Spontaneous pattern classification by a dynamical system

W. Banzhaf

Tome 48 N° 12 DÉCEMBRE 1987

LE JOURNAL DE PHYSIQUE

J. Physique **48** (1987) 2027-2035 DÉCEMBRE **1987,** PAGE 2027

Classification
Physics Abstracts
03.20 - 46.10

Spontaneous pattern classification by a dynamical system

W. Banzhaf

Institut für Theoretische Physik der Universität Stuttgart, Pfaffenwaldring 57/IV, D-7000 Stuttgart, F.R.G.

(Reçu le 24 mars 1987, révisé le 24 août 1987, accepté le 25 septembre 1987)

Résumé.— Nous étudions les propriétés d'une architecture d'ordinateurs proposée récemment qui modélise un système dynamique. Nous généralisons le modèle et explicitons quelques questions importantes. Nous utilisons la complexité du bassin d'attraction comme mesure et illustrons l'utilité de ce système pour la reconnaissance de formes.

Abstract.— We study features of a recentrly proposed computer architecture which models a dynamical system. We generalize the model and give some central issues. We apply the basin complexity as a measure and exemplify the use of the system for pattern recognition.

Three years ago, an arrangement of computers has been proposed [1,2] which represents a kind of a dynamical system. A more detailed analysis was given by the same authors in [3].

If one models the following principles in a system of computers :

— many equally structured computing elements are forming a system,

— exchange of data is going on locally, i.e. between neighbours,

— the algorithms contain non-linearities,

— the whole system is open with respect to flow of information,

— the dynamics of interaction between computing elements dominates behaviour of the system,

then it should be expected that selforganizing processes could be observed. These processes and its utility for pattern recognition will form the subject of interest here.

The aforementioned principles play a prominent role in models of nature, where formation of an enormous number of different patterns is to be explained. By analogy it was concluded [4] that problems of pattern recognition could be solved by application of similar principles, if they were realized appropriately on computers. In fact, natural pattern recognition systems as, e.g. living organisms, make extensive use of these principles in coping with the challenges of its complex environment.

A general description could be stated as follows : the proposed arrangement of computers is one example of a multi-attractor system, where at the beginning given states $P_1, ..., P_n$ dissipatively

relax into (presumable) simple attractor states $A_1, ..., A_m$. One can see this evolution as processing of information [5-7], represented at the start of computation by some "input state" P_i, at the end by some "output state" A_j. The dissipative character of the system consists of destruction or compression of information easily seen if one compares numbers of possible states at the beginning of computation and at the end.
Although this is rather general, pattern recognition systems call for just this feature as one important ingredient.

Formulated more specifically : the classification of patterns requires firstly criteria of which information should be seen as features to be classified and which not (feature selection) and secondly the mechanism to compress information appropriately (feature extraction).

Throughout this paper we shall suppose the number of patterns is much greater than the number of classes.

Ordinary pattern recognition proceeds *via* dividing the classification task into two parts, where of a programmer often takes over the first part of feature selection using heuristic principles or alternatively running a program designed to cluster features and to transform the problem to a suitable basis. Then only the second part constitutes the problem of pattern recognition to be solved by a machine. Although the task as a whole can be executed by one program, there is, by contrast, no separation into two distinct parts in selforganizing systems, thus giving more flexibility to the entire system.

The real difference, however, between selforganizing pattern recogniton (PR) systems and deterministic or stochastic non-selforganizing ones is the fact that in a selforganizing system classes are determined by the nature of the dynamical system itself and the near-neighbour interactions of processing elements. Beyond that, the system respects to topological laws and automatically maps similar patterns into identical classes. This has to be seen in distinction to non-selforganizing systems, where all principles of the processing of patterns are to be set up explicitely.

Since in the case of the selforganizing systems considered here there is only a limited influence on the classification result from the outside, and in fact the system is able to classify just from the beginning of sample presentation, we shall call the classification "spontaneous". Spontaneous classification of patterns, however, poses serious questions which we shall address at the end : what are the semantics of classes and how can one use spontaneously emerging classes for controllable pattern recognition ?

1. The original system.

We now want to describe the abstract framework comprising as a special case the system [1].
Generally stated, we have a network of identical processing elements (PEs), which could be identified by its indices $PE_{ijk...}$ characterizing its location in a net.
Every PE possesses an OUTPUT value $OP_{ijk...}$, a vector of INPUT values

$$\mathbf{I}_{ijk...} = \left\{ I^{(l)}_{ijk...} \right\}, \; l = 1, 2, ..., v$$

and additionally the vector of MEMORY values

$$\mathbf{S}_{ijk...} = \left\{ S^{(m)}_{ijk...} \right\}, \; m = 1, 2, ..., s$$

Therefore, every processor has to its disposal several registers to hold memory values (sometimes, but not here, called processor state) and a couple of input lines to receive messages from others which are represented mathematically by $v-$ resp. s-dimensional vectors. On the basis of this information it computes one unambigeous output.
The algorithm which has to be executed by every PE each time step t reads globally (suppressing indizes)

$$OP(t+1) = f(\mathbf{I}(t), \mathbf{S}(t)) \qquad (1)$$

although a local version is reasonable as well

$$OP_{ijk..}(t+1) = f(\mathbf{I}_{ijk..}(t), \mathbf{S}_{ijk..}(t), i, j, k, ..) \quad (1a)$$

In this way depending on content of memory, on inputs and (eventually) on location in the net a new state (= output) will be computed.
The non-linearity necessary for selforganization is generated by an additional dependence of memory values on the state of the PEs

$$\mathbf{S}_{ijk..}(t+1) = g(\mathbf{S}_{\{opq..\}}(t), OP_{\{rsu..\}}(t)) \quad (2)$$

If the ouput of every processor is recycled as input for another one and functions f and g contain

some - e.g. saturation - non-linearities, the system exhibits features of a simple multi-attractor system.
The algorithms (1) and (2) are chosen in such a way as to ensure that state values OP are fast relaxing (stable) variables whereas memory values S are slowly varying (instable) quantities. Stated in the language of synergetics, the memory values are allowed to slave the state values [8].
When S is driven into saturation due to its non-linearity, also the OP.s reach limit values. Chosing, instead, saturation of OP which are considerably below these limit values results in a strong robustness of OP against perturbations in S. This will be studied in more detail below.
To summarize : we have two interactions of different kind, one concerning the transfer of output information of PE.s to input lines of other PE.s which is a one-way interaction, the other concerning internal quantities of the processing units which consists of exchange of messages between PE.s in both directions. Furthermore, these two interaction types are distinct with respect to the time scales of its evolution equations.
In the following we limit ourselves to meshes of two dimensions. The special realization of Huberman [1] can be given here in compact form :

$$I_{ij}^{1,2}(t) = OP_{i\pm1,j-1}(t) \quad \mathbf{S}_{ij}(t) \equiv S_{ij}(t)$$
$$OP_{ij}(t+1) = \sigma\left[S_{ij}(t).(I_{ij}^{1}(t) - I_{ij}^{2}(t))\right] \quad (3)$$

$$S_{ij}(t+1) = S_{ij}(t) + \rho\left[\frac{1}{2}.\mathrm{sign}(OP_{ij}(t+1) - OP_{i-1,j}(t+1) + \frac{1}{2}.\mathrm{sign}\ OP_{ij}(t+1) - OP_{i-1,j}(t+1)), S_{ij}(t)\right] \quad (4)$$

$$\sigma(x) = \begin{cases} x & |x| \le a \\ a.\mathrm{sign}(x) & |x| \ge a \end{cases} \quad (5a)$$

$$\rho(x,y) = \begin{cases} x & b \le y \le c \\ 0 & \text{otherwise} \end{cases} \quad (5b)$$

with $\quad a = 15 \quad b = 1 \quad c = 4$

This is one of many possible algorithms for contrast adaptation.
In what follows we shall see that - besides the flexibility in algorithms - also the non-linearities and the density of state space can be subject to variability without serious changes in the general behaviour of the system.

2. Some observations.

We consider a system of $M \times N$ processing elements arranged as in [1].
To extract the essentials of the model constituted by equations (3,4,5), we have to consider some points more closely.
Beginning with algorithm (1) we remove its non-linearity σ and test the OP behaviour. Since the following features

i) S values are slaving OP values
ii) S values possess saturations

are still valid, we conclude that OP will still have limit values, although they will be considerable higher than just the saturation values of OP implemented originally by σ.
Figure 1 shows $OP-$distances $d(P)$ of periodically applied inputs as measured by means of the stroboscopic method used by Hogg and Huberman [1].

After growing up to large values the distance also converges to zero as in the original net, signalling adaptation to the inputs. But adaptation is now only due to the fast that S values sattle down after a transient time.
Use as a PR system, however, is not adequate, since we have removed "dissipation" and therefore the formation compression capabilities from I/O behaviour.
To state it in other words : what physicists mean by dissipation, i.e. contraction in phase space, is one important constituent of a pattern classification system.
The second point concerns the time-scale of adaptation.
By fixing the step ΔS to $\pm 1{,}0$ the mutation of S is kept small against that of OP. We remove this limitation, also with respect to an analog computation of values chosing

$$\Delta S_{ij} = \varepsilon(OP_{ij} - OP_{i-1,j})(OP_{ij} - OP_{i+1,j}) \quad (6)$$

Here we get our first control parameter ε adjusting the rate of convergence of the net. It will be useful in context of different kinds of fluctuations applied to the net [9].
Another change is concerned with the density in state space. With the allowance of real numbers the density is enlarged by a factor of 10^9 (single precision computations). The behaviour resulting form this change is qualitatively the same. This demonstrates the possibility of an analog realization of the net by operation amplifiers [9].

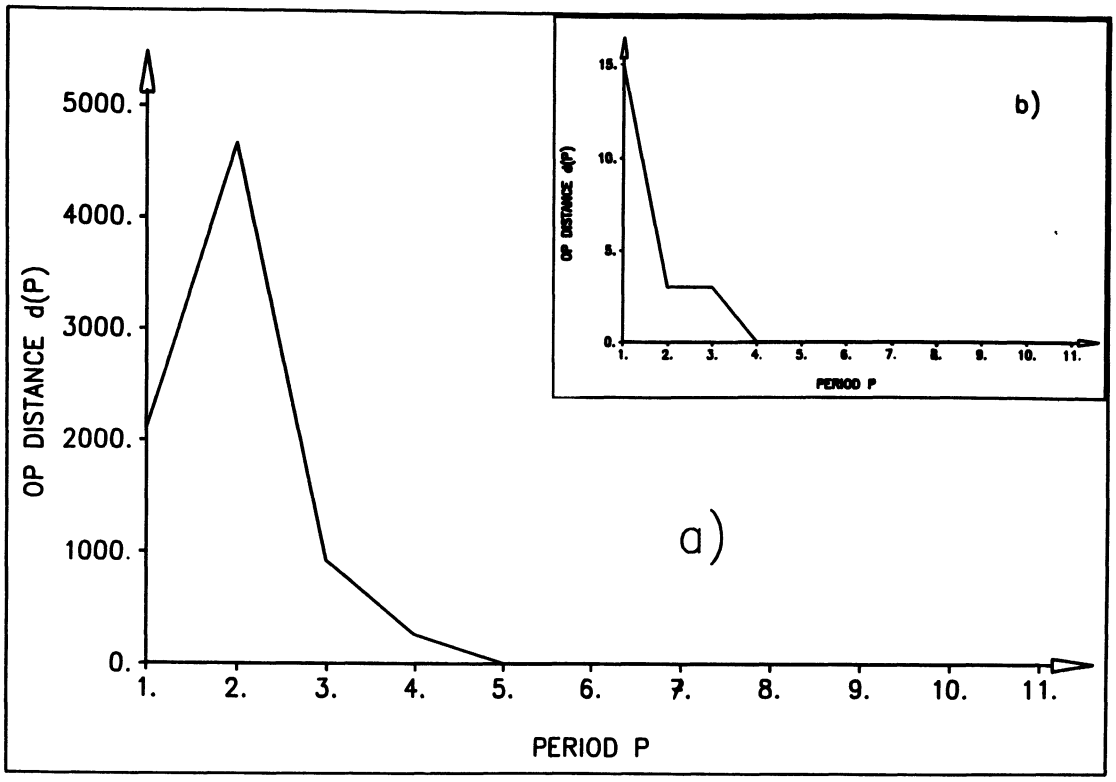

Fig.1.– Comparison of output distances $d(P) = \max_{\{i\}} [OP_i(tP) - OP_i(t(P-1))]$ over period number P in a 8×4 network with 4 input patterns (8 components each). $d(P)$ is going to zero, a) after growing up to large amounts due to displacement of saturations in OP values, b) just from beginning if OP values are limited to $[-15, +15]$.

Figure 2 shows the convergence behaviour, again measured in OP distances for different values of ε for a net with extended density in state space. In a further variation we change the algorithm (4) totally, substituting now the saturation non-linearity of S by some kind of limitation of resources. To this end the development of S must be made dependent on another control parameter S_C, which limits the average of S_{ij} thereby introducing a cooperative interaction

$$S_{ij}(t+1) = (1+R).S_{ij}(t) + \rho(...) \qquad (7a)$$

$$R = S_{ij}(t).(S_C - \left(\frac{1}{M.N}\right) . \sum_{ij} S_{ij}) \qquad (7b)$$

Here, for the sake of simplicity we let the damping R depend on the global average of S_{ij}. Although this seems to be a violation of the principle of local interactions, it is common ingredient to all natural systems to be limited with respect to its resources. On the other hand, dependence on the global state of a parallel computer system is a problem of global communication solutions of which are proposed in many text books on parallel computing, e.g. [10].

In contrast, different local algorithms are possible as well. E.g. these may perform summations of S_{ij} in a specified processor neighbourhood as in the case of cellular automata it is performed by totality rules [11].

Figure 3 shows a comparison of memory values S for the original (a) and the cooperative (b) algorithm with identical inputs. One can see the considerable concentration of S values deviating from 1. The important point about this is that with few deviations of values from the starting state in the cooperative algorithm case we obtain the same basins of attraction as in the saturation case, where nearly all memory values have to be changed.

By the way, removing non-linearities σ and ρ simultaneously results in a similar non-dissipative behaviour as described before.

What was the motivation for considering some issues more closely here ?

1. Modelling dynamical systems by computer networks is just at its dawn. To get a feeling of

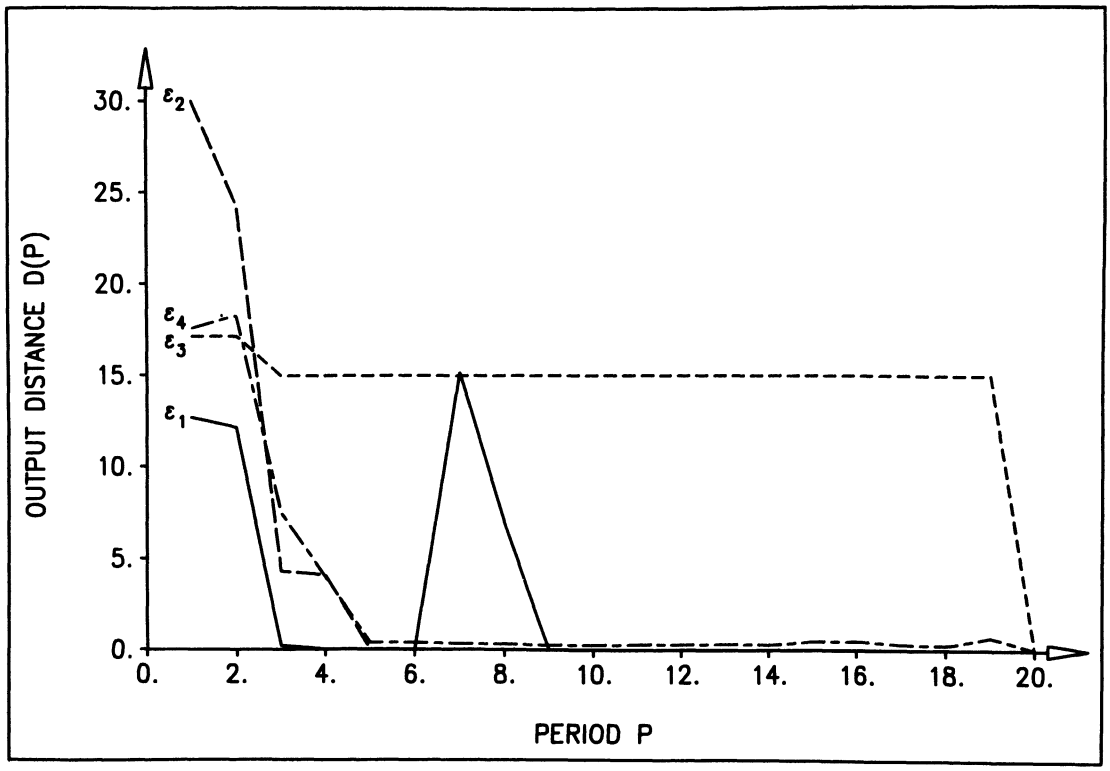

Fig.2.– Different rate of convergence $d(P)$ for different choice of control parameter ε, $\varepsilon_1 = 6 \times 10^{-4}$, $\varepsilon_2 = 6 \times 10^{-3}$, $\varepsilon_3 = 2 \times 10^{-2}$, $\varepsilon_4 = 1$ in a 8×4 network computed with real numbers.

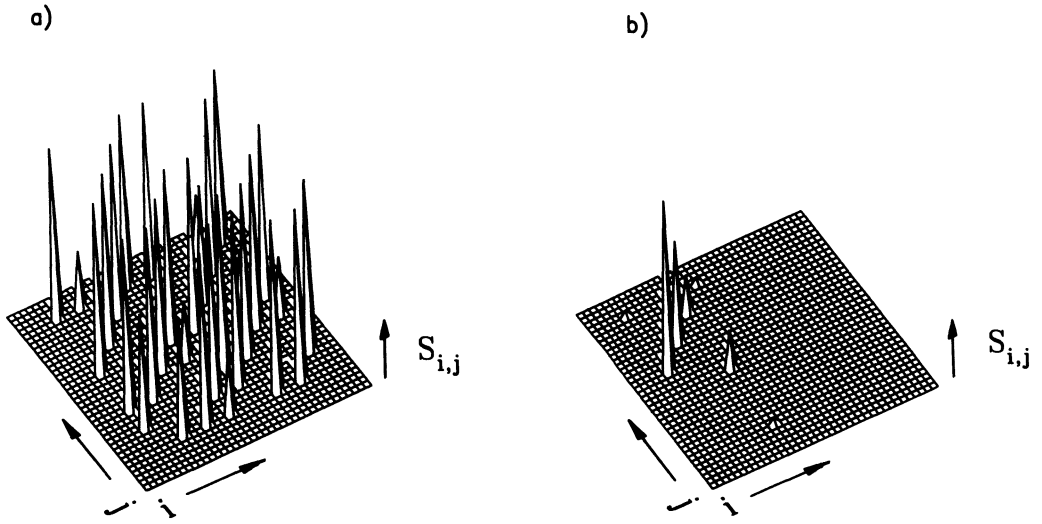

Fig.3.– Memory values $S_{i,j}$ differing from starting value 1 (a) Original algorithm of Eq. (3,4,5). (b) Cooperative algorithm, Eq. (3,7a,7b), $S_C = 2.0$ in a 8×8 network with 12 input patterns. Both exhibit approximately same basin structure.

the essential points one has to collect experiences with respect to the question, what properties would be dispensable for the behaviour as a dynamical system and what would be constituent. As dispensable were identified :

i) saturation non-linearities,

ii) choice of special points in phase space for the evolution of the system.

2. A second point is concerned with the influence from outside onto the dynamical system.

Computers should execute orders. If one is confronted with a non-transparent network at least some control parameters are necessary to be able to influence something.

3. The most fundamental argument is that we need more understanding of processes in networks. What seems to be sure is that modelling dynamical systems on computers will bring to light many new properties and will enable performances, which up to now only mark biological systems.

We can expect important contributions to the relation between information processing "devices" in electronics and nature.

We close with an observation possible in all nets : these are attractor cycles, where $d(t) \neq 0$ after relaxation. The appearance of two or more attractors for one input signals bifurcations and the possibility of generating more complicated series of outputs.

3. General behaviour of the meshes.

We now want to demonstrate the behaviour of meshes quite general. All experiments were done in nets with algorithm of equations (3,4,5).

As mentioned at the beginning the meshes show dissipation, i.e. contraction of volume in the state space of state variables. As an estimate, in a net of width M (with the same number of input components) and input data of integer numbers in [−15,+15] the number of possible states will be diminished by a factor of

$$K = (3/31)^{-M} \approx 10^{-M}$$

Here it was assumed that all state values develop toward three possible states : ±15, 0, generating 3^M attractors.

As a measure to examine the basins of attraction in state space we introduce the attractor complexity [12]

$$C_{\mathrm{B}} = - \sum_i p_i \ln p_i \qquad (8)$$

p_i is the fraction of volume in state space due to attractor A_i. Or, since the state space is accessible only statistically, p_i means the probability to find A_i as the output if the input is generated by an equally distributed random process.

For all measurements, the adaptation property of the nets should be turned off, i.e. S constant, otherwise it would strongly influence the result. This is a consequence of the very fast adaptation compared to other models, e.g. neural nets, [13]. Generally speaking the attractor complexity is an averaging measure of the basin structure of a net. One example of a real structure is shown in figure 4. Nets also exhibit spontaneous classification abilities if no training inputs were applied (see Fig.4).

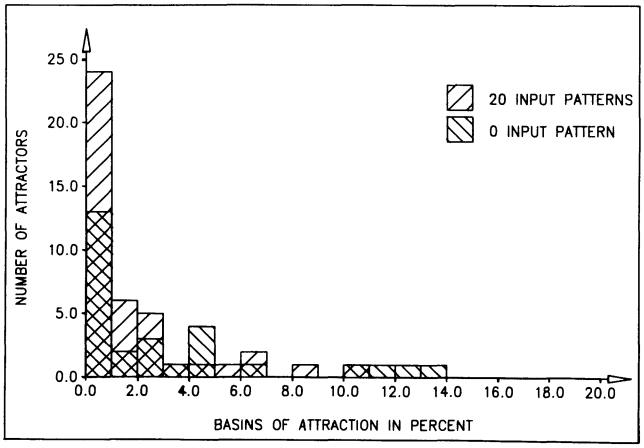

Fig.4.– Basin structure of a 8×8 net with 20 inputs and 0 inputs.

Observation of developments in this structure can also be used with profit.

In figure 5 one can see dependence of C_{B} on M, N and the number of inputs in a training sequence. The fact that C_{B} is growing with the number of trained inputs shows clearly a shrinking of the basins of attraction.

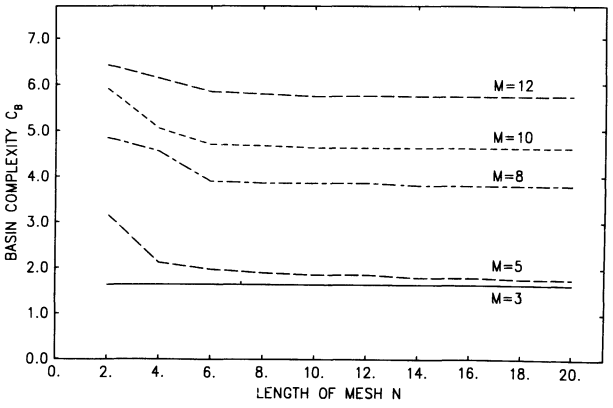

Fig.5.– Basin complexity $C_B = -\sum_i p_i . \ln p_i$ against length N of nets for different width M. C_B approaches limit values as N grows. (On the basis of 10000 input trials each).

Constant C_B for $M = 2, 3$ is due to border effects.

Falling C_B with increment in N, on the other hand, means that some possible reactions of the net at smaller N die out if length N approaches about the width M of the net. Whether this could be used as a parameter to control some kind of resolution during the classification process remains as an open question.

4. Pattern classification abilities.

As mentioned above one application of the nets discussed will be a pattern recognition system. Patterns are represented by strings of numbers, here taken from the interval $[-15, +15]$. Classes are also represented by strings of numbers, regularly consisting of saturation states ± 15, sometimes also of the neutral state 0 or another constant value.

The situation resembles that of the "brain-state-in-a-box"-model of Anderson *et al.* [14], where also non-saturation values appeared in the final states. Without adaptation, the system spontaneously classifies all input patterns into output classes and two problems arise :

i) In which way classes can be made flexible ?

ii) What are the meanings of classes ?

The first problem could be solved partially by turning on adaptation (S develops according to Eq. (4)). Then the behaviour of the net seems to exhibit generalization and differentiation, depending on presented training samples. Table I demonstrates that the considered computing structures exhibit exactly this characteristic which can be formulated quantitatively in the following way : suppose you have an ensemble of pattern vectors $\mathbf{I}_j$, $j = 1, J$ used as sample patterns for the systems training period and a corresponding set of output vectors $\mathbf{O}_j$ representing the classes. Then the variance of the $\mathbf{O}_j$ defined as

$$\mathrm{Var}(\mathbf{O}) = \langle \mathbf{O}.\mathbf{O} \rangle - \langle \mathbf{O} \rangle . \langle \mathbf{O} \rangle ,$$

$$\langle \mathbf{O} \rangle = \frac{1}{J} \sum_j \mathbf{O}_j$$

is up to marginal variations presumably due to random effects independent of the variance of the $\mathbf{I}_j$ analogously defined.

Different classes are formed also from similar patterns of the training period and the system is trained to amplify these small differences. Vice versa, large differences in training vectors also result in the same output variance indicating abstraction of the system (see Tab. I).

This is, however, only part of the story, since the adaptation algorithm restrains the behaviour of the net to contrast adaptation. Although there are algorithms which allow to manipulate the basins of attraction according to other criteria [15], conservation of topology, in general, will be a fixed constraint.

Table I.– *Abstraction and generalization abilities in nets with a different number of sample inputs.* (8×8 *net, original algorithm).*

Number of Sample Inputs	Var(IP)	Var(OP)
5	20,7	1103,8
5	468,2	1551,6
5	918,0	1296,0
5	1760,0	1566,0
10	24,0	1641,1
10	503,7	1494,0
10	1113,8	1401,8
10	2032,8	1390,8
15	15,1	1611,8
15	468,7	1640,0
15	1030,0	1502,0
15	2323,0	1656,0

This means that a classification totally at will is not possible in self-organizing nets.

The semantics of classe remains as the other problem of this approach. As a consequence of spontaneous classification the meaning of classes has to be fixed separately. Figure 6 shows a solution to this problem.

Patterns coming from outside the system are transformed to attractors by a dynamical system. A replica of the system with fixed memory parameters S_{ij} is available internally.

In (B) patterns are generated as long as there is no match between SA and GA, i.e. at the level of attractor states. This is done by a simple evolution algorithm according to the so-called two-membered (1+1) evolution strategy [16]. Starting from a given (parent) pattern in generation N, $G(P, N)$, a slightly modified child pattern $G\ (C, N)$ is generated by a random mutation in one of its components. In the next generation, only a single one will "survive", i.e. will

be $G(P, N+1)$. For to select between $G(P,N)$ and $G(C,N)$ a measure of quality of each pattern has to be introduced. This is done here by using the euclidian distance of the resulting attractor states $GA[G(P,N)]$ and $GA[G(C,N)]$, respectively, from SA. The pattern with smaller distance will be $G(P, N+1)$. The mutation process is stopped, when SA and GA are the same, i.e. its distance is zero.

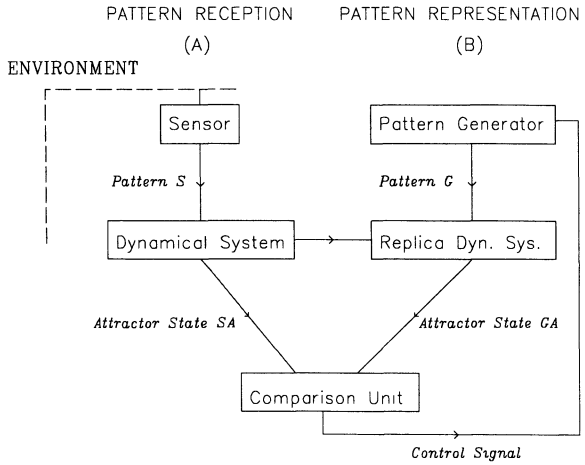

Fig.6.– Pattern recognition system.

The examples given in table II always start from $G(P,O) = (14,13,12,...,8,7)$. Number of mutations is equal to number of generations necessary to end up with the correct attractor state.

Comparison of external and internal patterns at the level of fixed points of the dynamical system determines therefore the semantics : patterns G in (B) generates the same result as pattern S in (A), or to state it in other words, G is the internal representation of S and an attractor $SA(=GA)$ means G for this dynamical system.

Comparison of patterns at the level of its attractor states helps to avoid production of much redundant information. It suffices to find one representative state of class SA in (B). Of course, there is a huge number of applicable evolution algorithms [16,17] varying greatly in speed and complexity. As a consequence of the fact, however, that many basins of attraction are of considerable size (of order $10^{-3}...10^{-2}$ of state space) even one of the simplest algorithms is successfull.

5. Conclusions.

Examination of computer modeled dynamical systems open new routes to pattern recognition. Although preliminary in results, the relevance of these models in application to special classification problems is shown. As many dynamical systems as possible, however, - only one example of which was presented here - should be subject to examination prior to comparison with conventional methods of pattern recognition. At the long run there is hope to find a flexible algorithms using selforganizing principles which is comparable in performance to traditional pattern discrimination algorithms.

Table II.– *Examples of recognition of patterns S. The evolution algorithm always starts from G=(14,13,12, 11,10,9,8,7) and modifies G the given number of times until the same attractor is reached.*

Pattern S	Attractor SA = GA	Pattern G	Mutations
+15 +12 −12 −8 −15 −14 −9 +12	+ + 0 − − − + +	+12 +9 +12 +15 +10 +2 +8 +7	74
−12 −5 −14 +1 +12 +6 −6 −10	+ − − 0 + + − −	+14 +13 +2 +11 +10 +9 +6 +7	40
−13 −4 +4 −14 −14 0 +12 −6	− + + − − + + −	+14 +13 +12 +11 +7 +9 +8 +5	7
−10 +2 +1 +2 +14 0 +12 +9	− + 0 + + − + +	+13 +13 +12 +11 +10 +9 +9 +7	15
+6 −13 +3 +15 +6 +9 −7 +4	+ − − + + − − +	+14 +8 +12 +12 +10 +5 +4 +10	55

Acknowledgments.

This research was supported by "Stiftung Volkswagenwerk". The author is grateful for discussions to Prof. H. Haken. He also expresses his gratitude to the Institute of Theoretical Physics for providing him with the necessary computer power.

References

[1] HOGG, T., and HUBERMAN, B.A., *Phys. Rev. Lett.* **52** (1984) 1048.

[2] HUBERMAN, B.A., *Phys. Scripa* **T9** (1985) 165.

[3] HOGG, T., and HUBERMAN, B.A., *Phys. Rev.* **A32** (1985) 2338.

[4] HAKEN, H., (Ed.), *Pattern Formation and Pattern Recognition by Dynamical Systems,* Springer Series in Synergetics, Vol. 5 (Springer, Berlin) 1982.

[5] EBELING, W., and FEISTEL, P., *Physik der Selbstorganisation und Evolution* (Akademie Verlag, Berlin) 1982.

[6] HAKEN, H. in : *Stochastic Phenomena and Chaotic Behaviour in Complex Systems*, P. Schuster (Ed.), Springer Series in Synergetics, Vol. 21 (Springer, Berlin) 1984.

[7] SHAW, R., *Z. Naturforsch.* **36a** (1981) 80.

[8] HAKEN, H., *Synergetics - An Introduction* (Springer, Berlin) 1983.

[9] in preparation.

[10] HOCKNEY, R.W., and JESSHOPE, C.R., *Parallel Computers* (Adam Hilger Ltd., Bristol) 1981.

[11] WOLFRAM, S., *Rev. Mod. Phys.* **55** (1983) 601.

[12] KANEKO, K. in : *Dynamical Systems and Nonlinear Oscillations*, G. Ikegami (Ed.) (World Scientific Publishers, Singapore) 1985.

[13] For a review see e.g. CLARK, J., RAFELSKI, J., and WINSTON, J., *Phys. Rep.* **123** (1985) 215.

[14] ANDERSON, J.A., and MOZER, M. in : *Parallel Models of Associative Memory*, J.A. Anderson, G.E. Hinton (Ed.) (Lawrence Erlbaum Ass., Hillsdale, N.J.) 1981.

[15] HOGG, T., and HUBERMAN, B.A., *J. Stat. Phys.* **35** (1985) 115.

[16] RECHENBERG, I., *Evolutionsstrategien*, in : B. Schneider, U. Ranft (Eds.) Med. Informatik und Statistik, Vol. 8 (Springer, Berlin) 1978.

[17] SCHWEFEL, H.P., *Numerical Optimization of Computer Models* (Wiley, Chichester) 1981.