Adaptive genetic algorithm for the binary perceptron problem

1990 J. Phys. A: Math. Gen. 23 L1265

(http://iopscience.iop.org/0305-4470/23/23/014)

J. Phys. A: Math. Gen. 23 (1990) L1265–L1271. Printed in the UK

LETTER TO THE EDITOR

Adaptive genetic algorithm for the binary perceptron problem

H M Köhler

Institut für Theoretische Physik der Georg-August-Universität, D-3400 Göttingen, Federal Republic of Germany

Received 18 July 1990, in final form 28 August 1990

Abstract. For neural networks with $\pm J$ couplings the perceptron problem for random unbiased patterns is considered. An algorithm that uses concepts of the continuous perceptron problem as well as ideas of biological optimization is proposed and investigated. The distribution of local stabilities and the critical storage capacity α_c are determined. While for N less than ≈ 50 the value of α_c is approximately 0.83, the storage capacity goes down to $\alpha_c \approx 0.7$ for $N = 255$.

Neural networks with two-state neurons $S_i = \pm 1$ have been analysed with the methods of statistical mechanics [1]. They can be used as distributed memories with associative abilities. Retrieval of information is a dynamical process, where in the simplest case this has deterministic dynamics defined by a sign transfer function:

$$S_i^{(t+1)} = \operatorname{sign} \sum_{j\,(\neq i)} J_{ij} S_j^{(t)} . \tag{1}$$

Information is stored during learning in the synaptic couplings J_{ij}. In attractor neural networks, the task is to find synaptic couplings J_{ij} such that a given set of configurations or patterns $\xi_i^\mu = \pm 1$, $i = 1, \ldots, N$ and $\mu = 1, \ldots, p$ are attractors of the network. Given the above dynamics the patterns are stationary states if and only if

$$E_i^\mu = N^{-1/2} \xi_i^\mu \sum_{j\,(\neq i)} J_{ij} \xi_j^\mu \geq \kappa > 0 \tag{2}$$

for every $i = 1, \ldots, N$ and $\mu = 1, \ldots, p$. In feed-forward networks the task is to find the right output configuration to a given input. In this case ξ_i^ν denotes the state of one output neuron and $\{\xi_j^\mu\}$ $(i \neq j)$ its input neuron states. A set of couplings $\{J_{ij}\}$, which satisfies (2) for every i and μ is called a perceptron solution. If the neurons S_i take the values of one of the patterns, the system is stable under zero temperature dynamics. So the given patterns are attractors of the system.

For simplicity only unbiased and randomly chosen patterns will be considered. For the case of arbitrary valued couplings J with spherical constraint $\sum_j J_{ij}^2 = N$, Gardner [2] calculated the greatest possible stability $\min_{i,\mu} E_i^\mu$ for given $\alpha = p/N$ in the limit of large N; the maximum storage capacity $\lim_{N\to\infty} p_{\max}/N$ is determined

0305-4470/90/231265+07$03.50

by $\alpha_c = 2$ [3]. Algorithms capable of producing optimal stability and of reaching the maximum possible storage capacity are known: the MinOver algorithm [4] is the most well known, while the AdaTron [5] is the fastest algorithm so far developed.

The case of binary couplings, taking only the values J or $-J$, is less well understood. The calculation of the maximum storage capacity in the thermodynamic limit $N \to \infty$ requires a solution with replica symmetry breaking (RSB) [6]. Krauth and Mézard [6] find a phase transition in one step RSB at $\alpha = 0.83$. At The same value of α the entropy of the replica symmetric solution turns negative. With a complete enumeration of all possible configurations for small systems ($N \leq 25$) a critical storage capacity of $\alpha_c = 0.82$ is found [7].

For binary couplings all learning rules that clip arbitrary valued coupling matrices to their sign, like the clipped Hopfield matrix

$$J_{ij} = \text{sign} \sum_{\mu} \xi_i^{\mu} \xi_j^{\mu} \tag{3}$$

and others like a clipped pseudo-inverse or a clipped optimal perceptron matrix do not stabilize the patterns exactly. One finds attractors that are strongly correlated with the patterns, but their overlap with those patterns is less than 1 when α is not zero. The same problem occurs with an algorithm that finds a coupling matrix by minimization of a quadratic form [8], although a higher storage capacity is possible ($\tilde{\alpha_c} \approx 0.4$ compared to 0.1 for clipped Hopfield [9] and 0.3 for clipped pseudo-inverse and clipped optimal perceptron [8]; $\tilde{\alpha_c}$ is defined as the greatest possible number of attractors with a macroscopic overlap with the patterns divided by N). Amaldi and Nicolis [10] propose tho algorithms for the problem; one uses simulated annealing and the other so-called 'tabu' search. They apply their algorithms to systems with $N \leq 81$ with the result $0.55 < \alpha_c(N = 81) < 0.66$.

The aim of this letter is to present an algorithm that produces ± 1 coupling matrices that stabilize patterns for much larger systems with higher storage capacities. Since the problem can be interpreted as a special integer optimization task, it is believed to be NP-complete. The optimal configuration can not be found in polynominal time, so approximate techniques are needed. For the purposes of learning it is sufficient to regard only one output neuron, so that the coupling matrix can be reduced to a coupling vector $J_{ij} \to J_j$. In attractor nets the patterns need to be rescaled for neuron i as $\zeta_j^{\mu} = \xi_i^{\mu} \xi_j^{\mu}$, while for feed-forward nets the patterns can be mapped on any $\eta^{\mu} = \pm 1$ with $\zeta_j^{\mu} = \eta^{\mu} \xi_j^{\mu}$.

The simplest possible way of constructing a coupling matrix is by optimization with a cost function, for example

$$\mathcal{H} = N^{-1} \sum_{\nu} (\lambda - E^{\nu})^2 \Theta(\lambda - E^{\nu}) . \tag{4}$$

A local minimum is found with methods of steepest descent or simulated annealing. In (4) Θ denotes the Heaviside function. Preliminary simulations show that the square in (4) should not be left out and that much higher exponents or exponential cost functions, which take $(\lambda - E^{\nu})$ in the exponent, are not appropriate. A cost function with exponent 1 or 0 does not take diffusion effects into account. For $E < \lambda - 2/\sqrt{N}$ one step to smaller field values compensates for one to greater field values. At the end the system is caught in a local minimum of its energy landscape with many of the E_i^{ν} having values far below λ. On the other hand diffusion should not be penalized too

much, because the system would be trapped too early (after one Monte Carlo step in zero-temperature optimization with an exponential cost function). During simulations with (4) and simulated annealing for N up to 99 the storage capacity could not be raised systematically above 0.6, although higher values for the capacity were observed for smaller N.

Much better results could be achieved with an algorithm that is split in two parts: an adaptive part, which is embedded in a framework of a genetic optimization program that searches a wider part of the phase space. In the first part patterns are adaptively stabilized towards stability λ, but only if the pattens are not yet well learned. As a measure of how much (more) pattern μ needs to be embedded, I calculate $\delta x^\mu = (\lambda - E^\mu)\Theta(\lambda - E^\mu)$. For continuous couplings J^c this would require the modification $J_j^c(t+1) = J_j^c(t) + \zeta_j^\mu \delta x^\mu$. Since the J-vector can only be changed in steps, by altering the sign of one or more of its entries, the influence of all patterns must be taken into account simultaneously. In order to have an absolute measure δI_j, how much every synapse would need to be changed, and which is independent of the individual signs, the sum over the necessary modifications is multiplied by $-J_j$:

$$\delta I_j = -J_j \sum_\mu \zeta_j^\mu \delta x^\mu. \tag{5}$$

Then all those J_j that are worst in the sense that δI_j have the greatest positive value, shall be flipped (couplings J_j with negative δI_j are not flipped). Usually there is more than just one flip necessary, because the E^μ and the δI_j are quantized in steps of $2/\sqrt{N}$. To be more precise, the set of J_i that will be flipped in one time step is given by

$$\mathcal{F} = \{ J_i | (i | \delta I_i = \max_j \delta I_j \wedge \delta I_i > 0)\} \qquad \forall i, j = 1, \ldots, N. \tag{6}$$

Starting from some initial configuration, $\mathcal{F}$ is repeatedly calculated and the corresponding spins are flipped, until a solution is found, or a maximum number of iterations is exceeded, or the system is caught in a cycle. This algorithm cannot be described by a cost function, which is minimized. The system is not a Hamiltonian system and consequently cycles appear. The resulting configuration of J can not be expected to be a linear combination of the patterns as in adaptive learning rules for continuous couplings, where $J_j^c = N^{-1} \sum_\mu x^\mu \zeta_j^\mu$. Furthermore, there is no guarantee that one of the configurations of the J-vector within one cycle is better than the previous one. So in the computer realization all configurations were stored, while it was checked first to see if the new configuration had occured before; this was done with the help of hash-coding techniques. In that case the algorithm was stopped and the best configuration found so far was kept. The result of this process depends of course on the start configuration used. For that reason simulations were done for a set of M different start vectors (for simplicity $M = N$). One of them was a clipped Hopfield vector, while the others were chosen at random with equal propability for $+1$ and -1. There was no evidence that the clipped Hopfield vector behaved any better than the other start configurations.

Now I describe the genetic part of the algorithm. After all M vectors had passed through the first part of the algorithm, they were sorted according to their stability $\kappa(k) = \min_\mu E^\mu(k)$, $k = 1, \ldots, M$. In the next step the first (the best) vector and the second were crossed as if the J-vectors would represent a genetic code (see figure 1) to

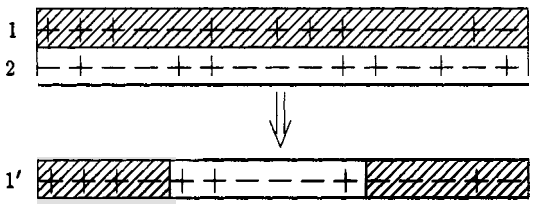

Figure 1. Creation of a new J-vector for optimization in the adaptive part of the algorithm from two vectors of the last generation.

produce a new vector, which brings the system out of the cycles that it was in before. Vector number one and three form the next new vector, then one and four, two and three and so on, so that the sum of the numbers is monotonously increasing. The vectors were sectioned at random cutting points, so that the average length l of the pieces was $l \propto N$. In fact, the scaling factor does not have a significant meaning in this model, as could also be seen in the simulations, because there is no relation of spatial order in the system; the output neuron is coupled with all the other neurons. The crossing over, or the mixing of the vectors, could have been done just as well differently. When M new vectors were created, they were used as start configurations for the first (adaptive) part of the algorithm. This method of pushing the system out of cycles or metastable states is clearly better, than a random tip as used in optimization with simulated annealing; the phase space is not smooth as a consequence of RSB and the fact, that the mean overlap of two solution vectors at α_c is $q \approx 0.5$ [6]. Instead of some small steps of hill climbing in random directions, bigger steps are taken in directions given by some evolutionary pressure, caused by the selection of only the best J-vectors.

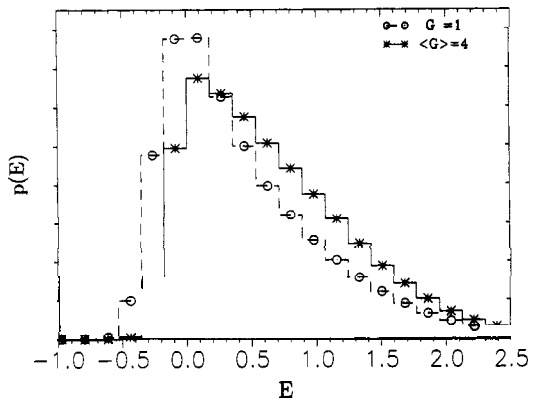

Figure 2. Field distributions of all members of a generation, after one generation and after the algorithm has stopped. The distributions are averaged over 400 independent runs of the algorithm with independent patterns and start configurations for $N = 127$ and $\alpha = 0.6$.

The adaptive and the genetic part of the algorithm must be iterated several times until the minimal field value of one of the configurations $\kappa(k) = \lambda$, or most of the vectors (90%) have the same $\kappa(k) < \lambda$, or a maximum number of iterations is exceeded; here 50 (or 60 for $N = 255$) were enough for at least 70% of all trials to be within that

limit. Let us say that one 'generation' G is completed whenever the adaptive part has finished.

All simulations were done for desired stability $\lambda = 0$, odd N and $M = N$ (an odd N permits no $E^{\mu}(k) = 0$). In figure 2 I have plotted the field distribution of $E^{\mu}(k)$ for all $k = 1, \ldots, N$ and $\mu = 1, \ldots, p$ after one generation G and after the algorithm has found one configuration with $\kappa > 0$. The distribution was averaged over 400 runs of the program with one set of patterns and one set of start configurations in each run. On the average it took 4.0 generations to find a solution vector. System size N was 127 and $\alpha = 0.6$. Due to the genetic optimization the distribution shifted to the right and the skewness increased. Much simulation work was done for determining the critical storage capacity α_c. In figure 3(a) the maximum κ of one generation averaged over many runs for six values of N from 49 up to 255 and $\alpha \in [0.65, 0.9]$. For $N = 49$ averaging was possible over 1000 samples, while for $N = 255$ not more than 60 systems could be simulated. The error for all data points is of the size of the symbols or less, except for $N = 255$, where an error bar is shown. The data of a given N are connected with spline interpolations to determine $\alpha_c(N)$ as the intersection with $\kappa = 0$.

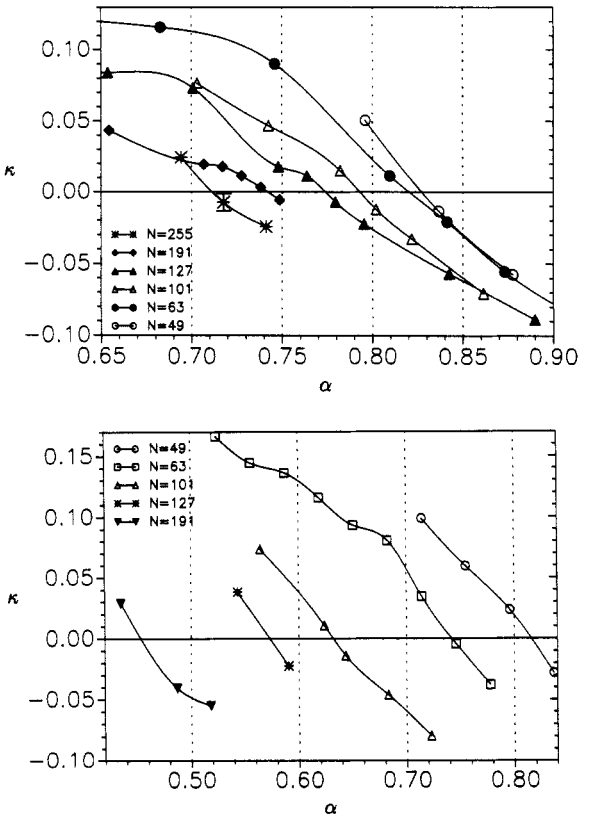

Figure 3. (a) Best κ of a generation at the end of the optimization program, averaged for $N = 49, 63, 101, 127, 191, 225$ over 1000, 600, 500, 300, 100, 60 independent runs. The intersection points of the lines with $\kappa = 0$ determine $\alpha_c(N)$. (b) Best κ of a generation after one pass through the adaptive optimization, averaged for $N = 49, 63, 101, 127, 191$ over 500, 300, 200, 200, 50 independent runs.

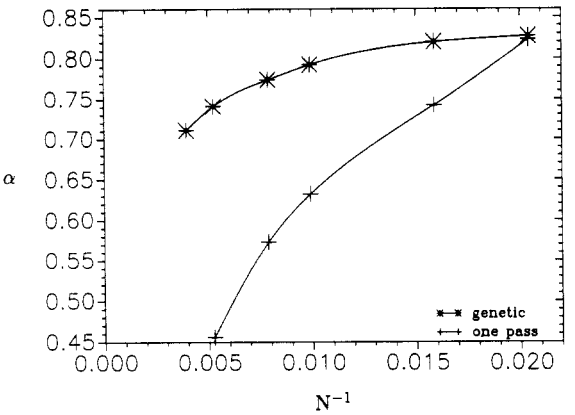

Figure 4. The critical storage capacity $\alpha_c(N)$ from figures 3(*a*) (asterisks) and 3(*b*) (crosses) plotted against N^{-1}. A normal finite-size scaling is not possible.

In order to see the effect of the genetic optimization more quantitatively, the simulations were repeated without the second part of the algorithm; that means the number of generations G was restricted to 1. At the same time the number of starting vectors was increased to get comparable results; M was chosen $M = NG_c^0$, where G_c^0 is the number of generations in the unresricted run at α_c. In figure 3(*b*) the corresponding stabilities are plotted against α. Since the first pass through the adaptive algorithm takes more time than the later passes, simulations could not be carried out for N=255. Averaging for the remaining N could be done over 500 ($N = 49$) down to 200 ($N = 191$) samples. Again the maximum storage capacity is taken as the intersection of a spline interpolation of the data and the line $\kappa = 0$. Now the α values of the intersection points of both simulations are plotted in figure 4 against N^{-1}. In the case of genetic optimization or unrestricted G (asterisks), the N-dependence is much weaker than in the case of no genetic optimization, or just one pass through the adaptive part (crosses). Instead of a dramatically decreasing storage capacity with growing N, α_c decreases more slowly with genetic optimization and stays above 0.7 for $N \leq 255$. A finite-size scaling, i.e. a linear approximation of α against $N^{-\gamma}$ is not possible for positive γ. That means that no prediction for α_c in the limit of infinite N can be given. For small N on the other hand, it seems that $\alpha_c \approx 0.83$.

I have also investigated the field distribution for those J-vectors that had $\kappa > 0$. Figure 5 shows those field distributions for $N = 63, 127, 191$ and α as close to $\alpha_c(N)$ as possible. No significant N-dependence can be determined. A Gaussian fit curve, which is cut off at negative values, gives a good approximation for the type of the distribution.

With the proposed algorithm it is possible to calculate binary perceptron coupling vectors or matrices for finite systems. The upper bound of the storage capacity $\alpha_c = 0.83$ as calculated in [6] for $N \to \infty$, which also seems to hold for small N as shown in [7], can not approximately be reached for N exceeding ≈ 50.

Genetic optimization has been suggested previously for all kinds of problems, see for example [11]. The kind of genetic optimization is different from the Darwinian program, because many good realizations of J-vectors are kept in a population and not only the fittest survive. In the model described there is a feed back from the phenotype space, which is associated with the space of local stabilities E^ν, to the genotype

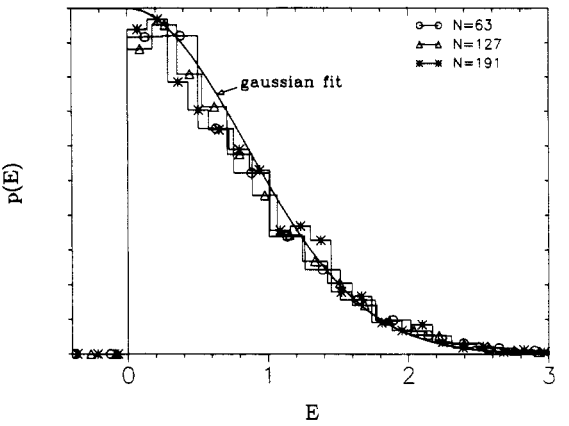

Figure 5. Field distribution for configurations with $\kappa > 0$ and for $N = 63, 127, 191$ averaged over 600, 300, 35 samples for $\alpha \approx \alpha_c(N)$. A first approximation for the curve type is a centrally clipped Gaussian fit.

space or the J-vector space. The same feedback mechamism has been proposed by Eigen [12] in his model of Hypercycles for molecular evolution. Here the J-vectors are changed in the adaptive part of the algorithm, so a biological interpretation of this part can not be phenogenesis. It should rather be interpreted as the search of a whole quasispecies, represented by one member of the population, for a good solution in some local minimum of an energy-like landscape.

The proposed algorithm is far from being optimal and there is lots of room for changes and improvement. For example the number of times a single vector participates in the creation of a new vector could be limited to a finite value, so that the variety of genetic information does not get lost too easily when N grows. The part of the phase space that can be searched should have a size that scales linear in N.

I would like to thank A Zippelius, W Kinzel and M Opper for inspiring discussions. The computer time for the simulations ($\approx$ 100 h on a Cray Y/MP) was granted by the Forschungszentrum in Jülich, Federal Republic of Germany.

References

[1] Amit D J, Gutfreund H and Sompolinski H 1985 *Phys. Rev. Lett.* **55** 1530
[2] Gardner E 1988 *J. Phys. A: Math. Gen.* **21** 257; 1987 *Europhys.Lett.* **4** 481
[3] Cover T M 1965 *IEEE Trans.* **EC14** 326
[4] Krauth W and Mézard M 1989 *J. Phys. A: Math. Gen.* **20** L745
[5] Anlauf J K and Biehl M 1989 **10** 687ff; 1990 *Europhys. Lett.* **11** 387
[6] Krauth W and Mézard M 1989 *J. Physique* **50** 3057
[7] Krauth W and Opper M 1989 *J. Phys. A: Math. Gen.* **22** L519
[8] Köhler H M, Diederich S, Kinzel W and Opper M 1990 *Z. Phys.* b **78** 333
[9] Amaldi E and Nicolis S 1989 *J. Physique* **50** 2333
[10] Sompolinski H 1987 The theory of neural networks *Heidelberg Colloquium on Glassy Dynamics* ed L van Hemmen and I Morgenstern (Berlin: Springer)
[11] Wang Q 1987 *Biol. Cybern.* **57** 95
Rechenberg I 1973 *Evolutionsstrategie* (Stuttgart: Fromman)
[12] Eigen M 1979 *The Hypercycle—A Principle of Natural Selforganisation* (Berlin: Springer)