form as

$$C^D = C - (1 - D)\mathrm{diag}(C_{ii}), \ 0 < D < 1, \qquad (24)$$

where $C = VV^+$ and $\mathrm{diag}(C_{ii}) = \mathrm{diag}(C_{11}, \ldots, C_{NN})$ is a diagonal matrix. A network designed with this rule is termed a desaturated network. Before presenting the theory behind desaturation, let us demonstrate how it works on a real problem - the recognition of letters. Fifty English and Ukrainian letters were stored in a PINN, a letter represented as a 10x10 binary pattern. Then desaturation (Eq. 24) was applied to the network (desaturating coefficient D varying

from 0 to 1), and the network was used to retrieve the letters when presented with 9% noise.

Figure 3 shows the retrieval of one letter as observed in the experiments. The overall experimental result was that all letters were retrieved successfully within 3-4 iterations by the network with partially reduced self-connections (D=0.15). The performance of the standard network (D=1) and of the network with removed self-connections (D=0) was poor. In both cases a lot of noise remained. Only one iteration was done in the first case and too many iterations and cycles occurred in the second case. More data from this experiment are given in [16], [17].

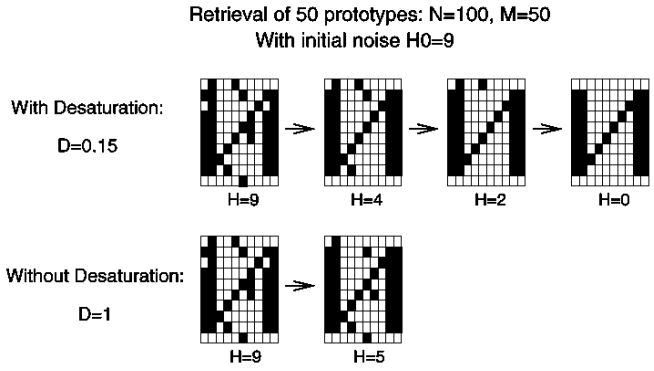

Figure 3 - The retrieval of a letter as observed in the experiments with and without desaturation. Three iterations are executed in the first case, and only one - in the second case

B. Energy considerations

The improvement of the performance can be explained from the "energy" point of view (Section III-C). It can be easily shown using Eq. 14 that changing the diagonal terms does not affect the location of the energy minima in the state space:

$$\begin{aligned} E^D(\vec{Y}) &= -\frac{1}{2}\vec{Y}^T C^D \vec{Y} = \\ &= -\frac{1}{2}\vec{Y}^T C\vec{Y} - \frac{1}{2}\vec{Y}^T(1-D)\mathrm{diag}(C_{ii})\vec{Y} = \\ &= E(\vec{Y}) - \frac{1}{2}(1-D)\sum^{N} C_{ii} = E(\vec{Y}) - \frac{1}{2}(1-D)M, \end{aligned} \tag{25}$$

as $(1-D)M$ is a constant. Yet, changing the weights changes the evolution of the network. In particular, the energy is no longer a monotonically decreasing function of time. This makes the network less prone to getting trapped in local minima and thereby increases the number of iterations. But this may also result in the occurrence of cycles.

Therefore, the changes of the network behaviour incurred by desaturation need to be investigated further.

C. Theory

We want to make sure that the modification does not erase the prototypes from the memory, and does not increase the number of spurious attractors.

Theorem 1: Desaturation preserves prototypes as stable states of the network.

Proof: This follows from Eqs. 24 and 8 and the fact that $D>0$ and $C_{ii}<1$ (Eq. 14):

$$\begin{aligned} \mathrm{sign}[C^D\vec{V}] &= \mathrm{sign}[C\vec{V} - (1-D)\mathrm{diag}(C_{ii})\vec{V}] = \\ &= \mathrm{sign}[\vec{V} - (1-D)\mathrm{diag}(C_{ii})\vec{V}] = \vec{V}. \end{aligned} \tag{26}$$

Theorem 2: Desaturation decreases the number of spurious static attractors.

Proof: We use the idea of Kanter and Sompolmsky [2]. Consider a state $\vec{Y}$, which differs from a prototype $\vec{V}$ in only one neuron i ($y_i = -v_i$). Vector $\vec{Y}$ will also be a static attractor if

$$y_i\left[\sum_{j=1}^{N} C^D_{ji}y_i\right] = y_i\left[\sum_{j=1}^{N} C_{ji}v_i - C_{ii}v_i + C^D_{ii}y_i\right] =$$

$$= v_i[v_i - C_{ii}v_i + DC_{ii}v_i] = v_i[v_i(C_{ii} + DC_{ii} - 1)] > 0, \tag{27}$$

which happens when $C_{ii}(1+D) > 1$.

By decreasing D, we decrease the probability that vector $\vec{Y}$ is a spurious attractor.

These theorems do not yet guarantee that the performance of the network is improved. First we need to know how desaturation affects the DAR of the network (Section III-B), and then we have to find out how many dynamic attractors occur in the desaturated network.

Theorem 3: The average DAR of PINNs increases with the decrease of the desaturating coefficient D as

$$\langle Hattr\rangle = \frac{\frac{1}{2}\left(1-(D+1)\frac{M}{N}\right)}{\sqrt{\frac{M(N-M)}{N^3}}}. \tag{28}$$

Proof: See [37].

The dependency in Eq. 28 is illustrated in Figure 4, which shows the effect of self-connection reduction on the size of the DAR. From the bottom up, the dashed lines correspond to the values D = 1.0, 0.4 and 0.2 in Eq. 28. For comparison, we also plot the DAR given by Eq. 18 by a solid line. As can be seen, decreasing self-connections increases the DAR, making it possible for the network to retrieve prototypes even for M =75 (as $\langle Hattr\rangle > 1$ when D = 0.2).

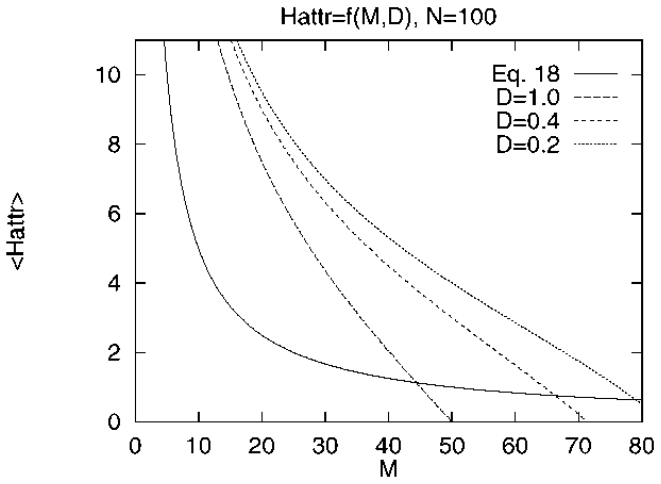

Figure 4 - The direct attraction radius as function of D

This theorem provides the theoretical basis for the observed improvement of the performance of the network caused by the modification, yet it does not explain it completely. This is because the main property of the modification, as mentioned in Section IV-B, is that it allows the network to escape spurious attractors, which increases the indirect attraction radius. However, as theoretical calculation of the indirect attraction radius is known to be hard [38], it can be discovered by simulations only[1]. The result of these simulations are presented in the next section, but first we will resolve the "occurrence of cycles".

D. Dynamic attractors

As has already been mentioned, desaturation results in the occurrence of dynamic attractors. These new spurious attractors degenerate the memory. Therefore, it is important to know when they occur and how to avoid them. This is studied in [8], and below we summarize the main results.

Theorem 4. The number of dynamic attractors in a desaturated PINN increases with the number of prototypes M and with the degree of weight reduction $\alpha = 1 - D$.

Proof: The proof from [8] is reproduced in Appendix.

Whether or not the network will be trapped in a dynamic attractor described by Theorem 4, depends on the number of states the network passes through during its evolution, i.e. the number of iterations. The further the initial state of the network is from its final state, the more iterations it takes to reach this state. This explains why the number of cycle occurrences increases not only with M and α, but also with the value of initial noise.

In order to find the area where cycles do not occur, a set of Monte-Carlo simulations has been carried out (see [8] for more details about the setup of the experiment). In Figure 5 the values of D and $H0$ below the curves are those for which cycles are observed with probability no greater than 1%. The simulations are carried out for a network of size $N = 100$, number of prototypes M =20, 40, 60, and different values of the desaturating coefficient D and initial noise $H0$.

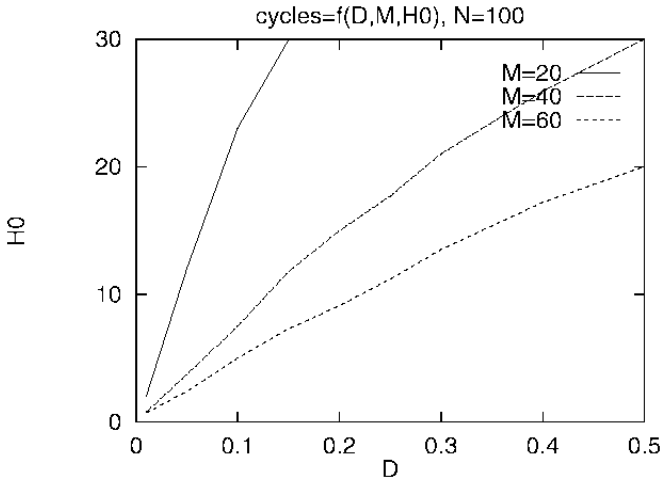

Fig. 5. In the area below the lines the probability of the occurrence of dynamic attractors is not greater than 1%

Simulations also show that when a cycle occurs, there are 2 oscillating neurons in most cases.[2] This allows us to estimate the value of D that causes the occurrence of such cycles (see Appendix):

$$D < \frac{\langle |C_{ij}| \rangle}{\langle C_{ii} \rangle} \approx \sqrt{\frac{N-M}{NM}}. \qquad (29)$$

For example, when N =100 and M =0.5N a cycle will most likely occur when D<0.1, which is in agreement with the experimental observations. However, as mentioned above, the occurrence of cycles also depends upon the initial noise. For example, from Figure 5 it can be seen that for M =0.6N and D =0.1 cycles are rarely observed unless the initial noise $H0$ is greater than 4%.

It should also be mentioned that when cycles occur, the performance of the network can be improved by switching from synchronous dynamics to asynchronous dynamics, as suggested in [2].

V. SIMULATIONS

As mentioned before, the increase of the direct attraction radius with desaturation does not fully describe the improvement of the network performance. Knowing how other network parameters are affected by the desaturation is desirable. These parameters are:

1. the error-correction ratio $H/H0$, which is defined as the ratio between the final noise H, measured after a net-

1.Another approach would be to use an approximate treatment for the description of the PINN dynamics as suggested in [39].

2. To be more exact, the number of oscillating neurons varies from 1 to almost N, but the number 2 is the prevailing number especially when D>0.05.

work has converged, and the initial noise $H0$;

2. the indirect attraction radius (hereafter called simply the attraction radius), which is the largest amount of noise that can be completely eliminated by network iterations;

3. the associative capacity, which is the greatest number of prototypes that can be stored by the network with the attraction radius being greater than zero; and

4. the number of iterations that the network executes before reaching an attractor.

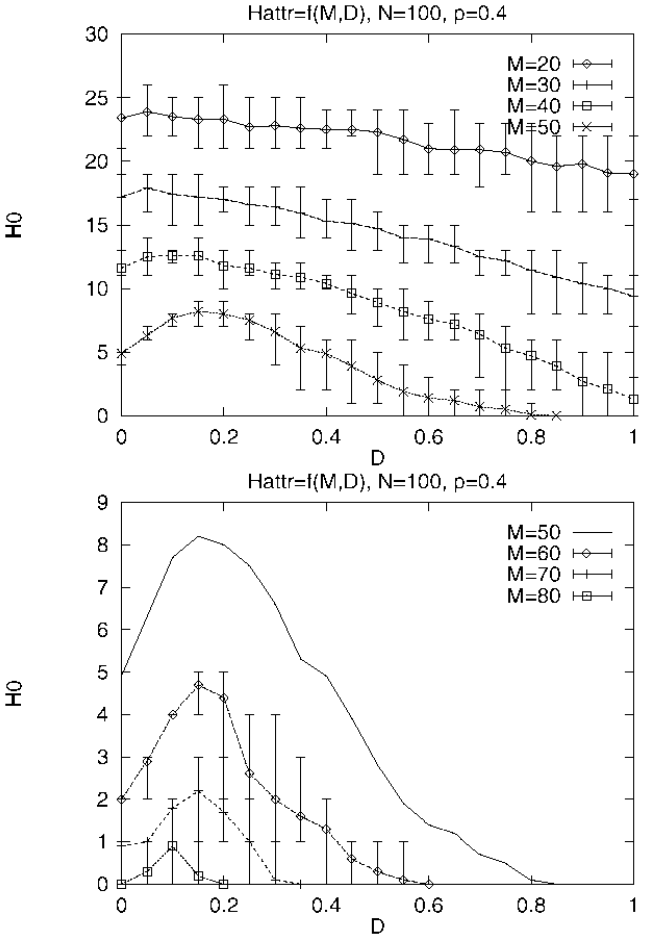

Figure 6 - The increase of the attraction radius as a result of the reduction of self-connection: $C_{ii} = D \cdot C_{ii}$, $p = 0.4$

Whereas the first three parameters describe the improvement of retrieval with desaturation, the last one indicates how much desaturation slows down retrieval. The results concerning the improvement of the error-correction ratio are given in [40], and below we present the results concerning the other parameters.

Extensive Monte-Carlo simulations have been carried out. Prototypes are random vectors, where the probability p of a neuron being in state +1 is the same for all neurons in all prototypes. The update flow technique described in Section II-A is used[1].

The measurements of the attraction radius are carried out as follows. A network of size N=100 is used[2]. We start with M=20 prototypes. The coefficient D in Eq. 23 is decreased from 1.0 to 0.0 with an increment 0.05. For each D, the initial noise $H0$ is increased until the final noise H is greater than 0.1, as averaged over 10 sets of prototypes and 10 implementations of the initial noise. The initial noise is created by randomly inverting $H0$ neurons of a prototype. Then, using Eq. 12, we add 10 new prototypes and repeat the procedure. We record the average value of noise $H0$ which can be completely eliminated, its minimal and maximal values, and also the number of occurred iterations.

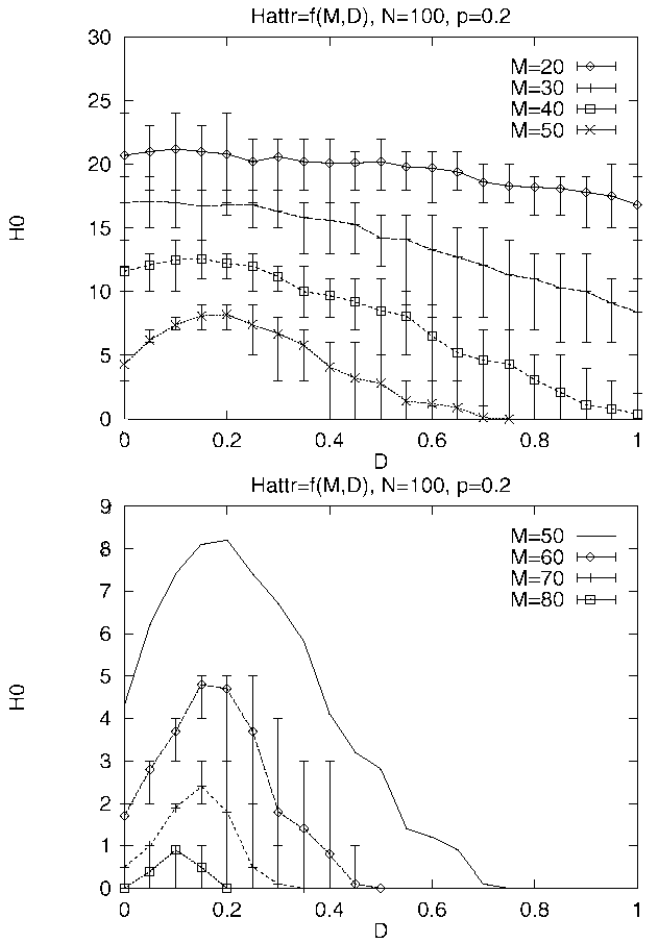

Figure 7 - The increase of the attraction radius as a result of the reduction of self-connection: $C_{ii} = D \cdot C_{ii}$, $p = 0.2$

Figures 6 and 7 plot the results for probabilities p=0.4 and p=0.2 respectively, where the average values of the attraction radius are drawn as curves and the minimal and

1. The source of the program used in the simulations is available at ftp://ftp.cs.ualberta.ca/pub/dmitri/PINN .

2. Data obtained for different N are given in the next subsection.

maximal values of the attraction radius are drawn as vertical bars.

The main results from the simulations in [40] and from the ones described above are as follows.

- While $D \geq 0.15$, a decrease of self-connections always results in a decrease of the error-correction ratio $H/H0$ and an increase of the attraction radius of PINNs.

- When the number of stored prototypes is small ($M \leq 0.2N$), the improvement is insignificant. Yet cycles may occur. Therefore, it is not worth using desaturation for small learning ratios M/N.

- The improvement is especially noticeable when $M \geq 0.4N$. More specifically, the desaturation increases the attraction radius of the network from zero (in the case of standard PINNs), to:

12.6 (on average) and to 12 (at worst) for $M = 0.4N$,

8.2 (on average) and to 7 (at worst) for $M = 0.5N$,

4.7 (on average) and to 4 (at worst) for $M = 0.6N$,

2.2 (on average) and to 1 (at worst) for $M = 0.7N$,

0.9 (on average) for $M = 0.8N$.

- The optimal value for the coefficient of self-connection reduction lies in the interval $0.1 \leq D \leq 0.2$.

- The number of iterations increases insignificantly: from 1 to between 2 and 4.

- Decreasing D below 0.1 is undesirable, as this results in the occurrence of numerous dynamic attractors, thus decreasing the attraction radius. For D=0 the performance, although not optimal, is still better than that of the standard model (D =1) confirming the result of [2].

- The associative capacity of PINNs is greater than $0.7N$ and is $0.8\ N$ on average.

A. Scaling a network

In order to use the data presented in the previous subsections, one needs to know how the parameters of the network change with the size of a network, and how they change in the case of non-random prototypes. Here the approach of Kanter and Sompolinsky [2] can be used. In order to compensate for the effect of prototype overlapping, they use another definition for the attraction radius of a network, which can be written for small values of the initial noise as:

$$R = \frac{R_N}{\langle R_V \rangle}, \qquad (30)$$

where R_N is the average attraction radius of a network of size N measured by experiments and $\langle R_V \rangle$ is the average distance between the prototypes stored in the network[1].

From this definition it follows that the measured attraction radius R_N increases with the size of the network and decreases when correlated prototypes are stored. This is exactly what is observed by experiment. Figure 8 shows the observed attraction radius $R_N = Hattr/N$ as a function of the self-connection reduction for different sizes of the network.

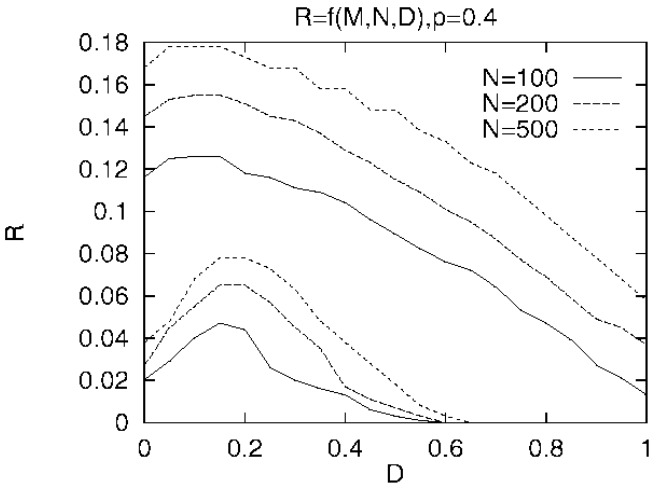

Figure 8 - The increase of the attraction radius $R_N = Hattr/N$ with desaturation for different network sizes N, $M = 0.4N$ (top three curves) and $M = 0.6N$ (bottom three curves)

VI. CONCLUSIONS

We described how reduction of self-connection affects the performance of pseudo-inverse neural networks. We summarized the results obtained on the phenomenon and showed that the best performance of the network is achieved when self-connection is reduced approximately 0.15 times, i.e. when

$$C_{ii} = D \cdot C_{ii}, \text{ where } D = 0.15.$$

In order to efficiently detect cycles, which may occur when self-connection is reduced, the update flow neuroprocessing technique described in the paper should be used.

Another contribution of the paper is the desaturation method which, when applied to a pseudo-inverse neural network, restores the balance of network weights, allowing thereby the network to store more prototypes. This method gives rise to developing non-iterative learning techniques for storing continuous information in real-time, which is described in [41].

ACKNOWLEDGEMENTS

The results presented in this paper originally appeared in the author's Ph.D. dissertation [42]. Some of them also appeared in [43].

1.In this section the distance and the attraction radius are scaled according to the size of the network. In particular, $R_N = Hatr/N$.

ADDENDUM

Since this work has been done, other worth mentioning results on pseudo-inverse networks and the influence of self-connection appeared.

Satoshi Matsuda presented results [44] on finding the optimal value of self-connection for Hopfield-like networks, where optimality is considered in terms of finding an optimal solution for an optimization problem the network is applied too.

Giuseppe Grassi et al implemented inverse Greville theorem [45], and also extended the application of pseudoinverse learning rule to heteroassociative and second-order neural networks [46].

Finally, a bulk amount of research on the influence of selfconnection on chaotic neural networks, which is another type of attractor-based networks, and its application to the combinatorial problems has been conducted by Masaya Ohta [47].

APPENDIX

From Proposition 1 it follows that if there exists a dynamic attractor in the network, it consists of exactly two states. Let us calculate the probability that two states of the desaturated network $\vec{Y}^{d1}$ and $\vec{Y}^{d2}$ are a dynamic attractor, i.e. that

$$\begin{cases} \vec{Y}^{d2} = \text{sign}[C^D \vec{Y}^{d1}], \\ \vec{Y}^{d1} = \text{sign}[C^D \vec{Y}^{d2}]. \end{cases} \tag{31}$$

Designating $\Omega = \{i : y_i^{d1} = -y_i^{d2}\}$ as the set of the indices of oscillating neurons in these states, we have

$$\begin{cases} y_i^{d1} \sum_{j=1}^{N} C_{ij}^D y_j^{d1} < 0, \\ y_i^{d2} \sum_{j=1}^{N} C_{ij}^D y_j^{d2} < 0, \end{cases} \qquad \forall i \in \Omega . \tag{32}$$

Adding one equation to another yields

$$y_i^{d1} \left(\sum_{j=1}^{N} C_{ij}^D y_j^{d1} - \sum_{j=1}^{N} C_{ij}^D y_j^{d2} \right) = 2 y_i^{d1} \sum_{j \in \{\Omega\}} C_{ij}^D y_j^{d1} < 0 . \tag{33}$$

Summing this equation over all oscillating neurons, we have

$$2 \sum_{i \in \Omega} y_i^{d1} \sum_{j \in \Omega} C_{ij}^D y_j^{d1} = 2 \sum_{i,j \in \Omega} C_{ij}^D y_j^{d1} y_i^{d1} < 0 \tag{34}$$

and using Eq. 23, we obtain

$$\sum_{i,j \in \Omega} C_{ij}^D y_j^{d1} y_i^{d1} < (1 - D) \sum_{i \in \Omega} C_{ii} . \tag{35}$$

Since for the pseudo-inverse learning rule $C_{ii} \approx \frac{M}{N}$ (Eq. 14), we obtain that the probability that $\{\vec{Y}^{d1}, \vec{Y}^{d2}\}$ is a dynamic attractor is proportional to M and to $\alpha = 1 - D$, which is the statement of Theorem 4.

Eq. 35 allows us to estimate the value of D, which produces the most number of dynamical attractors. If $\Omega = \{i, j\}$ consists of only two indices, then from Eq. 35 we have

$$2 C_{ij} y_i y_j + C_{ii} + C_{jj} < (1 - D)(C_{ii} + C_{jj}) . \tag{36}$$

From this equation, using the estimates of C_{ij} and C_{ii} (Eqs. 14-17) we obtain

$$D < \frac{\langle |C_{ij}| \rangle}{\langle C_{ii} \rangle} \approx \sqrt{\frac{N - M}{NM}} . \tag{37}$$

REFERENCES

1. L. Personnaz, 1. Gliyon and G. Dreyfus. Collective computational properties of neural networks: New learning mechanisms, Phys. Rev. A vol 34, pp. 4217-4228, 1986
2. I. Kanter and H. Sampolinsky. Associative recall of memory with out errors) Phys. Rev. A vol 35, pp. 380-392) 1987.
3. S. Amari. Learning patterns and pattern sequences by self organizing nets of threshold elements, IEEE Transactions on Computers' vol C-219, No.II, pp. 1197-1206, 1971.
4. W.A. Little. The existence of the persistent states in the brain, Mathematical Biosciences, Vol. 19, pp. 101-120, 1974.
5. S. Amari. Neural theory of association and concept formation, Biological Cybernetics, vol 26, pp. 175-185, 1977.
6. T. Kohonen. An adaptive associative memory principle, IEEE Transactions on Computers, C-23, 444-445, 1974.
7. T. Kohonen. Associative memory: A System-Theoretical Ap proach, Berlin:Springer, 1978.
8. D.O. Gorodnichy and A.M. Reznik. Static and Dynamic Attrac tors of Auto-associative Neural Networks, Lecture Notes in Com puter Science, Vol 1311 (Proc. of 9th Intern. Conf. on Image Analysis and Processing, Florence, Italy, Vol. II), pp. 238-245, Springer, Sept. 1997.
9. A.M. Reznik. Iterative projection learning algorithm for neural networks (in Russian), Kibernetika i Sistemny! Analiz, N 6, pp. 131-141,1993.
10. J.J. Hopfield. Neural networks and physical systems with emer gent collective computational abilities, Proc. Nat. Acad. Sci. USA.vol 79, pp. 2554-2558, 1982.
11. K. Cheung, L. Atlas, R. Marks. Synchronous vs asynchronous behavior ofHopfield's CAM neural net, Applied Optics, Vol. 26, No 22, pp. 4808-4813, 1987.
12. P. Peretto. Collective Properties of Neural Networks: A statisti cal Physics Approach, Biological Cybernetics, 50, 51, 51-62, 1984.
13. A. Frurnkin, E. Moses. Physicality of the Little model, Phys. Rev. A vol 34, No I, pp. 714-716, July 1986.
14. D.O. Gorodnichy, A.M. Reznik. NEUTRAM - A Transputer Based Neural Network Simulator, Proc. of 2nd Intern. Conf. on Software for Multiprocessors and Supercomputers Theory, Prac tice, Experience (SMS TPE'94), pp. 136-142, Moscow, Russia, 1994.
15. A. Albert. Regression and Moore-Penrose pseudoinverse. Aca demic New-York, 1972.
16. D.O. Gorodnichy. A way to improve error correction capability of Hopfield associative memory in the case of saturation, HELNET 94-95 International Workshop on Neural Networks Proceedings, vol. 1/11, pp. 198-212, VU University Press, Amsterdam, 1996.
17. D.O. Gorodnichy. Desaturating Coefficient for Projection Learn ing Rule, Lecture Notes in Computer Science, Vol. 1112 (Intern. Conf. on Artificial Neural Networks Proceedings), pp. 469-476, Bochum, Germany, Springer-Verlag, 1996.
18. A.M. Odiyzko. On subspaces spanned by random selection of

± l vectors, J. Combinatorial Theory A, vol. 47, pp. 124-33, 1988.
19. Y. Crama, P.Hansen and B. Jaumard. Detection of Spurious states of Neural networks, IEEE Transactions on NN, Vol. 2, No I, pp. 165-169,1991.
20. J.-D. Gascuel, B.Moobed, M.Weinfeld. An Internal Mechanism for Detecting Parasite Attractors in a Hopfield Network. Neural Computation, 6,5, pp.902-915, 1994.
21. M. Weinfield. A fully digital integrated CMOS Hopfield network including the learning algorithm. Proc. of Intern. Workshop on VLSI For Artificial Intelligence , E1-E10, Univ.of Oxford, 1988.
22. V.S. Dotsenko, N.D. Yarunin and E.A. Dorotheyev. Statistical mechanics of Hopfield-like neural networks with modified interactions, J. Phys. A: Math. Gen 24. pp. 2419-2429, 1991.
23. A.Engel and M. Weigt. Storage capacity of the truncated projection rule, J. Phys. A: Math. Gen 24, pp. 3707-3709. 1991.
24. A. Schultz. Five variations of Hopfield Associative memory networks, Journal of Artificial Neural Networks vol. 2, no 3, pp. 285-294, 1995.
25. B.Telfer and D.Casasent. Updating optical pseudoinverse associative memories. Applied Optics, vol. 28, no. 13. pp. 2518-2528, 1989.
26. S. Diederich. M. Opper. Learning of correlated patterns in spin-glass networks by local learning rules, Phys. Rev. Lett.-58. N9, 949-952, 1987.
27. E. Gardner. The space of interactions in neural network models, Phys. Rev. A vol 21, pp. 257-270. 1988.
28. J. Kratschmar and G. Kohringi Retrieval of neural networks constructed from local and nonlocal learning rules, Journal de Physique. 2, pp. 223-229, Feb. 1990.
29. A. Johannet, L. Personnaz, G. Dreyfus, J.-D. Gascuel, and M. Weinfeld Specification and Implementation of a Digital Hopfield-Type Associative Memory with On-Chip Training, IEEE Transactions on Neural Networks Vol. 3, N. 3, p. 529, July 1992.
30. M.H. Hassoun and A.M. Youssef. Autoassociative neural memory capacity and Dynamics. International Joint Conf. on NN (IJCNN'90) Proceedings vol 1, pp. 763-769. San Diego, USA, 1990.
31. A.N. Michel, J.Si and G. Yen. Analysis and Synthesis of a class of discrete-time neural Networks, IEEE Transactions on NN, Vol. 2, No 1, pp. 29-39, 1991.
32. S. Coombes and J. Taylor, Using Features for the Storage of Patterns in a Fully Connected Net. Neural Networks 9, pp. 837-844, 1995.
33. S. Coombes and J. Taylor, The Storage and Stabilisation of Patterns in a Hopfield Net, Neural Network World, 5. (1995) 133-150.
34. M.A.G. Abyshagur and A.M.Helaly. Neural network training using the bimodal optical computer, Proceedings of SP!E - The International Society for Optical Engineering, vol 1294, pp. 77-83, 1990.
35. G.R. Gindi, A.F. Gmitro and K. Parthasarathy. Hopfield model associative memory with non-zero diagonal terms in memory matrix. Applied optics 27, pp. 129-134, 1988.
36. H. Yanai and Sawada Y. Associative memory network composed of neurons with Hysteretic Property, Neural Networks Vol.3, N2, pp. 223-228, 1990.
37. D.O. Gorodnichy and A.M. Reznik. Increasing Attraction of Pseudo-Inverse Autoassociative Networks, Neural Processing Letters, volume 5, issue 2, pp. 123-127, 1997.
38. P. Floreen and P. Orponen. Attraction radii in binary Hopfield nets are hard to compute, Neural Computation 5. pp. 812-821, 1993.
39. R.D. Henkel and M. Opper. Parallel dynamics of the neural network with the pseudoinverse coupling matrix. J, Phys. A: Math, Gen 24, pp. 2201-2218, 1991.
40. A.M. Reznik, D.O. Gorodnichy. A.S. Sychov. Controling local self-connection in neural networks designed with the projection learning rule (in Russian). Kibemetika i Sistemnyi Snaliz, N 6, pp 153 - 162. 1996.
41. A.M. Reznik. Non-Iterative Learning for Neural Networks, CD-ROM Proceedings o IJCNN>99, Washington, July 12-17, 1999.
42. D.O. Gorodnichy. Investigation and design of high performance fully connected neural networks. PhD dissertation, National Academy of Sciences of Ukraine, Kiev, Ukraine (written in Russian). Available online at ftp://ftp.cs.ualberta.ca/pub/dmitri/PINN 1997.
43. D.O. Gorodnichy. The Optimal Value of Self-connection ("Best Presentation" award paper). CD-ROM Proceedings of IJCNN'99, Washington, July 12-17, 1999).
44. S. Matsuda. "Optimal" Hopfield network for Combinatorial Optimization with Linear Cost Function. IEEE Transactions on Neural Network., Vol. 9, No. 6, pp. 1319-1330, 1998.
45. M. Brucoli, L. Carnimeo, and G. Grassi. Hetereassociative memories via celluar neural networks, Int. Journal of Circuit Theory and Applications, 26, 231-241, 1998.
46. Ciuseppe Crassi and Giuseppe Acciani Cellular Neural Networks for Information Storage and Retrieval: A New Design Method CD-ROM Proceedings of IJCNN'99, Washington, July 12-17, 1999).
47. M. Ohta. On the self-feedback controlled chaotic neural network and its application to the N-Queen problem, CD-ROM Proceedings of IJCNN'99, Washington, July 12-17, 1999.

УДК 681.32:007.52

PARALLEL IMPLEMENTATION OF NEURAL NETWORK ALGORITHM USING PARALLEL VIRTUAL MACHINE (PVM) SOFTWARE

Jamil Ahmad

Для повышения скорости работы нейронных сетей предлагается использовать технологию параллельной реализации нейронных сетей. Рассмотрен пример решения задачи распознавания символов нейронной сетью, моделируемой с помощью программного комплекса параллельной виртуальной машины.

Although there are many implementations of artificial neural networks (ANN) are available on sequential machines, but most of these implementations require an immoderate amount of time to train or run ANNs, especially when the ANN models are large. It can be observed that the problem is related to computational power of computer machines. One possible approach for solving this problem could be a parallel implementation of ANNs. Hence, researchers have adopted a number of strategies since the rebirth of ANN in 1986 to implement ANN model in parallel environment. Very few of these strategies are using software platform for such implementations. Therefore, this paper presents a novel technique of implementing ANN for the recognition of characters using Parallel Virtual Machine (PVM) software package. PVM permits a heterogeneous collection of computers hooked together by a network to be used as a single large parallel computer. Thus large computational problems can be solved more cost effectively by using the aggregate power and memory of many computers. The nodes (neurons) of ANN model are distributed over on the participating computers in the parallel environment to do the needful calculation in parallel. Weights are also adjusted in the same way, if there is any discrepancies between computed and target outputs. Simulation of the study shows that parallel implementation of the ANN produces better results than sequential implementation.

1. INTRODUCTION

The growing importance of artificial neural networks is now widely recognized, it is critical that these models run fast and generate results in real time. As discussed above