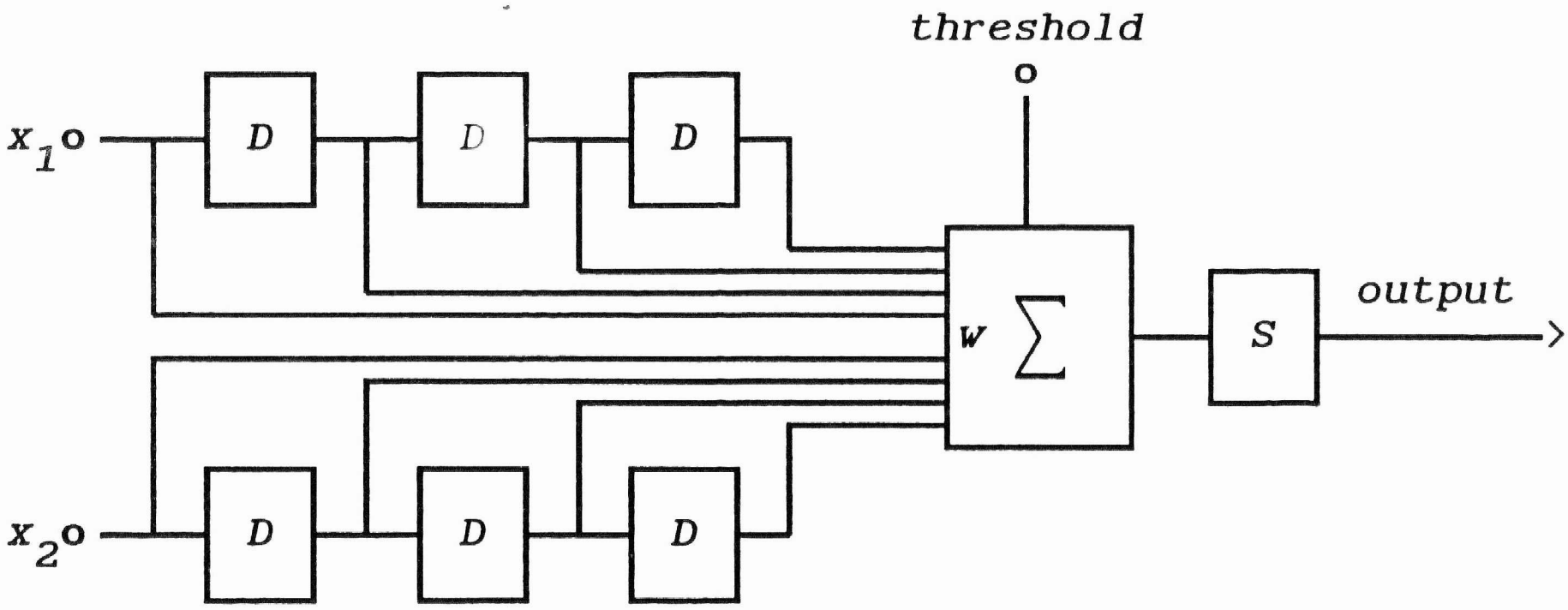
IDG

NEURAL NETWORK WORLD

International Journal on Neural and Mass-Parallel Computing and Information Systems

VOLUME 1 **1991** **NUMBER 5**



Hornik K.: **Functional Approximation and Learning in Artificial Neural Networks**
Kufudaki O., Hořejš J.: **PAB: Parameters Adapting Back-Propagation**
Herrmann M., Englisch H.: **Neural Nets as Heteroassociative Memories**
Kainen P.C.: **On Parallel Heuristics**
Hlaváčková K.: **Basic Algorithms for Maps**
Beran H.: **Time Delay Neural Networks**
Goltsev A.D.: **The Neuronlike Network for Brightness Picture Segmentation**
Fatton D.: **On the Uniform Training**
Hořejš J.: **A View on Neural Network Paradigms Development (Part 5)**

NEURAL NETWORK WORLD is published in 6 issues per annum by the ComputerWorld Company, Czechoslovakia, 12000 Prague 2, Blanická 16, Czechoslovakia, the member of the IDG Communications, USA.

Institute of Computer and Information Science, Czechoslovak Academy of Sciences, 18207 Prague, Pod vodárenskou věží 2, Czechoslovakia.
Phone: (00422) 82 16 39, (00422) 815 2080, (00422) 815 3100

Fax: (00422) 85 85 789,

E-Mail: CVS15@CSPGCS11.BITNET

Price Information:
Subscription rate 399 US$ per annum.
One issue price: 66.50 US$.
Subscription adress: ComputerWorld Co., Czechoslovakia, 120 00 Prague 2, Blanická 16, Czechoslovakia.

Scanning the Issue

Editorial p.257

Papers:

Hornik K.: **Functional Approximation and Leraning in Artificial Neural Networks** p.257
Rigorous theory of functional approximation and learning (estimation) is studied.

Kufudaki O., Hořejš J.: **PAB: Parameters Adapting Back-Propagation** p.267
New algorithm for back-propagation learning is presented.

Herrmann M., Englisch H.: **Neural Nets as Heteroassociative Memories** p.275
The ability of neural network to store information as heteroassociative memory devices is discussed.

Kainen P.C.: **On Parallel Heuristics** p.281
Parallel heuristic based on mathematical knowledge is considered.

Hlaváčková K.: **Basic Algorithms for Maps** p.287
Learning variants of self-organization in Kohonen map type networks are demonstrated.

Beran H.: **Time Delay Neural Networks** p.295
New adaption algorithm based on the equivalence of the time delay neural network structures to the classical back-propagation structure is presented.

Goltsev A.D.: **The Neuronlike Network for Brightness Picture Segmentation** p.303
Algorithm for brightness picture segmenation realized as a part of computer modeled neural network is descibed.

Fatton D.: **On the Uniform Training** p.309
New strategies of backpropagation training avoiding the danger of local minimum convergence are discussed.

Tutorial:

Hořejš J.: **A View on Neural Network Paradigms Development (Part 5)** p.313

Book Review p.276

Book Alert p.301

Comming Events p.311

Literature Survey p.266,274,294,302,308,320

ISSN 1210-0552

Editorial

Recent increase in the number of submitted manuscripts allows us to present a little wider spectrum of papers which we present to our readers in this issue. But we are still taking care of the balance between those contributions which are predominantly theoretical and those whose focus is more oriented toward applications. We have selected K. Hornik's paper concerning the still important problems of the neural networks learning and approximation as the first in this issue. The sequence of theoretical oriented paper follows by the contribution prepared by O.Kufudaki and J.Hořejš, presenting a new approach to the design of back-propagation learning algorithms for one - dimensional layered neural networks. The third paper included in this part of our Journal was written by M.Herrmann and H.English and deals with the problems of heteroassociative memories of the modified Hopfield structure type. P.C.Kainen address some very interesting questions about neural networks heuristics and K. Hlaváčková concerns her interest in mapping algorithms. The last three papers deal with some more practical problems: H. Beran address the applications of time-delay neural networks, A.D. Goltsev the problems of picture segmentation and D. Fatton some problems of training.

As in all the previous issues, we include here the continuation of J. Hořejš's tutorial on neural network paradigms.

We continue also in presenting information on interesting scientific conferences, symposium and other meetings and also on interesting books. We also include a selection of references concerning neural networks, neurocomputing and mass-parallel systems, selected from the Scientific Information System of the Institute of Computer and Information Science of the Czechoslovak Academy of Sciences, Prague.

M.Novák

FUNCTIONAL APPROXIMATION AND LEARNING IN ARTIFICIAL NEURAL NETWORKS

Kurt Hornik[1]

Abstract:

We discuss the potential of using artificial neural networks in problems of functional approximation and learning (estimation). It is argued that an analysis of this potential should be based on a rigorous theory rather than on the findings of particular simulation experiments. We survey some of the results which have already been established, discuss their relevance and indicate directions in which we think further research will be necessary.

1. Introduction

Within the last few years, artificial neural networks (ANN's) have received quite rapidly growing amounts of both academic and commercial interest, to a large extent due to the fact that they promised to be a universal and easy-to use means of "learning" relations and interactions in complex nonlinear systems for which similar means of understanding and modeling had previously been unavailable. With the aid of the popular generalized delta rule for adjusting the variable networks connections strengths to data sets, researchers and practitioners have applied ANN's to a huge number of problems in various fields. However, today it seems that the period of more or less uncontrolled experimentation is slowly coming to an end, and that a certain amount of the initial enthusiasm has already disappeared and been replaced by a continuously growing demand for a better-structured and increasable scientific approach to ANN's which in particular should make clear what the real advantages of this new technology are.

In fact, the functional "learning" capabilities of ANN's can conveniently be analyzed in a rigorous mathematical framework. For simplicity, let us restrict our attention to

[1]Kurt Hornik
Institut für Statistik und Wahrscheinlichkeitstheorie
Technische Universität Wien, Vienna, Austria

feedback-free architectures in which information is processed in only one direction. Then clearly, for each fixed set of interconnection strengths, the network implements a rule for computing values at, say, l output units, given values at, say, k input units, or in other words, a mapping from $\mathcal{R}^k$ to $\mathcal{R}^l$. Attempting to "learn" a function f from $\mathcal{R}^k$ to $\mathcal{R}^l$ means to use a collection of input-target patterns to determine a suitable feedforward network architecture and adjust the network weights in a way that the mapping implemented by the thus trained network is close to f. In other words, learning is a data-based rule for finding approximators to functions. (The distinction between mere approximation and learning has sometimes been neglected in the neural network literature. Clearly, a system can simultaneously possess extraordinary approximation and very limited learning capabilities, e.g. because it is so complicated that an algorithmic fine-tuning to a specific problem is virtually impossible).

How closeness between functions is measured depends significantly on the specific problem to be dealt with. In many applications, it is necessary to have the network perform *simultaneously* well on all input samples taken from some compact input set X in $\mathcal{R}^k$. In this case, closeness is measured by the uniform distance between functions on X, i.e.

$$\rho_{u,X}(f,g) = \sup_{x \in X} |f(x) - g(x)|.$$

In other applications, we think of the inputs as random variables with common distribution μ and are interested in the *average* distance, where the average is taken with respect to the "input environment measure" μ. In these cases, closeness can e.g. be measured by the $L^p(\mu)$ distances

$$\rho_{p,\mu}(f,g) = \left[\int_{\mathcal{R}^k} |f(x) - g(x)|^p d\mu(x)\right]^{1/p},$$

$1 \leq p < \infty$, the most popular choice being $p = 2$ corresponding to mean square error. In decision-making problems as in classification or pattern recognition where to each input pattern there corresponds one of finitely many possible actions, it may be desired to make the probability of incorrect decision small, which leads to the closeness measure

$$\rho_{\neq,\mu}(f,g) = \mu\{x \in \mathcal{R}^k : f(x) \neq g(x)\};$$

more generally, one could consider the expected loss resulting from incorrect decisions.

Of course, there are many more ways of measuring closeness of functions. For a more detailed account on this issue, cf.e.g. [3] and in particular, White [47, 48].

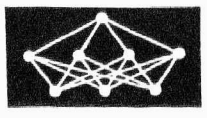
Which closeness measure is actually employed in a specific problem usually depends on the researcher's or practitioner's attitudes and preferences and is thus much more the result of a highly prejudiced choice than of a generally valid rule. (Of course, a measure can be infeasible in certain application; e.g. if the effect of avoidably incorrect medical treatment could be lethal, "average" closeness measures are clearly ruled out). Mostly based on the fact that it does not suppress significant differences on possibly large subsets of the input space by assigning little importance to them, it is often argued that the uniform distance is always to be preferred to average measures as $\rho_{p,\mu}$, see e.g. [50]. Contrary to that, we think that although the uniform distance is of considerable theoretical importance (if the input space X is compact, uniform closeness implies closeness as measured by $\rho_{p,\mu}$), its practical importance is significantly reduced by the fact that it usually entails unrealistically large requirements for network size and learning time, and definitely does not scale well, to say the least, with even moderately large problem sizes.

However, the subjectiveness in the choice of the closeness measure must not be mistaken as something unscientific. It simply reflects the fact that the user himself must judge the relevance of certain features to his problem and in particular, he must decide how the *performance* of a specific approximation or learning system is to be measured. In the context of pure approximation, this can be done by suitably trading off the complexity of the approximating system, e.g. quantified by the number of hidden neurons it utilizes for internal representation, to the accuracy of approximation it yields. In the sampling context of actual learning, one additionally has to take into account the number of training patterns necessary to guarantee a certain degree of approximation and/or generalization, as well as the computational requirements which arise during the training period. Hence, the user has to measure the performance of a learning system on a certain task relative to the "costs" of structural, sampling, and computational complexity.

Assuming that such a performance measure has been specified, one can start to compare the capabilities of feed forward network architectures to those of different methods of functional approximation and learning, as e.g. kernel smoothing or regression methods based on wavelets or more traditional systems such as polynomials, trigonometric polynomials, and splines.

It is therefore of fundamental importance to rigorously clarify in which functional learning tasks of practical interest, and in what sense, ANN's outperform their competitors. For these problems, they should become accepted as a powerful tool in applications and successfully be employed there; on the other hand, their use in problems to which other systems are better suited should come to an end within more of less soon. It should not be necessary to point out that such an analysis, as refined and challenging as it may be, has to be carried out systematically and cannot be based on the results of individual simulation experiments.

It is the aim of this presentation to contribute to this analysis, partially by providing several important results which have already been obtained, and partially by exhibiting problems of current interest and indicating possible directions of future research. It is not attempted to give a complete survey of all available results on approximation and learning capabilities of feedforward networks.



2. Approximation

2.1. Existence of Approximators

Probably until five years ago it seems to have been more or less commonly believed that in their famous book on *Perceptrons*, [40] had demonstrated that multilayer perceptrons of the type introduced by [39] were fundamentally incapable of performing certain computational tasks. Although the results of Minsky and Papert only apply to certain special classes of perceptrons - and one nowadays feels that this fact should immediately have become obvious to researchers - they nevertheless appear to have been the cause of a substantial reduction of neural network research intensity for more than a decade.

When finally researchers started to train simple feedforward architectures by the generalized delta rule ("error back-propagation algorithm"; [41], they were rather surprised by their success on basic learning problems, and they became interested again in the investigation of the computational capabilities of sufficiently complex multilayer feedforward networks.

A first result in this direction was obtained by [16] who used a theorem initially due to the famous Russian mathematician [27] to establish that, upon suitably choosing the activation functions, an arbitrary continuous function f can exactly be represented over compact subsets X of $\mathcal{R}^k$ by a finite neural network. More precisely, assume for simplicity that $X = [0,1]^k$ and $l = 1$, and let $\xi_1, \ldots, \xi_k$ be the components of the input vector x. Kolmogorov's result, as later on refined by [38] and [43], implies that there exist continuous one-variable functions ϕ and ψ_j, $j = 0, \ldots, 2k$, and a universal constant λ such that f can be represented as

$$f(x) = \sum_{j=0}^{2k} \phi\Big(\sum_{i=1}^{k} \lambda^{ij} \psi_j(\xi_i)\Big), \quad \forall x \in [0,1]^k.$$

The functions ψ_j are monotonically increasing satisfy a Lipschitz condition with exponent $\log(2)/log(2k+2)$, and are universal for the given dimension k; only ϕ depends on f. However, the above representation cannot directly be used in functional learning because it is by no means clear how ϕ should be parameterized to allow for adjusting to a specific f, see in particular [15]. As has recently been pointed out by [32], cf. also [31, 33], the techniques which have been used for deriving the above representation can also be employed to give a rather elementary proof that for an arbitrary sigmoidal function ψ, continuous functions can be approximated arbitrarily well (with respect to uniform distance) by sufficiently large feedforward networks with two layers of hidden units which all have the same activation function ψ. (A function $\psi : \mathcal{R} \mapsto [0,1]$ is called *sigmoidal* $\lim_{t\to-\infty} \psi(t) = 0$ and $lim_{t\to\infty}\psi(t) = 1$; if in addition ψ is nondecreasing, it is called a *squasher*).

Another early result has been given by [24] who showed that integrable functions f which the Fourier inversion formula holds can be represented by a single hidden layer network with a continuum of hidden units which all have the same integrable and nonzero activation function ψ. More precisely, let

$$\hat{f}(a) = (2\pi)^{-k} \int_{\mathcal{R}^k} e^{-ia'x} f(x) dx$$

be the Fourier transform of f (in what follows, ' denotes transpose such that $a'x$ is the dot product of a and x), and pick a nonzero real number λ such that $\hat{\psi}(\lambda) \neq 0$; such λ exists because if ψ is nonzero, its Fourier transform $\hat{\psi}$ cannot vanish identically on $\mathcal{R} \setminus \{0\}$. Then

$$f(x) = \int_{\mathcal{R}^k} \int_{\mathcal{R}} \psi(a'x + \omega) K_\lambda(a,\omega) da d\omega,$$

where

$$K_\lambda(a,\omega) = \Re\Big(\frac{|\lambda|^k e^{-i\lambda\omega} \hat{f}(\lambda a)}{2\pi\hat{\psi}(\lambda)}\Big).$$

However, the most popular activation functions, like the logistic or arctangent squasher or the Heaviside function, are not integrable. Therefore, it appeared at first glance that the Irie-Miyake integral representation was of limited practical interest, and its potential value was seriously underestimated.

For what follows, it will be convenient to introduce some notations. For compact subsets X of $\mathcal{R}^k$, $C(X)$ is the space of all continuous functions on X. A set $\mathcal{G}$ of functions on X is *dense* (with respect to the uniform distance $\rho_{u,X}$) if for all f in $C(X)$ and $\epsilon > 0$ there is a function $g = gf, \epsilon$ in $\mathcal{G}$ such that $\rho_{u,X}(f,g) < \epsilon$. Similarly, if μ is a finite measure on $\mathcal{R}^k$ and $1 \leq p \leq \infty$, $L^p(\mu)$ is the space of all (Borel measurable) functions for which $\int_{\mathcal{R}^k} |f(x)|^p d\mu(x) < \infty$, and a set functions $\mathcal{G}$ is dense in $L^p(\mu)$ (with respect to $\rho_{p,\mu}$) if for all f in $L^p(\mu)$ and $\epsilon > 0$ there is a function $g = g(f,\epsilon)$ in $\mathcal{G}$ such that $\rho_{p,\mu}(f,g) < \epsilon$.

Finally, for functions ψ defined on $\mathcal{R}$, let

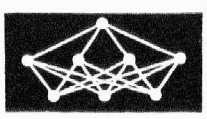
$$\mathcal{N}_k(\psi; n) = \Big\{g : \mathcal{R}^k \to \mathcal{R} : g(x) = \sum_{j=1}^{n} \beta_j \psi(a_j'x + \omega_j)\Big\}$$

and

$$\mathcal{N}_k(\psi) = \bigcup_{n=1}^{\infty} \mathcal{N}_k(\psi; n);$$

then $\mathcal{N}_k(\psi; n)$ and $\mathcal{N}_k(\psi)$ are the sets of all functions implemented by a single hidden layer feed forward network with k linear input units, n respectively an arbitrarily large number of hidden units with activation bias, and a single linear output unit. This architecture clearly constitutes a fundamental building block for more general multilayer multi output feedforward networks.

Equipped with this terminology, the fundamental question which functions can, to any desired degree of accuracy, be approximated by sufficiently complex ANN's can now be answered by determining in which function spaces $\mathcal{N}_k(\psi)$ is dense.



The first results of that kind were given in [13] for the particular case of ψ equal to the so-called cosine squasher. Quite unexpectedly, very general results were then obtained, independently and almost simultaneously, by [8], [12], and [21], who proved that for arbitrary compact subsets X of $\mathcal{R}^k$, $\mathcal{N}_k(\psi)$ is dense in $C(X)$, provided that ψ is squashing [21], continuous and squashing [12], or continuous and sigmoidal [8]. Similar results were obtained for $L^p(\mu)$ approximation for the case of compactly supported input environment measures μ .

Let us briefly sketch the ideas behind the respective proofs.

Hornik, Stinchcombe & White show that for arbitrary squashing ψ, the cosine function can be approximated by elements of $\mathcal{N}_1(\psi)$ uniformly compact subsets of $\mathcal{R}$. Their results then readily follow from well-known results on approximation capabilities of the trigonometric system. A similar argument was given in [17].

Funahashi observes that if ψ is squashing and $\gamma \neq 0$, then the function $t \longmapsto (\psi(t) - \psi(t-\gamma))$ is nonzero and integrable and can thus be used in the Irie-Miyake formula. The desired results on denseness of $\mathcal{N}_k(\psi)$ can then be obtained rather straightforwardly by providing Riemann sum approximations for the corresponding integrals. The same idea was uset lateron in [22] to establish, for sufficiently smooth and "l-finite" activation functions ψ, Sobolev-space denseness properties of $\mathcal{N}_k(\psi)$, i.e. capabilities of approximation with respect to closeness measures between smooth functions which also take into account how close their derivatives are, up to some order. Such smooth approximation results are of considerable importance in applications, e.g. in robotics (smooth movements, see e.g. [26], signal processing (determining the Lyapounov exponent of a chaotic time series, see e.g. [10]), and econometrics (elasticities of certain functions arising in the theory of the firm and the consumer); for more details, see [22]).

Cybenko applies powerful results from functional analysis - certain Riesz representation theorems for continuous linear functional and the Hahn-Banach theorem - to conclude that if $\mathcal{N}_k(\psi)$ is not dense in $C(X)$ respectively $L^p(\mu)$, there exists a nonzero, signed finite measure σ on $\mathcal{R}^k$ (actually, σ is concentrated on X respectively the support of μ) such that the function $L_\psi\sigma$, given by

$$L_\psi\sigma(a,\omega) = \int_{\mathcal{R}^k} \psi(a'x+\omega)d\sigma(x)$$

vanishes for all a in $\mathcal{R}^k$ and ω in $\mathcal{R}$. Hence, if a bounded function ψ is called *discriminatory* if no nonzero signed finite measure σ can exist for which $L_\psi\sigma(a,\omega) = 0$ for all a and ω, then establishing the desired results on approximation capabilities of $\mathcal{N}_k(\psi)$ is equivalent to showing that ψ is discriminatory, which Cybenko proves for the case of sigmoidal ψ.

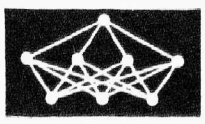
This elegant approach has been the source of even further generalizations of the basic approximation theorems.[19] shows that all *nonconstant bounded* functions are discriminatory; hence in fact, for arbitrary nonconstant and bounded ψ and arbitrary finite measures $\mu, \mathcal{N}_k(\psi)$ is dense in $L^p(\mu)$, and if in addition, ψ is continuous, then $\mathcal{N}_k(\psi)$ is dense in $C(X)$ for arbitrary compact subsets X of $\mathcal{R}^k$. Therefore, it is not the choice of a specific activation function, but rather the multilayer feedforward architecture itself which creates the potential of a universal learning system. Similarly general results can be obtained for smooth approximation capabilities. Apart from their apparent generality, the main importance of the theorems in [19] is the fact that they are the only currently available general results for $L^p(\mu)$ approximation capabilities of $\mathcal{N}_k(\psi)$ for the cases where μ is not compactly supported, which occurs e.g. if the input patterns are normally distributed.

It is quite natural to ask whether there exist activation functions which are discriminatory from a suitable *compact* subset Δ of $\mathcal{R}^{k+1}$, i.e., for which no nonzero, signed finite measure σ can exist for which $L_\psi\sigma(a,\omega) = 0$ for all (a,ω) in Δ. As a generalization of the above definition, such functions could be called Δ-discriminatory. This question is of considerable practical relevance because for such $\psi, \mathcal{N}_k(\psi)$ possesses universal approximation capabilities with uniformly bounded input-to-hidden weights. [44] showed that if ψ is a sufficiently kinky finite polynomial spline, or superanalytic at a point, then it is also Δ-discriminatory whenever Δ contains a sphere with sufficiently large radius; they also point out the cosine function does not have this property. In fact, they have recently succeeded in proving the quite remarkable fact that the logistic function is Δ-discriminatory whenever Δ has nonempty interior [45].

Remark. The results on universal approximation capabilities of $\mathcal{N}_k(\psi)$ do not mean that a single hidden layer is sufficient for computational task ANN's could possibly perform. In fact, in engineering applications, it is very often necessary to approximately invert a noninvertible function f. As has recently been shown by [42], this cannot be accomplished by functions in $\mathcal{N}_k(\psi)$, no matter which ψ is actually used, but is easily done by a network with two layers of hidden units with Heaviside activation function.

2.2 Rates of Approximation

Of course, the above results on universal approximation capabilities on multilayer feedforward networks do not at all imply that these networks are particularly well-suited to issues of functional approximation, not does the fact that the theorems are valid for more or less any possible activation function ψ mean that all such ψ are suited equally well to these issues. What is really of interest, in particular to allow for comparison with competing function classes which possess universal approximation capabilities, is the question how "easily", quantified e.g. by the number of hidden units or the number of adjustable interconnection strengths, a certain accuracy of approximation can be achieved. Therefore, it is of fundamental importan-



ce to identify the *rates of approximation* which are being achieved by ANN's.

Unfortunately, this appears to be a quite challenging and difficult task. Although clearly any constructive proof of denseness of $\mathcal{N}_k(\psi)$ is some function space $\mathcal{F}$ (with respect to some closeness measure ρ) can also be used to produce upper bounds for the rates of approximation of elements of $\mathcal{F}$ by functions in $\mathcal{N}_k(\psi; n)$, these bounds usually turn out to be discouragingly large. In our opinion, this is due to the fact that thus far, all constructive proofs have actually been "artificial" in the sense that approximators to f are not constructed directly from f, but rather with the aid of another system of approximators to f.

Let us illustrate this point in more detail. As already mentioned, the method used in [21] for establishing denseness of $\mathcal{N}_k(\psi)$ consists in first approximating by functions in $\mathcal{N}_k(cos)$, i.e. by trigonometric functions, and then constructing approximators in $\mathcal{N}_1(psi)$ to the cosine function. (Their proofs are sometimes referred to as nonconstructive because they more or less artificially employ the Stone-Weierstrass theorem to establish denseness of $\mathcal{N}_k(cos)$. However, this can also be obtained directly by explicitly approximating by suitable trigonometric polynomials). [12] and [22] construct Riemann sum approximations to the Irie-Miyake integral representation, in which the sole purpose of the integration with respect to ω is to cancel out the original ψ terms and replace them by cosine terms, with the effect that the thus obtained rates of approximation are necessarily of worse order than those which can be obtained by directly discretizing the Fourier inversion formula (i.e., by initially using the cosine as activation function). A similar situation occurs if the Radon transform representation introduced in [7] is discretized as suggested in [18]. Constructive proofs based on network architectures with two hidden layers usually proceed by first providing approximators to indicators of cubes respectively intersections of hyperplanes, i.e. by approximately implementing hidden units with $\sum\prod$ activation rules, see e.g. [50] and [32].

Summing up, as all methods which we are currently aware of for constructing approximators by ANN's with sigmoidal, squashing, or integrable activation functions rely on the use of other approximating systems, such as ANN's which use the cosine as activation function or hidden units with $\sum\prod$ activation rules, it is clearly very natural to ask what the benefits of not immediately using these "other" systems might possibly be. It is by no means clear in which sense (if any) ANN's based on e.g. logistic or arctangent activation functions should be superior to "Fourier networks" based on the cosine as activation function; to us, the careful investigation of this question should be one of the most important research topics of the next few years. (Of course, the results of [44], [45] show that the cosine cannot provide approximations with uniformly bounded input-to-hidden weights which the logistic function can; however, it is totally unclear how "nice" such approximations are, and which rates they can achieve).

Quite surprisingly, it is sometimes possible to obtain "good" bounds for rates of approximation without explicitly constructing good approximators. [25] shows that if in a Hilbert space H, an element f of H is contained in the closure of the convex hull of a bounded subset G of H, then the minimal distance between f and G_n, the space of all convex combinations of at most n points in G, can be bounded as

$$\min_{f \in G_n} \| f - g \|_H \leq \gamma/\sqrt{n},$$

where $\| \cdot \|_H$ denotes the usual Hilbert space norm and $\gamma = \sup_{g \in G} \| g \|_H$ Therefore, in order to apply this result to neural network approximation, we arrive at the problem of determining whether a function can be approximated arbitrarily well by elements in the convex hull of a bounded subset of $\mathcal{N}_k(\psi)$, e.g. sets of the form $\{\beta\psi(a'x + \omega)\}$ with β bounded in absolute value. This question could even be investigated at a very general level similar to Cybenko's setting for proving existence of (unconstrained) ANN approximators, using the fact that a closed convex subset and a point not contained in it can be separated by a hyperplane; however, this general approach has thus far been more or less untractable.

Relying heavily on Fourier analysis, [5] has recently applied Jones's result to obtain the following theorem. Let $\mathcal{F}_c$ be the set of all continuous and integrable real-valued functions whose Fourier transform $\hat{f}$ satisfies $c_f := \int_{\mathcal{R}^k} |a\hat{f}(a)| \leq c$, and let ψ be an arbitrary sigmoidal function. Then for every $f \in \mathcal{F}_c$, every $r > 0$ and $n \geq 1$, every finite measure μ on $\mathcal{R}^k$ and every $\gamma > 2rc$, there exists a linear combination of the form

$$g_n(x) = \beta_0 + \sum_{j=1}^{n} \beta_j \psi(a_j' x + \omega_j)$$

such that

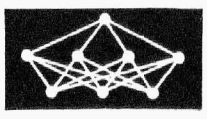
$$\int_{B_r} |f(x) - g_n(x)|^2 d\mu(x) \leq \frac{\gamma^2}{n} \mu(B_r),$$

where $B_r = \{x \in \mathcal{R}^k : |x| \leq r\}$. The coefficients in the approximating element g_n may actually be restricted to satisfy $\beta_0 = f(0)$ and $\sum_{j=1}^{n} |\beta_j| \leq 2rc$.

Barron's idea is to use the Fourier inversion theorem to obtain the representation

$$\begin{aligned} f(x) - f(0) &= \Re \int_{\mathcal{R}^k} (e^{ia'x} - 1) e^{i\omega(a)} |\hat{f}(a)| da \\ &= c_f \int_{\mathcal{R}^k} \frac{\cos(a'x + \omega(a)) - \cos(\omega(a))}{|a|} d\Lambda_f(a), \end{aligned}$$

where the phase $\omega(a)$, defined by $\hat{f}(a) = |\hat{f}(a)| e^{i\omega(a)}$ is a continuous function of a, and Λ_f is the probability measure on $\mathcal{R}^k$ with density $|a\hat{f}(a)|/c_f$. This shows that the mapping $x \longmapsto f(x) - f(0)$ is the limit of convex combinations of the functions $\beta g_{\kappa,\omega}(t)$, evaluated at $t = u'x$ with $|u| = 1$, where $|\beta| \leq c$ and for $\kappa \neq 0, g_{\kappa,\omega}(t) = \cos(\kappa t + \omega) - \cos(\omega))/\kappa$. By then cleverly constructing suitable functions in $\mathcal{N}_1(\psi)$ which approximate $g_{\kappa,\omega}$ uniformly over $[-r, r]$, the result can be established.



Clearly, the above method again is artificially constructive in the sense we have already explained, and again it is more than natural to ask why we should want to use some sigmoidal ψ rather than the cosine as activation function. In fact, for the latter choice, we could more directly write

$$\begin{aligned} f(x) &= \Re \int_{\mathcal{R}^k} e^{ia'x} e^{i\omega(a)} |\hat{f}(a)| da \\ &= \int_{\mathcal{R}^k} \cos(a'x + \omega(a)) |\hat{f}(a)| da \end{aligned}$$

to obtain the same $n^{-1/2}$ order of approximation for the much larger class of continuous and integrable functions with integrable Fourier transform, cf. [25].

Functional approximation by feedforward neural networks is very closely related to a statistical technique called Projection Pursuit (PP), see in particular the excellent survey paper of [23]. PP uses model functions of the form

$$\sum_{j=1}^{n} \phi_j(u_j' x),$$

where the u_j are unit vectors in $\mathcal{R}^k$ and the ϕ_j arbitrary functions on $\mathcal{R}$. Clearly, all elements of $\mathcal{N}_k(\psi)$ are of the above form, and in fact, $\mathcal{N}_k(\psi)$ can be thought of as a more or less conveniently parameterized basis for the class of PP model functions. PP was introduced by [11] in a sampling context as an attempt to cope with one of the problems usually encountered in the statistical analysis of (even only moderately) high-dimensional data sets, commonly referred to as the "curse of dimensionality": for regression techniques based on classical approximating systems, the number of parameters to be fitted explodes with the dimension of the problem, and the methods based on local smoothing (e.g. nearest neighbor methods) require very large samples, because otherwise, small neighborhoods of points of interest do not contain enough data points. Because they can reconstruct a function from its projections along directions of maximal variation, there is some hope that PP techniques could overcome these difficulties.

However, as in most cases best PP approximators cannot be written down explicitly, it has proved to be very hard to substantiate these expectations. An extensive comparison of kernel smoothing and PP approximation for $k = 2$ and the standard gaussian input environment measure has been carried out by [9].

By choosing the ridge functions ϕ_j to be of the form $\beta \cos(\kappa t + \omega)$ one can, as outlined above, obtain $L^2(\mu)$ approximation of order $n^{-1/2}$ with $O(nk)$ parameters for continuous and integrable functions with integrable Fourier transform [25]. Thus, for sufficiently regular functions f, the curse of dimensionality can in fact be avoided. However, the effect of allowing for more general ϕ_j, and in particular, how these should then be learned from samples, are far from being well-understood.

Let us summarize our considerations. Functional approximation by elements of $\mathcal{N}_k(\psi)$ can avoid the curse of dimensionality, at least for $L^2(\mu)$ distances. Nevertheless, the approximation performance of networks based on sigmoidal activation functions relative to Fourier networks still needs to be investigated thoroughly.

One possible approach might consist in finding "ψ-inversion theorems" of the form

$$f(x) = \int_{\mathcal{R}^k} \int_{\mathcal{R}} \psi(a'x + \omega) d\eta_f(a, \omega),$$

where η_f is some signed measure on $\mathcal{R}^{k+1}$ which can directly be computed from f. In the light of the above mentioned results of [25], η_f should really be a *finite* signed measure, unlike what is the case in the Irie-Miyake integral representation, and could probably be compactly supported. This would also allow to obtain results on rates of $L^2(\mu)$ approximation for gaussian input environment measures which have not been established thus far.

Another possibility could consist in providing representations

$$\sum_{j=1}^{n} \beta_j \psi(a_j' x + \omega_j) = \int_{\mathcal{R}^d} K_n(x, y) f(y) dy,$$

where the kernels K_n approximate the delta function as $n \to \infty$. This might in particular allow for improving the existing bounds for rates of uniform approximation.

3. Learning

3.1 On-line Learning Methods

As we have already mentioned, we feel that the increase in using ANN's in the context of functional learning may to a great extent be attributed to the fact that the BP algorithm seemed to provide the users with an easy-to-use method for network training, although its limitations were encountered soon, and a lot of additional features have meanwhile been suggested to improve on its performance.

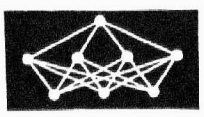
Many on-line learning algorithms for training the adjustable network weights θ (assuming for simplicity that the network architecture is kept fixed throughout learning) can compactly be written as

$$\theta_t = \theta_{t-1} + \eta_t Q(z_t, \theta_{t-1}),$$

where z_t is the training pattern presented at time t (in supervised learning, z_t consists of two components, an input x_t and a target y_t), η_t is the learning rate employed at time t, and $Q(.,.)$ is function characteristic of the algorithm; in the case of BP,

$$Q(z, \theta) = \nabla g(x, \theta)(y - g(x, \theta)),$$

where $g(x, \theta)$ is the output of a network with weight configuration θ from an input x, and $\nabla g(x, \theta)$ is the matrix of the partial derivatives of g with respect to θ.



In order to analyze the behavior of such algorithms, in particular for the case of large random training samples and sufficiently small learning rates, it is not too unreasonable to expect that one can, by averaging over all training patterns, relate the algorithm to the ordinary differential equation (ODE)

$$\dot{\theta} = Q(\theta), \quad Q(\theta) = EQ(z_t, \theta).$$

(The dot denotes the derivative with respect to time). In which sense such a replacement is possible, and which conclusions can be drawn concerning the on-line learning algorithm by analyzing its associated ODE, has been the subject of intensive research for the last few years. One proceeds by constructing so-called interpolated processes $\theta(s)$, $s \geq 0$, obtained by either piece wise linear or piecewise constant interpolation of the sequence of updates θ_t with interpolation intervals η_t, and then shows that the solution paths of the ODE are the limit points of the interpolated processes (with respect to uniform convergence on bounded time intervals). This can be done e.g. if the $\eta_t \to 0$ at a suitable rate and time is "large" (the so-called stochastic approximation case), see e.g. [46], [47], [29], or if $\eta_t \equiv \eta$ tends to zero (the case of constant, but very small learning rates), see [30].

Ideally, the asymptotic ODE should have one globally attractive equilibrium θ^*, which would then allow to infer that in some sense, the estimates generated by the algorithm are "close" to the optimal θ^*. However, contrary to classical learning algorithms from engineering applications (as the adaptive least squares algorithm and its extensions), this situation is typically not encountered in "new" neural network algorithms, for which a global asymptotic analysis of the associated ODE is extremely difficult; in particular, these ODE's are usually characterized by a multiplicity of equilibria, several of which are asymptotically stable.

In the case of BP, $Q(\theta) = -\nabla E(\theta$, where $E(\theta) = E|y_t - g(x_t, \theta)|^2/2$ is the (theoretical) error function to be minimized. As then clearly E is nonincreasing along the solution paths of the ODE, it is very often concluded that θ converges to the set of local minima of E, at least for random initial conditions with probability one. However, as has also been pointed out in [30], such a conclusion cannot be drawn unless we know that the level sets of the error function are compact subsets of the weight space, which is usually not the case in applications. (Of course, in order to prevent the estimates from diverging, one could confine them to some compactum, e.g. by truncating the entries of θ if they appear too large. But this clearly generates new artificial equilibria in at least parts of the boundary of the set which one confines to). This situation is particularly annoying in situations where it is known that no local minima exist, as in supervised learning in linear networks with a bottleneck layer [1]. Similarly annoying difficulties can occur in situations where the ODE is not a gradient system, but it is known that the desired limit points of the algorithm are the only asymptotically stable equilibria of the ODE, as is the case for a very general class of unsupervised feature extraction algorithms, see [20] and [30].

We feel that as the existing general results on "ODE methods", cf. e.g. [36], [35], [37], and [34], were really developed for use in classical engineering applications where the underlying dynamical system has a unique globally attractive equilibrium, they are thus not really well-suited to systems which do not have this property. Therefore, a rigorous and satisfactory analysis of the properties of some typical neural network on-line learning algorithms only be possible if the existing theory is substantially deepened. Currently, results on actual implementations of the algorithms usually out perform the results of their theoretical analysis. This situation is particularly inconvenient because, as explained in the introduction, what we should really be interested in is not so much the fact that the estimates generated by the algorithms converge to desired optimal weight configurations as the number T of training patterns tends to infinity, but rather at which rate this convergence occurs, i.e. how large T has to be to achieve a certain degree of accuracy; such results clearly cannot be obtained if convergence itself cannot be established.

3.2 Off-line Learning Methods

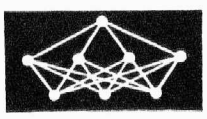
It is trivial that algorithms which only try to optimally adjust the weights of some fixed network architecture, e.g. by minimizing mean square error, cannot learn arbitrary mappings. Such a task can only be accomplished if the network architecture e.g. the number n of hidden units, may also be adjusted and increased without bounds.

In general, such methods can be thought of as trying to minimize some criterion function $Q(g)$ over a suitable collection $\mathcal{G}$ of functions. Usually, $Q(g)$ is a power of the distance between g and another function f, relative to some closeness measure. For example, an error function $Q(g) = E|y - g(x)|^p$ equals the $p-th$ power of the $L^p(\mu)$ distance between g and a suitable function f of the conditional distribution of y given x, see e.g. [2] and White [48] (as usual, we write μ for the distribution of x); in particular, for the most prominent choices of $p = 2$ and $p = 1$, f equals the conditional expectation of y given x and the conditional median of y given x, respectively.

However, in applications, $Q(g)$ itself is usually not explicitly known, because the joint distribution of x and y is unknown, and instead we have to use an estimate $Q_T(g)$ of $Q(g)$ obtained from a sample x_t, y_t, $t = 1, \ldots, T$. In the above example, we could e.g. take the empirical moment

$$Q_T(g) = T^{-1} \sum_{t=1}^{T} |y_t - g(x_t)|^p.$$

Clearly, unconstrained minimization of Q_T over $\mathcal{G}$ would now drastically overfit the sample and not allow for sufficiently good generalization on previously unseen patterns (networks with enough hidden units can exactly interpolate the training sample). Therefore, one has to make sure that the function g_T actually learned is sufficiently "regu-



lar", which can be achieved either by nonparametrically smoothing natural interpolatory estimates, by restricting the minimization to sufficiently "small" bounded subsets $\mathcal{G}_T$ of $\mathcal{G}$, or more generally, by trading off goodness-of-sample-fit achieved by g, as measured by $Q_T(g)$, to the complexity of g, measured by a suitable function $C_T(g)$, and actually obtaining an estimate $\hat{g}_T$ by minimizing the penalized criterion function

$$Q_T(g) + \lambda C_T(g),$$

where $\lambda > 0$ is a control parameter which weights the importance of in-sample-fit to complexity; restricting g to $\mathcal{G}_T$ can be described by letting $C_T(g) = 0$ if g is in $\mathcal{G}_T$ and ∞ otherwise. Notice that the same idea is employed for model selection in nested linear models.

Assume for simplicity that the problem of minimizing $Q(g)$ over the closure of $\mathcal{G}$ has a unique solution g^*; in our cases of interest where $Q(g) = E|y-g(x)|^p = (\rho_{p,\mu}(f,g))^p$ and $\mathcal{G} = \mathcal{N}_k(\psi)$, this is always the case with $g^* = f$, as by the results of section 2, the closure of $\mathcal{N}_k(\psi)$ (with respect to $\rho_{p,\mu}$) equals $L^p(\mu)$. In these cases, rates of convergence of $\hat{g}_T$ to g^* can be obtained at a very general level, by combining the errors incurred from approximating g^* by functions of certain complexity, and the error resulting from estimating these approximators from the sample. Such general results with learning error bounds given in terms of the "index of resolvability" have e.g. been given in [2], cf. also [6].

Combining these general methods with the results on (non-sample) rates of $L^2(\mu)$ approximation as discussed in section 2.2, [4] has recently given improved rates of convergence for neural network functional learning using complexity regularization. He shows that for sigmoidal ψ, if $n \approx (T/k/log(T))^{1/2}$, $\hat{g}_{T,n}$, is a function in $\mathcal{N}_k(\psi;n)$ learned from the sample by minimizing $Q_T(g) = T^{-1}\sum_{t=1}^{T}|y_t - g(x_t)|^2$ over a suitable subset of $\mathcal{N}_k(\psi;n)$, and if the underlying f lies in $\mathcal{F}_c$ for some c, then the expected mean square error can be bounded according to

$$E\int_{B_r}|f(x)-\hat{g}_{T,n}(x)|^2 d\mu(x) \leq const(f,c,r)\sqrt{\frac{k\log(T)}{T}}.$$

The same rate of convergence can be obtained by choosing n through some order selection procedure. Again, one might conjecture that a similar, if not better result, in particular for a larger class of functions f, could be obtained by training Fourier networks. This issue certainly deserves further investigation.

Methods which explicitly restrict the minimization to small bounded subsets of $\mathcal{G}$ are usually called "sieve methods". [49] has established that if $\mathcal{G}_T$ equals $\mathcal{N}_k(\psi;n_T,M_T)$, the collection of all functions in $\mathcal{N}_k(\psi;n_T)$ for which all weights are less than M_T in absolute value, and both n_T and M_T tend to infinity at a suitable rate as $T \to \infty$, then the $L^2(\mu)$ distance between the underlying f in $L^2(\mu)$ and the function $\hat{g}_T$ learned from a sample of size T by minimizing $Q_T(g) = T^{-1}\sum_{t=1}^{T}|y_t - g(x_t)|^2$ over $\mathcal{N}_k(\psi;n_T,M_T)$ tends to zero as $T \to \infty$. [14] establish similar results for Sobolev-space convergence, i.e. with respect to smooth closeness measures, provided that f is sufficiently smooth. Rates of convergence have not been identified thus far; the recently obtained results on rates of (non-sample) approximation should prove to be most valuable for this purpose.

References

[1] Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: learning from examples without local minima. **Neural networks, 2**, 53-58.

[2] Barron, A.R. (1989). Statistical properties of artificial neural networks. In **Proceedings of the 28th Conference on Decision and Control.** New York: IEEE.

[3] Barron, A.R., & Barron, R.L. (1989). Statistical learning networks: a unifying view. In E.Wegman (Ed.), **Computing Science and Statistics: Proceedings of the 20th Symposium on the Interface** (pp.192-203). Washington, DC: American Statistical Association.

[4] Barron, A.R. (1991a). Complexity regularization with applications to artificial neural networks. To appear in: G. Roussas (Ed.), **Proc. NATO ASI on Nonparametric Functional Estimation**. Dordrecht, The Netherlands: Kluwer Academic Publishers.

[5] Barron, A.R. (1991b). **Universal approximation bounds for superpositions of a sigmoidal function.** Technical Report # 58, Department of Statistics, University of Illinois at Urbana-Champaign.

[6] Barron,A.R. (1991c). **Approximation and estimation bounds for artificial neural networks**. Technical Report # 59, Department of Statistics, University of Illinois at Urbana-Champaign.

[7] Carroll,S.M., & Dickinson,B.W. (1989). Construction of neural nets using the Radon transform. In **Proceedings of the 1989 International Joint Conference on Neural Networks** (pp. I:607-611). San Diego: SOS Printing.

[8] Cybenko,G. (1989). Approximation by superposition of a sigmoidal function. **Mathematics of Control, Signals and Systems, 2,** 303-314.

[9] Donoho,D.I., & Johnstone,I.M. (1989). Projection-based smoothing, and a duality with kernel methods. **Annals of Statistics,17**, 58-106.

[10] Farmer,J.D., & Sidorovich,J.J. (1988). **Exploiting chaos to predict to future and reduce noise.** Technical Report LA-UR-88-901. Los Alamos, NM: Los Alamos National Laboratory.

[11] Friedman,J.H. & Stuetzle,W. (1981). Projection pursuit regression. **Journal of the American Statistical Association, 76**, 817-823.

[12] Funahashi,K (1989). On the approximate realization of continuous mappings by neural networks. **Neural Networks, 2**, 183-192.

[13] Gallant,A.R., & White,H. (1988). There exists a neural network that does not make avoidable mistakes. In IEEE **Second International Conference on Neural Networks** (pp. I: 657-664). San Diego: SOS Printing.

[14] Gallant,A.R., & White,H. (1989). **On learning derivatives of an unknown mapping with multilayer feedforward networks.** Discussion Paper 89-53, Department of Economics, University of California, San Diego.

[15] Girosi,F., & Poggio,T. (1989). Representation properties of networks: Kolmogorov's theorem is irrelevant. **Neural Computation,1**, 465-469.

[16] Hecht-Nielsen,R. (1987). Kolmogorov mapping neural network existence theorem. In IEEE **First International Conference on Neural Networks** (pp.III: 11-13). San Diego: SOS Printing.

[17] Hecht-Nielsen,R. (1989). Theory of the back propagation neural network. In **Proceeding of the 1989 International Joint Conference on Neural Networks** (pp. I:593-606). San Diego: SOS Printing.

[18] Hornik,K. (1990). Smooth approximation capabilities of multilayer perceptrons. In **Proceedings of the International Symposium on Neural Networks and Neural Computing Neuronet'90**. Prague: SAV.

[19] Hornik,K. (1991). Approximation capabilities of multilayer feedforward networks. **Neural Networks**, forthcoming.

[20] Hornik,K., & Kuan,C.-M. (1990). **Convergence analysis of local feature extraction algorithms.** BEBR Discussion Paper 90-1717, College of Commerce, University of Illinois, Urbana-Champaign.

[21] Hornik,K, Stinchcombe,M., & White,H. (1989). Multilayer feedforward networks are universal approximators. **Neural Networks,2**, 359-366.

[22] Hornik,K., Stinchcombe,M. & White,H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. **Neural Networks,3**, 551-560.

[23] Huber,P.J. (1985). Projection pursuit. **Annals of Statistics,13**, 435-475.

[24] Irie,B., & Miyake,S. (1988). Capabilities of three layer perceptrons. In IEEE **Second International Conference on Neural Networks** (pp.I:641-648). San Diego: SOS Printing.

[25] Jones,K.L. (1991). A simple lemma on greedy approximation in Hilbert space an convergence rates for projection pursuit regression and neural network training. **Annals of Statistics**, forthcoming.

[26] Jordan,M. (1989). Generic constraints on underspecified target trajectories. In **Proceedings of the 1988 IEEE International Conference on Neural Networks** (pp.I:641-648). New York: IEEE Press.

[27] Kolmogorov,A.N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. **Doklady Academii Nauka USSR,114**, 953-956. **American Mathematical Society Translations,28**, 55-59 [1963].

[28] Kuan,C.M. (1989). **Estimation of neural network models.** Ph.D. thesis, Department of Economics, University of California, San Diego.

[29] Kuan,C.M., & White,H. (1990). **Recursive M-estimation, nonlinear regression and neural network learning with dependent observations.** BEBR Working Paper 90-1703, College of Commerce, University of Illinois, Urbana-Champaign.

[30] Kuan,C.M., & Hornik,K. (1991). Convergence of learning algorithms with constant learning rates. To appear in IEEE **Transactions of Neural Networks.**

[31] Krkov,V. (1990a). 13th Hilbert's problem and neural networks. In **Proceedings of the International Symposium of Neural Networks and Neural Computing Neuronet '90** (pp.214-216). Prague: SAV.

[32] Krkov,V. (1990b). **Kolmogorov's theorem and multilayer neural networks.** Preprint, Institute of Computer Science, Czechoslovak Academy of Sciences.

[33] Krkov,V. (1990c). **Kolmogorov's theorem is relevant**. Preprint, Institute of Computer Science, Czechoslovak Academy of Sciences.

[34] Kushner,H.J. (1984). **Approximation and weak convergence methods for random processes.** Cambridge, MA: MIT Press.

[35] Kushner,H.J., & Clark,D.S. (1978). **Stochastic approximation methods for constrained and unconstrained systems.** New York: Springer Verlag.

[36] Ljung,L. (1977). Analysis of recursive stochastic algorithms. **IEEE Transactions on Automatic Control, AC-22**, 551-575.

[37] Ljung,L.,& Soenderstroem,T. (1983). **Theory and practice of recursive identification**. Cambridge, MA: MIT Press.

[38] Lorentz,G. (1962). Metric entropy, width, and superposition of functions. **American Math. Monthly.69**, 469-485.

[39] McCulloch,W.S., & Pitts,W. (1943). A logical calculus of the idea immanent in nervous activity. **Bulletin of Mathematical Biophysics,5**, 115-133.

[40] Minsky,M., & Papert,S. (1969). **Perceptrons.** Cambridge, MA: MIT Press.

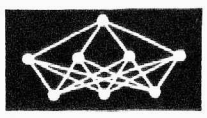
[41] Rumelhart,D.E., Hinton,G.E., & Williams,R.J. (1986). Learning internal representations by error propagation. In D.E.Rumelhart, J.L.McClelland, & the PDP Research Group, **Parallel distributed processing: Explorations in the microstructures of cognition** (Cambridge, MA: MIT Press.

[42] Sontag,E.D. (1990). **Feedback stabilization using two-hidden-layer nets.** report SYCON-90-11, Rutgers Center for Systems and Control, Rutgers University, New Brunswick, NJ.

[43] Sprecher,D.A. (1965). On the structure of continuous unctions of several variables. **Transactions of the American Mathematical Society,115**, 340-355.

[44] Stinchcombe,M. & White,H. (1990). Approximating and learning unknown mappings using multilayer feedforward networks with bounded weights. In **Proceedings of the 1990 International Joint Conference on Neural Networks** (pp.III: 7-16). New York: IEEE Press.

[45] Stinchcombe,M., & White,H. (1991). **Using feedforward networks to distinguish multivariate populations.** Preprint, Department of Economics, University of California, San Diego.



[46] White,H. (1987).Some asymptotic results for back-propagation. In **IEEE First International Conference in Neural Networks** (pp.III: 261-166). San Diego: SOS Printing.

[47] White,H. (1989a). Some asymptotic results for learning in single hidden-layer feedforward network models. **Journal of the American Statistical Association,84**, 1003-1013.

[48] White,H. (1989b). Learning in neural networks: A statistical perspective. **Neural Computation,1**, 425-464.

[49] White,. H. (1990). Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. **Neural Networks,3**, 535-549.

[50] Williamson,R.C., & Paice,A.D.B. (1990). **The number of nodes required in a feed-forward neural network for functional representation**. Preprint, Department of Systems Engineering, Australian National University, Canberra, Australia.

Literature Survey

The literature on neuroscience increases last few years extremely fast. At present some estimations of more than 20000 existing papers, conference and symposium talks, books and research reports are made. Evidently it is not possible to inform the readers about all the interesting publications, which currently appear. However, we would like to use the existence of the computer oriented Scientific Information System of the Institute of Computer and Information Science in Prague for to present here almost regularly the short survey of the last year records of this base.

Of course, the readers are asked for to be so kind and inform the Editors or the Institute about any publication, which they recommend to insert in this literature survey.

Alexandre F.,Guyot F.,Haton J.P.,Burnod Y.: The Cortical Column: A New Processing Unit for Multilayer Networks
Neural Networks Vol.4, 1991 No.1 pp.15-26

Abstract: We propose in this paper a new connectionist unit that matches a biological model of the cortical column.

Alspector J., Gannett J.W., Haber S., Parker M.B., Chu R.: A VLSI-Efficient Technique for Generating Multiple Uncorrelated Noise Sources and Its Application to Stochactic Neural Networks
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.1 pp. 109-123

Abstract: Here we present a method for generating multiple arbitrarily shifted pseudorandom bit streams from a single LFSR. Each bit stream is obtained by tapping the outputs of selected LFSR cells and feeding these tapped cell outputs trough a set od exclusive-OR gates. This enables many neurons to share a single LFSR, resulting in an acceptably small overhead for VLSI implementation.

Avitabile G., Forti M., Manetti S., Marini M.: On a Class of Nonsymmetrical Neural Networks with Applications to ADC
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.2 pp. 202-209

Abstract: A class of neural networks with nonsymmetrical connections is considered. The analysis of the dynamical behavior shows that this type of neural network is characterized by a unique equilibrium point whose attraction domain is the entire space. Such properties do not depend on the shape of neuron nonlinearities.

Barnard E., Casasent D.: Invariance and Neural Nets
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.498-508

Key words: neural networks; nets; invariance.

Abstract: Invariance with respect to certain transformations is one of the main tasks of pattern-recognition systems. We study various techniques for obtaining this invariance with neural net classifiers and identify the invariant-feature technique as the most suitable for current neural classifiers. A new formulation of invariance in terms of constraints on the feature values leads to a general method for transforming any given feature space so that it becomes invariant to specified transformations.

Barnard E., Cole R.A., Vea M.P., Alleva F.A.: Pitch Detection with a Neural-Net Classifier
IEEE Transactions on Signal Processing Vol.39, 1991 No.2 pp.298-307

Abstract: The extens of generalization attainable with neural nets is first examined, and it is shown that a suitable choice of features is required to utilize this property.

Baum E.B.: Neural Net Algorithms That Learn in Polynomial Time from Examples and Queries
IEEE Transactions on Neural Networks Vol.2, 1991 No.1 pp.5-19

Key words: neural networks.

Abstract: Back propagation and many other neural net algorithms use a data base of labeled examples to train neural networks.Here examples are pairs $(x, t(x))$, where x is an input vector and $t(x)$ is the value of the target function for input x. We propose an algorithm which trains networks using examples and queries.

PAB: PARAMETERS ADAPTING BACK-PROPAGATION

O. Kufudaki [1], *J. Hořejš* [2]

Abstract:

A new method for back-propagation is suggested. It uses the parameterized transfer sigmoid function $S(\xi, \lambda, \sigma) = \sigma/(1 + exp(-\lambda, \xi))$ where ξ is the net income, λ the gain and σ a scaling factor. Each neuron has its own λ and σ. Besides the weights, λ and σ are also automatically adapted by the steepest descent method. The algorithm performs considerably better than standard BP both in avoiding (apparent) local minima and in convergence speed. It was experimentally compared to SuperSAB [6] with encouraging results. Moreover, because the basic ideas of the two approaches are independent, they can be combined to achieve even better performance. Estimations for components of the gradient of error function E, $\{\partial E/\partial w, \partial E/\partial \lambda, \partial E/\partial \sigma\}$, are given and confronted with experiments. Because σ can in a sense theoretically substitute weights, consequences for the "weightless" neural networks problem are commented on. In the introduction, historical motivation for the idea and possible explanation of distinguished properties of PAB are given; sect. 1 and 2 thus can be omitted by those interested in BP applications only. On the other hand we draw the reader's attention there to a combinatorial geometry result, which may be of interest for the further theory of multilayered networks in general.

Key words: speed-up of BP, local minima avoidance, parameterized transfer functions, weightless neural networks, the generalization problem

1. Historical motivation

We first met the back-propagation (BP) about 4 years ago. After succeeding with well-known examples like the symmetry problem, we tried to realize the identity mapping to transfer the vertices of the 4-dimensional cube over the $4-2-4$ net, being prepared for the instance that due to the sigmoid transfer function of the usual form $[1/(1+exp(-i))]$,the 0/1 vectors will not be mapped accurately; but it did not work at all. Only then did we notice the number of regions, dissected in a $m-k-n$ net by n k-dimensional hyperplanes, and derived (see [3]) the formula

$$\sum_{i=0}^{k} \binom{n}{i} \tag{1}$$

limiting the number of possible different boolean vectors passing through the net, especially when the identity mapping is to be realized (i.e. if $m = n$ and $\mathbf{y} = \mathbf{x}$ (see Fig.1 for the case $m = n = 4, k = 2$)). This formula, now often ascribed to [1], is however just a special case of the known equality from combinatorial geometry [2] (treated also in [5]):

$$\sum_{i=0}^{d} \binom{n}{k-i} \binom{k-i}{d-i} \tag{2}$$

giving (in general) the number of d-dimensional convex subspaces arising from the dissection of a k-dimensional space by n k-dimensional hyperplanes.

If we denote by $\mathbf{z} = [z_1, \ldots z_k]$ the point in k-dimensional space in the "upper part" of the net (if input is the bottom and output the top one), corresponding to an input pattern $\mathbf{x} = [x_1, \ldots, x_m]$ in the lower part, then potentials (outputs before taking the final nonlinear transformation) of all output neurons y_i ($\mathbf{y} = [y_1, \ldots, y_n]$) are scaled distances from corresponding hyperplanes, these being tangential to hyperspheres with center $\mathbf{z}$ and y_i corresponding to the radius of the hypersphere. Because of nonlinear transforms in the hidden layer, the scales on the k axis may be mutually different (and varying in time as will be described in section 3), and it is better to speak about k-dimensional quadrics, which we call here hyperspheroids. This geometrical view enables us to see why more hidden neurons make an easier solution of the mapping problem: it is surely more probable to find (case $k = 3$) a common tangential plane to three randomly placed spheres than to find (case $k = 2$) a common tangential

[1] Dr.Olga Kufudaki
Institute of Information and Computer Science, Czechoslovak Academy of Sciences, 182 07 Prague 8, Pod vodárenskou věží 2, Czechoslovakia
[2] Prof.Dr.Jiří Hořejš
Department of Computer Science, Charles University, 118 00 Prague 1, Malostranské nám. 25, Czechoslovakia

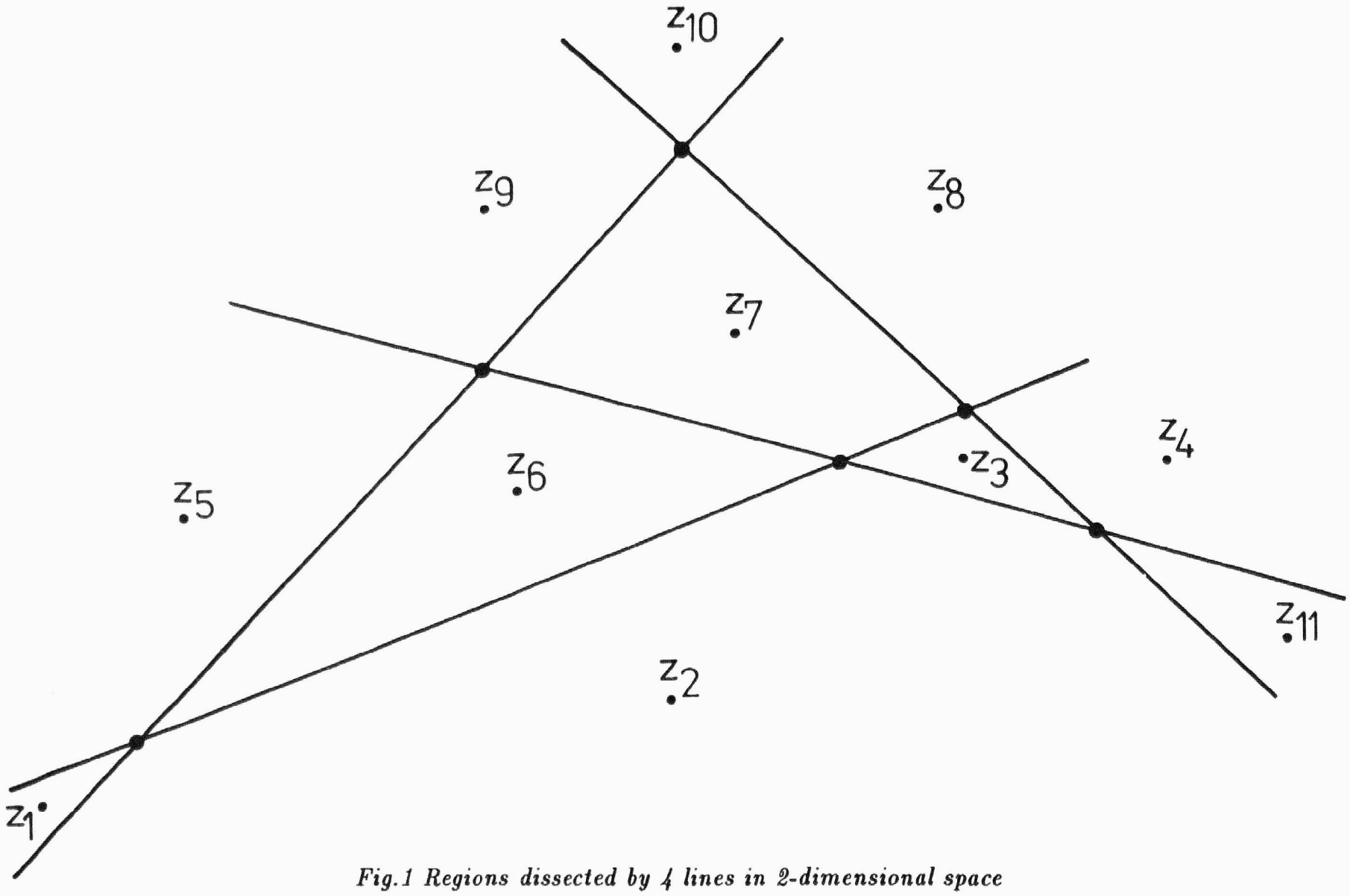


Fig.1 Regions dissected by 4 lines in 2-dimensional space

line to three generally positioned circles with radii prescribed by expected outputs. For the simplest (classification-like) tasks we want that outputs of all output neurons tend to 0 or 1, and all points z coming from the lower parts when a particular pattern from the training set has been presented on the input, to be as far from the separating hyperplanes as possible (corresponding hyperspheroids radii being "infinite"). Thus the outputs of the mentioned points should have the largest possible value to be far away from the 0 on the first axis of the transfer function S (thus tending to $\mp$ infinity). It follows that in such cases we can consider only full dimensional convex subspaces and set $d = k$ which shows how many different 0/1 correct output vector responses we can maximally expect. For a net $* - 2 - 4$ we get the number 11.

It is seen that transfering all 16 of the vertices of a 4-dimensional cube over the net $4 - 2 - 4$ is an impossible task. If you try to do it, some of the 16 inputs necessarily lead to the same output. Actually, there exist even such choices of the 11 vectors, for which the problem is unsolvable (using decadic codes, the set $\{1, 2, 3, 4, 5, 7, 8, 9, 10, 12, 15\}$ is an example). Formula (2) gives only an upper bound. Believing, that the solution for this randomly chosen example exists and blaming an (apparent) local minimum for unsuccessful convergence , also had its positive effect; it stimulated further investigation of BP, see [12].

By the way, from (2) we can also derive some limitations on the appearance and number of input vectors mapped to identical outputs if we consider the three-valued "logical" functions over $\{0, 0.5, 1\}$: output of a top-layer neuron equals 0.5 if and only if z lies on the hyperplane; thus for the case of Fig.1 we can transfer vectors with no more than 6 pairs of values 0.5; for higher dimensions of the uppermost hidden layer you can similarly count the upper bound for the number of vectors having two or three 0.5's and so on. All these considerations are not of course restricted to the given examples, but should attract researchers to potentials of combinatorial geometry in the study of NNs.

2. Orientation of hyperplanes

Moreover, which 11 out-of 16 mentioned vectors will be successfully transmitted over the net also depends on the orientation of the hyperplanes. If, for a given input pattern, z comes into the position depicted in Fig.2, where the numbers labeling hyperplanes correspond to particular output neurons, the resulting output vector will tend towards 1010. To get another output, say 1110, one of these 3 possibilities should happen:

(i) in further computation, z will cross one of the hyperplanes, in this case that of number 2; if it stays finally in the positive halfspace with respect to (w.r.t.) hyperplane

No 2 (of III), it will always produce 1 in the position of the second output neuron. If it is the hyperplane, which moves (to the dashed line, I), other results may become spoiled.

If this happens at the beginning of computation and the 11 $z's$ are nearby, the situation may still be saved by a big migration of z-points and hyperplanes. If most of them are however sufficiently far from the hyperplanes (irrespective of whether from the very beginning or near the end of the adaptation process), then troubles may arise, because some of the outputs are near zero or one and, as is well known,

$$\Delta w \approx y.(1-y) \tag{3}$$

('$\approx$' means ' is proportional to') so that the adaptation ability of the upper part of the net decreases, and its lower part should try to remedy the situation;

(ii) some of the hyperplanes revert their orientation II; to do it within the mechanism of traditional BP, you can either diminish the normal vector to zero an then let it increase in the opposite direction or turn the whole hyperplane 180 degrees; because the first possibility leads to proportional change of all of k their coordinates, both possibilities are improbable. Moreover as the distance of z from the hyperplane increases and corresponding outputs tend towards 0 or 1, (3) again intervenes so that drastic changes in the normal vectors are hardly to be expected. Taking finally into account the fact that

(iii) the mutual switch of one of more pairs across the hyperplane in the upper half of the net requires an analogical change of orientation of a hyperplane in the lower half of the net, we see that the quickest solution of the problem would be to revert the hyperplane orientation by some new mechanism. We have suggested such a mechanism: let the gain λ (originally constantly 1) of the transfer function change during the computation; if λ changes its sign, the orientation of the corresponding hyperplane automatically reverts. This idea, based on a calculation of $\Delta\lambda$ similar to the calculation of Δw in standard BP, then evolved in the GAB *[Gain Adapting BP]* with a better chance of avoiding unpleasant configurations, otherwise often leading to getting stuck. It should however be emphasized that GAB mostly solves the problem by somehow envisaging these situations even before they occur, so that actually changing the sign of λ is not always necessary.

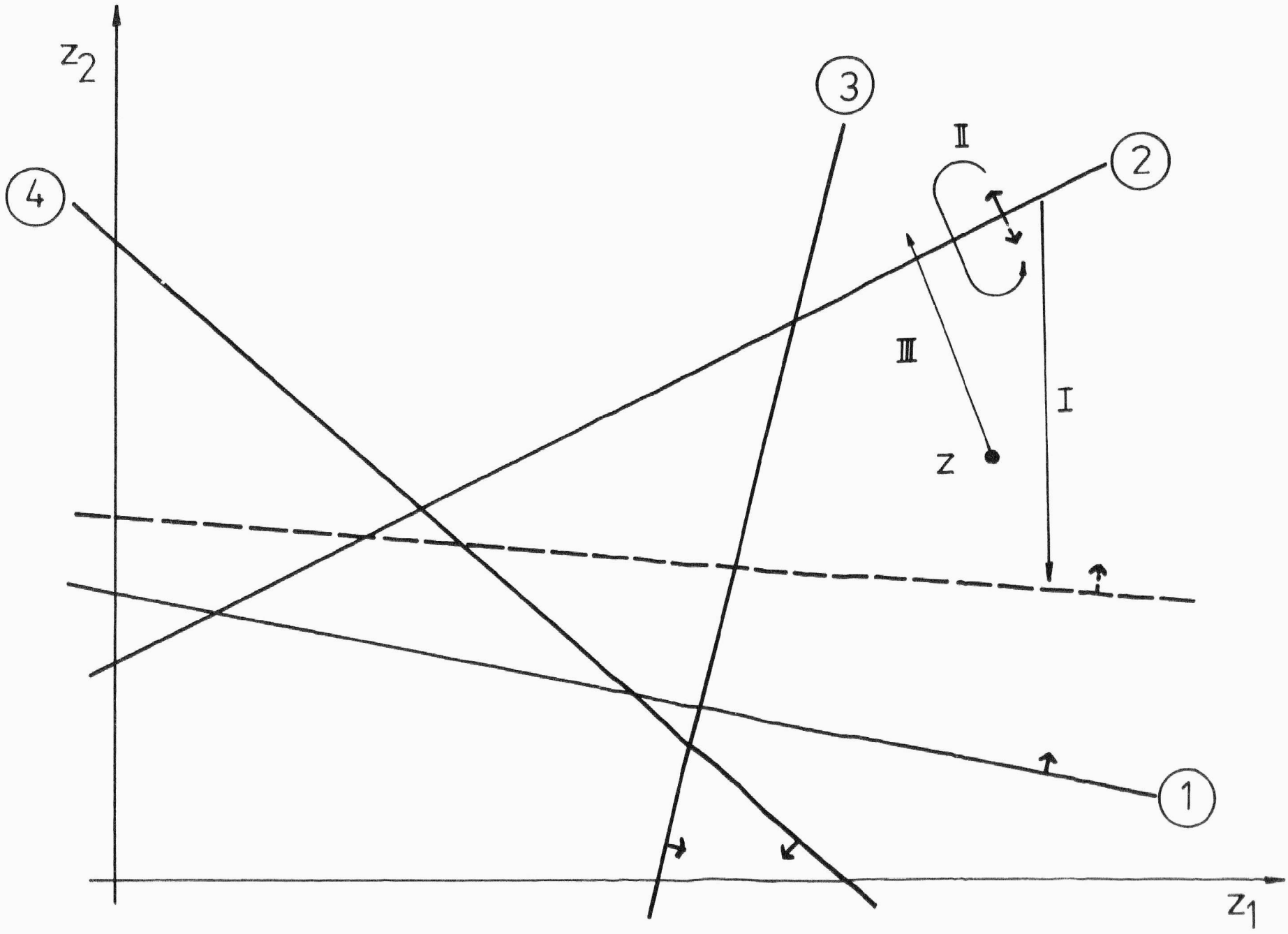


Fig.2 Possible rearrangement of hyperplanes and points

3. The idea of PAB

GAB uses an unhomogeneous net in which every neuron has its own time-dependent λ specifying the gain of S. Let us now generalize the idea of GAB still a bit more, taking the whole parameterized family of transfer functions, e.g.

$$S(\xi, \lambda, \sigma) = \sigma.(1/(1 + exp(-\lambda\xi)) \tag{4}$$

where λ gives the slope of the sigmoid at point 0 (if we have $\lambda = 0$, we get just a straight line (constant 0.5), while λ tending to infinity derives from S the hard nonlinearity used in perceptrons) and specifies a scale in the vertical coordinate. In the thus-obtained PAB *[Parameters Adapting Back-propagation]*, the mapping performed by the net depends not only on the weights (including thresholds), but also on all the λ's and σ's.

It is important that the individual parameters can also be adapted automatically, interleaving in a suitable way the usual adaptation of w according to formulas, given by the gradient

$$grad\ E = \{\partial E/\partial\sigma, \partial E/\partial\lambda, \partial E/\partial w\} \tag{5}$$

Note that $S(\xi, \lambda, \sigma)/\sigma = S(\xi, \lambda, 1)$ and $S(\lambda\xi, 1, \sigma) = S(\xi, \lambda, \sigma)$.

We shall now compute the gradients for the net $m-k-n$, and distinguish again the upper part of the net (from layer **1** to output layer **2**) and the lower part (from the input layer **0** to first (and the only) hidden layer **1**). However, the recursive part of the computations remains valid for any number of hidden units, except that indexing would be more complicated. Note that we took into account the fact that outputs of **z**'s (= outputs of neurons in layer **1**) are composite functions of σ, λ and w. The effect on the reverting hyperplanes' orientation occurs whenever λ or σ passes 0 from positive to negative value or vice versa.

Using the LMS criterion for global error given by the well known formula $E = 1/2\ .\sum\sum(y-d)^2$, where **y** is the actual output and **d** the desired one, the inner sum taken over all output neurons and the outer one taken over all members of the training set, we derive for a particular value of σ, λ and w (considering the others as constants in a given sweep) the following (the input vector **x** is submitted in layer **0**, ξ are potentials of corresponding neurons before the S is applied):

For the upper part we have:

$$\partial E/\partial\sigma_{\mathbf{2},i} = \delta_{\mathbf{2},i}.y_{\mathbf{2},i}/\sigma_{\mathbf{2},i} \qquad (i = 1, 2, \ldots, n) \tag{6}$$

where $\delta_{\mathbf{2},i} = y_{\mathbf{2},i} - d_i$ (difference between actual and desired value of output of neuron i; note there is no danger of possible division by zero of the value of σ, because y is of the form (4) and this divided by σ simply gives $S(\xi, \lambda, 1)$);

$$\partial E/\partial\lambda_{\mathbf{2},i} = \delta_{\mathbf{2},i}.y_{\mathbf{2},i}.(1 - y_{\mathbf{2},i}/\sigma_{\mathbf{2},i}).\xi_{\mathbf{2},i} \tag{7}$$

where $\xi_{\mathbf{2},i}$ is the inner potential of neuron i;

$$\partial E/\partial w_{\mathbf{1},j,i} = \delta_{\mathbf{2},i}.y_{\mathbf{2},i}.(1 - y_{\mathbf{2},i}/\sigma_{\mathbf{2},i}).\lambda_{\mathbf{2},i}.z_{\mathbf{1},j} \tag{8}$$

where $w_{\mathbf{1},j,i}$ is the weight coming from $j-th$ neuron of **1** to $i-th$ neuron of **2** and $z_{\mathbf{1},j}$ is the output of $j-th$ neuron of **1**.

Finally, for connections between **1** and **2** we have - in the role of $\delta_{\mathbf{1},j}$:

$$\partial E/\partial z_{\mathbf{1},j} = \sum \delta_{\mathbf{2},i}.w_{\mathbf{1},j,i}.y_{\mathbf{2},i}.(1 - y_{\mathbf{2},i}) \tag{9}$$

with the sum taken over $i = 1, \ldots, n$ in the layer **2**, j is a selected neuron in **1** and for the rest of lower part:

$$\partial z_{\mathbf{1},j}/\partial\sigma_{\mathbf{1},j} = z_{\mathbf{1},j}/\sigma_{\mathbf{1},j} \tag{10}$$

$$\partial z_{\mathbf{1},j}/\partial\lambda_{\mathbf{1},j} = z_{\mathbf{1},j}.(1 - z_{\mathbf{1},j}/\sigma_{\mathbf{1},j}).\xi_{\mathbf{1},j} \tag{11}$$

$$\partial z_{\mathbf{1},j}/\partial w_{\mathbf{0},p,j} = z_{\mathbf{1},j}.(1 - z_{\mathbf{1},j}/\sigma_{\mathbf{1},j}).\lambda_{\mathbf{1},j}.x_{\mathbf{0},p} \tag{12}$$

with $j = 1, \ldots, k$, $p = 0, \ldots, m$. For fictive neurons simulating thresholds (z_0 and x_0) we assume the constant value -1.

Having the partial derivatives, we can set as usually,

$$\begin{aligned} \Delta\sigma &\simeq -\pi.\partial E/\partial\sigma \\ \Delta\lambda &\simeq -\mu.\partial E/\partial\lambda \quad \text{for some } \pi, \mu, \eta \\ \Delta w &\simeq -\eta.\partial E/\partial w \end{aligned} \tag{13}$$

or in more detail:

$$\Delta\sigma_{\mathbf{2},i} \simeq -\pi.\delta_{\mathbf{2},i}.y_{\mathbf{2},i}/\sigma_{\mathbf{2},1} \qquad i = 1, \ldots, n$$

$$\Delta\sigma_{\mathbf{1},j} \simeq -\pi.\partial E/\partial z_{\mathbf{1},j}.\partial z_{\mathbf{1},j}/\partial\sigma_{\mathbf{1},j} = -\pi.\delta_{\mathbf{1},j}.z_{\mathbf{1},j}/\sigma_{\mathbf{1},j} \qquad j = 1, \ldots, k$$

$$\Delta\lambda_{\mathbf{2},i} \simeq -\mu.\delta_{\mathbf{2},i}.y_{\mathbf{2},i}.(1 - y_{\mathbf{2},i}/\sigma_{\mathbf{2},i}).\xi_{\mathbf{2},i} \qquad i = 1, \ldots, n$$

$$\Delta\lambda_{\mathbf{1},j} \simeq -\mu.\partial E/\partial z_{\mathbf{1},j}.\partial z_{\mathbf{1},j}/\partial\lambda_{\mathbf{1},j} = -\mu.\delta_{\mathbf{1},j}.z_{\mathbf{1},j}.(1 - z_{\mathbf{1},j}/\sigma_{\mathbf{1},j}).\xi_{\mathbf{1},j} \qquad j = 1, \ldots, k \tag{13'}$$

$$\Delta w_{\mathbf{1},j,i} \simeq -\eta.\delta_{\mathbf{2},i}.y_{\mathbf{2},i}.(1 - y_{\mathbf{2},i}/\sigma_{\mathbf{2},i}).\lambda_{\mathbf{2},i}.z_{\mathbf{1},j} \qquad i = 1, \ldots, n \quad j = 1, \ldots, k$$

$$\Delta w_{\mathbf{0},p,j} \simeq -\eta.\delta_{\mathbf{1},j}.z_{\mathbf{1},j}.(1 - z_{\mathbf{1},j}/\sigma_{\mathbf{1},j}).\lambda_{\mathbf{1},j}.x_{\mathbf{0},p} \qquad j = 1, \ldots, k \quad p = 0, \ldots, m$$

possibly including moreover the momentum terms (with coefficients $\pi 2$, $\mu 2$ and α, respectively).

Let us note that the derived relations hold not only for the family of sigmoid functions, but can be adapted for any differentiable parametric transfer function $F =$

$\int\int f(\mathbf{t},\chi)dtd\chi$, where $\mathbf{t}$ is a vector of parameters and $\chi = \xi(\mathbf{w},\mathbf{x})$ is a function giving the presynaptical activity of the neuron, while $x = F(\chi)$ is its postsynaptical activity. Then for the upper part of the net,

$$\begin{aligned}\Delta t_i \approx -\partial E/\partial t_i &= \partial E/\partial F.\partial F/\partial t_i = \delta_i.\int f d\chi,\\ \Delta w \approx -\partial E/\partial w &= \partial E/\partial F.\partial F/\partial \chi.\partial \chi/\partial w\\ &= \delta_i.\int f dt.\partial \chi/\partial w\end{aligned}$$

and similar formulas hold for the lower part (cf.[7]).

4. Estimation of gradients

To estimate values of the PAB parameters gradients, the following ratios are needed:

$$\Delta\sigma/\Delta w^*,\ \Delta\lambda/\Delta w^*,\ \Delta w/\Delta w^*,$$

where Δw^* is the change of w_{ji} in the net with constant values of $\sigma = \lambda = 1$. Similarly, the asterisk $*$ here always refers to values in standard BP computations.

From (13'), it follows that in all parts of the parametric net, for equally valued δ_i and λ_i :

$$a)\ |\Delta\sigma_i/\Delta w_{ji}| = y_i/(\sigma_i.\kappa_i^*.z_j^*) = y_i(\lambda)/\kappa_i^*.z_j^* \qquad (14)$$

where $\kappa_i^* = y_i^*.(1-y_i^*)$ and $y_i(\lambda) = y_i/\sigma_i$.

Because y_i^*, $y_i(\lambda)$ as well as z_j^* all are within [0,1], we see that $\kappa_i^* \in [0,0.25]$ with maximum in $y_i^* = 0.5$ and hence also $\kappa_i^*.z_j^* \in [0,0.25]$. Now let us see when (14) is grater than 1:

$$|\Delta\sigma_i| > |\Delta w_j^*| \text{ if and only if } y_i(\lambda) > \kappa_i^*.z_j^*$$

If $y(\lambda) \in [0.25,1]$ then the right-side inequality holds, because $\kappa_i^*.z_j^* \leq 0.25$. If however $y_i(\lambda) < 0.25$, then for corresponding y_i^* we have $\kappa_i^* < 0.19$ and thus left-side inequality is always true, irrespective of values of σ_i and $y_i(\lambda)$. It follows than $\partial E/\partial\sigma > \partial E/\partial w^*$.

$$b)\ |\Delta\lambda/\Delta w^*| = |\xi_i.\sigma_i|.z_j^* \qquad (15)$$

In this case

$$|\Delta\lambda| > |\Delta w^*| \text{ if and only if } |\xi_i.\sigma_i| > z_j^*.$$

Because $|\xi_i.\sigma_i| > 1$ and usually $|\xi_i| \gg 1$ as well as $|\xi_i| \geq w_{ij}.z_j^*$, it most frequently holds $\partial E/\partial\lambda > \partial E/\partial w^*$.

$$c)\ |\Delta w/\Delta w^*| = |\lambda_i\sigma_i|.z_i/z_j^* = |\lambda_i\sigma_i\sigma_j|.z_j(\lambda)/z_j^* \qquad (16)$$

In this case

$$|\Delta w| > |\Delta w^*| \text{ if and only if } |\lambda_i\sigma_i\sigma_j|.z_j(\lambda) > z_j^*$$

which holds at least for every $|\lambda_i\sigma_j| > 1$. This corresponds to the fact that nets with harder nonlinearity usually converge faster, if at all (i.e. they may have stability problems). Using PAB, the $\lambda's$ can reach high values continuously without causing problems so big that the adaptive process can't stabilize.

From (14), (15) and (16) we see why the gradients of PAB are often much higher then these of standard BP.

Assuming $\sigma \in (-s,s)$, $\lambda \in (-k,k)$ and $\xi \in (-r,r)$, for some numbers s,r,k, then from the easily seen facts $y^* = (0,1), z^* \in (0,1)$, $\kappa^* \in [0,0.25]$ and for $y^* = 0.5$ (when $\max\delta_i = 0.5$) we deduce the following restrictions on the values of gradients (for $\eta = \pi = \mu = 1$):

$$\begin{aligned}\Delta w^* &\in (-0.125,0.125)\\ \Delta\sigma &\in (-s,s)\\ \Delta\lambda &\in (-s^2r,s^2r)\\ \Delta w &\in (-s^2k,s^2k),\end{aligned} \qquad (17)$$

where usually $r \gg k > s$.

It is seen that most dramatic changes during adaptation should be expected in λ, and w, in this order, which is in accordance with our experimental findings. Because the maximal value of σ usually exceeds 1, its gradient can be according to (17) in order of magnitude greater than Δw^*, which is most conservative.

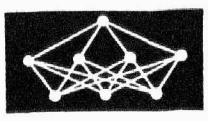
These estimations give us a hint at how to choose the initial parameter values and ratios of learning and momentum rates. For the σ, the computations are especially sensitive if you approach some upper limit, after which the net immediately overflows); this can be explained by the fact that according to last formula of (17), Δw and $\Delta\lambda$ depend on σ quadratically.

5. The assessment of PAB

The automatic and incessant modification of parameters may perform not only hyperplane orientation reversal (which then sometimes manifests itself as a little "jump" from a positive to a negative value or vice versa), often observed in our experiments, especially for λ ([8] confirmed this effect), but finds under such a "threat", an often more elegant way how handle the situation. Generally, we found many times that it succeeded when the traditional form of BP failed or in other cases speeded up the convergence considerably.

Note that due to the arbitrary values of σ and λ, the range of values of output neurons is theoretically unlimited, preserving at the same time the top layer's nonlinearity.

Also, it is clear that there are other possibilities for further extension of the parameterization technique, like shifting the range of (4) by an additive or multiplicative number, either a constant or a value depending on σ. From the geometrical point of view you can imagine that under parameterization, the hyperspheroids become more "plastic" and flexible.



Finally note one important feature of σ modification: it is not dependent on the factor mentioned in (3).

It was shown by extensive use over a span of 3 years that the special form of PAB, namely GAB, is a more powerful than standard BP in two respects: First, it will not get stuck in poor (apparent) local minima in many cases where the traditional BP does and, second, it is, as a rule, faster. The idea of GAB was implemented, applied, graphically demonstrated and systematically pursued by E.Pelikan [4], especially in his investigations and construction of a system for neural signal analysis NESP. Sima [8] and Fatton [12] also confirm the effect. PAB inherited and made more apparent these properties, having one more degree of freedom. No wonder then that it is yet better in many tasks, like those mentioned above (11 out of 16 0/1 vectors). Its efficiency can be seen partially from the obligatory XOR test. In Table 1, the Global Error (further denoted as GE) and $\sum$, which is the sum of absolute differences over all patterns, are shown after 2000 and 5000 iterations. The first row gives the results of the traditional BP together with the choice of learning step η and momentum coefficient α. In the third row, automatic adaptation of λ has been included (μ as in (13), $\mu 2$ momentum coefficient) and in the fifth row, adaptation of σ was added (with coefficients π and $\pi 2$). Final values of λ and σ for the topmost neuron are shown as well. During the computation they changed considerably.

The parameterized BP often behaves in a different way than the "normal" one does. The GE curve may several times abruptly increase to relatively high values, then fall down below the level reached before the peak. These oscillations are interleaved with longer periods of slow decrease. The rapid decrease of GE is observed namely in times of these "abruptions" (this might be compared to a sort of self-annealing, or relaxing after enough frustration leads to a revolting move) and is usually signalized by strange behavior of the parameters. Also, PAB is rather sensitive to the initial configuration setting. The values given in the odd lines of the table correspond to the same random choice of weights, with all $\lambda's$ and $\sigma's$ having the initial value 1. Changing both these values to 1.1 (even lines in the table), the "abruption" effect occurs and the results improve. This is seen in Fig.3, where the GE graph for the case $\sigma = \lambda = 1.1$ is shown. The curves "Lambda" and "Sigma" represent values of one selected neuron.

Setting initially all the $\sigma's$ to 1.5, while retaining previous values of initial $\lambda's$, the $\lambda's$ (!! - cf. discussion of (17)) increase enormously and the convergence process stabilizes at $GE \cong 2.0$. Note that the effect of initial values persists even if the parameters do not change at all (row 2 in the table, where the standard BP is used).

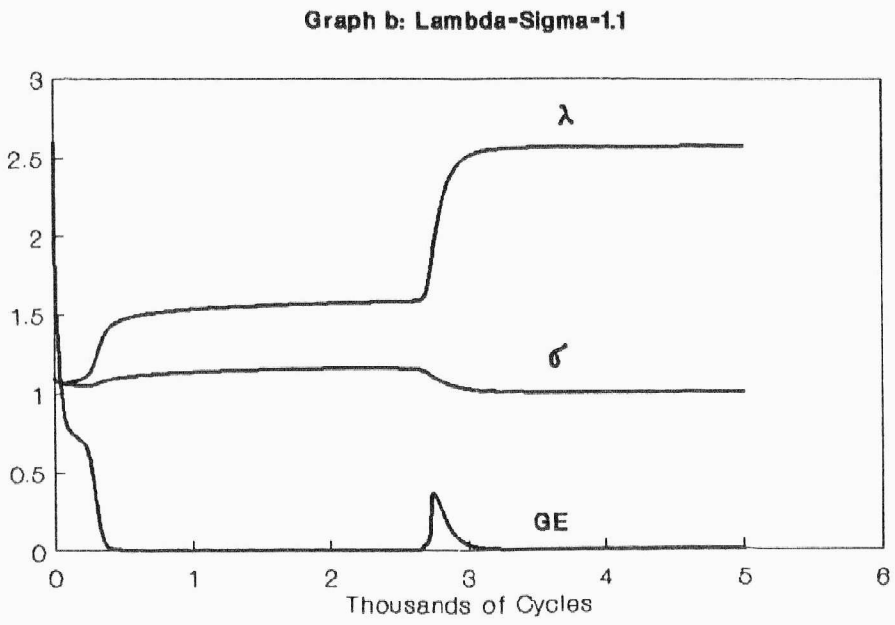


Fig.3 Dependency of GE, Lambda and Sigma on No of cycles

We have compared the results of PAB with those of SuperSAB with self-adapting learning rates η ([6]) on the motivational task of identity mapping of various quadruplets of 0/1 inputs, where the danger of the convergence process is often encountered in the form of relatively good decrease of global error till reaching values slightly above 1 or 2 (which usually means that the mapping behaves well except that it insists on one or two pattern/output neuron pairs producing complementary answers). As a rule, SuperSAB reached (under the same randomly generated initial weights) the critical values near 1 or 2 more quickly, but then started to remain in one of its two modes (increasing/decreasing the learning rates η) for a long period, sometimes practically decreasing η to 0. Then further decrease of GE stopped. PAB approached the critical region more slowly, but often crossed it and continued with decrasing GE.

From these and other comparisons we deduced that SuperSAB is pretty quick when moving over an error landscape with relatively shallow valleys and low hills; PAB seems better when there are deep holes around (with narrow paths out to escape apparent local minima). This gave us few ideas for further research. First, to consistently join

	2000 cycles				5000 cycles			
Lear.rate/moment	GE	$\sum$	λ	σ	GE	$\sum$	λ	σ
$\eta = 0.7\ \alpha = 0.3$	0.0044	0.13	1	1	0.00125	0.07	1	1
dtto	0.001	0.09	1.1	1.1	0.006	0.05	1.1	1.1
$\mu = 0.03\ \mu 2 = 0.02$	0.0012	0.07	1.6	1	0.00025	0.03	1.7	1
dtto	0.003	0.03	1.6	1.2	0.0001	0.02		1.1
$\pi = 0.0025\ \pi 2 = 0.0002$	0.0002	0.03	1.5	1.2	0.00006	0.01	1.6	1
dtto	0.0002	0.03	1.6	1.1	0.00001	0.008	2.6	1

Tab. 1

these two techniques. In fact we already used a mixture of both, but only η was adapted during the computation in some examples. The possible objection to PAB, namely that of introducing other parameters to handle, weakens with the introduction of the SuperSAB technique.

Second, we want to use the idea of SuperSAB and other weapons against the growth of individual (pattern / output neuron) pairs of errors, fighting most intensively with the (always changing) worst pair. In other words, we would like to develop a "Uniform BP", in which the errors in all outputs (in all output neurons and for all patterns) will always be approximately the same. This may be another way tackling overfitting, which is otherwise often of selective nature: for some pattern/output neuron pairs results can be very accurate, while a few exceptions can still produce significant global error.

Remark. During his stay at Charles University, D.Fatton has contributed to a partial solution of these tasks; a note on his work is submitted [12].

Note also that if λ or $\sigma \cong 0$ for a relatively long time, the corresponding neuron is a candidate for the removal.

No matter what Table 1 indicates, we believe that the major practical contribution of PAB is not its speed, but its ability to adapt more easily to inhospitable landscapes.

Another phenomenon of PAB deserves attention. What seems at first sight surprising is that we were able to solve some problems only by adapting the parameters λ and σ, without any modification of weights ($\eta = \alpha = 0$)! (e.g. a few 0/1 vectors from the introductory problem were transmitted over the (same!) net with constant mutually equal weights with a good accuracy comparable with those given in Table 1 for traditional BP).

Thus we claim as a hypothesis that with the adaptive parameters λ and σ every neural net is equivalent to a net with fixed mutually equal weights.

If you take into account that any weight can be simulated by an interneuron "sitting" on the connection, with arbitrary output caused by proper increase of σ and/or changing its sign by converting the orientation of λ, the claim need not be interpreted as a direct objection to connectionism (where adaptive weights play the crucial role), while still supporting the idea of weightless neurocomputing [11]. It is one way of avoiding direct weight modification. Although this would generally cause an increase of the number of neurons (afterwards perhaps being minimized by an analogy to various now often discussed techniques), some implementation aspects could give it some sense, e.g. by avoiding 3-dimensional arrays of weights (which on some computers lead to poor memory utilization).

6. Comments on generalization

There are plenty of modifications of BP, accelerating convergence, usually by giving the net more degrees of freedom (more parameters for tuning it, or adding hidden neurons, say). They are potentially dangerous from the point of view of generalization, as they admit more internal representations. This caution should also be taken into account when assessing the speed-up techniques and PAB is in this respect no exception.

On the other hand this does not mean that generalization of such nets is necessarily worse; it depends on many factors including task, data and others. In [10], the authors performed studies on the influence of various choices of σ and λ (they use the same transfer function (4) as we do, their parameters however are not adaptive during computation) and found out that their proper selection improves greatly generalization, relative to their task.

For dynamically changing parameters you could reach similar results under the assumption that you use satisfactorily converged set frozen in a more less stable configuration, not during an oscillatory phase.

Also a Uniform BP might contribute to good generalization by facing the danger of unjustified preference of some hidden laws before the others, all being biased by the same degree of fuzziness.

Finally, because PAB is (at present at least allegedly) able to eliminate weights, emphasizing the principle of locality where each neuron can express its own "weight" (by the value of σ) and its own decisive strength (by the value of λ), it can also in a perspective contribute to the generalization problem along the lines indicated above.

This research is further developed in [13].

References

[1] G.Mirchandani and Wei Cao: On Hidden Nodes in Neural Networks, IEEE Trans. Circ. Syst., vol.36, No5, 661-664.

[2] H.Edelsbrunner: Algorithms in Combinatorial Geometry, Springer Verlag 1987.

[3] J.Horejs, O.Kufudaki: A study on NNs, Res.Rep. Dept Comp.Sci. Charles University 1989 [in Czech].

[4] E.Pelikan: A model of multilayered NN with adaptive transfer function, Res.Rep. Comp. Center CSAV 1989 [in Czech].

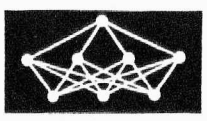
[5] J.Horejs, O.Kufudaki: A Geometrical Model of Layered NNs, Proc. Int. Symp. on NNs, Prague, Sept.1990, 122-124, extended version in: Theoretical Aspects of Neurocomputing, World Scientific Co, London 1991, pp.195-212.

[6] T.Tolleanaere: SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties, Neural Networks, vol.3 (1990), 561-573.

[7] K.Hornik: Functional Approximation and Learning in Artificial Neural Networks, Neural Network World, (this issue).

[8] J.Sima: Neural Nets as Expert Systems, Diploma thesis, Charles University Prague, 1991.

[9] D.E.Rumelhart, G.E.Hinton, R.J.Williams: Learning representations by back-propagating errors, Nature, vol.323, Oct.1986, 533-536.



[10] J.B.Hampshire, A.H.Weibel: A Novel Objective Function for Improved Phoneme Recognition Using Time Delay Neural Networks, Proc. IJCNN 89, Washington, I- 235-241.

[11] I.Alexander: Connectionism or weightless programming?, in Artificial NN, T.Kohonen et al Eds., North Holland 1991, 991-1000.

[12] D.Fatton: On the Uniform Training (submitted to NNW).

[13] O.Kufudaki, J.Horejs: Weightless and Threshold-controlled Neural Networks, submitted to 11th European Meeting on Cybernetics and System Research.

Literature Survey

Burrascano P.: A Norm Selection Criterion for the Generalized Delta Rule
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp. 125-130

Key words: training algorithm.

Abstract: The derivation of a supervised training algorithm for a neural networks implies the selection of a norm criterion which gives a suitable global measure of the particular distribution of errors.

Cherkassky V., Lari-Najafi H.: Constrained Topological Mapping for Nonparametric Regression Analysis
Neural Networks Vol.4, 1991 No.1 pp.27-40

Key words: constrained topological mapping; curse of dimensionality; nonparametric regression; self-organization.

Abstract: The idea of using Kohonen's self-organizing maps is applied to the problem of nonparametric regression analysis, that is, evaluation (approximation) of the unknown function of N-1 variables given a number of data points (possibly corrupted by random noise) in N-dimensional input space.

Chua L.O., Tichonicky I.: 1-D Map for the Double Scroll Family
IEEE Trans. on Circuits and Systems Vol.38, 1991 No.3 pp.233-243

Abstract: We use the 1-D map pi an approximation of the 2-D Poincare map to study the periodic windows of the Double Scroll family. First, using an algorithm based on the kneading theory, we determine the structure and the order of appearance of periodic orbits in the 1-D map pi.With this information, we then find the structure and the period of the corresponding orbits of the 3-D system.

Cilingiroglu U.: A Purely Capacitive Synaptic Matrix for Fixed-Weight Neural Networks
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.2 pp.210-217

Abstract: It is shown that the synaptic function of fixed-weight neural networks can be implemented with one capacitor only. The resulting synaptic matrix, being devoid of active devices, offers not only very high space-power efficiency and speed but also large synapse capacity with considerable analog depth.

Dembo A., Farotimi O., Kailath T.: High-Order Absolutely Stable Neural Networks
IEEE Transactions on Circuits and Systems Vol. 38, 1991 No.1 pp.57-65

Abstract: The stability properties of arbitrary order continuous-time dynamic neural networks are studied in the spirit of an earlier analysis of a first-order system by Cohen and Grossberg.

Drago P.G., Ridella S.: An Optimum Weights Initialization from Improving Scaling Relationships in BP Learning
"Artificial Neural Networks" North-Holland, Elsevier Sci Publ.B.V.(Kohonen,Maekisara,Simula..), 1991, 4 p.

Key words: learning; neural nets.

Abstract: An algorithm for fast minimum search is proposed, which reaches very satisfying performances by making both the learning rate and the momentum term adaptive in an optimum way, and by executing controls and corrections both on the possible cost function increase and on eventual moves opposite to the direction of the negative of the gradient.

Guo H., Gelfand S.B.: Analysis of Gradient Descent Learning Algorithms for Multilayer Feedforward Neural Networks
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.8 pp. 883-894, ISSN 0098-4094

Key words: neural networks; feedforward.

Abstract: We investigate certain dynamical properties of gradient-type learning algorithms as they apply to multilayer feedforward neural networks. These properties are more related to the multilayer structure of the net than to the particular threshold units at the nodes. The analysis explains the empirical observation that the weight sequence generated by backpropagation and related stochastic gradient algorithms exhibits a long term dependence on the initial choice of weights, and also a continued growth and/or drift even long after the outputs have converged.

Huang S.Ch., Huang Y.F.: Bounds on the Number of Hidden Neurons in Multilayer Perceptrons
IEEE Transactions on Neural Networks Vol.2, 1991 No.1 pp.47-55

Key words: perceptrons.

Abstract: This paper investigates some fundamental issues concerning the capability of multilayer perceptrons with one hidden layer.

NEURAL NETS AS HETEROASSOCIATIVE MEMORIES

M.Herrmann, H.Englisch[1]

1. Introduction

The mathematical theory of neural networks is based mainly on the notion of a formal neuron introduced by McCulloch and Pitts [1], also known as a threshold logic unit (TLU) [2], which is able to perform a weighted sum over a set of inputs, to modify this sum nonlinearly, and to send the result to other neurons of the net. Some of the abilities of the brain such as recognition, learning or decision processes can be modelled as collective phenomena of an artificial neural network, i.e. a number of connected formal neurons.

Here we consider the network mainly as a memory device. Whereas classical memories return the content of a memory unit according to some code number or address which has to be put in exactly, networks can serve as content addressable memories, i.e. they return the content of the memory from incomplete or vague, "noisy" inputs. In the autoassociative case, the network produces an improved version of the input. An autoassociative memory appears as a special case of a heteroassociative one, where the prototypes of the input and the corresponding output pattern, respectively, need not to be identical. This can be interpreted as finding the answer to a posed question.

The perceptron [3] which will be considered in the next section presents itself as a suggestive solution for a heteroassociative memory. By means of a counterexample we will illustrate that the perceptron cannot solve every matching problem. Though examples of this kind can be dealt with by using multilayred perceptrons [4] or committee networks [2], it is useful to take the ideas of the Hopfield model [5] into consideration (section 3), which we consider subject to the condition that regions of different retrieval quality are given. In this case, an improvement in capacity (in comparison to the original Hopfield model) is obtained. The last sections are devoted to a special case of our model, namely to bidirectional networks as Hopfield type heteroassociative memories. As the output of such a network the state after infinitely many time steps is considered. Due to the relation between bidirectional and unidirectional (autoassociative, Hopfield models), some interesting aspects of the convergence arise, which will be considered in section 5. Further, in section 6 we consider the strongly diluted approximation of the bidirectional model in order to determine optimal values for the external field.

[1] M.Herrmann, H.Englisch
Sektion Informatik, Universität Leipzig, O-7010 Leipzig, FRG

2. The Perceptron

The perceptron [3] consists in simplified form of two layers. At the first layer of N_0 neurons an input configuration $\mathbf{S}(0) = (S_i(0), \ldots, S_{N_0}(0))$, where $S_i(0) \in \{-1, 1\}$, is given. Each of the N_1 neurons of the second layer is connected with every neuron of the first and appears as the output of a distinct TLU with identical input vectors $\mathbf{S}(0)$:

$$S_i(1) = sign\Big(\sum_{j=1}^{N_0} J_{ij} S_j(0) + h_i\Big), \quad i = 1, \ldots, N_1, \qquad (1)$$

where J_{ij} is the weight of the connection between the $j-th$ neuron of first layer and the $i-th$ neuron of the second one. By h_i, pixel dependent thresholds are included. For $N_0 = N_1$ the perceptron may be used as an autoassociative memory, for $N_0 \neq N_1$ as a heteroassociative memory only, and in particular for $N_1 = 1$ the perceptron (or simply the TLU) provides binary decisions for a "situation" represented by $\mathbf{S}(0)$. We remark that recognition as well as decision can be understood as a matter of classification, where there are in the former case as many classes as distinct memory contents and in the latter case as many classes as different decisions, and in particular for $N_1 = 1$ there are two classes. The weights of the connections J_{ij} will be adjusted in order to obtain a certain relation between input and output of the network.

The function of the perceptron is easily elucidated if we think of any possible input as a vertex of the hypercube $[-1, 1]^{N_0}$ in an N_0-dimensional space, which is called the pattern space (for $N_0 = 3$ see fig.1).

Each TLU corresponds to a hyperplane (orthogonal to the weight vector $\mathbf{J}_i = (J_{i1}, \ldots, J_{iN_1})$) , which divides the set of vertices into two parts. Depending on which side of the plane the input vectors is situated a different output $S_i(1) \in \{-1, 1\}$ is obtained.

The parity function usually serves as an example of where the perceptron fails. It requires output values $+1$ or -1 if the number of "-1" in the input is even or odd, respectively. For $N_0 = 2$ this corresponds to the XOR function (cf.[6]), where it is obvious that one cannot separate two

diagonally opposite vertices of a square from the other ones by only one straight line.

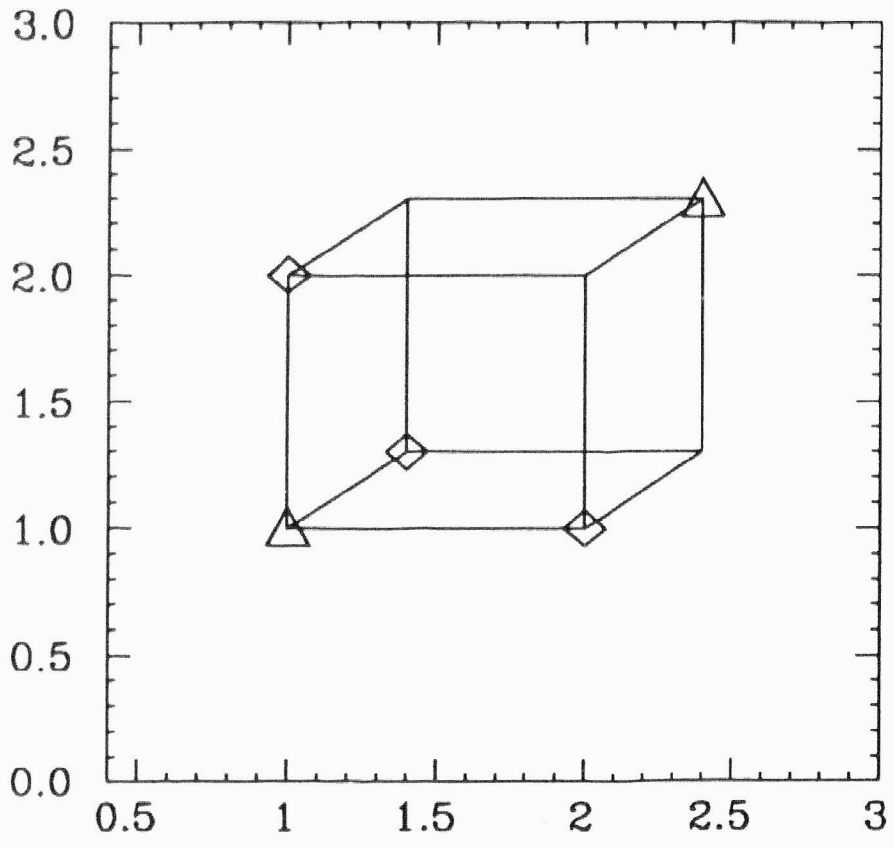


Fig.1.: Three-dimensional pattern space

Moreover, the portion of binary functions or classifications defined for all elements of $\{-1,1\}^{N_0}$ which are representable by a perceptron vanishes for large N_0. If, however,only $p = \alpha N_0$ randomly chosen vertices are considered, then for $\alpha < 2$ [7] the perceptron will learn to classify them correctly with a probability close to 1. For latter result [7], the assumption was made that the points in the space of the input patterns are in general position. This means that for each $K < N_0$ there is no hyperplane of dimension K which contains $K + 2$ or more input points. Only recently it was shown [8] that randomly chosen vertices are in general position with probability 1 for any finite α an N tending to infinity, such that $\alpha_{crit} = 2$ is the critical value also for binary patterns.

As an illustration we will consider the example from fig.1. The maximal number of binary patterns which can be arranged in general position at a cube is 5 and there is (up to rotations) only one such configuration (represented in fig.1). Obviously, it is impossible to find exactly one plane separating the points (-1,-1,-1) and (1,1,1,) (i.e. the Δ's in fig.1.) from the points (-1,-1,1), (-1,1,-1), (1,-1,-1) (the $\Diamond$'s in fig.1). This is the only one of the 16 possible decompositions of these five points into two sets which cannot be performed by a hyperplane.

Difficulties of this kind can be solved by using hidden units [4].

Another approach (or an interesting special case of networks with hidden units), the committee network, is also based on the idea of the pattern space.

Instead of relying on only one TLU it seems useful to form a committee [2] of such units. Their outputs are summed by a special vote-taking TLU that (put out) the majority decision of the committee.

The example of fig. 1. can be solved with a committee of two TLU's. Each TLU can be chosen in such a way that the corresponding plane cuts off one Δ-vertex and is oriented in such a way that this vertex gets the value $+1$. Thus, the Δ-vertices get one vote, therefore an output $+1$ is obtained, whereas the other patterns ($\Diamond$) are situated on the negative sides of both of the planes. For fairness, the inputs of the third vote taking TLU are connected to the outputs of the other two by equal strengths. Thus Δ's lead to $(+1)+(-1)=0$ and $\Diamond$'s to $(-1)+(-1)=-2$. A constant threshold $h = 1$, however, leads to an unambiguous decision of the committee.

3. Neural Nets with Regions of Different Retrieval Quality

Consider the autoassociative case for the perceptron, i.e. $N_0 = N_1 = M$ (rather than N which denotes the total number of neurons throughout the paper). The improvement of a noisy version $\mathbf{S}(0)$ of a vectors ξ^μ which has been stored in memory previously while running the network is measured by the retrieval quality defined as:

$$m^\mu(t) = \frac{1}{M}\sum_{i=1}^{M} \xi_i^\mu S_i(t) \qquad (2)$$

where $t \leq 1$. For independent ξ^μ and $\mathbf{S}(t), m^\mu(t)$ will be close to zero. If both are identical then $m^\mu(t) = 1$.

If an improvement of the input vector $m^\mu(1) \geq m^\mu(0)$, is reached by application of equation (1), it is tempting to iterate this rule several times.

$$S_i(t+1) = sign\Big(\sum_{j=1}^{M} J_{ij} S_j(t) + h_i(t)\Big); \qquad i = 1,\ldots,M; \quad t \geq 0. \quad (1')$$

In the Hopfield model [5], the weight J_{ij} are determined by the Hebb rule

$$\begin{aligned} J_{ij} &= \frac{1}{M}\sum_{\mu}^{p} \xi_i^\mu \xi_j^\mu, \quad i \neq j \\ J_{ii} &\geq 0, \end{aligned} \qquad (3)$$

where $\xi^\mu = (\xi^m u_1,\ldots,\xi_M^\mu);\ \mu = 1,\ldots,p$, are the vectors to be stored. The $\xi^m u_i$ are assumed to be independent and identical distributed random variables with zero mean.

Temporarily we will apply (1') only in a serial mode, i.e. for randomly or according to a certain rule chosen index i $S_i(t+1)$ is calculated.

If the argument of the sign function is equal to zero and for any other neuron $S_k, k \neq i$ the state remains unaltered.

Besides, the network can be simplified by using only one layer of neurons whose values change according to (1').

If the iteration procedure (1') converges for $t \to \infty$ (cf. below) then the limit state $\mathbf{S}(\infty)$ is said to be the output of the network.

Convergence, i.e. the existence of a definite limit state, depends essentially on the weight J_{ii} for the value of the $i-th$ neuron at the time before, i.e. on the stability of the state of the neuron, and on the symmetry of the connection matrix which is guaranteed by (3). It is obvious that for large J_{ii} the self-interaction term in (1') dominates the other ones, whereas for a negative self-interaction the opposite sign is prefered at time $t+1$. It turned out that $J_{ii} \geq 0$ (as in eq. (3)) is sufficient for the convergence of the dynamics (1') in the serial mode [9].

It has been shown [10] that a vector with $m(0) < 0.97$ is improved by iteration of (1') if the ratio α of the number p of stored vectors and the number M of neurons does not exceed a critical value which is close to 0.14. It should be remarked that a temporary improvement occurs also for higher α but this does not effect the retrieval quality of $\mathbf{S}(\infty)$ [11].

In the following paragraphs, we consider the input vector being composed of several regions or non-overlapping receptive fields each containing a different amount of information on the vector ξ^μ which is to be retrieved. The additional knowledge given in this way will us enable to focus attention on the important parts by a suitable weighting of the input vector $\mathbf{S}(0)$.

This situation is reminiscent to the one striking some parts of the surroundings with higher, but other parts with lower, accuracy.

The different reliability of different regions is expressed by the partial retrieval quality (4), i.e. overlap with one of the stored items within the corresponding part.

Let the input vector $\mathbf{S}(0)$ be a noisy version of the stored item ξ^μ. It splits into n parts of length N_k, where $k = 0, \ldots, n-1$. The $k-th$ part is situated between the neurons with number $A_{k-1}+1$ and A_k, $k = 0, \ldots, n-1$. A_k is given by $\sum_{l=0}^{k} N_k = N.$, hence $A_{n-1} = \sum_{k=0}^{n-1} N_k = N$. Further, we set $A_{-1} = 0$. The partial retrieval quality in the $k-th$ region is defined as

$$m_k^\mu(t) = \frac{1}{N_k} \sum_{i=A_{k-1}+1}^{A_k} \xi_i^\mu S_i(t). \tag{4}$$

Then the total retrieval quality (2) is a weighted sum of the partial retrieval qualities.

$$m^\mu(0) = \sum_{k=0}^{n-1} \frac{N_k}{N} m_k^\mu(0) \tag{5}$$

For each region we introduce a dynamic variable a_k, expressing the attention paid to it a_k is to be given as a function of m_k. Thus, the process of recognition is described by the rule

$$S_i(t+1) = sign\Big(\sum_{k=0}^{n-1} a_k \sum_{j=A_{k-1}+1}^{A_k} J_{ij} S_j(t)\Big), \tag{6}$$

which, obviously, becomes for $n = 1$ the dynamics (1') of the original Hopfield model [5], since the constant a_1 is in efficacious due to the sign function. We remark that all considerations in the following are exact for $t = 0$ in the Hopfield model and for $t \geq 1$ in the strongly diluted model [12, 13], which can also be extended to a network with regions of different retrieval quality. In diluted models, the weights J_{ij} are given by (3) (or (10)) with probability C/N and $J_{ij} = 0$ with probability $1 - C/N$, where C^{2t-2} is required to be small in comparison to N, but large compared to 1. The symmetry $J_{ij} = J_{ji}$ is destroyed by the dilution.

Apparently, these features are more similar to the properties of the brain than those of the original Hopfield model since natural neurons exhibit a rather low connectivity, but the mathematically favorite effect of uncorrelatedness would not allow for meaningful processing.

The retrieval quality $m^1(1)$ for an input $S(0)$, being a perturbation of ξ^1, can be calculated from

$$\begin{aligned} S_i(1) &= sign\Big(\sum_{k=0}^{n-1} a_k \sum_{j=A_{k-1}+1}^{A_k} \frac{1}{N}(\xi_i^1 \xi_j^1 S_j(0) + \\ &\qquad \sum_{\mu=2}^{p} \xi_1^\mu \xi_j^\mu S_j(0))\Big) \\ &= sign\Big(\sum_{k=0}^{n-1} a_k \frac{N_k}{N} m_k^1(0) + a_k R_k\Big), \end{aligned} \tag{7}$$

where $m_k(0)$ is the retrieval quality of the $1-st$ pattern restricted to the $k-th$ part of $S(0)$. $R = \sum_{k=0}^{n-1} a_k R_k$ is, after normalization by the reciprocal of $\sqrt{\sum_{k=0}^{n-1} a_k^2 \alpha \frac{N_k}{N}}$, a standardized Gaussian random variable. Therefore, we can write $m^1(1) = \frac{1}{N} \sum_{i=1}^{N} \xi_i^1 S_i(1)$ as

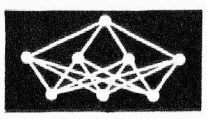
$$m^1(1) = erf\Big(\frac{\sum_{k=0}^{n-1} a_k \frac{N_k}{N} m_k^1(0)}{\sqrt{\alpha \sum_{k=0}^{n-1} a_k^2 \frac{N_k}{N}}}\Big), \tag{8}$$

where $erf(x) = \sqrt{\frac{2}{\pi}} \int_0^x exp(-x^2) dx$. From the Cauchy-Schwarz inequality for the weighted scalar product $\langle (a_k), (m_k) \rangle = \sum_{k=0}^{n-1} a_k \frac{N_k}{N} m_k$, the optimal value $a_k = c m_k(0)$ is obtained, where c can be set equal to 1.

In order to interpret this result, we rewrite equation (8):

$$m^1(1) = erf\Big(\sqrt{\frac{\sum_{k=0}^{n-1} \frac{N_k}{N} (m_k^1(0))^2}{\alpha}}\Big). \tag{9}$$

The (weighted) quadratic mean as in equation (7) can better overcome the noise term R_k than the arithmetic mean which is performed in the ordinary case, where $a_k = \frac{1}{n}$ for all k. The factor of the improvement of $m(1)$



in our model upon the ordinary case is plotted in fig. 2. for some special parameters. For the special case $n = 2$ with $m_0^1(0) \geq 0$ and $m_1^1(0) = 0$ the argument of the *erf*-function in (9) is improved by a factor of between 1 (for $N_0 \ll N_1$) and 1 (for $N_0 \gg N_1$) at the first

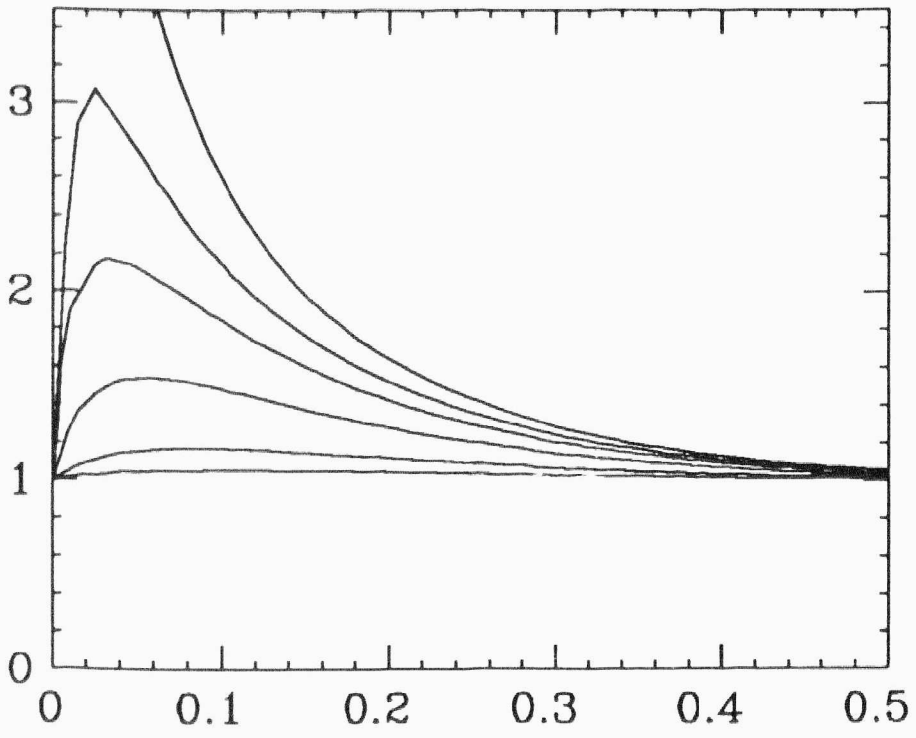


Fig 2.: Factor of improvement of $m(1)$ *from eq.(9) upon* $m_{Hopfield}(1)$ *vs.* N_0/N_1 *for* $n = 2$, $m_0(0) = 1$, $m_1(0) = 0$, *0.025, 0.05, 0.1, 0.2, 0.3 (from top to bottom),* $\alpha = .14$.

time step compared with the ordinary case. For regions of equal size a factor of up to $\sqrt{2}$ can be obtained.

4. The Bidirectional Associative Memory

A more recent approach [14] to heteroassociative memories consists in a kind of unification of the ideas of the perceptron with that of the Hopfild model. The bidirectional associative memory (BAM) consists (like a perceptron) of an input layer and an output layer of neurons which are mutually connected, but (like in the Hopfield model) the output of the network is used as the input for the next time step, where the information flows in opposite direction.

Bidirectional networks are especially useful for optical implementations of neural networks, where devices with up to 10^{12} interconnections have been proposed [15].

Thus, we are interested for the above case ($m_0(0) \geq 0$ and $m_1(0) = 0$) in the unknown second part of the vector $\mathbf{S} = (\xi, \eta)$, where we may obtain a connection matrix M given by a analogue of the Hebb rule.

$$M_{ij} = \frac{1}{N} \sum \eta_i^\mu \xi_j^\mu \qquad (10)$$

It stores correlations between the slots of the vectors ξ^μ, η^μ; $\mu = 1, \ldots, p$. Let N_0 and N_1 be the dimensions of ξ^μ and η^μ, respectively, with $N_0 + N_1 = N$, then $\mathbf{M}$ is an $(N_0 \mathrm{x} N_1)$-matrix. For $N_0 = N_1$ and $\xi^\mu = \eta^\mu$ for all μ (10) is simply the Hebb rule. Otherwise our network may serve as a heteroassociative memory and is equivalent to the bidirectional associative memory (BAM) [14] which can also be understood as a Hopfield model with a special connection matrix

$$\mathbf{J} = \begin{pmatrix} 0 & \mathbf{M}^T \\ \mathbf{M} & 0 \end{pmatrix}, \qquad (11)$$

where $N_0^2 + N_1^2$ of the N^2 connections have been wiped out. As stated in [13, 16] the introduction of nonlinear synapses (instead of (10)) in the BAM improves the information capacity. Besides, the modification of the synapses does not influence the bidirectional stability.

5. Convergence of the BAM

Since the matrix (11) is a symmetric one and has a zero diagonal, it fulfils the conditions for convergence [9] of the serial Hopfield model, i.e. the network provides a stable pair $(\mathbf{S}^{(0)}(\infty), \mathbf{S}^{(1)}(\infty))$ from the possibly noisy version of the input vector $(\xi^\mu, \mathbf{O})$, where the first operations calculate only elements of the $\mathbf{S}^{(1)}$-part. If we would like to correctly introduce an unambiguous state, we have to deal with ternary neurons (cf. [17]).

The parallel or synchronous mode is of more interest for applications, but there the sufficient condition [9] for convergence in the Hopfield model sharpens to $J_{ii} \geq \frac{p}{N}$, which the matrix (10) does not fulfill. In [14] it was shown, with the help of a Lyapunov function, that by running the network several times a so-called bidirectionally stable pair $\{(\mathbf{S}^{(0)}(\infty), \mathbf{O}); (\mathbf{O}, \mathbf{S}^{(1)}(\infty))\}$ wil be reached and that this is the case for any matrix $\mathbf{M}$. But the proof in [9] implies immediately this bidirectional stability: The serial mode applied alternatively to every $\mathbf{S}^{(0)}$-unit once and to every $\mathbf{S}^{(1)}$-unit once leads to the same result as the parallel mode, due to the block structure of the matrix (J_{ij}) to the same result as the parallel mode, due to the block structure of the matrix (J_{ij}). Thus, convergence holds for the matrix (10), too.

These ideas can be applied to the original Hopfield model, again. With the help of a sightly more complicated Lyapunov function [13] one can show that in those cases where the parallel dynamics do not converge, limit cycles with period 2 occur instead of a definite fixed point. Now, there is a way to understand this behaviour from another point of view. If the matrix (10) is a Hebb type, then also only bistable limit cycles can occur. The Lyapunov function in [13] for the parallel Hopfield model is nothing other than the simple serial Lyapunov function of a BAM.

Additionally, as the Hopfield model can be considered as a set of perceptrons with only one output neuron, the bidirectional network is related to the committee network [12]: Each slot of the output can be understood as being produced by a committee of neurons. The weights of the first layer are the rows of the matrix $\mathbf{M}$ and the weights of the vote-taking TLU have to be chosen as the columns of $\mathbf{M}$ if an improved version of the input is to obtained.

6. The Optimal Threshold for the BAM

For $m_1(0) = 0$ the formula (6) leads to a trivial result for the parameter a_k, but the retrieval quality depends essentially on the difference between N_0 and N_1. The loss of information at the larger of both sides can be overcome by the introduction of stability terms [18], i.e. weights $h_0(t)$ and $h_1(t)$ for the state of the network at the time before. Hence, they lead to a partial conservation of the information, especially while passing the larger of both sides (let $h_1(-1) = 0$):

$$S_i^{(1)}(2t+1) = sign\Big(\sum_{j=1}^{N_0} M_{ij} S_j^{(0)}(2t) + h_1(2t-1) S_i^{(0)}(1)\Big); \quad i = 1, \ldots, N_1,$$

$$S_i^{(0)}(2t+2) = sign\Big(\sum_{i=1}^{N_1} M_{ji}^T S_i^{(1)}(2t+1) + h_0(2t) S_j^{(0)}(0)\Big); \quad j = 1, \ldots, N_0 \tag{12}$$

In the context of the strongly diluted model, we obtain for the retrieval qualities of both layers

$$m_1^1(2t+1) = \frac{1+m_1(2t-1)}{2} erf\Big(\frac{m_0^1(2t)+h_1(2t-1)}{\alpha_0}\Big) + \frac{1-m_1(2t-1)}{2} erf\Big(\frac{m_0^1(2t)-h_1(2t-1)}{\alpha_0}\Big)$$

$$m_0^1(2t+2) = \frac{1+m_0(2t)}{2} erf\Big(\frac{m_1^1(2t+1)+h_0(2t)}{\alpha_1}\Big) + \frac{1-m_0(2t)}{2} erf\Big(\frac{m_1^1(2t+1)-h_0(2t)}{\alpha_1}\Big) \tag{13}$$

We obtain as optimal values for the sequences $h_1(2t+1)$ and $h_o(2t+2)$ in the parallel mode for random, independent, identical distributed patterns with expectation value zero

$$h_1(2t+1) = \frac{\alpha_1}{m_o(2t)} \log\Big(\frac{1+m_1(0)}{1-m_1(0)}\Big),$$

$$h_0(2t+2) = \frac{\alpha_0}{m_1(2t+1)} \log\Big(\frac{1+m_0(0)}{1-m_0(0)}\Big), \tag{14}$$

respectively, where $\alpha_0 = p/N_0$, $\alpha_1 = p/N_1$ correspond to the parameter α and m_0, m_1 are the retrieval qualities according to (4) of the $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}$ state vector, respectively.

Fig. 3 shows the improvement reached by optimal local fields. In the strongly diluted version of the original model the retrieval quality vanishes for $\alpha \geq \alpha_{crit}$, whereas local fields at least keep the initial retrieval quality. For medium α, an improvement of $m_0(\infty)$ as well as of $m_1(\infty)$ is visible. The output retrieval quality, however, will tend to zero as $\alpha \to \infty$ due to the information loss in each single step. The two diagrams of fig. 3 show the influence of differences in size between both sides. Leading to an improvement for different (a) as well as equal (b) sides in each case

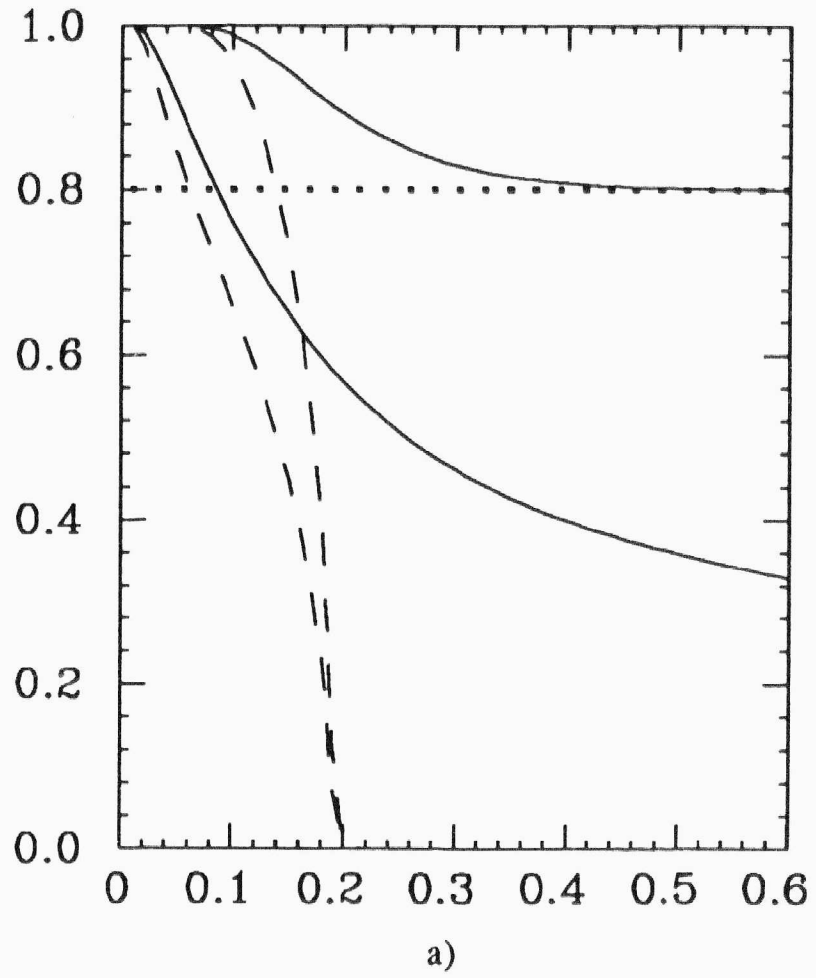


Fig. 3.: Retrieval quality as a function of α for a) $N_0/N_1 = 0.1$ and b) $N_0/N_1 = 1$. BAM with (full curves) and without (dashed curves) local fields. Upper curves represent $m^{(0)}$, lower ones $m^(1)$. For b) the dashed curves coincide. The initial overlap is $m^{(0)}(0) = 0.8$ (dotted line).

the local fields can, however, not completely overcome the worse retrieval for great differences in size.

7. Conclusion

Several approaches to heteroassociative memory based on networks of formal neurons have been discussed. Particular attention was paid to the usage of a Hopfield neural network as a heteroassociative memory. Such a modification of the original model as is presented here is known as the bidirectional associative memory. Comparing the conditions for convergence in the Hopfield model and the BAM it is easily shown that the convergence of the serial Hopfield model implies the bidirectional stability of the BAM. From this - *vice versa* - we find immediately that in the parallel Hopfield model only cycles of length of less than two can occur.

We analyzed the BAM from a more general point of view which has as a simple attention mechanism its own interest. In the strongly diluted approximation we demonstrated the positive effect of local input-dependent thresholds for the BAM.

References

[1] McCulloch, W.S.; Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. **Bull. Math. Biophys. 5** 115-133.

[2] Nilsson, N.J. (1965). **Learning machines**. New York: McGraw-hill.

[3] Rosenblatt, F. (1962). **Principles of Neurodynamics.** New York: Spartan.

[4] Rumelhlart, D.E., McClelland, J.L. and the PDP Research Group (1986). **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1: Foundations.** Cambridge: MIT Press.

[5] Hopfield, J.J. (1982). Neural Networks and Physical System with Emergent Collective Computational Abilities. **Proc. Nat. Acad. Sci. 79** 2554-2558.

[6] Brown, R.J. (1987) Committee Networks. **Dr. Doobs J.126** 16-27.

[7] Cover, T.M. (1965) Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. **IEEE Trans.Electron.Comput. EC-14** 326-334.

[8] Budinich, M (1991) On linear separability of random subsets of hypercube vertices. **J. Phys. A: Math. Gen. 24** L211-L213.

[9] English, H., Herrmann, M. (1989) The Limit Behaviour in the Hopfield Model. **Wiss. Beitr. MLU Halle-Wittenberg 54** 36-40.

[10] Chrisanti, A., Amit, D.J., Gutfreund, H. (1986) Saturation Level of the Hopfield Model for Neural Network. **Europhys. Lett. 2** 337-341.

[11] Gardner, E., Derrida, B., Mottishaw, P. (1986) Zero Temperature Parallel Dynamics for Infinite Range Spin-Glasses and Neural Networks. **J. Phys. (Paris) 48** 741-755.

[12] Derrida, B. Gardner, E., Zippelius, A. (1987) An Exactly Soluble Asymmetric Neural Network Model. **Europhys. Lett 4** 267-173.

[13] English, H., Engel, A., Schütte, A., Stcherbina, M. (1990) Improved Retrieval in Neural Nets with Thresholds and Nonlinear Synapses. **Stud. Biophys. 136** 37-54.

[14] Kosko, B. (1988). Bidirectional Associative Memory. **IEEE Transactions on systems, man and cybernetics 18** 49-60.

[15] Kinser, J.M., Caulfield, J., Shamir, J. (1988). Design for a Massive All-optical Bidirectional Associative Memory. **Appl. Opt. 27** 3442-3444.

[16] English, H., Herrmann, M. (1989). Neural Network with Nonlinear Synapses. **Studia biophys. 132** 145-152.

[17] Herrmann, M. (1989). Neural Networks with Ternary Neurons. To appear in **Neurocomputers & Attention, Proc. Int. Workshop, Moscow.**

[18] Engel, A., English, H., Schütte, A. (1989). Improved Retrieval in Neural Networks with External Fields. **Europhys. Lett 8** 393-397.

Book Review

Neural Networks (Computers with Intuition)

S. Brunal, B. Lautup

World Scientific 1990,
ISBN 0971-50-939-3

This marvelous popularization book goes far beyond the scope of its title. In 9 Chapters [Profession: Computer, The Brain' s Wetware, Knowledge, Information Processing, Computer Architecture, Binary Neural Networks, Perceptron, Computo, Ergo Sum?] the reader will find a lot of interesting facts and views about the Universe surrounding us as well as its map within us, philosophical reflections, witty parables, excursions to history and exciting remarks. You can expect everything: from XOR to free will problem.

Although intended as an amusing reading for laymen, it is probable that even a specialist will find there a simplified yet correct description of what his colleague searches for in a lab next door.

The second half of the booklet covers - in the same fashion - Hopfield and multilayered networks, without any formulas but still giving a good understanding of the authors were able to introduce some techniques even with problems of proper input coding, description of learning the hyphenation problem etc.

Annotated selected bibliography and an index conclude the book.

Prof.Dr. Jiří Hořejš
Department of Computer Science,
Charles University, Prague

ON PARALLEL HEURISTICS

Paul C. Kainen[1]

Abstract:

We consider parallel heuristics based on mathematical knowledge. Several examples are surveyed from optimization and problem solving with emphasis on how an abstract insight leads to efficient calculation. In some cases the resulting methods are error-tolerant and are problem-size independent. We give instances in visual perception in artificial, mathematically defined environments which suggest that heuristics may be utilized in some types of biological computation, and an experiment to test this idea is proposed.

I. Introduction

The word "heuristics" (from Greek *heure* - to find) is used by psychologists and computer scientists to denote random searching guided by educated guesses. Heuristics, as compared with deterministic algorithms, have a distinctly biological (rather than mechanical) feel. To the extent that a heuristic is an educated guess, it can embody the education represented by an organism's genetic, somatic or neural memory as well as nondeterministic aspects of living choice. When these choices are approximately independent, they may be carried out in parallel.

For references to heuristics e.c. see, on the pure mathematics end, Polya [35], [36]. There are countless examples in engineering – recently, for instance, [27], [28], [9], [1], [7] and [10]. Heuristics are often employed by experts to solve difficult problems. This amounts to an a priori decision to ignore most of the variables with efforts focused on the critical few. Here wisdom consists in knowing the small core of important factors.

Such techniques have been used in machine-aided calculation from the beginning (e.g. in Monte Carlo methods). As early as 1957 Simon and Newell embedded heuristics in a computer program called Logic Theorist, which could prove theorems from Principia Mathematica (in one case Logic Theorist gave a more elegant proof than the one used by Russell and Whitehead).

Our chief criterion for heuristics is that some mathematical computation is involved which allows a short cut in the abstract structure so that an entire class of concrete calculations can be avoided. Not only are heuristics invaluable in optimization and problem solving, but we believe they also may play a key role in some biological computations. These methods tend to be noise-immune and naturally parallel.

Our thesis that empirical efficiency and abstract shortcuts are strongly related is supported by examples from computer science and mathematics in sections 2 and 3. In section 4 psychophysics is considered, primarily in terms of phenomena in visual perception requiring mathematical computation. Section 5 proposes an experiment to test a heuristic theory of perception. Section 6 is a discussion of approximate independence. The references follow.

2. Combinatiorial Search

The difference between a deterministic algorithm and a heuristic is that the heuristic will usually work very fast (but may not always do so!) while the deterministic algorithm must always provide an answer but may take longer. A deterministic algorithm is supposedly 100% certain. This means that there is a mathematical proof showing how the steps of the deterministic algorithm are carrying out a theoretically rigorous data transformation. A heuristic, on the other hand, may have only a probabilistic certainty of approximate optimality.

On deeper reflection, however, these alternative approaches are actually much closer than it may seem at first. Aside from the possibility of errors in the mathematical proofs (or inconsistencies or inaccuracies in the constraining axioms and laws of the system) there is unavoidable presence of error in the data.

For instance, the well-known Kruskal minimal total-length spanning tree MST algorithm finds such a tree in a graph (when one exists) by "greedily" adding edges, shortest first, unless a cycle would result. This assumes that the length of all edges are known exactly (or at least up to ordinal certainty). If edges actually have uncertain length, then we could choose incorrect (non-shortest) edges in applying the algorithm to our data and so the resulting spanning tree could be non-minimal.

Nonetheless, we would heuristically expect the resulting total tree length to be stable since small errors only affect the order of the shortest edges and so have negligible impact on the MST.

Binpacking is a particularly clear instance of efficient computation by heuristics.

[1]Paul C. Kainen
Industrial Math 3044 N St. NW Washington, D.C. 20007

Suppose given a list L of weights (each between 0 and 1); it is required to load them into the smallest possible number of "bins" (each of capacity 1). Needless to say, the computation of the absolute minimum number of bins is NP-hard [12; p. 124-127]; of course the sum of the weights is a lower bound. Nonetheless, a simple good-guess procedure called "FFD" turns out to be (1) probably within 11/9 of best possible even in the worst case of a list constructed to be difficult; (2) as the length of the list goes to infinity, asymptotic to the best with probability 1; (3) actually optimal 75% of the time simulations for lists of length 10^5 (see Bentley et al [3]).

The FFD procedure merely puts the weights in decreasing order and then places each weight into the first bin in which it fits without exceeding the weight limit of 1. That this common sense approach works so incredibly well is typical of good heuristics. Moreover, FFD can be easily adapted to a parallel processor solution even when the data is on-line.

Heuristics can perform better than expected, but we suspect that frequently such exceptionally effective computation occurs as a result of underlying mathematical facts though their rigorous statement and proof may not be known. For instance, the Dantzig simplex method has such seemingly gratuitous power.

3. Intelligent Problem-Solving

In this section, computation problems are presented which can be solved through search based on abstract principles. Intelligence is involved in selecting the appropriate abstraction.

Example 1. Measuring by coffee spoons. Consider the notion of a "preserved quantity".

Suppose one has a container of coffee and one of milk. What happens when you first dilute the coffee with a little milk, then take some of the resulting mixture and put it back into the milk? When the amounts exchanged are equal, the respective concentrations of coffee and milk depend on the initial volumes. After the dual exchange, the total volume of liquid in each container is preserved so, at each container, exports equal imports. Hence, both imports are so dilution ratio is inversely proportional to volume ratio. Note that "perfect mixing" is not needed.

When the volumes of coffee and milk are identical, it follows that the resulting dilutions must be equal.

A parallel computer might use this approach to compare two unknown processing element populations by making a pair of random transfers and then sampling in parallel to see which population is more concentrated.

Physical models, such as above, may help to explain the "implicit parallelism" of Holland's genetic [15], simulated annealing [14], [11],[24] and Hopfield networks [16], [17]. We think that a full theory of heuristics will relate all of these ideas in the context of geometric probability (as in, e.g., Solomon [40]).

Example 2. (P.A.M. Dirac) Less can be more. Expanding the mathematical structure which describes a physical problem can greatly simplify the solution. A popular puzzle involves successive divisions of a pile of coconuts subject to modularity conditions (remainder of 1 on division by 6). Each of 6 pirates takes 1/6th of the coconuts, one after the other in order of rank (or ferocity!). Then all 6 divide the remainder equally. At each division, there is precisely one coconut left over which is given to a monkey. What is the least positive number N of coconuts for which such a process is feasible?

Dirac noticed that the procedure could be defined for negative N as well and that clearly -5 coconuts would then work! For each of the first division, a pirate "takes" -1 coconuts from the pile; that is he gives the pile one of his own! This now goes to the monkey and leaves the pile with -5 again. At the last division, all 6 of them "take" their share, that is, they collectively add 6 to the -5 which leaves one for the monkey, as required.

But if $N = -5$ works, this solves the original positive integer problem too. For if N is any solution, surely $N+6^7$ is also a solution (check that the modularity condition still holds at each division). Hence the minimum solution is $-5+6^7$. Calling -1 coconut an "anti-coconut", the reader may better appreciate Dirac's theoretical prediction of the anti-electron.

Extending the domain can facilitate parallelism. For example, real numbers are usually represented within a computer using floating point notation. But Schroeder [38] has pointed out the advantages of more sophisticated number-theoretic techniques which, in addition to permitting operators to act simultaneously on all of the components of a representation (without worrying about "carry bits"), also lead to error-free computation.

Example 3. (A. Renyi) **When is ignorance bliss?**

If one has absolutely no information regarding a random graph process, then one may as well regard the process as uniformly random. Let V be any sequence of n labeled vertices. Define a random process TREE which produces with equal likelihood any of the n^{n-2} distinct labeled trees on V. Recall that for any graph a vertex is called an end point if it is adjacent to exactly one other vertex in the graph.

Theorem (Renyi). For any $h > 0$ there is an integer, N, such that when $n > N$ if v is chosen uniformly at random in V and T is an outcome of TREE, then |Pr (v is an end point of T) $-e^{-1}| < h$. Here, $e = 2.718\ldots$, and in fact the convergence is very fast. Thus, a random vertex in a random tree has a chance of about e^{-1} of being an end point.

Now suppose one were asked to exhibit an end point for some given tree T. If T is small, of course this is trivial. But if T. (i.e., n) is very large, then the deterministic solution for this problem requires accessing a huge amount of data, and in fact it can be shown [45] that the smallest deterministic algorithm requires on average log n steps. This algorithm starts at a random vertex in the random

tree and takes a "walk" until, since there are no cycles in such a tree, the walk ends at an end point of the tree.

But Renyi's Theorem says that k random guesses would have a probability of $(1-e^{-1})^k$ of all being wrong (sampling without replacement). Thus, for any $h > 0$ there exists K such that, for $k > K$, k random guesses have probability $< h$ of not finding an end point. This is **independent** of n. Even 10 random guesses almost surely suffice for any random tree.

Note that the heuristic is not only infinitely faster but it is also more robust. Suppose data describing the tree contains an extra edge. If the deterministic algorithm is unlucky enough to use this edge, then it would never escape from the resulting cycle. However, the heuristic procedure uses less information (it is not trying to follow a path in the graph) and so a few extra edges (or even a lot of them) don't significantly affect its rapid convergence.

To be fair we should mention that if the tree on n vertices happens to be a path, the guessing heuristic needs on the order of n tries – but then so does the deterministic algorithm. Due to their minimal requirements for information, heuristics lend themselves to parallel and decentralized implementation, without the need to pass auxiliary variables among the elements of a decomposition. In this example, processor could independently seek an endpoint.

4. A Look at Psychophysics

When they are guided by mathematical structure, heuristics can throw away most of the information because it is irrelevant. This seems to us very much like the behavior of the human visual system. For example, a pixelized photograph can often be more easily recognized when one squints at the image, thereby blurring it to obscure the irrelevant detail of the irregular black and white digital checkerboard in order to respond to the statistical pattern of light and dark density. The computational efficiency of vision has long been studied as part of the mathematical psychology of perception otherwise known as "psychophysics". See, e.g., [2], [6], [32].

Many years ago N. Bernstein [4] and, independently, G. Johannson [20] showed that subjects (of the experiments) could describe complex bodily movements based only on the trace made by small lights attached to performers who were otherwise entirely invisible. Individual performers could be identified from the characteristic nature of their traces. (This should not be surprising to anyone who has recognized an approaching person from the rhythm of the foot steps).

In this section we shall give four other examples in which pattern perception appears to occur accurately based solely on mathematical information. We believe that the psychophysics of these perceptual phenomena is heuristic. The speed and "effortless" nature of the perception strongly suggest parallel computation.

Example 1. (L. Glass) Take a random pattern of opaque spots on a transparency and superimpose another copy with the same center but a small rotation. The subject immediately perceives a family of concentric moiré circles. As the rotation is increased, circles disappear (largest first) until all correlation is lost and a random field of twice the original density appears. Similarly, expansion and contraction appear as explosions and implosions. See [13], [43].

Example 2. (B. Julesz) This experiment also involves random dot patterns but here it is the statistics of their distribution which are compared. In normal stereoscopic vision, both eyes focus on a common object. The left and right view have the same key "landmarks" appearing in slightly shifted parallax with respect to one another and background.

In the Julesz variant the subject is shown a left/right artificial image pair (normally created by computer but a TTL-version has been proposed). Each image consists of an array of variable blob shapes called "textons" each element of which can be modulated as to geometric type, position and orientation. By controlling the statistical distribution of the pair, it is possible to cause the subject to experience the perception if an "abstract" mathematical object as a 3-dimensional figure in space, that is, as a virtual stereogram. For instance, cubes, tori and hyperbolic surfaces are shown in Vol. VII: Perception, Handbook of Sensory Physiology. See [21], [18], [37], [33]. Some effort may be necessary to fuse the stereo pair.

Example 3. (E.H.Land) The human sense of color somehow retains remarkable fidelity under extreme distortions of lighting quality; this can be seen even without contextual clues. There are two different illustrations of the effect.

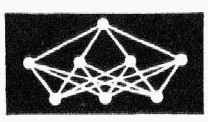
Suppose a scene is photographed twice on black and white film with one picture taken through a red filter and one through a green filter. Slides are made and the red-filtered black and white film is projected back through a red filter but the green filtered black and white is projected without a filter, as shades of gray. Scenes with context, like a table with bowls of fruit, are immediately perceived in "natural" color.

In a different manifestation of psychophysical color constancy, the subject sees a "Mondrian-painting-like" assemblage of small overlapping rectangles with specified Munsell values (hue, saturation and intensity). Even when the assemblage is illuminated by two narrow-band light sources (e.g.,lasers), the subject is able to identify the underlying Munsell value of a rectangle. See [29].

Example 4. (J.A.Lissajous) A beam of light which is deflected by two vibrating mirrors will trace a repeating geometric figure, called a Lissajous figure, when the rates of oscillation of the mirrors is in a fixed whole-number ratio – for instance, the 2 to 3 figure is shown in Fig.1.

When the frequencies are identical, then the relative phase of the two oscillations determines the figure which the subject observes (Fig.2).

The only psychophysics involved so far is just persisten-



ce of vision; one continues to see the trace of the bright spot after it has moved on. The shape seen is the geometry scanned.

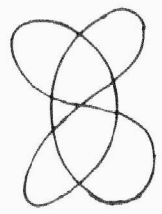


Fig.1

But as soon as there is any distortion in the perfect frequency ratio of the oscillators, Lissajous noted that the perceived figure appears to take on a 3-dimensional quality as though a wire-frame structure were rotation in space. Furthermore, mechanical impedance or other factors which affect the internal dynamics of the mirror cause a distortion in the figure but not a change of topology. See Figs. 1 and 2.

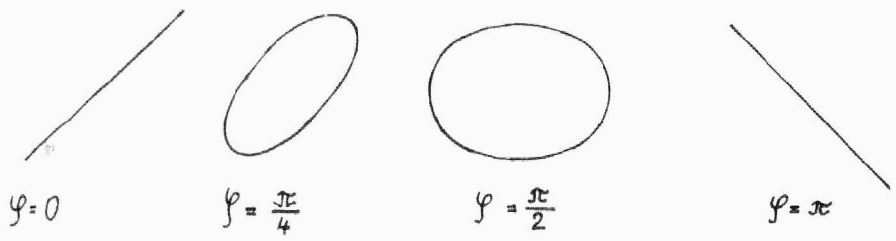


Fig. 2

Now psychophysics is involved since the actual trace is a quasi-periodic, non-closed figure but it is perceived through a kind of "envelope" as a moving closed figure. The shape seen is not what a photograph would show but represents a strong closure effect. For instance, when the two frequencies start equal and at $\pi/2$ ("quadrature") phase then as one oscillator varies slightly in frequency the initial circle seems to be embedded in 3-space and to be rotating about an axis at $\pi/4$ or $-\pi/4$ depending on which frequency is larger. See [23].

Acoustic pattern recognition. The above examples all relate to vision but certainly analogous mathematical analysis is performed within the sense of hearing. For example, (see [38], p. 22) when two tones at frequencies f1 and f2 are simultaneously presented, a subject may hear the "greatest common divisor" frequency, yet the tone heard changes smoothly with small changes in f1 and f2. This is just like the Lissajous phenomenon, and may also be involved in echolocation by bats [41].

5. Is the visual perception of connectivity achieved heuristically?

The perception of connectivity has been proposed as a basic element of Gestalt visual perception. How difficult is such a computation from an information-theoretic standpoint?

It can be shown that if one is only allowed to ask about the existence of one edge at a time, the problem of determining whether or not a graph on n vertices is connected requires $n(n-1)/2$ questions in the worst case. (See Bollobas [5].

But nevertheless connectivity appears to be "effortlessly" perceivable. Contrast this to map coloring problem (MCP) where even experts proceed slowly and sequentially. The MCP is quite difficult and has to be done step by step, possibly with backtracking. (See, e.g., [39]).

There are many perceptual tasks which appear to be inherently sequential (like the MCP); for instance, looking for the letter "w" in a paragraph. On the other hand, detecting that a line of type is incorrectly set an angle to its fellows is effortless and immediate.

Perceptual solution of combinatorial and geometric problems fits a heuristic model. For example, the famous "traveling salesman" problem (to find a shortest total-length cycle through a set of N cities in the plane) has no known algorithmic solution which does not grow exponentially with N. Human ability to obtain reasonably good tries needs to be explained. For an interesting approach, see Durbin and Willshaw [8].

To find a heuristic for determining connectivity, we propose using the theory of random graphs due to P. Erdöes and A. Renyi (see, e.g., Spencer [42]). The following is a typical result of the theory:

Theorem. If G is a random graph on n vertices where each edge occurs with some uniform independent probability $p = p(n)$, then there is a threshold for connectivity at $p = log\ n/n$ (logarithm base e).

Given a large random graph, one could quickly test it for connectivity by taking a sample of the possible vertex pairs and determining what percentage are edges of the graph. If the percentage is significantly greater than $log\ n/n$, then the graph is almost surely connected.

Note that sampling heuristics of this sort are ideally parallizable. Many vertex pairs can be checked simultaneously and independently. The resulting edge density estimate tells us almost surely a great deal about the graph: connectivity, planarity, etc. (see [42]).

It would be quite feasible to measure psychophysically how the speed and accuracy of visual perception of test a theory of heuristic perception.

A good way to study any computational procedure is to examine its failure cases. Suppose subjects are presented with the image of a disconnected graph where the statistics of edge density, if the graph had been randomly generated without precondition, would "guarantee" connectedness. If subjects perceive the graph to be connected, then they could not be using any sort of internal deterministic algorithm, but would seen to be using a heuristic based on the Erdöes-Renyi theory.

6. Approximate Independence

In dividing a problem into separate pieces for a set of parallel processors, we would prefer not to need any commu-

nication between the processing elements. Heuristics based on random sampling often have the desirable property that the outcome of each sample is nearly independent of the others, and such heuristics will thus be well-suited to parallel implementation.

The mathematical issues involved in generalized statistical independence are far from resolved, but the practicality of the concept has already found substantial application in communication networks. For example, Kleinrock [25] notes that in packet-switching networks, the message length distribution at distinct nodes are approximately independent.

Approximate independence is the assumption underlying many heuristics. We have argued elsewhere ([22], [39]) that this assumption is physically reasonable. At the present time, theory is not adequate to explain the overwhelming success of heuristics based on approximate independence. For instance, something more than luck must have been involved in the successful performance of such arguments in predicting convergence time for the computer reduction techniques used by Appel, Haken and Koch in their solution of the Four Color Problem. (See [39]).

Classically (see Kolmogorov [26], p. 10]), a set of events are mutually independent when the probability of the intersection of any non-empty subset is the product of the probabilities of its members. Recent nonstandard models take a different approach – see, e.g., [34]. Independence certainly fails when there is some causal link between the events (when it rains, the probability that the street is wet increases). Yet when the causal links become sufficiently weak, the whole corpus of physics and engineering shows that we may effectively assume independence. For instance, the theory of Maxwell-Bolzmann statistics uses such an argument (see, e.g., [44], p. 164).

A simple but useful example of approximate independence occurs for gas thermodynamics. Although individual molecules do not move independently of one another, since there are about 10^{23} of them in a cubic centimeter of gas, the chance that any particular two interact strongly is essentially zero. (Of course there is also always the very weak force of gravity). Assuming independence, one can derive the gas laws relating temperature, pressure and volume.

The theory of chaos might be quite helpful in justifying approximate independence. If molecular kinetics cannot be exactly computed because of the continual influence of other particles (through collisions, EM fields, gravity or whatever) then the chains of exact causation, which are required to prevent independence, themselves become highly improbable.

One way to insure approximate independence is to add random noise (as in simulated annealing). Adding noise can be cooled below zero [C].

A clear example of approximate independence was found by D. Matula [30]. Let G be a random graph with n vertices and edge probability p. Fix k a non-negative integer and let $\mathcal{F}$ be the set of all k-element vertex sets in G. Obviously, for $S \in \mathcal{F}, P(G(S)$ is not complete) $= 1 - p^{b(k,2)}$, where $b(k,2)$ is the binomial coefficient. The events $\{G(S)$ is not complete $S \in \mathcal{F}\}$ are **not** independent but they are approximately so when k $\ll n$. A rigorous argument somewhat like that for the Chebyshev inequality guarantees that for those values of k for which $(l - p^{b(k,2)})^{b(n,k)}$ acts like an $0-1$ variable, the result is nearly identical to P (G has no k-element compete subgraph). The only "bad" values for k make $b(n,k)p^{b(k,2)}$ approximately equal to 1 so in general the chance that any such k exists is zero (see McDiarmid [31] or [42] for the details). Thus, the random variable which measures the number of vertices in the largest complete subgraph has a very nearly Dirac delta distribution!

References

[1] R.E.Boring, M.Andrews and F.Lam, Towards portable microcode, IEEE Region 5 Conf. 1988 (88CJH2567-6), Electrotechnology, 1-5.

[2] H.B.Barlow, Reconstructing the image in space and time, **Nature**, **279** (1979), 189-190.

[3] J.L.Bentley, D.S.Johnson, F.T.Leighton, C.C.McGeoch, L.A.Mc Geoch, Some unexpected expected behavior results for bin packing, preprint, 1984.

[4] N.Bernstein, **The Coordination and Regulation of movements**, Pargamon, Oxford, 1967.

[5] B.Bollobas,**Extremal Graph Theory**, Academic Press, 1978.

[6] A.T.Bahill, L.Stark, The trajectories of saccadic eye moments, Sci.Am., January 1979, 108-117.

[7] J.Cong, C.L.Liu, Over-the-cell channel routing, IEEE Int'l Conf. on CAD, ICCAD-89, 80-83.

[8] Durbin, D. Willshaw, An analogue approach to the /travelling salesman problem using an elastic net method, Nature 326 (1987), 689-691.

[9] B. Efron, **The Jackknife, the Bootstrap and other Resampling Plans** (CBMS 380), SIAM, Philadelphia, 1982.

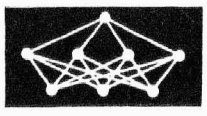
[10] A. Gamst, Homogenous distribution of frequencies in a regular, hexagonal cell system, IEEE Trans. Veh. Tech., VT-31 (1982), 132-144.

[11] S. Geman, D. Geman, IEEE Trans. on PAMI, 6 (1984),721-741.

[12] M.R. Garey, D.S. Johnson, **Computers and Intractability**, Freeman, San Francisco, 1979, pp.124-127.

[13] L. Glass, R. Perez, Perception of random dot interference patterns,**Nature246** (1973), 360-362.

[14] G.E. Hinton J.J. Sejnowski, IEEE Proc on Comp. Vision and Pattern Recognition, 1983, 448-453.



[15] J. Holland, **Adaptation in natural and artificial systems.** U. of Mich. Press, Ann Arbor, 1975.

[16] J.J. Hopfield, PNAS 79 (1982), 2554-2558; 81(1984),3088-3092.

[17] J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems, Biol. Cybernetics **52**, (1985),141-152.

[18] B. Julesz, J.J. Chang, Interaction between pools of binocular disparity detectors tuned to different disparities, **Biol Cybernetics22** (1976), 107-119.

[19] D.J. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Computing, **3** (1974), 299-325.

[20] G. Johansson, Studies on visual perception of locomotion, **Perception 6** (1977), 365-376.

[21] B. Julesz, Cooperative phenomena in binocular depth perception, **Amer. Sci. 62** (1974), 32-43.

[22] P.C. Kainen, The significance of the four color theorem, in **Proc. Humboldt State U. Conf.**, P.Z. Chinn and D. McCarthy, Eds., Util. Math. Winnipeg, 1979, pp.49-66.

[23] . . . , Lissajous arithmetic, in Eine Kleine Lichtmusik, G. Edwards, Ed., AAAS, Houston, 1979.

[24] S. Kirkpatrick et al., Science, 229(1983), 671-679.

[25] L. Kleirock, Communication Nets, Dover, New York, 1972.

[26] A.N. Kolmogorov, **Foundations of the Theory of Probability.** Chelsea Publ. Co., New York, 1956.

[27] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Eds., **The Traveling Salesman Problem**, Wiley, Chichester, 1985.

[28] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. **21** (1973),498-516.

[29] E.H. Land, J.J. McCann, Lightness and the retinex theory, **J. Opt. Soc. of America 61** (1971), 1-11.

[30] D.W. Matula, On the complete subgraphs of a random graphs, in Combinatorial Mathematics and Its Applications, Chapel Hill, 1970, pp.356-369.

[31] C. McDiarmid, On the method of bounded differences, in **Sureys in Combinatorics**, J.Siemans, Ed. (London Math Soc. LNS 141), Cambridge University Press, 148-188.

[32] N.H. Mackworth and A.J. Murandi, The gaze select informative details within pictures, **Perception & Psychophysics, 2** (1967), 547-552.

[33] D. Marr, T. Poggio, Cooperative computation of stereo disparity, **Science, 194** (1976), 283-287.

[34] E. Nelson, **Radically Elementary Probability Theory** (Annals of Math Studies 117), Princeton University Press, Princeton, NJ, 1987.

[35] G. Polya, **Mathematics and Plausible Reasoning** (2 vols.), Princeton University Press, 1954.

[36] G. Polya, **How To Solve It!**, Princeton University Press, 1973.

[37] S.S. Shetly, A.J. Brodersen, R. Fox, **System for generating Dynamic Random Element Stereograms**, Technical Report N14-1101/79C-0003, Energy Psychol. Pgms., Office of Naval Research, 1979.

[38] M.R. Schroeder, **Number Theory in Science and Communication**, Springer, Berlin, 1986.

[39] T.L. Saaty and P.C. Kainen, **The Four Color Theorem**, Dover, New York, 1986.

[40] H. Solomon,**Geometric Probability** (CBMS 28), SIAM, Philadelphia, 1978.

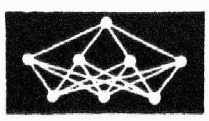
[41] N. Suga, Cortical computational maps for auditory imaging, **Neural Networks 3** (1990), 3-22.

[42] J. Spencer, **Ten Lectures on the Probabilistic Method** (CBMS 52), SIAM, Philadelphia, 1987.

[43] J. Walker, The amateur scientist columm, **Sci.Am.**, May 1980, 176-189.

[44] G.H. Wannier, **Statistical Physics**. Dover, New York, 1966.

[45] D.L. Wong, On the average path length between two modes in a randomly generated tree, in: **Algorithms and Complexity**, J.F.Traub, Ed., Academic Press, New York, 1976, p.504.



ON SOME VARIANTS OF ADAPTIVE RULES OF FEATURE MAPS

K. Hlaváčková[1]

Abstract:

Several learning variants of self-organization in a map-type neural network are demonstrated. Algorithms based on terms of neighbourhood are discussed. A new conception of neighbourhood - the ordering neighbourhood - is introduced.

1. Introduction

The self-organizing map neural network was developed by Teuvo Kohonen during the period 1979-1982 [8-16]. His work was strongly inspired by the pioneering self-organizing map studies of von der Malsburg [18].

Historically, the self-organizing map was one of the most important neural networks discovered before the major expansion of neurocomputing in the second half of the 1980's.

This work deals with several versions of the self-organizing learning algorithm. The second section explains basic definitions, the third describes Kohonen's learning rule and includes an original discussion about it. The minimal spanning tree algorithm is discussed in the fourth section; the idea being Kohonen's, Kangas's and Laaksonen's [7]. Finally, the new concept of an ordering neighbourhood is proposed in the fifth section.

2. A Map Type Neural Network

Let us describe the model of the neural network that is called a *map* (topological map, feature map). A map is a two-layered network: the first layer represents n input units (neurons); let the input vector be denoted $\underline{x} = (x_1, \ldots, x_n)$. The domain $\underline{x}$ of is called the *input space*.

The second layer represents output units and is called the *(neuron) field*. Let the number of output units be m. There may be given edges among them that are called the *topology of the network*. The space where the neuron field is located is called the *output space*.

This space, in contrast to other types of neural networks, is identical with weight space - the domain of *reference vectors* $\underline{w}_i$. Every unit from the first layer is connected with every unit from the second one. If we consider a given topology of the network, the a map maps reference vectors in time into the output space, so it is "mapping" output over time with respect to the topology.

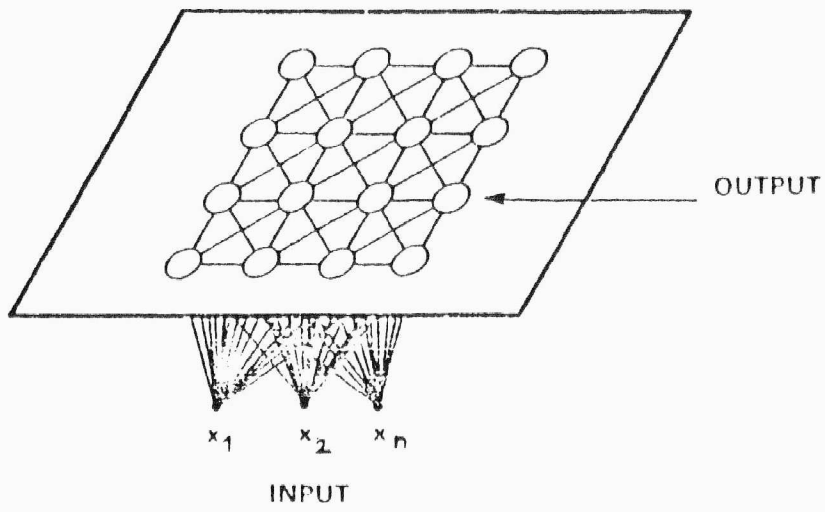


Fig. 1.

Example of a map with n input units and a neuron field in a two-dimensional output space.

In contrast to maps, other neural networks do not consider the mutual space relations of units in the field, and space coordinates of the output have no relation to the input space. We will call those network "maps with a zero-order topology".

In neural networks of the map type, the units in the field reply to the input signals as if the "curved coordinate system" (which reflects topological changes over time) was drawn over the field of neurons with a given topology.

If the dimension of the coordinate system over the field of neurons is s, then we say the network map *has an s-order topology.*

Remark:

Input and output (weight) spaces usually have the same dimension n. We can generally take the output space to be a subspace of an s-dimensional space, where $s \geq n$ and s is a natural number. The network is then an n-dimensional map with an s-order topology.

As a neural network, the map has two phases the adaptive phase (learning) and the active phase. In the adaptive phase the map "learns" given input patterns according to

[1] Kateřina Hlaváčková
Institute of Computer Science, Czechoslovak Academy of Sciences, P.O.Box 5, 18207 Prague 8, Czechoslovakia

given rules. In the active phase that follows, the map provides the output (reply) to the input (question). The map has the same rules for both phases (in contrast to some neural networks, for example the Hopfield network).

3. Learning Methods in Maps

3.1. Competitive Learning

Competitive learning is an adaptive process, when the neurons of the neural network are "tuned" according to input values. The basic principle of competitive learning ensues from work on several problems in mathematical statistics, especially cluster analysis.

The basic idea is the following:

Let $\underline{x}(t) \in \mathcal{R}^n$ be a sequence of samples of vectors, where t is the time coordinate, and a set of variable reference vectors

$$\{\underline{w}_i(t) | \underline{w}_i(t) \in \mathcal{R}^n, \quad i = 1, 2, \ldots, k\}.$$

Assume that the $\underline{w}_i(0)$ have been initialized (for instance by random selection). Competitive learning means: if input $\underline{x}(t)$ is simultaneously compared with each $\underline{w}_i(t)$ at each successive instant of time (taken here to be integer $t = 1, 2, 3, \ldots$) then the best-matching $\underline{w}_i(t)$ is to be adapted to match $\underline{x}(t)$ even more closely.

If the comparison is based on some distance function $d(\underline{x}, \underline{w}_i)$, the adaptation must be such that if $\underline{w}_c$ is the best-matching reference vector, then $d(\underline{x}, \underline{w}_c)$ decreases and all the other reference vectors $\underline{w}_i$, $i \neq c$ are without change.In this way the different reference vectors tend to be "tuned" to different domains of the input $\underline{x}$. If the distribution of the input is according to the probability density function $p(\underline{x})$, then $\underline{w}_i (i = 1, \ldots)$ tend to describe "clusters".

The first work on competitive learning follows from the method that we will now describe .

3.2. Vector Quantization (VQ) Method

Vector Quantization is a classic method that produces an approximation of a continuous function using a finite number of "reference" vectors [4,17]. We will use this method for a continuous probability density function. First, we will describe this method according to Kohonen [11].

Assume that X is a compact subset of $\mathcal{R}^n$ the Euclidean space, and let a vector variable $\underline{x}(t) \in \mathcal{R}^n$ have the probability density function $p(\underline{x})$ on X(t is the time coordinate). Let reference vectors $\underline{w}_i$, $i = 1, 2, \ldots, k$ be located in X and let $\underline{x}$ be approximated by the closest reference vector $\underline{w}_c$, $c = c(\underline{x})$ obtained from the criterion:

$$\| \underline{x} - \underline{w}_c \| = \min_{i=1,\ldots,k} \{\| \underline{x} - \underline{w}_i \|\}, \tag{1}$$

where $\| \cdot \|$ is the Euclidean norm ($\| \underline{x} \| = \left(\sum_{i=1}^{n} (x_i)^2\right)^{1/2}$).

This criterion assigns *a competition winner* to each $\underline{x}$ and we call it "the competitive criterion" and the learning process using this criterion *competitive learning.*

The average discretization error with respect to a given power r (r natural number) is defined by the functional

$$E(\underline{w}_{c(\underline{x})}) = \int \| (\underline{x} - (\underline{w}_{c(\underline{x})} \|^r \, dp(\underline{x}), \tag{2}$$

where $p(\underline{x})$ is the probability density function taken as a measure on X.

Kohonen [8-11] introduced the so-called adaptive rule of the (VQ) method

$$\begin{aligned} \underline{w}_c(t+1) &= \underline{w}_c(t) + \alpha(t)[\underline{x}(t) - \underline{w}_c(t)] \\ \underline{w}_i(t+1) &= \underline{w}_i(t) \qquad\qquad\qquad\qquad for\ i \neq c, \end{aligned} \tag{3}$$

where $\alpha(t)$ is a suitable monotonically decreasing sequence of scalar coefficients satisfying

$$0 < \alpha(t) < 1, \lim_{t \to \infty} \alpha(t) = 0, \sum_{t=1}^{\infty} \alpha(t) = \infty.$$

Kohonen took as a corresponding function in VQ method the function

$$E(\underline{w}_{c(\underline{x})}) = \int \| \underline{x} - \underline{w}_{c(\underline{x})} \| \, dp(\underline{x}). \tag{4}$$

The equations (3) are the simplest description of competitive learning.

Although successful in application [13, 16], this method still lacks a deeper theoretical analysis.

Our following theorem could show the theoretical background, which the adaptive rule might have-been be inspired by.

Theorem 3.2.1.:

Let $\underline{w}_c = \underline{w}_c(t)$ be the closest reference vector to $\underline{x} = \underline{x}(t)$ in the sense of the criterion (1). Let a function E (some given "error function") have continuous first partial derivatives. Let $\{\alpha(t)\}$, $t = 0, 1, 2, \ldots$ be a sequence satisfying:

$$0 < \alpha(t) < 1, \lim_{t \to \infty} \alpha(t) = 0, \sum_{t=1}^{\infty} \alpha(t) = \infty.$$

Then the optimizing process expressed by the equation

$$\underline{w}_c(t+1) = \underline{w}_c(t) + \alpha(t) \; grad \; E(\underline{w}_c)$$

produces a sequence of vectors $\underline{w}_c(t)$ such that

$$E(\underline{w}_c(t+1)) < E(\underline{w}_c(t)).$$

This sequence is either finite and its last member $\underline{w}_c^*$ is a stationary point (i.e. $grad \; E(\underline{w}_c^*) = 0$) or it is infinite and each of its limit points is a stationary point.

Proof:

This statement is a consequence of the theorem about the convergence of Cauchy's steepest-descent gradient method for function $E(\underline{w}_c)$ [20, p.29]. As it has continuous partial derivatives in the compact set X, the statement holds.

Remarks:

1) Function (2)'s first partial derivatives are not generally continuous. For this reason, describing the stationary points of (3) is theoretically difficult.

2) Approximation of a probability density function depends on the choice of residual error of convergence algorithm and on the r-power .

3) The speed of the convergence of the algorithm given by rule (3) is given by the speed of the gradient method (3) and depends on the choice of the parametric function $\alpha(t)$ in (3). As a variant of the the method (and in general for all competitive learning), - another iterative minimizing method of (2) can be chosen, for instance the method of conjugate gradients. Comparing these two methods, the convergence speed of the steepest-descent gradient method is quadratically lower then the speed of the conjugate gradients method.

3.3. The Self-Organizing Map Algorithm

The principal idea of self-organization is that if input signals are given by the probability density function, then weight (reference) vectors try to imitate it (Kohonen [11]).

In the previous part, we introduced the adaptive rule, which did not consider space relations among units. Units behaved independently. So the order in which their indices were assigned to the ranges of input vectors was more or less random and depended on the initial values $\underline{w}_i(0)$.

In 1973, von der Malsburg [18] published his work on computer simulations, where he demonstrated local ordering of units into small subsets that correspond roughly to the so-called columns of the cortex. The units in a small subset were "tuned" more closely than were the more remote units.

Amari [1] constructed a corresponding system of differential equations for two-dimensional continuous vector variables. These works are of great theoretical importance because they deal with a self-organizing tendency.

Kohonen, as early as 1981, experimented with various architectures and systems of equations and introduced the description of a process that seems to be general for the creation of globally self-organizing maps, whose algorithm we present here.

Assume a map whose output layer is a two-dimensional neuron field as in Figure 2.

The arrangement of the neuron field (that is the topology) can be hexagonal, rectangular, octagonal, etc. (i.e. a unit in the field is connected with six, four or eight of its nearest neighbours, respectively; Figure 1 shows the last example.

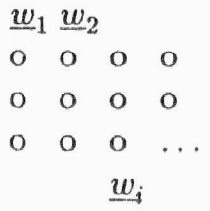


Fig. 2

Let $\underline{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{R}^n$ be an input vector and let it be connected in parallel with all units in the field (the second layer).

Let the weight vector of the $j-th$ unit be $\underline{w}_j = (w_{j1}, \ldots, w_{jn}) \in \mathcal{R}^n$.

It is important in these maps that units in the field are not altered independently by themselves but as sets of topologically connected units where similar changes are made. Changes of every unit in the field (or, more accurately, of its weight vector) will tend to be smoothed out in the long run.

In biologically inspired neural network models,learning of neighbouring cells can be implemented using various kinds of lateral feedback connections and other lateral interactions.

Now we shall explain a conception of these interactions (which are the mutual interaction of units in the field) for an arbitrary topology of the network by introducing the concept of a *neighbourhood set* N_c of a unit c.

The neighbourhood set of unit c N_c is defined as a set of units (or their indices) in the field whose distance from unit c (in the sense of the shortest path in the graph given by the topology) is at most the predefined radius.

At each learning step, all units within N_c are adapted but units outside N_c are without change. This neighbourhood is around the unit c, whose weight coordinates are the closest to the vector $\underline{x}$ in the space. Thus, c is determined from

$$\| \underline{x} - \underline{w}_c \| = \min_j \{ \| \underline{x} - \underline{w}_j \| \}, \qquad (5)$$

where $\underline{x}, \underline{w}_j \in \mathcal{R}^n$
and $\| \cdot \|$ is a norm in $\mathcal{R}^n$.

Unit c is called the *winner* of the competition. The magnitude of the radius of N_c can be time-variable.

Remark:

During his experiments, Kohonen [11] found that it is advantageous to let a radius of neighbourhood be rather large in the beginning (for instance covering half of the units in the field) and then shrink it monotonically with time. The reason (briefly said) is that a wide initial neighbourhood, corresponding to a course placement in the space in the learning process, first induces a coarse global ordering in the $\underline{w}_i$ values. It is also possible that it can be

$N_c = \{c\}$ in the end, which would mean that only the winner is adapted and the self-organizing process is reduced to simple competitive learning (see equations (3)).

The adaptive rule can be written:

$$\begin{aligned} \underline{w}_j(t+1) &= \underline{w}_j t) + \alpha(t)[\underline{x}(t) - \underline{w}_j(t)] \quad for\ j \in N_c(t) \\ \underline{w}_j(t+1) &= \underline{w}_j(t) \qquad\qquad\qquad\qquad for\ j \notin N_c(t), \end{aligned} \tag{6}$$

and similar to (3), the monotonically decreasing sequence $\alpha(t)$ must satisfy

$$0 < \alpha(t) < 1, \quad \lim_{t\to\infty}(t) = 0, \quad \sum_{t=1}^{\infty} \alpha(t) = \infty.$$

Remarks:

1) The system of two types of equations can be expressed by the system of one type of equations:

$$\underline{w}_j(t+1) = \underline{w}_j(t) + h_{cj}(t)[\underline{x} - \underline{w}_j(t)],$$

$$\begin{aligned} \text{where } h_{cj}(t) &= \alpha(t) \qquad \text{if } j \in N_c(t), \text{ and} \\ h_{cj}(t) &= 0 \qquad \text{otherwise.} \end{aligned}$$

The definition of h_{cj} and of a neighbourhood set is more general; a biological lateral interaction often has the shape of a "bell curve". If we note the coordinates of the unit c as $\underline{w}_c$ and coordinates of the unit i as $\underline{w}_i$, we can use a function

$$h_{ci} = h_o exp(- \parallel \underline{w}_i - \underline{w}_c \parallel^2 / \rho^2), \tag{7}$$

where $\parallel . \parallel$ is a norm in $\mathcal{R}^n$, and $h_o = h_o(t)$ and $\rho = \rho(t)$ are suitable decreasing functions of time.

2) Rule (6) is applied to a set of vectors with indices in $N_c(t)$. There is an interesting question whether a function minimized by (6) exists and if so, how it depends on set $N_c(t)$.

It seems (Kohonen [8]) that the general type of local interactions is the type where the closest neighbouring units reinforce (excite) each other, whereas the distant weaken each other (inhibit). The more mutually distant units influence each other more weakly, mostly in an excitative way.

This form of local interaction is often expressed by the so called "function of a mexican hat" (thanks to its shape in the three-dimensional space). The use of this function in the model of self-organization was inspired by some physiological model. We will simplify local interactions by using the neighbourhood set.

3) After large enough number of iterations if $N_j = \{j\}$ for all j, the algorithm producces a map with zero order topology (there are no connections among units in the field).

An important question is under what conditions will the algorithm produce a map with zero-order topology (after a suffcient number of iterations, of cource).

3.4. A Variantion of the Algorithm with Normalization

The following is another variant of the algorithm. By normalizing the input vector $\underline{x}$ before applying the algorithm and weights during the algorithm, it can improve numerical accuracy because the reference vectors have the same dynamic range as the weights have. But the normalization will not affect the idea of the algorithm.

In [11], Kohonen introduced an algorithm which normalizes the reference vectors at each step. The normalization, unfortunately, slows down the learning algorithm significantly. Let $\underline{x}$ be the input vector and $\underline{w}_i(0)$ be normalized initial weight vectors, which means members of $[0,1]^n$. In $[0,1]^n$, define a norm by the scalar product

$$(\underline{x}, \underline{y}) = \underline{x}^T \underline{y} \tag{8}$$

and then $\qquad \parallel \underline{x} \parallel = (\underline{x}, \underline{x}).$

The criterion of the winner in the learning algorithm is replaced by the criterion

$$(\underline{x}(t), \underline{w}_c(t)) = \max_j \{(\underline{x}(t), \underline{w}_j(t))\}, \tag{9}$$

and the adaptive rules are

$$\underline{w}_j(t+1) = \frac{\underline{w}_j(t) + \beta(t)\underline{x}(t)}{\parallel \underline{w}_j(t) + \beta(t)\underline{x}(t) \parallel}, \qquad if\ j \in N_c(t) \tag{10}$$

$$\underline{w}_j(t+1) = \underline{w}_j(t) \qquad\qquad \text{otherwise;}$$

$0 < \beta(t) < \infty$; for example $\beta(t) = 100/t$.

In the following Proposition, we show a relationship of this method to the method described in paragraph 3.3.

Proposition 3.4.1.:

a) If the input vector $\underline{x}$ and initial weight vectors $\underline{w}_i(0)$ are normalized, then the criteria (9) and (1) are equivalent.

b) After introduction of normalization for $\underline{w}_j(t+1)$, $j = 1, 2, \ldots$ from the adaptive rule (6) after every step. This rule is equivalent to (10).

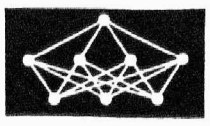
Proof:

a) According to the definition (8),

$\parallel \underline{x} - \underline{w}_j \parallel = (\underline{x} - \underline{w}_j,\ \underline{x} - \underline{w}_j) = (\underline{x},\ \underline{x}) - (\underline{x}, \underline{w}_j) - (\underline{w}_j), \underline{x}) + (\underline{w}_j, \underline{w}_j)$.
If $\parallel \underline{w}_j \parallel = \parallel \underline{x} \parallel = 1$, then $\parallel \underline{x} - \underline{w}_j \parallel = 2 - 2(\underline{x}, \underline{w}_j)$ and looking for a min $\parallel \underline{x} - \underline{w}_j \parallel$ is equivalent to looking for a $\max(\underline{x}, \underline{w}_j)$ over $j = 1, 2 \ldots$.

b) Assign $\beta(t) = \frac{a(t)}{1-\alpha(t)}$ in (10) for $j :\in N_c(t)$:

$$\underline{w}_j(t+1) = \underline{w}_j(t) + \frac{a(t)}{1-a(t)}\underline{x}(t)$$



$$= \frac{1}{1-a(t)}\{[1-a(t)]\underline{w}_j(t) + a(t)\underline{x}(t)\}$$
$$= \frac{1}{1-a(t)}\{\underline{w}_j(t) + a(t)[\underline{x}(t) - \underline{w}_j(t)]\}.$$

After normalization we get the adaptive rule (6).

4. The Spanning Tree Neighbourhood

In this part of the paper, we will consider a neuron field to be a set of vertices in the graph and their topology (their mutual arrangement) to be edges in an undirected graph. A new approach to the neighbourhood was originally introduced in [7].

The local neighbourhood is defined "dynamically" during the learning. The neighbourhood is defined according to the mutual magnitude of vector differences among the vectors $\underline{w}_i$. The minimal spanning tree (MST) algorithm assigns the edges between two vertices so that all vertices in the graph are connected, the connection does not create a cycle and the total sum of lengths of the edges is minimal.
Here, the lengths of the edges are defined to be Euclidean norms of the vector differences between the corresponding reference vectors.

Let $H_c(t)$ be a minimal spanning tree in the complete graph of the vertex set $N_c(t)$. The neighbourhood set at time $t+1$ $N_c^*(t+1)$ is then defined as a set of vertices of $N_c(t+1)$ which are connected to the unit c via a path in $H_c(t)$ passing vertices from $N_c(t)$ only.

As in the original learning algorithm, there is a wide neighbourhood set around each unit here, and this means considering more vertices of the field including the unit and the edges between them according to the topology of the field; the MST algorithm is then applied to this neighbourhood. The neighbourhood $N_c(t)$ (and therefore also $N_c^*(t)$) shrinks during the learning process.

Remark:

According to [7] weights are changed very "smoothly" over time in some neighbourhoods (thanks to the parameter $\alpha(t)$) and therefore it is not necessary to compute a minimal spanning tree at each step of learning for the input vector $\underline{x}(t)$. Kohonen experimentally found that it suffices to compute a new minimal spanning tree every 10 to 100 learning steps [7].

On the basis of ideas from [7] we introduce the following algorithm:

Let t be a variable for discrete time values $t = 0, 1, 2, \ldots$
Let z be the maximum number of iterations of the algorithm.

1. Let $\underline{x}(t)$ be an input vector and $\underline{w}_j(0)$ initial weight vectors.
Let the neighbourhood $N_j(t)$ be initialized for every j (units in the field) and every $t = 0, 1, \ldots z$.
Let $t := 1$;

2. Assign $\underline{x} := \underline{x}(t)$, where $\underline{x}$ is the variable for the input vector.

3. Compute the distance $d_j = d(\underline{x}, \underline{w}_j(t))$ for every index j in the neuron field;

4. $d_c := \min d_j$, $j = 1, 2, \ldots$

5. Let $H_c(t)$ be a minimal spanning tree of the complete graph on the vertex set $N_c(t)$ in the sense of the sum of distances over a graph,
where a distance of edges from E is given by the function

$$f : E \longrightarrow R^+$$

$$f(\{\underline{w}_i(t), \underline{w}_j(t)\}) = \| \underline{w}_i(t) - \underline{w}_j(t) \|,$$

where $\underline{w}_i(t)$ are weight vectors in time t, $\{\underline{w}_i(t), \underline{w}_j(t)\}$ is an (undirected) edge from E, and $\| \cdot \|$ denotes the Euclidean norm.
Let $H_c^*(t)$ be defined as before.

6. Adapt the weights

$$\begin{aligned} \underline{w}_j(t+1) &:= \underline{w}_j(t) + \alpha(t)[\underline{x} - \underline{w}_j(t)] && j \in N_c^*(t) \\ \underline{w}_j t+1) &:= \underline{w}_j(t) && j \notin N_c^*(t) \end{aligned}$$

7. $t := t + 1$

8. If $t < z$, go to step 2.

9. End.

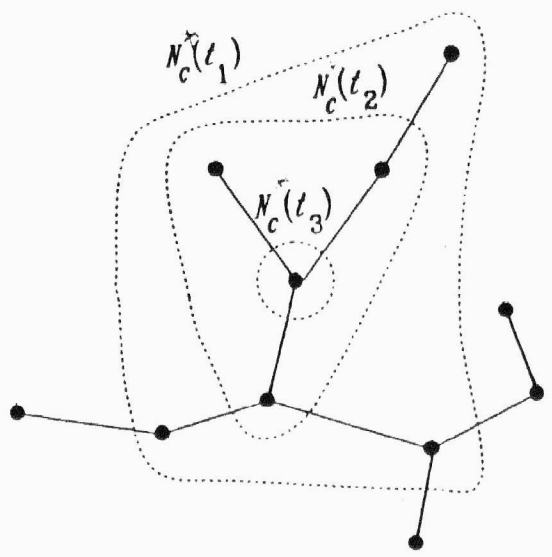


Fig. 3.

Examples of neighbourhood sets of the neuron c in the minimal spanning tree topology (According to [7].)

Remarks:

1) $|N_c^*(t)| \leq |N_c(t)|$ and so the adaptation rule in Step 6 is used for at most as many units as in set $N_c(t)$.

2) The originators of the idea of the MST-algorithm are Kangas, Kohonen and Laaksonen [7], who experimented with this algorithm on the computer. But so far no work

has been published which studies this algorithm theoretically with the aim of approximating the probability density function. There also have not been published any comparisons with other topologies in the light of this question.

5. The Ordering Neighbourhood

In the following we introduce a new adaptive rule.
Let a neuron field with its topology be given in the output space. We considered the concept of the neighbourhood of a unit in various ways. But in general, the closeness of index values in the neighbourhood of some unit in some topology in the field need not preserve the closeness of weight vectors corresponding to those indices during the iterative run. It depends on how the unit field is indexed and what the initial weights $\underline{w}_j(0)$ are.

The adaptive rule works on every unit in the neighbourhood of a chosen unit to bring their weight vectors near the weight vector of this unit in due time. It can happen that some weight vectors "closer" than the weight vectors inside the neighbourhood can lie outside the neighbourhood. We will try to improve this situation with another conception of a neighbourhood. We want the best-matching neighbourhood both in indices and in weight vectors. We will use information that has not been used before. During the searching of the weight with minimal distance from the vector $\underline{x}$ we went through all weights, but only the best-matching weight was considered and the others were forgotten and the others were forgotten.

However, now we arrange them in an increasing (according to the distance from the vector $\underline{x}$ sequence and we will define the closeness for corresponding neurons in the field in this way.

We change the neighbourhood dynamically by assigning the units with the closest weights to it.

We now show this learning algorithm in a more detailed form than in the algorithms so far.

5.1. Ordering neighbourhood learning algorithm (ON-algorithm)

Assume $[0,1]^n$ to be the input space of the vector $\underline{x}$ (t). Assume the neuron field with indices $\{1,\ldots,m\}$ to be located in the Cartesian power of some interval in $\mathcal{R}^n$, denote it K^n. Assume $[0,1]^n$ to be a subset of K^n. Let $r_1,\ldots,r_m$ be corresponding n-dimensional indices of weight vectors in K^n. A neighbourhood set is taken as an index set.

The learning algorithm:

1. Assign random values to the initial weights

$$w_{ij}(0) := u, \quad u \in K \qquad i = 1,\ldots,n \qquad j = 1,\ldots,m.$$

Let $q(t)$ be the number of units in $N_j(t)$ in time t for every $j = 1,\ldots m$ in the unit field.(A decreasing sequence of sets in time.) Let z denote the number of iterations of the algorithm.

$t := 1$;

2. Get the input $\underline{x}$ (t)

3. Compute

$$d_{r_j} = d(\underline{x}(t), \underline{w}_{r_j}(t)) = \sum_{i=1}^{n}(x_i(t) - w_{r_j i}(t))^2 \qquad j = 1,2\ldots,m.$$

4. Arrange d_{r_j} increasingly;
without loss of generality we can suppose

$$d_{r_1} <= d_{r_2} <= \ldots <= d_{r_m}.$$

Assign $N_{r_1}(t) := \{d_{r_1},\ldots,d_{r_{q(t)}}\}$.

5. Adapt weight in the neighbourhood $N_{r_1}(t)$

$$\begin{aligned} \underline{w}_j(t+1) &:= \underline{w}_j(t) + \alpha\underline{x}(t)[\underline{x}(t) - \underline{w}_j(t)] \quad if\ j \in N_{r_1}(t) \\ \underline{w}_j(t+1) &:= \underline{w}_j(t) \qquad\qquad\qquad\qquad\qquad\quad if\ j \notin N_{r_1}(t) \end{aligned}$$

6. $t := t + 1$

7. If $t <= z$, go to step 2

8. End.

Remarks:

It is obvious that the assumption about $[0,1]^n$ and K^n is not substantial from the algorithm's point of view.

The reasoning behind this assumption was to have initial weight vectors $\underline{w}_j(0)$ in the same space as vectors $\underline{x}(t)$.

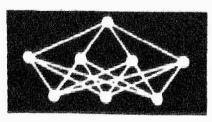
There are many theoretical questions about this conception of neighbourhood, but they are topics of future work.

One of them is to study the influence of the dynamically-taken neighbourhood for the approximation of a probability density function compared with the described methods.

The existence of a minimizing function which would describe both MST and ON-algorithms is not known.

6. Discussion

We have demonstrated several variants of a self-organizing learning algorithm. Let us note that this work was not aimed at exhausting all known variants. It is also important to note that there are a lot of mathematical questions regarding the variants discussed: for instance comparison of the probability density function approximation, the convergence speed of self-organizing processes, et al. The authors of the MST-algorithm [7] claim that this algorithm provides a far better and faster approximation of prominently structured probability density functions, but there has not yet been published any theoretical analysis on this topic.



References

[1] Amari, S.I.: Topograhic organization of nerve fields, Bull. Math. Biology, Vol.42, pp.339-354, 1980.

[2] Mc Dermott, E. and Katagiri, S.: Shift-invariant multicategory phoneme recognition using Kohonen's LVQ2. Proc.Int.Conf. on Acoustics, Signals, and Speech, ICASSP 89 (Glasgow, Scotland), pp.81-84.

[3] Fuller, J., Farsaie, A.: Invariant target recognition using feature extraction (as in [17], pp.595-598).

[4] Gersho, A.: On the Structure of Vector Quantizers, IEEE Trans.Inform.Theory, Vol.IT-28, pp.157-166, March 1987.

[5] Hopfield,J.J.: Neural networks and physical systems with emergent collective computational abilities, Proc.Nat.Acad.Sci. USA, Vol.79, pp.2554-2558, 1982.

[6] Iwamida, H., Katagiri, S., McDermott,E., and Tohkura,Y.: A hybrid speech recognition system using HMMs with an LVQ-trained codebook, ATR Technical Report TR-A-0062, ATR Auditory and Visual Perception Research Laboratories, 1989.

[7] Kangas, J., Kohonen,T., Laaksonen, J.: Variants of Self-Organizing Maps, IEEE Trans., on Neural Networks, Vol.1, No.1, March 1990.

[8] Kohonen, T.: Self-organized formation of topologically correct feature maps, "Biolog.Cybern.", Vol.43, pp.59- 61, 1982.

[9] Kohonen, T.: Automatic formation of topological maps of patterns in a self-organizing system, Proc.2nd Scand.Conf.on Image Analysis (Espoo,Finland, 1981), pp.241-220.

[10] Kohonen, T.: Clustering, taxonomy and topological maps of patterns, Proc.Sixth Int.Conf. on Pattern Recognition (Munich, Germany, 1982) pp.114-128.

[11] Kohonen, T.: The Self-Organizing Map. Proc. of the IEEE, Vol.78, No.9, September 1990.

[12] Kohonen, T.: Improved Versions of Learning Vector Quantization. In San Diego I, June 1990.

[13] Kohonen, T.: The 'neural' phonetic typewriter, Computer, Vol.21, pp.11-22, March 1988.

[14] Kohonen, T., Makisara, K. and Saramaki, T.: Phonotopic maps-insightful representation of phonological features for speech recognition, "Proc.Seventh" Int.Conf. on Pattern Recognition (Montreal,Canada, 1984), pp.182-185.

[15] Kohonen, T., Tokkola, K., Shozakai, M., Kangas, J., and O.Venta, O.: Microprocessor implementation of a large vocabulary speech recognizer and phonetic typewriter for Finnish and Japanese,"Proc.European Conference on Speech Technology" (Edinburgh, 1987), pp.377-380.

[16] Kohonen, T., Makisara, K. and Saramaki, T.: Phonotopis maps-insightful representation of phonological features for speech recognition, Proc.Seventh Int.Conf.on Pattern Recognition (Montreal, Canada, 1984), pp.182-185.

[17] Makhoul, J, Roucos, S., and Gish, H.: Vector Quantization in Speech Coding. Proc. IEEE, Vol.73, No.11, pp.1551-1588.

[18] Malsburg, Ch.v.d.: Self-organization of orientation sensitive cells in the scriate cortex. Kybernetik, Vol. 14, pp.85-100, 1973.

[19] Max, J.: Quantizing for Minimum Distortion. IRE Trans. Inform. Theory, Vol.IT-28, pp.157-166, Mar.1982.

[20] Polak,E., Computational Methods in Optimization, Academic Press 1971, New York.

[21] Orlando, J., Mann, R., Haykin, S., Radar classification of sea-ice using traditional and neural classifiers, Proc.Int.Joint.Conf. on Neural Networks, IJCNN-90-WASH- DC, 1990, pp.2. 263-266.

[22] Rumelhart, D.E, Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In Parallel Distributed Processing: Explorations in the microstructure of Cognition. Vol1: Foundations, Eds. Cambridge, Mass.: MIT Press, 1986, pp.318-362.

[23] Waibel, A., Hanarawa, T., Hinton, G., Shikano, K., and Lang, K.J.: Phoneme recognition using time-delay neural networks. IEEE Trans.Acoust. Speech and Signal processing Vol. ASSP-37, pp.382-399, 1989.

[24] Zador,P.L.: Asymptotic Quantization Error of Continous Signals and the Quantization Dimension. IEEE Trans. Inform. Theory, Vol.IT-28, No.2, pp.139-142, March 1982.

Floreen P.: Worst-Case Convergence Times for Hopfield Memories
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.533-535

Key words: neural networks; convergence; memories.

Abstract: The worst-case upper bound on the convergence time of Hopfield associative memories is improved to half of its previously known value. Also the consequences of allowing "don't know" bits in both the input and the output are considered.

Fogel D.B.: An Information Criterion for Optimal Neural Network Selection
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.490-497

Key words: neural networks; optimal.

Abstract: Neural networks have been used to resolve a variety of classification problems. The computational properties of many of the possible network design have been analyzed, but the decision as to which of several competing network architectures is "best" for a given problem remains subjective. A relationship between optimal network design and statistical model identification is described. A derivative of Akaike's information criterion (AIC) is given. This modification yields an information statistic which can be used to objectively select a "best" network for binary classification problems. The technique can be extended to problems with an arbitrary number of classes.

Frye C.R., Rietman E.A., Wong Ch.C.: Back-Propagation Learning and Nonidealities in Analog Neural Network Hardware
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp.110-117

Key words: adaptive learning.

Abstract: Authors present experimental results of adaptive learning using an optically controlled neural network. They have used example problems in nonlinear system identification and signal prediction to study the capabilities of analog neural hardware.

Grosh J., Hukkoo A., Varma A.: Neural Networks for Fast Arbitration and Switching Noise Reduction in Large Crossbars
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.8 pp. 895-904, ISSN 0098-4094

Key words: neural networks.

Abstract: A neural network-based controller is presented for the real-time arbitration of routing paths in large crossbar switches constructed from one-sided crosspoint chips. This controller is suitable for a synchronous environment where a number of connection requests are simultaneously presented to the switch.

Hwang N.J., Choi J.J., Oh S., Marks R.J.: Query-Based Learning Applied to Partially Trained Multilayer Perceptrons
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp.131-136

Key words: query-based neural network learning.

Abstract: This paper presents a novel approach for query-based neural network learning. Consider a layered perceptron partially trained for binary classification.

Johnson J.L.: Globally Stable Saturable Learning Laws
Neural Networks Vol.4, 1991 No.1 pp.47-51

Key words: global stability; optical memory; adaptation; saturable learning; Lyapunov function; temporal competition; code stability; storage capacity.

Abstract: This article discusses minimal saturable learning laws that incorporate deassociative as well as associative behavior. The requirement of global stability results in new terms for the nodal activity equation.

Johnson L.G., Jalaleddine S.M.S.: Parameter Variations in MOS CAM with a Mutual Inhibition Network
IEEE Transactions on Circuits and systems Vol.38, 1991 No.9 pp.1021-1029

Key words: parameter variations; neural networks.

Abstract: A MOS implementation of mutual inhibitin is combined with a standard content addressable memory (CAM) to produce a relaxative content addressable memory (RCAM) that automatically relaxes to the nearest match to an input. The operation of the RCAM is similar to the Hamming neural net. A simple model to predict the dynamic behavior of the mutual inhibition circuit is proposed.

Kong S.G., Kosko B.: Differential Competitive Learning for Centroid Estimation and Phoneme Recognition
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp.118-124

Key words: competitive learning.

Abstract: The Authors compared a differential-competitive-learning (DCL) system with two supervised competitive-learning (SCL) systems for centroid estimation and for phoneme recognition.

Kosko B.: Stochastic Competitive Learning
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.522-529

Key words: neural networks; stochastic; learning.

Abstract: We examine competitive learning systems as stochastic dynamical systems. This includes continuous and discrete formulations of unsupervised, supervised, and differential competitive learning systems. These systems estimate an unknown probability density function from from random pattern samples and behave as adaptive vector quantizers.

TIME DELAY NEURAL NETWORKS

H. Beran[1]

Summary:

The Time Delay Neural Network (TDNN) is a perceptron- like structure with several layers of neurons. Each neuron receives not only the instantaneous part of information from all the neurons of the previous layer, but also the past part of information from the predefined delays. The full equivalence between the Time Delay and Back Propagation (BP) is prooved, where the TDNN structure is equivalent to the classical BP structure with some empty connections. The original approach to the adaptation algorithm based upon this equivalence is derived. Simple examples of the network performance are demonstrated.

1. Introduction

Spatiotemporal pattern recognition is one of the typical neural network tasks. Several approaches have been developed up to now, taking into account either various physiological analogies or mathematically-derived laws. Two major groups of models are studied in the world, Avalanches [4,6,7] and Time Delays [5,12,13]. These models are often used for real-time signal processing.

The most simple signal processing method was developed for the one-dimensional signal, where the signal moves across the input window of the Back-Propagation network [1,3,9]. This method is very suitable for proper signal processing because of its phase-invariance. Most real signals, however, are multidimensional, i.e. vectors varying in time. As a result, we receive a spatiotemporal matrix (after digitalisation of the signals) which is to be analyzed. Several difficulties occur when trying the Back-Propagation approach in this case. The most simple solution is to use the matrix like a vector for training of the BP network. The network is very large in this case because of the number of the matrix elements. The overlearning effect may often occur, and the network requires more computer capacity and speed. Time-Delay Neural Networks solve the problem of multidimensional signal processing in a very elegant manner.

[1]H. Beran
Institute of Physiological Regulations,
Bulovka pav.11, 180 85 Prague 8,
Czechoslovakia

2. Network Description

The Time-Delay Neural Network is a Back-Propagation network with special neurons. Its formal structure is the same as the BP. The typical TDNN neuron is shown in Fig. 1. In comparison with the classical Back-Propagation, it takes into account not only the up-to-date input value, but also several of the last input values in pre-defined time delay intervals. The output function of the neuron is sigmoidal. All the layers of the TDNN network are composed of such elements, i.e. the next layer also takes into account the delayed values of the inner signals.

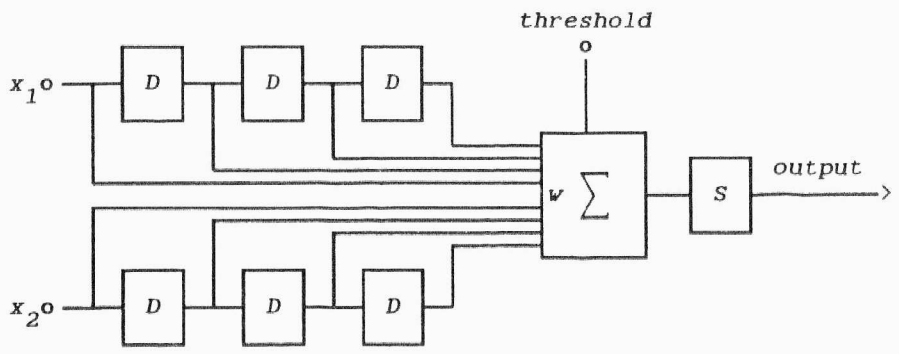


Fig. 1. An example of a TDNN neuron. The neuron can have several inputs (this example has 2, x_1 and x_2). Each input is delayed by one ore more delay units D (their number is equal for all the inputs in one layer). The inputs and all their delays are passed through the weighted sum to the sigmoidal output function S.

Each layer has its own number of delays, which can be different from the other layers, but this number must be equal between the neurons of the same layer. For the activation of the hidden and output TDNN layers some predefined amount of input delays of the hidden and output neuron inputs is necessary. We can imagine this temporal activation like the influence of the neurons in the previous layer in the past, or the influence of the temporal images of that layer (i.e. the previous layer outputs in the past). The values of all the network activities can be reached only from the input of the network and its values in the past. The total input delay required for the activation of all the network with all its inner layers and their delayed outputs exceeds the pre-defined delay of the input layer, which is only the delay necessary for the activation of the first hidden layer.

3. Learning Algorithm

The TDNN is adapted by gradient descent, like the classical BP. Like in the BP, the input given to the input layer causes the output, which can be compared with the desired output via an error function. The algorithm for the BP is derived and described by Rumelhart and his coworkers [11]. In this article, the same algorithm and similar notation are used being modified for the purposes of TDNN networks. The error function is

$$E = \frac{1}{2} \sum_{c} \sum_{j} (y_{j,c} - d_{j,c})^2, \qquad (1)$$

where y is the actual and d the desired network output, j and c are the indices representing all the output neurons and all the inputs in the learning set respectively.

This non-negative function is to be minimized via changing of the network parameters, i.e. weights and thresholds.

The output signal of the j-th neuron is given by the sigmoidal transfer function

$$y_j = \frac{1}{1 + e^{-x_j}} \ , \qquad (2)$$

where x_j is the value of the input summation of the $j-th$ neuron. There is a difference between TDNN and BP, because the TDNN also takes into account the temporal factor of the input matrix. The input sum can be expressed as

$$x_j = \sum_{i} \sum_{d} w_{i,j,d} * y_{i,d} \ , \qquad (3)$$

where $y_{i,d}$ are the outputs of the previous layer, i means the spatial index, d the temporal one, i.e. the signal delay, and $w_{i,j,d}$ are the weights from the i-th neuron of the previous layer from its d-th temporal delay to the j-th output neuron. "$*$" means simple multiplication (everywhere in this article).

Thus the output neuron, in comparison with the classical BP has an input spatiotemporal matrix of size d $*$ i instead of an input vector of length i.

The formula for the output layer gradient is very similar to that one of the BP:

$$\delta E / \delta w_{i,j,d} = y_{i,d} * y_j * (1 - y_j) * (y_j - d_j) \ . \qquad (4)$$

The situation in hidden layers is much more complicated due to the temporal structure of the network. The weights leading to the particular neuron have their spatial and temporal coordinates and the algorithmisation of the problem becomes very difficult. There exists a possibility to develop the algorithm individually for each network's topology, which is not suitable for experiments on the computer.

The adaptation method developed by the author is described in the following paragraphs. It is based upon the similarity between the TDNN and the BP.

3.1. The decomposition of the TDNN and the equivalent network

The classical approach is to consider the TDNN to be a dynamic system with inner variables, represented by the delay units of its neurons. The input vector, varying in time, is sampled by the pre-defined sampling interval being passed throughout the network from input to output. The layers of the network add the inner information about the past values of all the signals. Some period is necessary for filling in all the inner variables when starting the network. This approach is the "on-line" one. It corresponds to the classical TDNN definition by Waibel's research group [5,12,13]. This author's approach is the "off-line" one, using spatiotemporal matrices instead of the inner variables of the network.

For this approach it is necessary to define the temporal images of the neurons. The temporal image of the neuron is the same neuron with the same weights and threshold, but supplied by different input values than the actual neuron. Thus, the output from this temporal image is different. The values of the inputs of the temporal image (i.e. all the inputs including the delayed ones) are simply temporarily-shifted values of the input signal of the actual neuron. The actual neuron takes into account its input signals and their pre-defined number of delays, its temporal image calculates the same from the same number of the values, but shifted to the past. A very simple example of the TDNN with the temporal image is shown on Fig. 2.

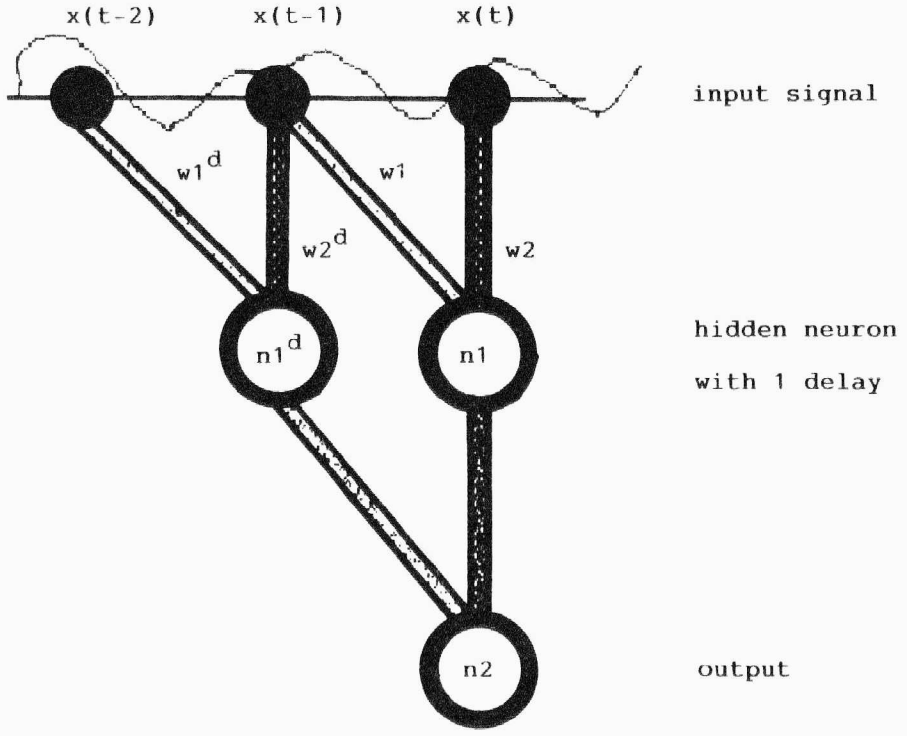


Fig. 2. A very simple TDNN structure. The network has one input, one hidden neuron n1 and one output neuron n2. X(t), x(t-1) and x(t-2) are the samples of a one-dimensional input signal. Both the hidden and the output neuron take into account one delay of their input signal. For the full input information of the output neuron the delayed output of the hidden neuron temporal image $n1^d$ is necessary. $N1^{d\prime}s$ input has its origin in the more delayed samples of the input signal (x(t-1); x(t-2)). Taking into account the fact that $n1^d$ is nothing more than the temporal image of n1, the weights w1 and $w1^d$ (also w2 and $w2^d$) must be equal.

The example also shows that the spatiotemporal matrix necessary for the full activation of the network is larger than the input of the dynamic system with inner variables.

If we consider all the neurons and their temporal images to be a new neural network structure, we obtain the BP network with some empty connections, its function being equal to the original TDNN. This BP network is called the equivalent network. The equivalent network has the following features:

a) The basis of the equivalent network is the original TDNN.

b) Instead of the inputs from delay units, each neuron has inputs from temporal images of the previous layer (except the input layer, which directly uses the past values of the input signals). The number of temporal images necessary for the definition of the layer's actual output is equal to the number of delay units in the layer.

c) The outputs of the temporal images of the layer are reached via temporal images of the previous layer (TDNN is a feedforward network). Enough temporal images must exist in the previous layer to supply the temporal images of the next layer. The total number of temporal images in the previous layer is thus greater than that necessary for the definition of the actual layer output (the previous output must be also defined). This rule raises the number of nodes in the equivalent network.

d) The total number of temporal images in particular layers must be enough to define the output of the network without any inner variables and without changing (shifting) the network input.

e) The inputs of the equivalent network and its particular layers are spatiotemporal matrices with temporal size greater than or equal to the number of delays in the corresponding layer of the original TDNN network.

f) In comparison with the BP of the corresponding size there are some empty connections in the equivalent network (except singular cases).

g) The equivalent network is only the transformation of the TDNN dynamic system (with inner variables) into the simple combinatorial system (with non-linear transfer functions) with a larger input matrix. Both of these networks (systems) are fully equivalent, no loss of features and properties occurs when using one of them instead of the other.

The number of delays of the network and its slabs can be simply calculated from the formula:

$$T_i = \sum_{m=i}^{L} D_m + 1 \ , \tag{5}$$

where T_i is the number of temporal images in the i-th layer of the equivalent network, D_m is the pre-defined delay of the m-th layer of the original TDNN network ($D_m = 0$ for no delay; the classical BP has $D_m = 0$ for all m), and L is the number of the layers. This formula is a logical result of the previous rules. Its derivation is very clear in Fig. 2. The formula also shows how the requirements for more temporal images rise when raising the number of layers of the network.

The comparison between the equivalent TDNN structure and the BP of corresponding size is shown Fig. 3. We can see some restrictions resulting from the fact that some hyperplanes are parallel to some axes (a logical result of empty connections).

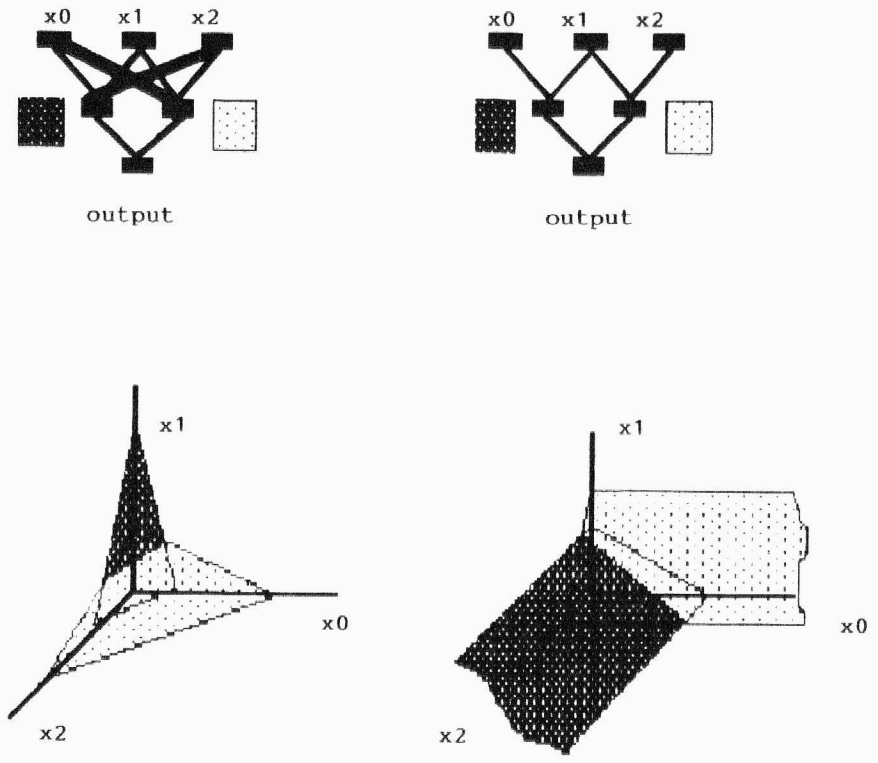


Fig. 3. A comparison between a full BP structure (left) with 3 inputs, 2 hidden neurons and 1 output neuron and the TDNN structure (right) shown on fig. 2. Both networks have the same number of elements but different number of weights. The 3-dimensional input space with axes x1, x2 and x3 is in both cases separated by the 2 hyperplanes of the 2 neurons in the hidden layer. One missing connection (bold line) causes the separating hyperplane of the hidden neuron to be parallel to the axis representing the missing input.

3.2. The adaptation of the equivalent network

The TDNN structure is adapted by the gradient descent algorithm developed by Rumelhart et al. [8,11]. The equivalent structure is adapted by the same manner. This method is very simple for the purposes of algorithmisation and computer simulation. In the classical BP algorithm without time delays the gradient can be expressed as

$$\delta E / \delta w_{i,j} = y_i * \delta_j \ , \tag{6}$$

where $w_{i,j}$ is the weight from the i-th neuron of the previous layer with the actual output y_i to the j-th neuron of the next layer, δ_j is a recursive "starvation" of this neuron, i.e. its demand to change the weights. These recursive members can be calculated in the output layer as

$$\delta_j = y_j * (1 - y_j) * (y_j - d_j) \ , \tag{7}$$

where y_j and d_j are the desired and actual output values respectively, and in the other layers

$$\delta_j = y_j * (1 - y_j) * \sum_k \delta_k * w_{j,k} \tag{8}$$

where $w_{j,k}$ is the weight leading from the j-th neuron to be calculated to the k-th neuron of the next layer; it is multiplied by the recursive member δ_k having been calculated for the k-th neuron in that layer. This is the recursive rule for calculating all the changes throughout the network from output to input (back-propagation of the error).

In the case of the equivalent TDNN structure the output layer has the same formula (7) because the output is considered without its time delays. In hidden layers, taking into account the spatiotemporal structure of an equivalent network, we obtain

$$\delta_{j,d} = y_{j,d} * (1 - y_{j,d}) * \sum_{k} \sum_{n=L}^{U} \delta_{k,n} * w_{j,k,d-n} \quad , \qquad (9)$$

where $\delta_{j,d}$ is the starvation of the j-th neuron in its d-th time delay (i.e. the starvation of the d-th temporal image of the equivalent network), $y_{j,d}$ is the output of this neuron, $\delta_{k,n}$ is the starvation of the k-th neuron of the next layer in the n-th temporal image, $w_{j,k,d-n}$ is the weight from the j-th neuron of a given layer to the k-th neuron of the next layer from the time delay d-n, where L and U are the lower and upper limit of the temporal summation respectively.

The situation is shown in Fig. 4. The lower and upper limits of the temporal summation must follow the incomplete structure of the TDNN. Some neurons receive less temporally delayed connections than the others. Formulas for calculating limits are the following:

$$L = \max\ (0, d - D_i)\ \ and \qquad (10)$$

$$U = \min\ (d, T_{i+1}) \qquad (11)$$

where d is the given time delay for calculation of the starvation $\delta_{j,d}$ (see eq. 9), D_i is the number of delays of the given TDNN layer (i.e. the original number of delays, without other temporal images for calculating of the equivalent structure, see eq. 5) and T_{i+1} is the total number of delays in the next layer of the equivalent network. The function of these limits can be seen on Fig. 4. In the parts of the network with a full structure the temporal summation is from $d - D_i$ to d; all the temporal images of the given layer are taken into account. In the parts with an incomplete structure, the limits have the bounds 0 (lower limit) and T_{i+1} (upper limit).

When calculating the change of a given weight we must take into account the fact that this weight is not only the weight of the actual neuron but also that of its all images. To follow the rules of the equivalent network we must adapt this weight synchronously, so as not to change the images from the actual neuron. The starvations, however, are influenced by different input values at different images, and so each image can have different need to change the same weight. For weight in a single image we receive from eq. 6:

$$\delta E/\delta w_{i,j,d,n} = y_{i,d} * \delta_{j,n} \quad , \qquad (12)$$

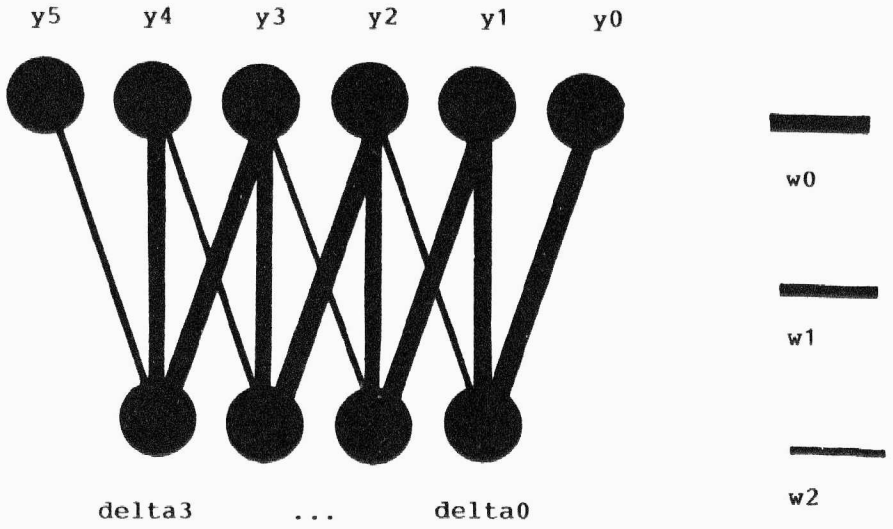


Fig. 4. The situation of the backward propagation of the errors on temporal images of neurons. The neuron in the first layer has 5 and the neuron in the second 3 temporal images. That means that 3 temporal images of the second layer are necessary to supply the next layer, the second layer takes into account only 2 delays of the first layer, but 5 temporal images of the first layer are necessary to supply the second one with all its temporal images. Actually, there are only 3 weights w0, w1 and w2 belonging to the neuron of the second layer. In comparison with a BP structure, some of the connections are empty due to the incomplete temporal structure. The starvations delta0 .. delta3 are back-propagated through the network along its incomplete connections, being multiplied by the outputs of the first layer y0 .. y5. The back-propagation of the errors has as a result the need to change the weights w0, w1 or w3 depending on which weight it has been back-propagated.

where $w_{i,j,d,n}$ is the weight in a single temporal image from the i-th neuron in his d-th temporal image in a given layer to the j-th neuron of the next layer in its n-th temporal delay, $\delta_{j,n}$ is the starvation of j-th neuron in n-th temporal delay and $y_{i,d}$ is the corresponding output of the i-th neuron. This formula expresses only the local change, i.e the change of a single (n-th) temporal image of the weight $w_{i,j,d}$. The global weight change can be determined by averaging

$$\delta E/\delta w_{i,j,d} = \frac{1}{T_{m+1}} \sum_{n=0}^{T_{m+1}} \delta E/\delta w_{i,j,d,n} \ ,$$

$$\delta E/\delta w_{i,j,d} = \frac{1}{T_{m+1}} \sum_{n=0}^{T_{m+1}} y_{i,n+d} * \delta_{j,n} \ , \qquad (13)$$

where T_{m+1} is the number of temporal images in the next layer (i.e. the layer emitting starvations) of the equivalent network (see eq. 6). We can see that the starvation with the temporal index n corresponds to the output of the previous layer $n + d$ (see Fig. 4).

Practically, the adaptation is performed with the momentum term

$$\Delta w = -\varepsilon \delta E/\delta w(t) + \alpha(w(t) - w(t-1)) \ , \qquad (14)$$

where Δw is the weight change, w(t) is the weight value in the time t, w(t-1) is the weight value in t-1, and where ε and α are constants.

Several modifications are possible in each neural network. The first modification, which is not a pure TDNN, is to adapt the equivalent network without synchronizing weights in temporal images. This is more general than the algorithm described here in the sense of input space division, but the result is only the incomplete BP without the equivalence to the TDNN. The second modification which was implemented and experimentally used is the adaptive sigmoidal slope. This method, developed by Pelikán [10], adapts not only the weights and thresholds of the network, but also the shapes of the sigmoidal transfer function. This improvement shows faster convergence and is more robust in the cases where the classical approach fails. The third improvement is batching of the weights changes, i.e. calculation of the needs to change the weight in each adaptation step and adapting the network after several steps to the average value of all these partial contributions. This method was partially implemented by calculating the average value of the temporal images (see eq. 13).

4. Examples

The first task was designed to study the basic features of the TDNN and to compare the equivalent network with the classical BP. The goal was to detect any front edge in the one-dimensional input signal, which was presented to the input in 3 points. The input received sequences of numbers between 0 and 1 and the network was trained to detect those sequences containing the front edge.

The network used for this experiment was that one shown in Fig. 2. This is one of the most simple ones. A comparison was carried out between a TDNN and a full BP. These two networks are shown in Fig. 3.

Several trials were carried out to train both networks (i.e. TDNN and BP) being started from different random starting weights within the interval -0.3 .. +0.3. In all the cases the TDNN converged very rapidly to the solution. The most typical of them is shown on fig. 5. When being compared with the BP of the same size, the training of the TDNN to the same learning set required 1/2 to 1/3 of the iterations than the training of the BP, depending on the training set and initial conditions (the algorithm parameters and learning strategy were equal). The TDNN, however, shows less generalization properties than the BP. This case is very simple and so the counterexamples can be simply constructed using Fig. 5, taking into account the fact that in the TDNN case, input hyperplanes are parallel to some axes in the 3-dimensional input space (i.e. 3 input points).

This example is very simple but it illustrates how the TDNN works in comparison with the BP.

The second, more complicated task, was to distinguish the typical pattern in the 4-channel EEG signal. Alpha activity was chosen as the typical pattern, which is a very typical EEG component of sinusoidal shape with a frequency

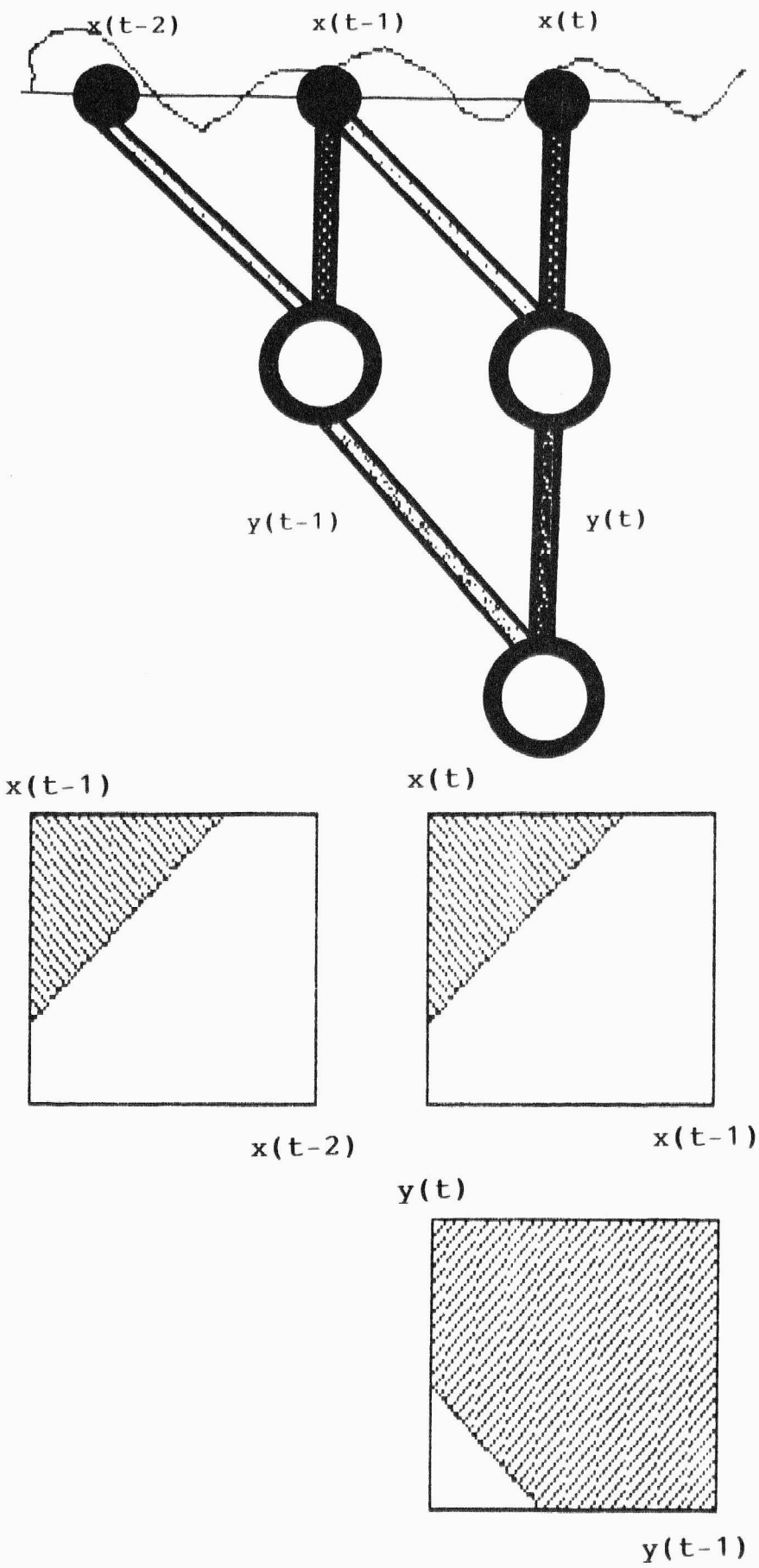


Fig. 5. The simple network from fig. 2. adapted to front edge detection. The neuron of the first layer - including its temporal image - converged to the equation $x_{t+1} > x_t + K$, which means that the output of the first layer neuron or its image is active only if the following signal is greater than the preceding one (front edge). The function of the neuron in the output layer is similar to OR, i.e. the front edge must be present in at least one of its preceding units.

of about 10 Hz and a relatively high amplitude. The goal was to distinguish this activity everywhere it occurs in any channel.

The network used had 4 input channels, each of which contained 5 delayed samples. The EEG was sampled with the frequency 50 Hz. The first layer of the network, connec-

ted to this input, had 3 neurons and the second (output) layer had one neuron with the binary input connected to 3 neurons of the previous layer and to 3 past samples of this layer.

The network was firs trained to the artificial signals, where the alpha activity were simulated by the sinusoidal curve and the non-alpha by noise. In the second step it was trained to the real EEG.

This simple network was able to distinguish 95% of the real alpha activity signals after training. Each part of the experiment required approximately 10 hours of adaptation on an IBM/PC/SX computer. In both steps, several learning strategies were used to reach a satisfactory final error. The most common of them was step-by-step training, where only a part of the whole training set is given to the input after randomly starting the initial weights. Then, the larger training set is used or another subset (part) is selected depending on the convergence features. This slightly complicated strategy prevents overlearning in the initial steps of adaptation and the change of the learning subset has an effect very similar to the simulated annealing.

When trying to adapt a corresponding BP network (i.e. the BP network with the same number of nodes) by similar means there was no success, even when re-starting the network and repeating the same learning strategy from different starting weights. The whole learning set did not converge while a part of it (10 input patterns) did. When switching this part with another one, it converged too, but the full learning set did not reach the satisfactory error value (function E) and there were some particular patterns which the model was unable to distinguish properly. There still exists the possibility that using a more complicated learning strategy (e.g. adaptive gradient step or very slow gradient step parameters) could result in a satisfactory solution. The main goal of this experiment, however, was to show that a simple TDNN network can be adapted to a task in which the BP network with the same number of nodes fails due to overlearning. The typical error curves of this experiment are shown on Fig. 6.

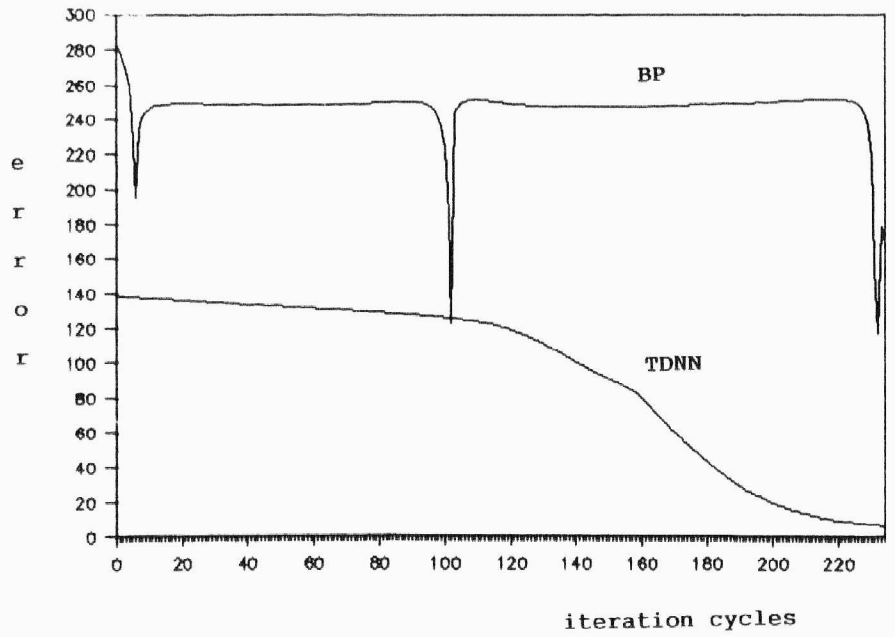


Fig. 6. A comparison between the TDNN and BP convergence. The BP had the same number of nodes like the TDNN (i.e. its equivalent network). The BP did not fully converge while the BP did. These convergence curves have been chosen from several trials and represent the typical situation.

5. Conclusions

The main advantage of this structure is the reduced number of weights in the temporally-oriented architecture. In comparison with the BP, the TDNN has less parameters to be adapted. On the other hand, the equivalent network has less possibilities to divide the input space than the corresponding BP network. TDNN performs the same operation in different time delays, i.e. the weights of the temporally-shifted images of the neurons remain the same, changing only the inputs. This fact causes two things: 1. The network is invariant in the temporal sense to the shift of any important phenomenon which it has been adapted for and 2. The network is insensitive to time sequences in the sequential point of view. Simple experiments show more rapid convergence of this network compared with the classical BP. When solving more complicated tasks and having larger structures TDNN is more resistive against overlearning than the BP. When comparing the generalization phenomenon, the TDNN has from the theoretical point of view less parameters and less abilities. However, when overlearning occurs, the TDNN is more advantageous because there are less parameters to be overlearned, its structure being also designed for the spatiotemporal tasks.

References

[1] Beran, H.: Physiological signal processing by the Back-Propagation neural network. Research report, Inst. of Physiological Regulations, Prague 1989.

[2] Beran, H.: Time-Delay neural network - a study. Proceedings of the international conference Neuronet 90, Prague 1990.

[3] Beran, H.: Physiological signal processing by means of layered neural network. PhD theses, Prague 1991.

[4] Grossberg, S.: The adaptive brain. North-Holland 1987, pp. 89-106.

[5] Hampshire, J.B., Waibel, A.H.: A novel objective function for improved phoneme recognition using time delay neural networks. Proc. III. Int. Conf. INNS, Washington, 1989, Vol. I, pp. 235-241.

[6] Hecht-Nielsen, R.: Nearest matched filter classification of spatiotemporal patterns. Applied optics, Vol. 26, No. 10, pp. 1892-1899. May 1987.

[7] Hecht-Nielsen, R.: Neurocomputing. Adison-Wesley, New York 1990.

[8] Lipmann, R.P.: An Introduction to Computing with Neural Nets. IEEE ASPP Magazine, pp. 4-22, April 1987.

[9] Pelikán, E., Beran, H.: The exploration of the back-propagation neural network in the field of physiological signal processing. International workshop Neurocomputers and Attention, Moscow, 18.-22.9.1989. Extended abstracts, Manchester univ. press, 1990.

[10] Pelikán, E.: The Back-Propagation neural network model with the adaptive transfer function. Research report, General Computing Center, Nr. V-404, Prague 1989.

[11] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature, Vol. 323, No. 9, pp. 533-536. October 1986.

[12] Sawai, H., Waibel, A., Haffner, P., Miyatake, M., Shikano, K.: Parallelism, hierarchy, scaling in time-delay neural networks for spotting Japanese phonemes/CV- syllables. Proc. III. Int. Conf. INNS, Washington, 1989, Vol. II, pp. 81-88.

[13] Waibel, A., Hazanawa, T., Hinton, G., Shikano, K., Lang, K.J.: Phoneme recognition using time-delay neural networks. IEEE ASSP, Vol. 37, No. 3, pp. 328-339. March 1989.

Book Alert

The following books can be interesting for the readers of our Journal

Decentralized AI. Proceedings of the First European Workshop on Modelling Autonomous Agents in a Multi-Agent World,Cambridge, UK, 16-18 August 1989. Edited by Y. Demazeau and J.-P.Mueller. Amsterdam, Elsevier Science Publishers 1990. 280 p. ISBN 0-444-88705-9. US $ 91.50. .

Much research in Artificial Intelligence deals with a single agent having complete control over the world. A variation of this is Distributed AI (DzAI), which is concerned with the collaborative solution of global problems by a distributed group of entities. This book deals with the activity of an autonomous agent in a multi-agent world.

Formal Techniques in Artificial Intelligence. A Source book. Edit. R.B.Banerji. (Studies in Computer Science and Artificial Intelligence, 6). Amsterdam, Elsevier Science Publishers 1990. 438 p. US $ 91.50. ISBN 0-444-88130-1.

Contrary to general opinion, Artificial Intelligence research has often been carried out from a mathematical point of view, and frequently incorporates techniques of theoretical computer science. This book surveys various areas of Artificial Intelligence research, describing formal techniques. The areas chosen are most those which have been or can be - discussed with mathematical precision and clarity.

Future Directions in Artificial Intelligence. IFIP TC 12 Collected Papers. Edited by P.A.Flach and R.A.Meersman. Amsterdam, Elsevier Science Publishers 1991. 190 p. US $ 80.00. ISBN 0-444-89048-3.

The contributions to this book are from Artificial Intelligence specialists from all over the world. Their ideas on the future of AI are original, thought-provoking and sometimes controversial.

Life, Brain and Consciousness. New Perceptions through Targeted Systems Analysis. G.Sommerhoff. (Advances in Psychology, 63). Amsterdam, Elsevier Science Publishers. 1990.336 p. US $ 128.50. ISBN 0-444-88436-X.

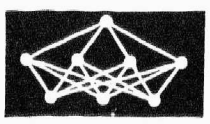
The relation between mind and brain can never be understood by science until the nature of consciousness and self-consciousness is clearly perceived as specific system-properties.

Mental Models and Human-Computer Interaction - 1. Edited by D.Ackermann and .J.Tauber. (Human Factors in Information Technology,3). Amsterdam, Elsevier Science Publishers. 1990. 280 p. US $ 91.50. ISBN 0-444-88705-9.

The important role of psychological research in the field of human-computer interaction (HCI) is becoming more and more recognized. The principles of how to design a user-oriented system cannot be worked out without the knowledge of how users work with systems.

Neural Networks. Advances and Applications. Edited by E.Gelenbe. Amsterdam, Elsevier Science Publishers. 1991. 274 p.US $ 94.50. ISBN 0-444-88533-1.

It is expected that Neural Networks will find their niche among the methods and techniques that computer scientists use for intrinsically difficult problems. An attraction of Neural Networks is the dialog established between computer science, biology, physics, psychology, numerical and non-linear analysis, and other areas.



Literature Survey

Kruschke J.K., Movellan R.: Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Back-Propagation Networks
Transaction on Systems, Man and Cybernetics Vol.21, 1991 No.1 pp.273-280

Abstract: The objective of the work is to explore the benefits of adaptive gains in back propagation networks.

Kung CH.-M., Hornik K.: Convergence of Learning Algorithms with Constant
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.484-489

Key words: neural networks; learning; algorithms; convergence.

Abstract: We investigate the behavior or neural network learning algorithms with a small, constant learning rate, in stationary, random input environments. It is rigorously established that the sequence of weight estimates can be approximated by a certain ordinary differential equation, in the sense of weak convergence of random processes as e tends to zero. As applications, back-propagation in feedforward architectures and some feature extraction algorithms are studied in more detail.

Kurkova V.: Kolmogorov's Theorem is Relevant
Neural Computation Vol.3, 1991 pp.615-620

Key words: Kolmogorov's theorem.

Abstract: We show that Kolmogorov's theorem on representations of continuous functions of n-variables by sums and superpositions of continuous functions of one variable is relevant in the context of neural networks. We give a version of this theorem with all of the one-variable functions approximated arbitrarily well by linear combinations of compositions of affine functions with some given sigmoidal function.We derive an upper estimate of the number of hidden units.

Leahy M.B., Johnson M.A., Rogers S.K.: Neural Network Payload Estimation for Adaptive Robot Control
IEEE Trans. On Neural Networks Vol.2, 1991 No.1 pp.93-100

Abstract: This paper proposes a new concept for utilizing artificial neural networks to enhance the high-speed tracking accuracy of robotic manipulators.

Lee B.W., Sheu B.J.: Hardware Annealing in Electronic Neural Networks
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.1 pp.134-137

Abstract: A novel hardware annealing in electronic neural circuits is derived from the analogy between the temperature in a Boltzmann machine and the apmlifier gain in a VLSI chip.

Lee B.W., Sheu B.J.: Modified Hopfield Neural Networks for Retrieving the Optimal Solution
IEEE Trans.on Neural Networks Vol.2, 1991 No.1 pp.137-142

Key words: Hopfield Networks.

Abstract: Due to the rugged energy function of the original Hopfield networks, the output is usually one local minimum in the enrgy function. This paper presents ana analysis on the locations of local minima in Hopfield networks and describes a modified network architecture to eliminate such local minima.

Merrill J.W.L., Port R.F.: Fractally Configured Neural Networks
Neural Networks Vol.4, 1991 No.1 pp.53-60

Key words: learning evolution; innate knowledge.

Abstract: A model is described for one method of instantiating constraints in neural networks, such as are required to account for nativist assertions in an associationist context. An implementation, inspired by some of the processes of biological development, is presented for the construction of networks with such constraints.

Michel A.N., Si J., Yen G.: Analysis and Synthesis of Class of Discrete-Time Neural Networks Described on Hypercubes
IEEE Transactions on Neural Networks Vol.2, 1991 No.1 pp.32-46

Key words: neural networks - discrete-time.

Abstract: The paper presents a qualitative analysis for a class of synchronous discrete-time neural networks defined on hypercubes in the state space. Next, the authors utilize these analysis results to establish a design procedure for associative memories to be implemented by the present class of neural networks.

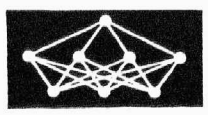
Miller I, Roysam B., Smith K.R., O'Sullivan J.A.: Representing and Computing Regular Languages on Massively Parallel Networks
IEEEE Transactions On Neural Networks Vol.2, 1991 No.2 pp.56-71

Abstract: A general method in proposed for incorporating rule-based constraints corresponding to regular languages into stochastic inference problems,thereby allowing for a unified representation of stochastic and syntactic pattern constraints.

Miyake S., Kanaya F.: A Neural Network Approach to a Bayesian Statistical Decision Problem
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.538-540

Key words: neural networks; Bayesian statistical decision.

Abstract: This letter proposes generalized mean-squared error (GMSE) objective functions that can be used in neural networks to yield a Bayes optimal solution to a statistical decision problem characterized by a generic loss function.



THE NEURONLIKE NETWORK FOR BRIGHTNESS PICTURE SEGMENTATION

A.D. Goltsev[1]

Abstract:

The algorithm of brightness picture segmentation is described. The algorithm is realized as a computer model of a neural network. The results of the model's runs are presented in photographs.

Keywords: neuronlike elements, contour (edge), brightness value, excitatory connections.

1. Introduction

Nowadays, the problem of picture segmentation is widely recognized as the most challenging task in efforts to create machine vision for robots [1]. Various approaches have been proposed [2-5] for picture segmentation (splitting an image into domains differing from one another by their visual characteristics). Partial image segmentation is performed by singling out homogeneous brightness domains within a picture, each of which can be defined as a local-contrast formation whose internal brightness heterogeneity is considerably lower than its distinction from the surrounding part of the picture. The term brightness segmentation is used to designate the process of partition of spots of uniform brightness in the picture. The method of brightness segmentation can be subdivided into two main steps:

1. revealing of the internal points of homogeneity domains by one or another variant of comparing the threshold value to each element of the matrix of the initial pattern brightness [4], and

2. tracing of the domain boundaries in order to markthem [5].

Neural networks of a visual analyzer are at present the only devices as far as we know, capable of complex visual pattern recognition. This fact testifies to the plausibility of the assumption that neuronlike networks represent the most adequate mechanisms for the visual recognition construction of algorithms for visual data processing in the form of organization of interaction among different parts of a structured neuronlike network. Naturally, this is also true for algorithms of brightness analysis, even to a greater extent than for some other functional subsystems in the pattern recognition problem. The fact is that the brightness analysis should undoubtedly be performed at the very first stage of visual data processing. In the process of neurophysiological investigation the detailed and valid data were obtained concerned with organization of just those neural networks that are immediately adjoining the light sensitive receptors of an eye of various living organisms [6]. In this way there is a possibility to make use of certain principles of structural organization of a visual analyzer when constructing artificial neuronlike networks intended for solution of the problem of preliminary image processing.

[1]Goltsev A.D.
Institute of Cybernetics, Kiev, USSR

2. General features of the model

The neuronlike network represented by the model considered below has a constant structure of connections that do not undergo any modifications during the processing of visual information. To say it in other words, any process of learning is absent in the network.

The model of the neuronlike network considered in this paper, which is intended for preliminary brightness segmentation of a picture, is designed as a computer program. For the sake of the better understanding of the model's functioning, meaningful description of the mechanisms of its operation is accompanied by the formal description of computation procedures through which the model is realized. In this paper the programmed simulation of a neuronlike network is organized on the basis of averaging (defocussing) by the square with a side n. Let (x_{ij}) be a matrix with a row containing p elements and a column containing m elements: i $= 1, 2, 3, \ldots, p$; $j = 1, 2, 3, \ldots, m$. Note, that all matrices introduced below are of one and the same dimension p x m. The value of ij-th element of the matrix obtained as a result of averaging by the square with a side n of a matrix $(x_i j)$ will be denoted by $f_{ij}^n(x)$. The averaging operation will be defined by the formula

$$1 = +\frac{n-1}{2} \quad k = +\frac{n-1}{2}$$

$$f_{ij}^n(x) = \sum \sum x_{i+1,j+k} \,, \qquad (1)$$

$$1 = -\frac{n-1}{2} \quad k = -\frac{n-1}{2}$$

$n = 3, 5, 7, \ldots, 31$

To avoid unnecessary complication in the description of computation procedures with quite unessential details associated with the effects occurring when the averaging square falls over the edges of a matrix, all formulas are written on assumption that the following inequalities are valid : $1 \leq i+k \leq p$; $1 \leq j+l \leq m$.

It should be mentioned that, at present, most of the time involved in completing an averaging operation is taken up by network counting. There is no difficulty in constructing a specialized high-speed device to carry out the procedure of defocussing an matrix of an assigned dimension.

3. Preliminary processing of the brightness picture

The model starts its work with two successive transformations of the initial picture. The point is that the initial picture can have an excessive or an insufficient light level. The purpose of this transformations consists in obtaining a picture that is optimal for contours (edges) extraction.

To compensate for defective illumination of the initial picture the model first of all reduces an interval of the brightness values to the fixed range. Let the initial picture be designated by (I_{ij}). The initial picture is defocussed. In the defocussed picture we seek the maximum brightness value M'

$$M' = \max_{i,j} f_{ij}^{n}(I) \; . \tag{2}$$

Let us designate by M the maximum possible brightness value in the picture. Then, this transformation may be described by

$$I'_{ij} = \frac{M}{M'} I_{ij} \; . \tag{3}$$

By means of this procedure the maximum if the brightness values of the matrix (I'_{ij}) becomes not less then M. All another details of the picture proportionally change their brightness.

The second operation for image transformation is based on the assumption that the weak brightness heterogeneities situated within the areas with rather high or low illumination level must not be marked out as contours. In the model, special elements with sigmoid output characteristics are used to blot out the aforesaid heterogeneities. Let us designate by S the procedure performed by each of these elements. Thus the described operation may be expressed as

$$I''_{ij} = S(I'_{ij}) \; . \tag{4}$$

The picture transformed by the two operations described above is considered normalized. The contours are extracted from this picture.

In the implemented computer model the contours were marked out by the subtraction of one defocussed picture from another. The degree of defocussing of these pictures is different: it is made by n_1 x n_1 and n_2 x n_2 squares.

Then, the operation of marking out contours in the model is

$$c_{ij} = 1(f_{ij}^{n_1}(I'') - f_{ij}^{n_2}(I'') - 0) \tag{5}$$

where (c_{ij}) - is the resulting contour points matrix; 0 - value of the threshold; $n_1 > n_2$; $1(x)$ - the unity step-function [7]:

$$1(x) = \begin{cases} 1, & for \;\; x' \;\;) \;\; 0 \\ 0, & for \;\; x \leq 0 \end{cases}$$

From the foregoing it follows that the marking of the contours in the model is performed by a boundary integro-differential operator similar to the ∇ G - filter suggested by Marr [8].

4. Description of the functioning of the basic part of the model

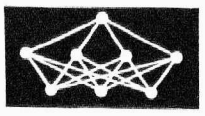
The base of the model is represented by a layer of neuronlike elements associated with one another by the regular structure of excitatory connections. Let us call this layer the layer with close connections. Mutual excitatory connections with the same weight R connect each neuronlike element (not a marginal one) of a layer with its neighbours situated within the limits of the square; the centre of this square is the neuronlike element being considered. The size of the square of excitatory connections is n , and the weight of excitatory connections is R . Due to the structure of the connections, the neuronlike elements of the layer situated apart from each other at distances less than $(n_B + 1)/2$ are mutually growing in their activity. The elements situated still further from one another are not directly interacting. During the calculation, the values of n_B and R_B are constant.

The dynamics of the output potential $P_{ij}(t)$ of the ij-th neuronlike element of the layer are described with equation

$$dP_{ij}(t) = \frac{1}{\tau}(\lambda E_{ij}(t) - P_{ij}(t)dt \tag{6}$$

where λ - is static amplification coefficient of the neuronlike element; τ - is the time constant; $E_{ij}(t)$ - is the summarized input of the ij-th neuronlike element.

The contours marked out in the previous stage of pattern analysis are used in the model for exerting an inhibitory influence over the layer of neuronlike elements with close connections. This may be structurally represented by inhibitory connections with weight r, which transfer inhibitory action from the matrix (c_{ij}) to the inputs of their respective neuronlike elements in the layer with close connections. The excitatory action of the brightness matrix of the normalized picture (I''_{ij}) is also fed to the layer with close connections. Stimulating influence proportional to the brightness in the element of the matrix (I''_{ij}) is admitted to the input of the respective neuronlike element of the layer with close connections through a^n excitatory



connection with weight ρ . In this way and in accordance with the above description, the total action at the input of the ij-th neuronlike element of the layer with close connections is calculated at an arbitrary moment of time t by the formula

$$E_{ij}(t) = \rho I''ij - rc_{ij} + R_B n_B^2 f_{ij}^{n_B}(P(t)) \; . \qquad (7)$$

Structure of the model is shown in the Fig.1.

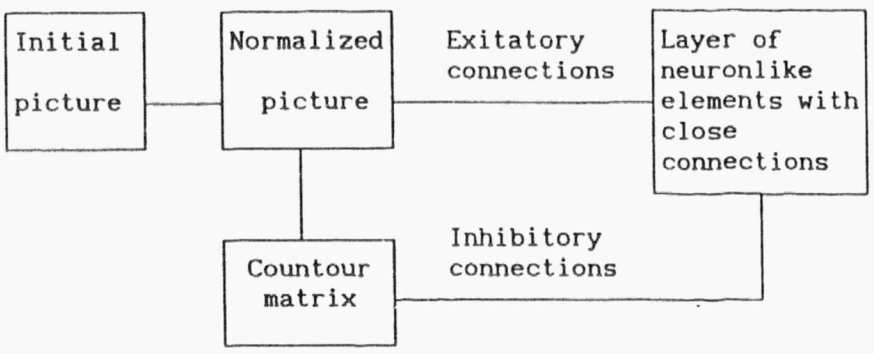


Fig.1

As a consequence of the mixed excitatory-inhibitory external influence on neuronlike elements in the layer of close connections and interaction between its elements, the total level of activity starts growing in the layer. As this takes place, the activity is distributed in the layer in a nonuniform way; the highest level of activity is attained in those parts of the layer where the image has the most bright, compact, contourless regions of comparatively large size and convex shape. The activation level of small, poorly illuminated and multicontour regions must be considerably lower. Some time after the neuronlike network is acquainted with a picture, the marking out of the desired domains is performed by a horizontal shear of activity relief in the layer with close connections. Thus, due to the combination of inhibitory and excitatory effects on the layer of closely-linked neuronlike elements and interaction among them, some quantity of compact parts corresponding to the most bright, large, contourless domains in the initial picture are marked in the layer.

To simplify the neuronlike network computations, $\lambda = \tau$ is assumed. Then Eq. (6) converts to

$$dP_{ij}(t) = E_{ij}(t)dt - \frac{P_{ij}(t)}{\tau}dt \; . \qquad (8)$$

For numeric solution of a system of p x m equations (8), discrete time t is introduced with the time interval Δt. As a result, from Eq. (8) we obtain

$$P_{ij}^t = \frac{\tau - \Delta t}{\tau}(P_{ij}^{t-1} + E_{ij}^{t-1}\Delta t) \; . \qquad (9)$$

Substitution of (7) into (9) gives the expression for the sequential recomputation of output potentials of the neuronlike elements of the layer with close connections (elements of matrix (P_{ij}^t))

$$P_{ij}^t = \frac{\tau - \Delta_t}{\tau}\{P_{ij}^{t-1} + \Delta t(\rho I_{ij}'' - rc_{ij} + R_B n_B^2 f_{ij}^{n_B}(P^{t-1})\} \; . \qquad (10)$$

Discriminated by threshold, vertices of the relief of activity in the layer of closely-linked neuronlike elements are represented by the units in the binary matrix (a_{ij}^t) according to the formula

$$a_{ij}^t = 1(P_{ij}^t - L) \qquad (11)$$

where L - is the constant threshold.

Recomputation of the network stops after a fixed number of program steps at time $t*$.

The function of excitatory relations among neuronlike elements in the layer with close connections consists (as seen in the foregoing description of the network's operation of the joint activation of compact groups of elements in the layer with close connections which are relatively strongly stimulated from outside. The role of inhibitory influence over the elements of the layer with close connections is to restrict the activity propagation in the layer and to prevent the merging of activated domains into a single domain. To put it otherwise, the effect of inhibitory influence of contours over the layer of the close connections permits the "pupation" of brightness spots in the picture.

It is necessary to note that the tuning of the model (including the picking out the values of n_B, R_B, r, ρ, $t*$) is performed using the criterion of the optimal marking out of the convex brightness domain of the assigned size. All parameters of the model mentioned above are constant during the computations. The marking out of domain with much smaller size requires retuning of the model.

For the computer input of pictures, the control of intermediary results, and the output of the results of model functioning, use was made of a system containing a video camera, 128 x 128 6-bit words of frame memory, computer, and monitor DIS. With the aid of this system the brightness analysis algorithm was verified using pictures of natural objects. Photographs taken from the monitor are shown in Figs. 2,3,4: the initial picture is placed in the

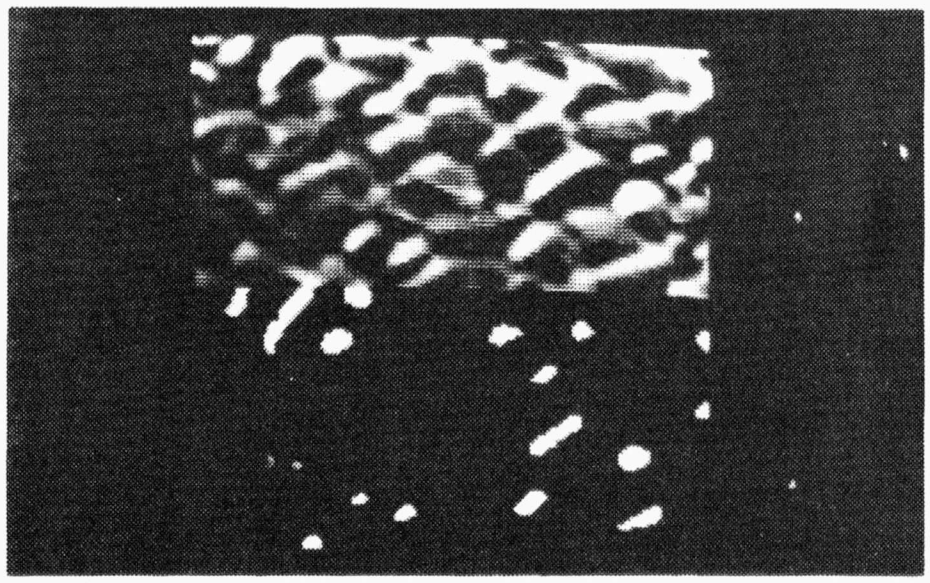


Fig.2

upper semi-frame, the domains marked out by the program are placed in the lower semi-frame. Fig.2 depicts a picture of a stone surface. Fig.3 shows a path in the grass. Fig.4 shows patterns of two textures.

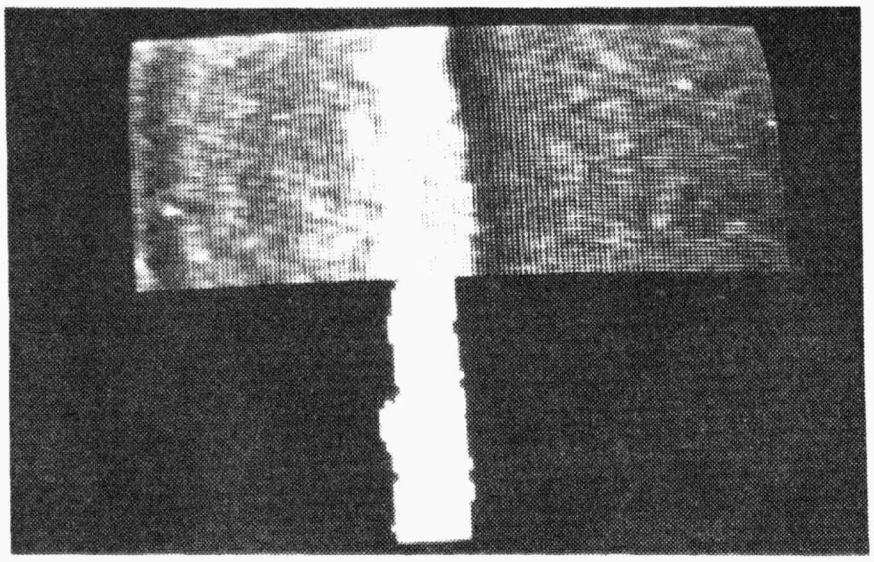


Fig.3

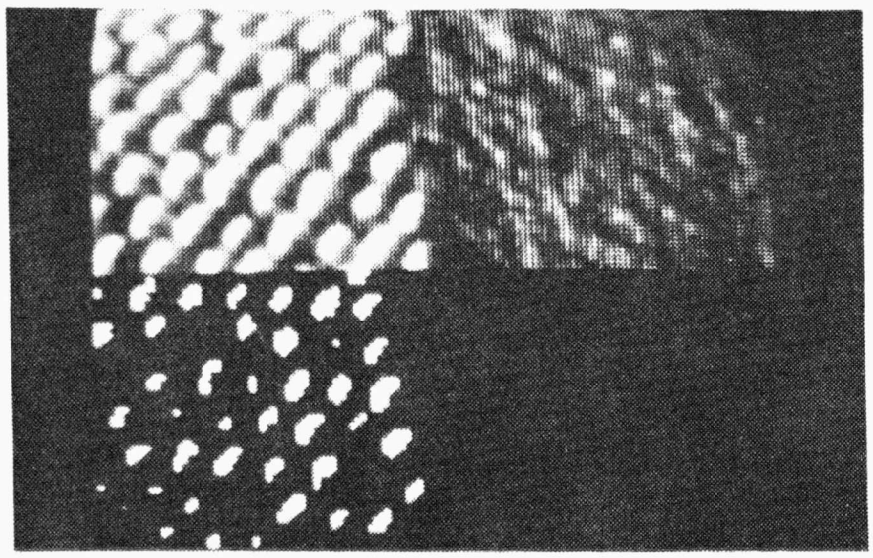


Fig.4

5. Algorithm to successively specify the shapes of brightness spots as they are found

According to the above description, the brightness analysis permits a parallel process to mark out all the brightest and most continuous spots in the picture. However, if the model parameters are chosen correctly, then each domain marked out by a network resides mostly within the actual boundaries of a spot. And here, the lower the brightness of a spot is, the greater is the difference between marked out domains both in area and in shape. Respectively, the greater the brightness of a spot is, the closer the domain marked by a network is to the actual shape of the spot. From the latter statement in follows that the network acts most accurately when it singles out the shape of the brightest spot in the picture. The purpose of brightness picture segmentation is to outline the boundaries of all uniformly illuminated domains of a picture. This suggests that if we artificially transform the initial picture in such a way that the brightness of the chosen spot changes and attains the maximum possible level, and then perform the brightness analysis on this picture, the shape of this spot would be consequently marked out much more accurately than in the case when the preliminary brightness correction were absent. This operation can be successively applied to each of the uniformly illuminated domains of the picture and in this way it solves the problem of comparatively precise outlining of the boundaries of all uniformly illuminated picture domains. So, in a brief formulation, the sense of the algorithm described is in the successive attracting of attention to all uniformly illuminated domains in the picture.

The algorithm for brightness spot shape outlining consists of some cycles whose quantity equals the number of uniformly illuminated domains found in the picture. Each cycle represents the process of brightness analysis of the picture treated in detail above, to which only minor alterations were made.

According to description of the brightness analysis algorithm, the initial picture is transformed by the model with the aim of creating the optimal conditions for its further processing. Then contours are marked out in the picture. Inhibitory-excitatory action from the contours and the transformed picture is sent to the layer of neuronlike elements with close connections. Activation of neuronlike elements of the layer begins. However, as soon as the output potentials of the first neuronlike elements of the layer attain threshold (L), the course of brightness analysis procedure is disturbed. The search for the most stimulated neuronlike elements in the layer with close connections is performed. The coordinates of this element are memorized by the special binary matrix (b_{ij}) where only one element equals unity and all the rest are zeros. The functional sense of the given operation is that the corresponding raster pixel is used henceforth as a domain identifier over which the model's attention is concentrated at the current cycle of algorithm operation. This makes it possible to approximately estimate the brightness of the considered spot. Let us determine the value (designated by M''') by the formula

$$M''' = \sum_{i=1}^{p} \sum_{j=1}^{m} b_{ij} f_{ij}^{n}(I) \,. \tag{12}$$

The formula (12) means that brightness of a domain is evaluated by the value of defocussed image illumination in the singled out raster pixel.

Then, the initial picture is transformed according to the formula

$$I_{ij}''' = \frac{M}{M'''} I_{ij} \,. \tag{13}$$

The transformation described by formula (13) converts the initial picture in such a way that the brightness estimate of the examined spot approaches, but does not surpass the value of M.

After the picture is transformed according to formula (13) it is considered by the model as the initial picture and the whole process of brightness analysis starts all over again, and is carried out to the full extent and this time it

is brought up to the end, i.e., to singling out of several domains in matrix $(a_{ij}^{t^*})$. So far, as matrix $(I_{ij}^{\prime\prime\prime})$ in the general case differs considerably from the initial pattern, the form of certain domains marked out in $(a_{ij}^{t^*})$ may be partially incongruous with the initial brightness spot. However, the form of the domain, for the sake of which this cycle of analysis is performed, appears to be marked out in a more precise way than in the case of the initial picture analysis.

It is necessary to introduce the operation of the single expansion of the binary matrix (x_{ij}). This matrix describes an algorithm for detection of the desired domain in matrix $(a_{ij}^{t^*})$. Let the matrix contain an arbitrary number of zeros and unities. Each unit element not situated on the edge of the matrix has 8 neighbours. The operation consists in conferring unity values to all 8 neighbours of each unit element of a matrix. Denote the operation of expansion of an arbitrary binary matrix (x_{ij}) by $\Phi(x_{ij})$.

By (u_{ij}^1), (u_{ij}^2), (u_{ij}^3),..., (u_{ij}^{φ}) and (s_{ij}^1), (s_{ij}^2), (s_{ij}^3),..., (s_{ij}^{φ}) we shall denote the sequences of intermediary matrices calculated in the process of the singling out among the plurality of domains contained in the matrix $(a_{ij}^{t^*})$, of that domain upon which the model's attention is focussed in the given cycle.

The above sequences are calculated by these formulas:

$$\begin{aligned}
(u_{ij}^1) &= \Phi(b_{ij}) \\
s_{ij}^1 &= u_{ij}^1 \ \& \ a_{ij}^{t^*} \\
(u_{ij}^2) &= \Phi(s_{ij}^1) \\
s_{ij}^2 &= u_{ij}^2 \ \& \ a_{ij}^{t^*} \\
(u_{ij}^3) &= \Phi(s_{ij}^2) \\
s_{ij}^3 &= u_{ij}^3 \ \& \ a_{ij}^{t^*} \\
&\dots \\
(u_{ij}^{\varphi}) &= \Phi(s_{ij}^{\varphi-1}) \\
s_{ij}^{\varphi} &= u_{ij}^{\varphi} \ \& \ a_{ij}^{t^*}.
\end{aligned} \tag{14}$$

The matrix (s_{ij}^{φ}) is determined from the condition

$$\sum_{i=1}^{p} \sum_{j=1}^{m} (s_{ij}^{\varphi} - s_{ij}^{\varphi-1}) = 0. \tag{15}$$

The desired raster domain is fixed in this matrix in the form of the unity values of its elements.

The results of the algorithms functioning (with a view to their further utilization) are transferred to the special binary matrix (z_{ij}). All elements of this matrix have zero values at the initial moment of time. Let us designate the sequence of cycles of picture analysis by a number of indexes $q = 1, 2, 3, \dots, v$. Then, in the q-th cycle of analysis, matrix (z_{ij}^q) is defined by the formula

$$z_{ij}^q = z_{ij}^{q-i} \vee s_{ij}^{\varphi} \ . \tag{16}$$

The next cycle of analysis starts after the marked out domain had been transferred to matrix (z_{ij}^q). The model "concentrates attention" on the new brightness spot in the picture.

Fig.5 illustrates the above process. The initial picture is at the top of Fig.5. The second semiframe at the top demonstrates the domains marked out in the picture during the single brightness analysis procedure; the figure at the bottom shows the same domains marked out as a result of multiple application of the mentioned procedure according to the algorithm described here.

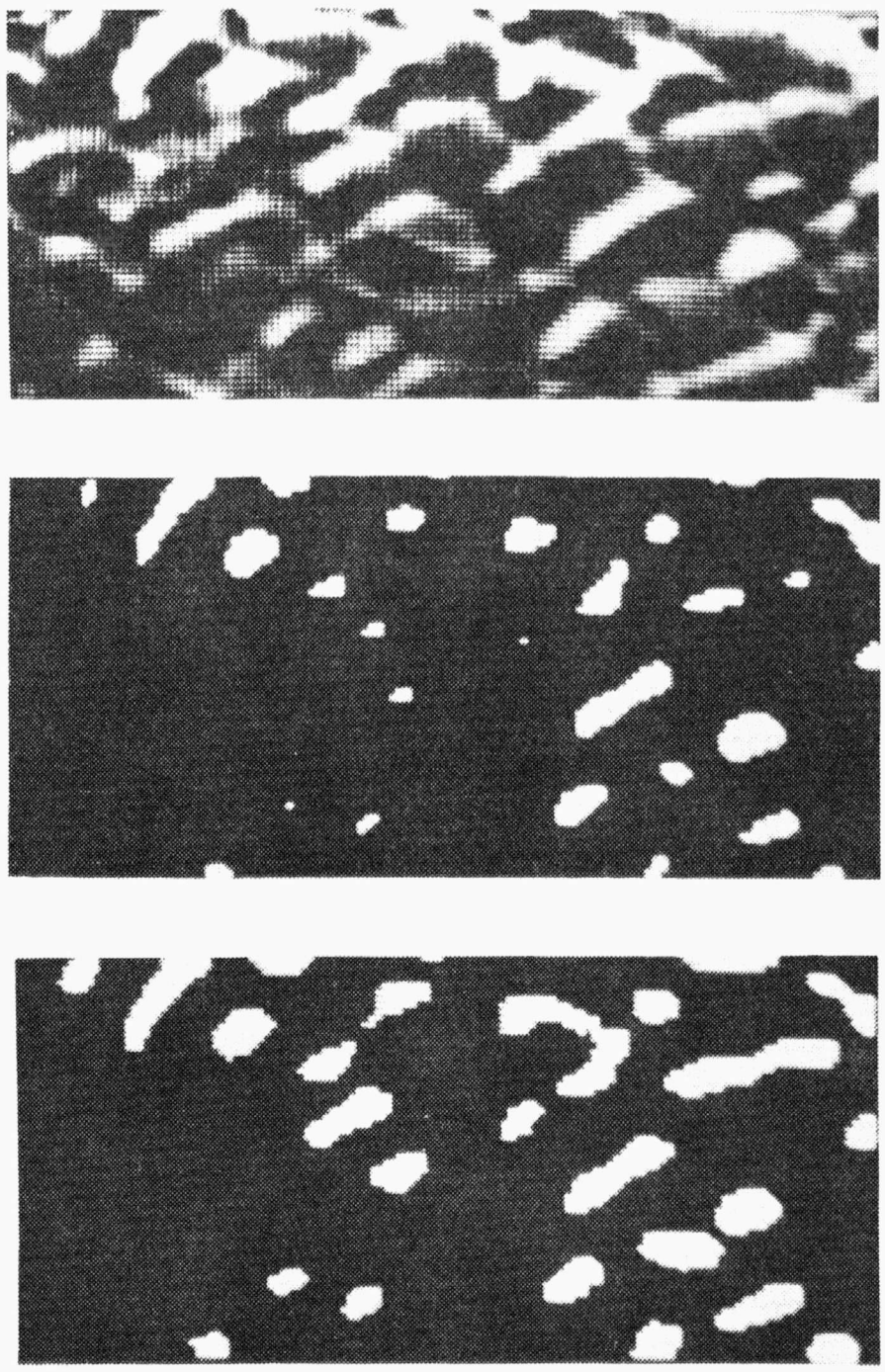


Fig.5

In conclusion it will be noted that the results of application of the suggested algorithm to a picture are to be considered as preliminary. In future they are supposed to be used as a learning sample for solution of the problem of precise definition of the shapes of brightness spots within a picture [9,10].

Literature Survey

Leung H., Haykin S.: The Complex Backpropagation Algorithm
IEEE Transactions on Signal Processing Vol.39, 1991 No.9 pp.2101-2104, ISSN 1053-587X

Key words: backpropagation; algorithm.

Abstract: The backpropagation algorithm provides a popular method for the design of multilayer neural networks. This correspondence generalizes the backpropagation algorithm to include complex coefficients and complex signals.

Lin Ch.-S., Kim H.: CMAC-Based Adaptive Critic Self-Learning Control
IEEE Transactions on Neural Networks Vol.2, 1991 No.5 pp.530-533

Key words: neural networks; self-learning; control.

Abstract: This article presents a technique that integrates the cerebellar model articulation controller (CMAC) into a self-learning control scheme developed by Barto et. al.

Lin W.Ch., Liao F.Y., Tsao Ch.K., Lingutla T.: A Hierarchical Multiple-View Approach to Three-Dimensional Object Recognition
IEEE Transactions On Neural Networks Vol.2, 1991 No.1 pp. 84-92

Key words: recognition systems.

Abstract: This paper proposes a hierarchical approach to solving the surface and vertex correspondence problems in multiple-view-based three-dimensional object recognition systems.

Livesey M.: Clamping in Boltzman Machines
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp.143-148

Key words: machine learning procedure.

Abstract: This paper investigates a certain assumption that appears in the proof of correctness of the standard Boltzmann machine learning procedure.

Malki H.A., Moghaddamjoo A.: Using the Karhunen-Loe've Trasformation in the Back-Propagation Training Algorithm
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp.162-164

Key words: Training Algorithm.

Abstract: A new training approach based on the back-propagation algorithm is introduced. In the proposed approach, initially, a set of training vectors is obtained by appluing the Karhunen-Loe've transform on the training patterns.

Meador J.L., Wu A., Cole C., Nintunze N., Chintrkulchai P.: Programmable Impulse Neural Circuits
IEEE Trans. On Neural Networks Vol.2, 1991 No.1 pp.101-109

Key words: artificial neural network.

Abstract: This paper describes CMOS electronic circuits, which emulate natural neurons at a more detailed level than that typically employed by artificial neural network models.

Morgan D.P., Scofield Ch.L.: Neural Networks and Speech Processing
"Bridging the gap between Neural Nets and Speech Processing" Hingham, USA, Kluwer Academic Publishers, 1991, 416 p.ISBN 0-7923-9144-6

Key words: neural networks; speech processing.

Abstract: Neural Networks and Speech Processing discusses artificial neural networks as well as biological neural networks in the auditory system. The focus of this text is automatic speech recognition (ASR), particularly large-vocabulary continuous speech recognition (CSR). This text bridges the gap between two disciplines - neural science and speech recognition. It will serve as a handbook which describes biological auditory processing, conventional speech recognition techniques, and current ANN paradigms applied to speech recognition. The book may also be used as a text for an advanced course on this topic.

Perfetti R.: A Neural Network to Design Neural Networks
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.9 pp. 1099-1103

Key words: neural networks.

Abstract: The design of the Hopfield's associative memory is reformulated here in terms of a constraint satisfaction problem and an electronic "neural" net, which is capable of solving this problem in real time, is proposed. Circuit solutions correspond to symmetrical zero-diagonal matrices that possess few spurious stable states.

Perlovsky L.I., McManus M.M.: Maximum Likelihood Neural Networks for Sensor Fusion and Adaptive Classification
Neural Networks Vol.4, 1991 No.1 pp.89-102

Key words: maximum likelihood; adaptive efficiency; cramer-rao bounds; sensor fusion; adaptive classification; phase transitions; attetion.

Abstract: A maximum likelihood artificial neural system (MLANS) has been designed for problems which require an adaptive estimation of metrics in classification spaces. Examples of such problems are an XOR problem and most classification problems with multiple classes having complicated classifier boundaries.

Salam F.M.A., Bai S.: A New Feedback Neural Network with Supervised Learning
Neural Networks Vol.2, 1991 No.1 pp.170-175

Key words: feedback neural network.

Abstract: We introduce a model for continuous-time dynamic feedback neural networks with supervised learning ability.

ON UNIFORM TRAINING

D. Fatton[1]

1. Introduction

During the training phase of an artificial neural network with the backpropagation method, it may happen that the algorithm converges to a local minimum. To avoid this, we thought of new strategies concerning the manner of presenting the patterns (to be learnt) to the network.

At any given time, the idea is to present the "worst pattern" where worst means that the difference between the desired and the effective output is the biggest when compared with the differences of outputs from other patterns.

The term "uniform training" comes from the fact that the expected result, if you present the "worst pattern" to the network at each training step, is that the differences (errors) between desired and effective outputs for the set of patterns will uniformly decrease, avoiding the observed facts during a convergence to a local minimum, which are that every pattern tends to be well learned (their errors tend to zero), except for one or two patterns which have much higher error than the average. In classification tasks, one bit is reversed causing the error to be one.

In the following paragraphs, we describe the considered problem to study this strategy and the used algorithms. Finally the results and comparisons are presented.

2. The Problem Considered

The problem considered is one of auto-association. The network used is 4-2-4 (4 inputs, 2 neurons in the hidden layer and 4 outputs). The desired outputs of the patterns are the same as their inputs and they have binary values. The maximum number of patterns which can be learned by such a network is eleven [1] but one must be careful when choosing them.

There are sets of eleven patterns which cannot be learned due to the fact that it is impossible to represent them by suitable hyperplanes partition (this was demonstrated by manual drawing of a graphic representation). The four outputs of the network represent four lines in the plane R^2. One line is described by the weights and the threshold corresponding to an output neuron. In the plane the convex subspaces arising from the crossing of these four lines represent the patterns. By way of example, the pattern 1-0-1-0 means that its corresponding subspace is in the strictly positive side of the first and third lines while in the negative side of the second and fourth lines. Here are three sets of eleven patterns. The first two are used in this study, while the third is impossible:

1101	1001	0001
1111	1011	0010
1011	1111	0100
1001	0011	0111
1000	0111	1000
1010	1110	1001
0010	0110	1111
0000	0010	1100
0001	0100	0011
0101	0000	1010
0100	0001	0101
set A	*set B*	*trap set*

3. The Algorithms Used

The basic algorithm is standard backpropagation with momentum strategy as proposed by Rumelhart, Hinton and Williams in 1986:

$$\xi_i = \sum_j \omega_{ij} x_j + \vartheta_j$$

$$x_i = S(\xi_i)$$

$$S(\xi_i) = \frac{1}{1 + \exp(= \lambda \xi_i)}$$

$$E = \frac{1}{2} \sum_j (d_j - y_j)^2$$

where:
- ξ_i is the potential of a neuron i.
- ω_{ij} is the weight on the connection between the neuron j and the neuron i.
- ϑ_i is the threshold of the neuron i.
- x_i is the output of the neuron i, while
- y_i is the output of the neuron i which is on the output layer.

[1] D.Fatton
Institute d'Informatique, University of Neuchâtel,
Chantemerle 20, Switzerland

S is the sigmoid with the parameter λ.
E is the error for one given pattern.
d_j is the desired output of the neuron j for a given pattern.

Finally the correction of the weights is the following:

$$B\omega_{ij}^n = -\eta \frac{\partial E}{\partial \omega_{ij}^n} + \alpha B\omega_{ij}^{n-1}$$

where we have two parameters:
η, the step size or learning rate.
α, the momentum parameter.

To this basic algorithm, we added the superSAB strategy [2], which consists of having one parameter η for every weight and adapting these parameters in the course of training. It can be summarized in this way: while the gradient does not change sign, η is exponentially increased. On the other hand, if the gradient changes sign, the weight is not corrected and η is exponentially decreased faster than the increase.

The next extension of the algorithm comes from PAB [2]. Each neuron has its own λ (parameter of sigmoid) and these parameters are corrected with the same idea as the weight's correction.

Finally we used the SuperSAB strategy for the correction of the lambdas too.

4. Results and Comparisons

The way the different results were compared is the following. We used two parameters. The first is called the "Global Error" (GE) and the second is called the "Number of Training steps" (NT). GE is the sum of the quadratic differences between desired and effective outputs for all patterns. NT is the number of times that the network was adapted by presenting a pattern to it.

We have analyzed also whether the problem was solved or not. This means that we tested every pattern by presenting it to the network. Then pattern is considered as learned if the difference between effective and desired output is less than 0.5 for every neuron on the output layer (this is equivalent to the use of hard nonlinearity on the output layer of the network).

Furthermore, two ways of presenting the patterns to the network were used during the learning phase. The first way is called "classic" and consisted of choosing each pattern after eachother (cycle) and each pattern is presented a certain number of times (iterations). Here the iteration was 15.

The second way was the goal of this study and is called "uniform". The idea of this "uniform" training was to always present the worst pattern to the network, but with this strategy: the network never converged. So "uniform" training means presenting the worst pattern to the network UNTIL the error for this pattern gets below a given limit.

Error is the quadratic sum of the difference of each couple's "desired-effective output". Here is an example of "uniform" training. The limit is first 0.5, the network is trained by presenting the worst pattern to it until it returns an error below the limit. Afterwards, we present the new worst pattern until its error is under the limit value (the error of the previous pattern is perhaps again bigger than the limit), and so on. The limit is decreased to 0.4 when all patterns have their errors below the limit of 0.5. The training continues by decreasing the limit to final value 0.1 and can be stopped when the 11 patterns are correctly learned.

[ ] Standard backpropagation with momentum:

With this simple algorithm, the "classic" training met an apparent local minimum after about 20000 training steps, the GE was 2.02. After 3300 training steps, GE was the 4.83. The GE did not decrease very fast to 2.

The "uniform" training was better at the beginning (GE=2.67 with NT = ~2000) but it seemed very difficult to solve the problem.

The conclusion is that the problem was solved with neither technique.

[ ] SuperSAB was added:

Again "classic" training met a local minimum and there were not many changes from the previous algorithm except that the local minimum ended with GE = 1.

With "uniform" training, GE decreased again quite rapidly at the beginning (GE = 2.69 for NT = 4575) but again it seemed very difficult to solve the problem afterwards.

[ ] Every neuron had its own lambda and these lambdas were adapted:

"Classic" training converged again to a local minimum (GE = 1).

"Uniform" training now was able to solve the problem. After 9896 training steps, GE = 1.8302 and the 11 patterns are well learnt. To decrease the GE to under 1, however, it was very difficult: it required 30 000 steps to have GE = 0.9858.

[ ] SuperSAB was also added to modify the parameters (lambdas) :

The "classic" training is again locked in a local minimum (GE = 2). However, GE decreased more rapidly than with other algorithms.

The "uniform" training is this time very good compared to other variants because after NT = 2917, GE = 2.12 and the problem was solved (in the sense that if hard nonlinearity is used, all patterns are correctly learned). Furthermore, continuing learning process, after NT = 7389, GE = 0.71 and the solution could still be improved.

All the experiments mentioned were performed with the patterns of set A. With set B, the results are in the same line. With "uniform" training, the GE did not decrease monotonously. GE tended to grow again every time the error limit was decreased.

The conclusion is that the adaptation of each parameter seems to be important during the learning phase but the way of presenting the patterns seems to have the same im-

portance. Effectively, the problem could only be solved by using the "uniform" strategy and the basic algorithm of backpropagation extended with SuperSAB and PAB techniques.

Intuitively, I am convinced that this "uniform" training can be generalized to any kind of training with backpropagation technique. The reason is that this strategy does not allow a pattern to stay apart, with a neuron having a large potential far from the threshold of the sigmoid, and being more and more difficult to adapt.

References

[1] O.Kufudaki, J.Horejs, "PAB: Parameter Adapting Backpropagation", Neural Network World (IDG Co. Czechoslovakia), (this issue).

[2] Tom Tollenaere, University of Edinburgh, "SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties", Neural Networks, vol. 3, pp. 561-573, 1990.

Interesting and Coming Events

In this section of our Journal the information on some interesting and coming conferences, symposiums and seminaries are given:

OCTOBER 1991

EPIA-91 5th Portuguese Conf on AI, Oct.1-3, 1991, Albufeira, Algarve PT. Info: Prof. Pedro Barahona, Dep de Informtica, Univ Nova de Lisboa, 2825 Monte da Caparica, Portugal, fax +351 1 2955641, phone ...295 4464. pb@fctunl.rccn.pt.

IEEE 32nd Annual Foundations of Computer Science, Oct. 1-3, 1991, San Juan,Puerto Rico. Contact: IEEE Computer Society Conference Dept. 1730 Massachusetts Ave., N.W. Washington, DC 20036-1903 (202) 371 1013 202) 728 0884 (FAX).

Course on Intelligent Systems for Signal and Image Understanding, Oct.1-5, 1991, Udine, Italy. Contact: V.Roberto, Dipartimento di Indormatica, Via Zanon, 6 Udine, Italy, phone +39 – 432 – 297169, fax +39 – 432 – 510755, e-mail:roberto@uduniv.cineca.it.

IEEE Workshop on Visual Motion, Oct.6-9, 1991, Princeton, N.J. Contact: Thomas S.Huang, Coordinated Science Lab, Univ. of Illinois, 1101 W.Springfield Ave., Urbana, IL 61801, phone (217) 333-6912.

11th IEEE Symposium on Mass Storage Systems, October 7-10,1991, Monterey, California, Sponsor: IEEE Computer Society Technical Committee on Mass Storage Systems and Technology, Contact: B.T.O'Lear, NCAR, P.O.Box: 3000, Boulder,CO 80307, phone (303)497 1268, fax (303) 497 1137.

Sixth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, Oct.6-11, 1991, Banff, Canada. Contact: John Boose, Advance Technology Center, Boeing Computer Services, 7L-64, PO Box 24346, Seattle, WA 98124; (206)865-3253.

11th IEEE Symp. on Mass Storage Systems, Oct.7-10, 1991, Monterey, Calif. Sponsor: IEEE Computer Soc. Technical Committee on Mass Storage Systems and Technology. Contact: Bernard T.O" Lear, NCAR, PO Box 3000, Boulder, CO 80307, phone (303)497-1268, fax (303)497-1137.

First Int'l Conf. on Artifical Intelligence Applications on Wall St., Oct.9-11, 1991, New York City. Sponsor: Polytechnic Univ., Brooklyn NY 11201, phone (718)260-3360, fax (718)260-3136.

Workshop on Experimental Distributed Systems, Oct.12, 1991, Huntsville, Ala. Contact: Raif M Yanney, TRW, 1 Space Park, DH2/2328, Redondo Beach, CA 90278, phone (213)764-6033.

Special Session on Multisensory Computer Vision, 1991 IEEE International Conference on Systems, Man & Cybernetics, Oct.13-16,1991, Charlottesville, Virginia. ContactN.Nandhakumar, Dept. of Electrical Engineering, University of Vwerginia, Charlottesville, VA 22903-2442, phone 804-924-6108, e-mail nandhu@virginia.edu.

ICCD'91. IEEE International Conference on Computer Design: VLSI in Computers & Processors, Oct.14-16, 1991, Hyatt Regency Cambridge, Cambridge, Mass. Sponsor: IEEE Computer Society and IEEE Circuits and Systems Society. In Cooperation with: IEEE Electronic Device Society. Contact: Dwight Hill, AT& T Bell Laboratories 3D-446, Murray Hill, NJ 07974, phone 201-582-7766, e-mail: dwight@research.att.com.

ISMIS"91: Sixth International Symposium on Methodologies for Intelligent Systems, Oct. 16-19, 1991, Hilton Hotel at University Place, Charlotte, North Carolina. Contact: Z.W.Ras, ISMIS"91, UNC-Charlotte, Comp. Sci., N.C.28223, phone 704-547-4567, fax 704-547-2352, e-mail:ras@unccvax.uncc.edu.

European Conference on Industrial Applications of Knowledge-Based Diagnosis, Oct.17-18, 1991, Milano, Italy. Contact: A.Camnasio, CISE, PO BOX 12081, 20134 Milano, Italy; (39)2-21672400, fax (39)2-26920587.

1st Int Conf Practical Application of Prolog, Oct.28-31, 1991, Edinburgh UK. Info: A1 Roth, 31 Bexley Avenue, Blackpool, Lancs FY2OTE, UK. fax +44 253 53811, phone ... 253 58081. alroth@cix.compulink.co.uk.

AI * IA 2nd Nat Congres on AI, Oct.29-31, Palermo, Italy. Info: Prof.Salvatore Gaglio, CRES, Centro per la Ricerca Electronica in Sicilia, Viale Regione Siciliane 49, 90046 Montreale (Palermo), Italy. fax +39 91 640 6200, phone ... 640 6192/619/4501.

NOVEMBER 1991

IEEE 1991 Medical Imaging Conference, Nov. 2-9, 1991, Santa Fe, New Mexico. Contact: S.E.Derenzo, Lawrence Berkeley Laboratory, Mail Stop 55-121, 1 Cyclotron Road, Berkeley, CA 94720, phone 415-486-4097, fax 415-486-4768.

25th Asilomar Conference on Signals, Systems, and Computers, Nov. 4-6 1991, Asilomar Hotel & Conference Grounds, Pacific Grove, CA. Contact Dr.Neil K. Jablon, AT& T Bell Laboratories, 200 Laurel Avenue 07748-4801. Tel: 908/957-2011.

Neuro-Nimes 91, Int'l Conference on Neural Networks and Their Applications, Nov. 4-8, 1991, Nimes, France. Contact: Jean-Claude Rault, EC2-269-287, rue de la Garenne, 92024 Nanterre Cedex, France; phone 33(1)47-80-70-00; fax 33(1)47-80-66-29.

TAI 91, Third IEEE Computer Soc. Conf. on Tools for Artifical Intelligence, Nov.5-8, 1991, San Jose, Calif. Contact Benjamin Wah, Coordinated Science Lab, MC 228, Univ. of Illionis, 1101 W. Springfield Ave., Urbana, IL 61801-3082, phone (217)333-3516, fax (217)244-1764, e-mail wah% aquinas@cso.unicu.edu; or Nikolaus G.Bourbakis, 4138 Moonflower Ct., San Jose, CA 95135, phone (408)284-6494.

Advances in Intelligent Robotics Systems, Nov. 10-15, 1991, Boston. Contact SPIE, PO Box 10, Bellingham, WA 98227, phone (206)676-3290.

ANNIE'91: Artifical Neural Networks in Engineering, Nov. 10-12 1991, Saint Louis, Missouri. Contact C.H.Dagli, General Chair, ANNIE'91, Engineering Management Department, 205 Engineering Management Building, University of Missouri-Rolla, Rolla, Missouri 65401-0249; phone 314-341-4374.

Fourth Int'l Symposium on AI Applications on Informatics: Sowftware Engineering, Database Systems, Computer Networks, Programming Environments, DIS, and DSS, Nov.13-15, 1991. Cancun, Mexico. Contact: Hugo Tersashima, ITESM Centro de Inteligencia Artificial, Sucursal de Correos "J," Monterrey NL, 64849, Mexico.

Frontiers of Computer Science:The Bledsoe Symposium, Nov. 15-16, 1991. Austin, Texas. Contact: Joanne Click, Office of External Affairs, Computer Sciences Department, University of Texas at Austin, Austin, Texas 78712-1188, phone, 512-471-9729, fax 512-471-8815.

Connectionist Models in Biomedicine, Nov. 17-20, 1991. Washington, DC. Contact P.D.Clayton, SCAMC Program Chair, AMIA, 11140 Rockville Pike, Box 324, Rockville, MD 20852.

Int'l Joint Conference on Neural Networks'91, Nov.18-22, 1991. Singapore. Contact: Teck-Seng Low, Communicationn Int'l Associates, 44/46 Tanjong Pagar Rd., Singapore 0208, (65)226-2838, fax (65) 226-2877, e-mail mpeangh@nusvm.

Supercomputing 91, Nov.18-22, 1991, Albuquerque, N.M. Cosponsor: ACM. Contact: Raymond L. Elliott, Computing and Comm. Div., MS B260, Los Angeles Nat'l Lab, Los Alamos, NM 97545; or Supercomputing 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202)371-1013.

DECEMBER 1991

3rd IEEE Symposium on Parallel and Distributed Processing, Dec.1-5, 1991. Dallas, Texas. Contact: Ian Watson, Department of Computer Science, University of Manchester, Oxford Rd., Manchester M13 9PL, England, phone +4461 275 6248, e-mail: iwatson@cs.man.ac.uk.

Neural Information Processing Systems, Natural and Synthesic (NIPS'91), Dec.2-5, 1991. Denver, Colorado. Contact: S.J.Hanson, Siemens Research Center, 755 College Road East, Princeton, NJ 08540.

Int'l Conf. on Parallel and Distributed Information Systems, Dec.4-6, 1991, Miami Beach, Fla. Cosponsors: IEEE Computer Soc. et al. Contact: Amit Sheth, Bellcore, IJ-210. 444 Hoes Ln., Piscataway, NJ 08854, phone (908)699-9011, e-mail amit@ctt.bellcore.com.

Int'l AMSE Conference on Signals, Data, and Systems. Dec.9-11, 1991. New Delhi, Indiana. Contact the Association for the Advancement of Modeling and Simulation Techniques in Enterprises, 16 Av. de Grange-Blanche, 69160 Tassin-la-Demi-Lune, France, phone 33(7)83-43-604, fax 33(7)83-45-417.

Computer Architecture for Machine Perception (CAMP 91), Dec. 16-18, 1991. Paris, France. Contact Louis Wendel, Ecole National Superieure de Physique de Strasbourg-LSIT, 7 rue de l'Universite, 67000 Strasbourg, France, phone, Sylvette/33-1-42-31-97-21, fax 33-88-35-31-76.

A VIEW ON NEURAL NETWORKS PARADIGM DEVELOPMENT

(Part 5)

J.Hořejš [1]

Here we continue in the tutorial paper concerning the neural network paradigm, which first part was published in the Neural Network Word. No. 1, 1991.

8A. BP for the last time *[an invitation to a deeper study]*

Back-propagation and similar adaptation strategies for multilayered network surely deserve a special care due to their generality, proved capabilities and many unresolved problems, of both theoretical and practical nature. In this a bit more advanced exposition we give (a) a certainly incomplete survey of various modifications, intended mostly but not solely to speed up the convergence and/or to avoid (apparent) local minima, (b) a similar sketch how to automatically reduce the number of hidden neurons [which often leads to another function to be minimized, so that we speak more generally about *objective function* instead of error function] as well as (c) few advices and comments which may be useful when solving problems by means of the (standard) BP. Exceptionally we shall give in this section direct references for the readers who already decided to work with BP.

As for theoretical foundations of approximation capabilities of multilayered networks, the interested mathematically oriented reader finds an excellent survey by K.Hornik [1] in this journal.

(a)

Before you start to read, how to reach a good local minimum, recall the concept of the error landscape which gives for any weight vector $\mathbf{w}$ and the input/output pair $[\mathbf{x}, \mathbf{y}]$ the function $E_{\mathbf{w}}(\mathbf{y})$, and keep in mind that every change of any weight w changes the whole error landscape in the neighbourhood of present position of $\mathbf{w}$. Thus if for example you seem to walk downward on a pretty plain surface, too big a step (learning rate η) can convert the nice landscape into a wall arising before you at once.

Interesting and useful modifications can be found e.g. under the following names:

[1] Prof. Dr. Jiří Hořejš,CSc.,Department of Computer Science, Charles University, 118 00 Prague 1, Malostranské nám.25, Czechoslovakia

QUICK-PROP [2]: each weight update moves the system toward the center of a more shallow error landscape in the weight space along a linear combination of previous and presently suggested directions (conjugate gradients). If assumed that local valleys are of parabolic shape this enables a fast search of a point with zero derivative.

DELTA-BAR-DELTA [3]: each weight w has its own learning rate $\eta(w)$, which (linearly) increases as far as direction of its update does not oscillate too quickly; otherwise the $\eta(w)$ exponentially decreases.

SUPER-SAB [4]: the basic idea as in [3]; as far as the sign of Δw for some particular w remains unchanged [so that w "walks down" in the error landscape, maybe, however, unnecessary slowly], the learning rate, i.e. the length of the pace, is prolonged by setting $\eta(w)^{new} = K.\eta(w)^{old}$ for some positive constant K which is recommended to slightly exceed 1. When the update of this individual weight w changes sign (i.e. if $\Delta w^{old}.\Delta w^{new} < 0$) the just performed update of w is canceled (which can be done by setting $\Delta w := -\Delta w$), the system waits one step without any change of the particular weight [not to repeat the same problem; this can be realized by the statement $\Delta w := 0$, be however careful on its proper placement in the program] and afterwards [being now in another, "shifted" position in the weight space, caused by other w's updates] it shortens its learning rate setting say $\eta(w)^{new} = \eta(w)^{old}/2$ and tries again. Do not forget that $\eta(w)$ is no more constant (it depends both on w and time although you specify the initial values $\eta(w)$, common to all weights, by the programer-specified η as before) and should be represented by an array.

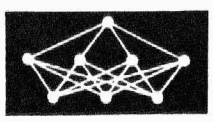
GAIN-BP [5]: error function minimization is based both on current and past data, the effect of which exponentially decays.

CHIR [6]: in this "choice of internal representation", suggestions are made in which you choose (to every input pattern) hidden layer neurons outputs so that upper layers will more easily meet their task, asking the same from lower layers. We have suggested (see [8]) a somewhat similar idea, introducing into the hidden layers "hints" derived from observed discrepancies between actual and expected values in the output layer and changing (only) thresholds of introduced "interneurons" accordingly.

CONE [7]: uses mixture of differentiable transfer functions and hard nonlinearities.

PAB [24]: parameters σ and λ of the sigmoid transfer function, which is of the form $S(\eta) = \sigma./(1/1+exp(-\lambda\eta))$, are continually updated according to similar laws as weights in the standard BP. By setting learning and momentum terms equal 0, you obtain the standard BP. On the other hand, setting $\eta = 0$, you arrive at "weightless" paradigm with all weights (maybe even mutually equal) which is allegedly equivalent to the usual one.

GEMINI [9]: *perturbations* [little changes] of the net incomes ξ_i of every neuron are introduced and responses in the error function measured in order to establish $\partial E/\partial \xi_i$; once this derivative is known, $\partial E/\partial w_{ij}$ is calculated. Because $\xi_i = \sum w_{ij} x_j$ [sum over all $x'_j s$ from a layer be-



low - see (11a) for the special case of a m-k-n net] and $z_i = S(\xi_i)$, it is $\partial E/\partial w_{ij} = x_j.\partial E/\partial \xi_i$; the idea is in [9] further improved.

BACKPERCOLATION [10]: uses the symmetry of net income $\mathbf{w}.\mathbf{x}$ and asks for changes of $x_i's$ in the above layers; in this way it assigns each neuron its own error surface. It also uses gradient descent, but on the local level, avoiding thus too much "wasted motion". Reducing local activation errors permits the system to "tunnel through" the global error surface.

AMBP [11]: In SuperSAB and PAB we meet automatic adaptation of parameters; AMBP (Adaptive Momentum BP) gives hints how to treat the momentum term and to estimate the range of initial weights values.

All these and many other improvements of the BP claim to perform better in some cases, often using some typical easy formulated problems [like the XOR problem, symmetry problem, parity problem (distinguishing inputs with even and odd number of $1's$)] in which they are superior to others, and/or having other advantages. In cases, where the main ideas of two such improvements are independent, it brings usually good results to combine them retaining advantages of both. It should be said however that the author is not aware of any "standard" benchmark and detailed environment requirements (about the random choice of initial values of weights, say) which could be accepted as the ultimate judge ranking or ordering all these improvements in a unique indisputable way.

You may also meet strategies which perform very well as far as the number of required cycles (iterations, epochs) concerns, but each cycle is computationally more expensive than in standard BP.

If you are already involved in a work with BP, you will of course hardly have enough time to see all the mentioned papers, which anyway are far from covering completely the area. Therefore we tried to describe few of them to the extent that the ideas could be hopefully understood and programmed; on the other hand more complex ideas are only vaguely sketched. Our experience shows that combination of SuperSAB and PAB (which shall be in detail described in a forthcoming issue of this journal) gives a relatively good toole if you choose their proper combination.

(b)

Another often discussed problem is that of the best size of the network. Because it is up to the designer, how many hidden layers and hidden neurons to choose (while the number of neurons in input and output layers are prescribed by the task), several strategies have been developed and many (sometimes opposing) views proclaimed. It is clear that more hidden elements give a better chance to reach a global minimum of the error function: if you add a neuron, which could be harmful, the algorithm has always the possibility to neglect it by establishing all connecting weights to zero. It is also clear that if you choose too few neurons, you can put severe restriction on the mapping to be realized (see e.g.(15)). The nonsystematic way then usually consists of some guess, adding or deleting hidden neurons dependently on whether the net is not able to converge or the computation is too slow, respectively.

However there are even more serious reasons why we prefer nets which converge, but not too easily. Experience as well as intuitive feeling shows that if you solved some nontrivial problem too easy, you have not solved the proper problem; in terms of NN: you found one of many trivial solutions to the approximation problem, which is very poor in knowledge extraction and thus generalization. Remember Fig.10!

Majority of NN-people believes that by reducing the number of hidden neurons to the absolutely necessary minimum which does still permit convergence, increases generalization capability of the net, although there are some indications that this need not be always true, especially if you use different measures of generalization abilities: e.g. you may distinguish performance of the NN on quite new inputs, not included in T or inputs from T, which are however corrupted by some *noise*, i.e. changed by some little random *corruptions* (distortions), see [12]. To express it another way: if there are too many neurons and thus too many degrees of freedom, then there are too many "solutions" to the inner laws represented by the net (too many "*models*" [in the sense of set theory] of the constraints - "*axioms*" extracted from the set T).

Still one important reason that asks for possibly small nets is related to the so called *overfitting*. If the net becomes *oversized*, it may start to be too good in a noisy environment, where some inaccuracies should be neglected. The net then can fit the mapping specified by the training set in all unnecessary details - it becomes overfitting; if the data given are much less precise than the errors reached during adaptation, we at least waste the time of computation and the welcome effect of noise filtering, typical for well-designed nets is lost.

Several techniques have been developed to find the "reduced" net. Those, which start with assumed subminimal number of neurons, adding in case of necessity other ones, fight with the problem how (when) to recognize that the net will not (can not) converge. You can adopt simple criteria like: if during 100 cycles E does not decrease more than 1% you are on a wrong way.

Or you can try to formalize the conviction that E would follow an exponential-like decrease approaching asymptotically an unacceptably high value (which is very often the case). Under this category falls solution offered in [13], where the author proposes to add a new neuron when a flattening of the average squared error curve, specified by a few parameters, is detected. This is in fact suggestion of a general *stopping criterion*, which can lead you to another conclusion as well: e.g. to start once more with another initial weights or so.

There is one more stopping criterion to be mentioned. Besides the training and testing sets T and Q, we have at our disposal also a *validation set* [test training set] V, consisting of some known pairs of input/output vectors $\{[\mathbf{x}^v, \mathbf{y}^v]\}$, not used during the adaptation, but checking

whether further training should take place. As far as testing the net on members of V gives good results, we continue in adaptation over T; otherwise we deduce that there is a danger of overfitting and stop further adaptation.

Another problem of adding new neurons concerns the weights of connections leading to/from them. Usually they are initialized randomly or to zero [which then does not change quickly the net response].

More promising seem to be approaches starting with sufficiently rich number of neurons, eliminating "superfluous" ones.

One of the first ways how to do that, has been suggested in [16], which proposes to eliminate some connections, diminishing successively their weights up to zero by extending the usual global error function (6) to more complex objective function, adding a *cost* function depending on weights greater than some norm. In combination with the validation set approach, the *weight − elimination*, which ultimately cuts off some "superfluous" neurons, we have a good chance to find a properly trained smallest net which still covers the training set T and avoids overfitting.

One such an objective function is suggested in [16]:

$$E = \sum\sum(y - \overline{y})^2 + \kappa . \sum\sum (w_{ij}^2/w_0^2)/(1 + w_{ij}^2/w_0^2) \tag{22}$$

where the first sum is taken over all output neurons and all members of T (cf.(6)) and the second sum is taken over all weights (over all connections in the net); w_0 is a scale factor (for activations between 0 and 1 we set $w_0 = 1$) and the cost κ applies if and only if $|w_{ij}| \gg w_0$. Of course, the adaptation then uses the new value $\partial E/\partial w$, calculated from (22).

Other objective function has been proposed as well. In [17], objective function is formed by adding to the error function (6) the so called *energy* function which is the sum $\sum\sum e((z_i^j)^2)$, j taken over all input patterns and i over *all hidden* neurons, where e is a positive monotonic function representing "energy", spent by the hidden neuron in solving a problem. If a neuron has a constant output independently of the pattern submitted, it contributes only to the energy term and will be suppressed by the algorithm. The method works well primarily in some cases of boolean functions.

There is known a plenty of methods, which estimate functioning of particular hidden neurons and cross them out as soon as they are purposeless, either not working at all or copying work of others. Here we mention a relatively "old" paper on *pruning* [18] and the technique of *skeletonization* [19] which assesses the *relevance* of a hidden neuron for proper functioning of the net, the main idea of which is to compare error function for the net including the particular hidden neuron and for the net with the neuron removed. A formula for α_i in the sum $\xi_j = \sum w_{ij}\alpha_i x_i$ is derived; here α_i gives the relevance measure for contribution of x_i to ξ_j and thus also for x_j. For $\alpha_i = 0$, output of x_i is not needed, while for $\alpha_i = 1$, the usual net income for ξ_j is formed.

Using PAB, a suitable candidate for removal (if it is not an output neuron) might be that neuron, which for a long time has λ near 0 with σ fixed [produces a constant value, which can be covered by the threshold] or even $\sigma \approx 0$ [giving thus a zero response].

Another interesting method appears in [20]. The method does not excise hidden units like many others do, but forms clusters of similar weight vectors in lower-dimensionality space. This yields a functional bottleneck, distributed across many units.

(c)

In nontrivial applications of (standard) BP we often encounter as the main problem that it seems that we are caught in a local minimum or the convergence is too slow or that the global error function starts to oscillate.

Unfortunately there is no exact way (or at least a good heuristic) what to do in such cases, even that some expert systems has been suggested ([14],[15]); the referred papers will however hardly satisfy a reader, looking for a practical solution. Some year ago, the main problem was to find or to manually control the convergence parameters, such as the learning rate η. With the discovery of their run time control (of the SuperSAB type), this seems to be no more the primary question, although the troubles of mentioned sort are not excluded even by using the described techniques. If the properly dimensioned net still does not converge properly, we can give only a few general hints without warranty, especially for those, who can not afford to implement the improvements mentioned above; some cases are however inherent to any BP method and some of the hints address not solely the problem of getting stuck or wildly oscillating:

-let us first note that "small" oscillations need not be harmful and that they may even help to overcome the dead state, when the decrease is practically stopped. If you (still) control the parameters η and α manually, try to (temporarily) decrease η and α. Generally it is recommended to do any change of parameters gradually rather than abruptly.

[Here it is assumed that you have implemented Save and UnSave procedures, which enable you to modify the parameters after temporarily halting the run (for smaller prototype programs the best way is to be able to modify the source code at all) and continue from the point, where you interrupted the computation. The Save/UnSave procedures should of course cover (almost) all (global) variables - do not forget to Save the "old" values if you need them.]

The experience and some research shows that the danger of being trapped in a local minimum [first part of Fig.9] is much less than the case, where the decrease of trajectory is very slow [last, flat part of Fig.9], sometimes so slow that it looks like being caught in a local minimum. One of the leading authorities in NN theory and practice, Robert Hecht-Nielsen, has even been for a long time convinced that all local minima we meet are just apparent; that we simply are not patient enough or that we work with unsui-

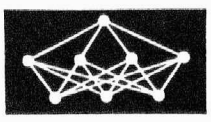


table values of parameters to let the net find the narrow path leading out of a deep gorge in the error landscape, into which we went astray. Only relatively lately he himself found a provably true local minimum.

There are some more "over's" connected with NN adaptation; we have already mentioned the overtraining case which seems to be related to the initial weight vector **w** being randomly placed so unluckily that the landscape around is like a plateau, where there are small slopes in every direction and when we eagerly ask to deform it to adapt to the training set, it may happen that too many wrinkled waves evolve ([see[23]). This may again lead the computation to a local minimum, be it true or apparent.

Well, we know that one of the possibility how to tackle the problem of not sufficiently decreasing error function is to extend the number of hidden neurons, which is undesirable from other reasons. Thus better first

-try to check data for consistency; if the error function decreases from the beginning of adaptation process satisfactorily and then remains hanging on some almost constant value, there could be in the set T two or more pairs requiring different answers **y** for the same input **x**; this does not generally mean that the effort of the net is meaningless: it may still offer you a sort of compromise, satisfying most of constraints in the best possible way;

-or you may have forgotten that the output is limited by the shape of nonlinearity function; using sigmoidal transfer function when trying e.g.to predict time series, some negative number may appear as the desired output, because they occur in the input series;

-because moreover the usual LMS criterion will not always best fit the task at hand, do not forget that the form of both the error function E and/or transfer function [sigmoid] are by no means obligatory; all what is required that you are able to compute $\partial E/\partial w$ [or $\partial E/\partial \lambda$ etc.]

-try to choose another initial values of weights. It is a custom to initialize weights to random "small" numbers and keep them small during the whole process; on the other hand, in many cases you can afford slightly "bigger" weights and the convergence improves;

-occasionally it may help to introduce a weight decay of about 0.01% each update;

-try both integer and high-precision arithmetic whenever the task permits;

-let us note that the weight change can be further smoothed by setting $\Delta w^* = (1-\alpha).\Delta w(t) + \alpha.\Delta w(t-1)$ [actual new change $= (1-\alpha)$. proposed new change $+\alpha$. actual old change].

-several experiences on the (initial) choice of parameters η an α have been reported; one of them recommends to choose the initial value of η sufficiently small, $\alpha \cong \eta/5$ and to increase it continually if the convergence behaves well; afterwards we may increase it slightly, while in case of divergence or persistent oscillation, we decrease both. In case of a local minimum (the global error decrease tends to a nonnull constant), a temporary increase of η and α may help; when the divergence phenomenon occurs, it should be decreased back. If the convergence is anyway unsatisfactory, stop the adaptation process and start with another random initial weights;

-be careful to cover by the training set T the typical distribution of inputs expected during generalization. Think also about training strategy you used; when you present some input pattern too many times, it may happen, that it will be difficult to adapt the net for the others, to *re-learn* it;

-a special form of the later case appears when you try to extend the training set T to a set $T' = T \cup T''$ in the case that for T you have already achieved a successful convergence. The question then arises what is better: to start the adaptation w.r.t. T' from the very beginning or to use previously converged set for T as a starting point in learning T'. In general, the following strategy seems to work: if for every pattern $\mathbf{x}''$ from T'' there exists in T a pattern **x** close to $\mathbf{x}''$ [say, in the sense of Euclidean metric of the input space], the newly required changes of weights will be not too drastic and we may use the previously reached point in the weight space; otherwise re-learning can be more difficult than starting with T' from the very beginning;

-as long as this (task dependent) hypothesis is valid, we should use it for the real-time learning, in which we try (and are often able) to order the training set so that the patterns lie on a "continuous curve", each new pattern being close to the predecessing one;

-this method could be of course only used provided that additional pairs do not contradict either absolutely [asking two different answers **y** for the same argument **x**] or relatively, requiring to find quite another model developed by the net for good generalization so far;

-this is a special case of a more general problem: when it is natural to partition training set into classes of apparently similar training pairs, is it better to let first the net converge on a subset of representatives and only then to add the rest or conversely? It happened several times that the complementary training strategy - teaching one class after the other - was better. Sometimes the net behaves in such a way, that it hardly "forgets" the training patterns already learned;

-concerning testing strategies, be careful not to deduce premature decisions, especially in case when some training pairs are stubborn ones (showing for a long time worst neuron/pattern error) so that their individual errors exceed the average: exercising them more frequently may just shift the problem to other ones;

-try to add a small amount of noise (with standard deviation of about 2%) to both data and weights to make the net more "robust"; this *shaking* reminds techniques of various stochastic variants [to be discussed later on]; they also many times enable to find a better solution in better time. GEMINI in fact uses shaking of net inputs as a self-contained method;

Now few comments concerning the problem of generalization:

-you often meet such papers the power of which is de-

monstrated on examples with *complete* training set T [exhausting all possible pairs] so that the question of generalization is void; like the XOR or the symmetry problem [for a fixed number of letters], for which simple table look-up or traditional programming is the best way how to find the answer [reminding thus the so many formal proof techniques for greatest common divisor algorithm]. Of course, as Rumelhart, Hinton and Williams have shown in their classical paper on family trees (see preceding section), even finite sets of pairs may give remarkable results provided that the training set T *is not* complete.

-because we have generally so many weights unspecified, the size of the training set should be sufficiently large to fix them appropriately. If the net is expected to do some knowledge extraction and generalization, the training set is best to be several times larger than is the number of weights (connections) involved.

-the fact that use of higher order NNs [in which the sums of the form $\sum w_i x_i$ are replaced by sums like $\sum w_{ijk} x_i x_j x_k$ etc], helps to increase the generalization abilities was demonstrated five years ago on a relatively hard problem, called "contiguity problem": in a 0/1 sequence, the number of uninterrupted blocks/clumps of 1's is e.g. either 2 or 3 and the net has to distinguish these cases (see [21] and also [25])

-ideally, we should design a *generalization* function to be added to (22) [forming thus still more complex objective function] in order to promote better generalization abilities; the problem is that we do not know how to exactly and generally define what the "generalization" actually means; like in the IQ tests: given sequence of four pictures, choose "the right" fifth one! (but it should be that one, which the inventor of the test had in mind).

References

[1] Hornik K.: Functional Approximation and Learning in Artificial Neural Networks, Neural Network World, this issue.

[2] Fahlman S.: Fast-Learning Variations on Back-Propagation: An Empirical Study, Proc. 1988 Connectionist Models Summer School, Morgan Kaufmann, 1989.

[3] Jacobs R.: Increased rates of convergence through Learning Rate Adaptation, J.Neural Networks, vol.1, p.295-307, 1988.

[4] Tollenaere T.: SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties, Neural Networks, vol.3, pp.561-573, 1990.

[5] Gawthrop P., Sbarbaro D.: Stochastic Approximation and Multilayer Perceptrons: The Gain Backpropagation, J.Complex Systems vol.4, p. 51-74, 1990.

[6] Saad D, Marom E.: Learning by Choice of Internal Representations: An Energy Minimization Approach, J. Complex Systems vol.4, p.107-118, 1990.

[7] Fukumi M., Omatu S.: A New Neuron Model "CONE" with Fast Convergence Rate and its Application to Pattern Recognition, Trans. of Inst. Electronics, Information and Commun. Eng., Japan, vol.J73-D-II, No.4, p.648-653, 1990; adapted version also under the title A New Back-Propagation Algorithm with Coupled Neuron, in Proc. IJCNN Washington, 1989.

[8] Kufudaki O., Horejs J., Husek D.: Threshold-Controlled Backpropagation Learning, Proc. NEURONET'90, Prague Sept.1990, p.211-213; extended version under the title Coupled BP Learning Theoretical Aspects in Neurocomputing, World Scientific Co, London, p.195-212, 1991.

[9] Le Cun Y., Galland C.C., Hinton G.E.: GEMINI: Gradient Estimation through matrix inversion after Noise Injection, Proc. 1988 Connectionist Models Summer School, Morgan Kaufmann, p.141-148, 1989.

[10] Jurik M.: Backpercolation, Report of Jurik Research & Consulting Co, Aptos, CA, USA, 1991.

[11] Drago G.P., Ridella S.: An optimum weights initialization for improving scaling relationships in BP learning, Artificial Neural Networks, Elsevier Science Publishers (North-Holland), p.1519-1522, 1991.

[12] Sietsma J., Dow R.J.F.: Creating Artificial Neural Networks That Generalize, Neural Networks, vol.4, pp.67-79, 1991.

[13] Ash T.: Dynamic Node Creation in Backpropagation Networks, ICS Report 8901, Feb.1989, Univ. of California, San Diego.

[14] S.C.Kwasny: Rule Based Training of Neural Networks, Expert Systems with Applications, vol.2,p. 47-58, 1991.

[15] Steib,M.L. and Weidman S.T.: Expert Systems for Guiding Backpropagation Training of Layered Perceptrons, ibid, 73-81.

[16] Weigend A.S., Rumelhart D.E.,Huberman B.A.: Back - propagation, Weight Elimination and Time Series Prediction, Proc. 1990 Connectionist Models Summer School, Morgan Kaufmann, 1990.

[17] Chauvin I.: A Back-propagation Algorithm with optimal use of Hidden Units, Advances of Neural Information Processing I, Ed. Touretzky D.S., p.519-526, Morgan Kaufmann Publ., 1990.

[18] Sietsma J., Dow R.: Neural net pruning - why and how. Proc IEEE 1st Int.Conf. on NN, vol I p.325-333, IEEE San Diego, 1988.

[19] Mozer M.C., Smolensky P.: Skeletonization: A technique for trimming the fat from a network via Relevance Assessment, Advances of Neural Information Processing I, Ed. Touretzky D.S., p.107-115, Morgan Kaufmann Publ., 1990.

[20] Kruschke J.K.: Improving generalization in Back-propagation networks with distributed bottlenecks, Proc. IJCNN, I- 443-447.

[21] Maxwell T, Giles C.G., Lee Y.C.: Generalization in Neural Networks: The Contiguity Problem, Proc. IEEE First Annual Conf. on NNs, San Diego, June 1987.

[22] Durbin R., Rumelhart D.E.: Product Units with Trainable Exponents and Multi-Layer Networks, Neurocomputing, NATO ASI Series, Springer Verlag, p.15-26, 1990.

[23] Hecht-Nielsen R.: Neurocomputing, Addison Wesley 1989, pp.432.

[24] Kufudaki O., Horejs J.: PAB:Parameters Adapting BP, this issue.

9. Higher order separating surfaces.

Until now we used paradigm, which is close to neurobiological observations and has a simple mathematical description: the response of a neuron and its manifold consequences were dependent on its net income, i.e. on the value of $\xi = \sum w_j x_j$ which was then evaluated and postponed to further processing by (again a simple) nonlinear transfer function. Except a constant response, the linear dependency of ξ on $x_i's$ is the simplest possible. If we replace this "first order" relation by a more complicated one, we can sometimes achieve more: in generalization abilities, in applicability and even in simplicity!

I.Polynomials

We have already mentioned in sect.8A(c) that higher order nets usually better generalize than linear ones. Of course they have more complicated structure, which is even not so easy to depict. Thus e.g. for the formula

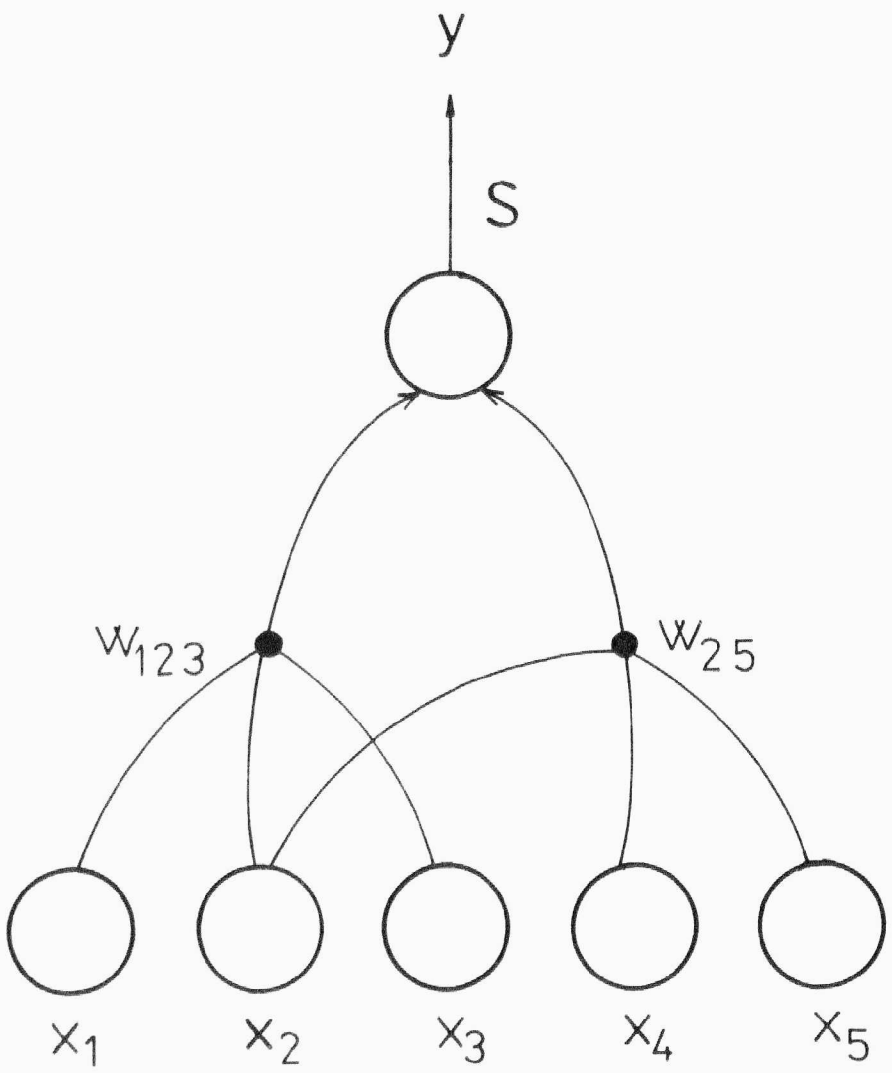


Fig.28 Weights in higher order NN

$$y = S(w_{25}x_2x_5 + w_{123}x_1x_2x_3) \qquad (23)$$

the following Fig.28 indicates how it is possible to illustrate that more than one neuron share the common weight (w_{25} and w_{123}), respectively.

Having only higher order neurons at our disposal, some of the task can be solved very simply. Thus e.g. the XOR problem is easily solved by the only second order neuron. With use of the encoding

x_1	x_2	y
1	1	1
−1	−1	1
−1	1	−1
1	−1	−1

and $S(\eta) = 1$ for $\eta > 0, S(\eta) =$ -1 for $\eta < 0$ we simply put

$$y = S(x_1x_2) \qquad (24)$$

But even for standard 0/1 codes and sigmoid S we have

$$y = S(w_1(x_1+\vartheta)+w_2(x_2+\vartheta)+w_{12}(x_1+\vartheta)(x_2+\vartheta)) \qquad (25)$$

where the total threshold is divided into several components, depending of course on the incoming pattern. Using now standard calculation of $\partial E/\partial w$, we can arrive at the result that whenever $w_1, w_2 \gg 0$ and $w_{12} \ll 0$, y realizes the XOR function.

Another interesting property of (symmetrical) higher order neurons was pointed out in works of Maxwell, Giles and Lee: if you perform any permutation on indices of $x's$, the output of considered neuron [cf.(25)] is the same. More generally, higher order NNs show certain invariance properties (w.r.t. a group of input vectors transformations), which can be utilized in many ways.

Durbin and Rumelhart recently proposed a general scheme in which besides usual $\sum$ - neurons (summing weighted inputs as known), also Π - neurons the output of which is given by

$$\xi = \Pi x_i^{p_i} \quad or \quad \exp(\sum p_i.ln\ x_i) \qquad (26)$$

are used (for the real parts of complex plane this converts to $\exp(\sum p_i.ln|x_i|).\cos\sum\ p_iI_i$, where $I_i = 0\ [\pi]$ for $i > 0\ [i < 0]$, respectively). The neurons are usually organized into multilayered way, outputs of Π neurons leading to summing $\sum$ neurons; this is a generalization of so called $\Sigma\Pi$ nets, dealt with already in the early stages of NN theory ($p_i = 0$ or 1). Besides generalization and other advantageous properties the author claim that logarithm and exponential scaling occurs in some sorts (e.g. Purkinje) cells and has some neurophysiological correlate.

II.RCE network.

Let us now more closely follow a useful and simple second order neurons. Let the role of the hyperplanes from sect.2 and others be replaced by a surface which is well known from the time of childhood: a ball, or to use more scientific terms so that not all will understand: the n-dimensional hypersphere (note that our parents preferred second order) balls rather than linear cubes in our plays); do not worry, I am not going to claim that balls have some special relevance to neurophysiology and neurons as such (even that we carry them in the ball-like heads). What I do claim that "neurons" with a rather simple activation rule depending on whether they are inside or outside a hypersphere are a material well suited to construction of

a very simple and yet very powerful NN [it is said that it immediately follows the multilayered back-propagation in the hit parade of most frequent used paradigms].

Let in a (cartesian) m-dimensional space $\mathbf{r} = [r_1, r_2, \ldots, r_m]$ denote the center of a hypersphere h with radius ρ, so that for any point $\mathbf{x} = [x_1, x_2, \ldots, x_m]$, $\mathbf{x} \in h$ if and only if

$$d(\mathbf{x}, \mathbf{r}) = \sum_{i=1}^{m} (x_i - r_i)^2 < \rho \qquad (27)$$

[if suitable, " $<$ " can be replaced by " $\leq$ "].

Now let the input space should be classified into n different *categories* $C_1, C_2, \ldots, C_n$. There is again a training set T, now consisting of pairs $[\mathbf{x}_j, C_j]$ and a teacher, who for every $\mathbf{x}_j$ from T (i.e. for every pair, the first component of which is $\mathbf{x}_j$) specifies the category C_j to which $\mathbf{x}_j$ belongs [the case when an object $\mathbf{x}$ belongs to more categories will be discussed later; at present we assume that the categorization is unique]. Construct a three-layered network $m - k - n$, where input layer accepts any m-dimensional vector of *features* (to recall or extend our terminology), characterizing an object from the input space. The top output layer consists of n binary neurons, the neuron i firing if an input x belongs to the category (class) C_i.

The hidden layer as well as the whole net is specific and in many respects differs from what we already learned.

First, the number of its neurons k is not specified before, but can successively be extended; initially there are no neurons at all. Second, there is no step of initialization of weights; no wonder, if initially there are no interconnections. All what misses is created in "real time", as the learning/adaptation process begins [there is thus of course no spreading activity possible before adaptation].

As soon as first input stimulus, vector $\mathbf{x}_1$ (where $[\mathbf{x}_1, C_j] \in T$) arrives, the net takes the first uncommitted hidden neuron $h1$ (of which there is no lack, supposedly), connects to it all the m input neurons and ascribes to the connections the weights $w_i(h1)$ $(1 \leq i \leq m)$, $\mathbf{w}_1 = [w_1(h1), \ldots, w_m(h1)]$, which will further represent the center of m-dimensional hypersphere (in a similar sense in which formerly weights represented a hyperplane). It will be done in such a way that the neuron - hypersphere is activated by $\mathbf{x}_1$. If we set $\mathbf{r}_1 = \mathbf{w}_1 = \mathbf{x}_1$, then for arbitrary positive ρ_1, this will indeed trivially hold: the center of a hypersphere is closer to its center than the radius distance $\rho 1 > 0$ prescribes. For specificity assume e.g. that initial values of all $\rho's$ are 1 if not specified otherwise.

If, afterwards, a second input $\mathbf{x}_2 \neq \mathbf{x}_1$ is presented, one of the three possibilities occurs:

-a) $\mathbf{x}_2$ is still closer to $\mathbf{x}_1 = \mathbf{r}_1 = \mathbf{w}_1$ then ρ_1 *and* the teacher considers $\mathbf{x}_2$ as another representant of the same category C_j. In this case all is O.K.: $h1$ fires and $\mathbf{x}_2$ is recognized as before, belonging to C_j - see Fig.29a.

-b) $\mathbf{x}_2$ falls inside the same hypersphere, i.e. $d(\mathbf{x}_2, \mathbf{r}_1) < \rho_1$, but the teacher assigns to it another category, C_k say. In this case $h1$ should *not* fire; the simplest way how to solve the problem is to *shrink* the hypersphere $h1$ (h's denote both *h*idden neurons and *h*yperspheres which they represent), setting the new value of ρ_1 to $\rho_1 \leq d(\mathbf{x}_2, \mathbf{r}_1)$. After this remedy, $h1$ no more covers $\mathbf{x}_2$, does not fire and does not declare that $\mathbf{x}_2$ belongs to the category C_j. This is however only the first part of the solution. Another hidden neuron $h2$ has to be attached to the hidden layer, $\mathbf{x}_2$ becomes (as an m-dimensional point) its center $\mathbf{r}_2 = \mathbf{x}_2$ and other m connections from the input neurons to $h2$ are created. Moreover they should be assigned weights, forming the vector $\mathbf{w}_2 = \mathbf{x}_2$; as before, $\mathbf{x}_2$ then surely causes $h2$ to fire and $h2$ starts to represent C_k. Of course, the radius ρ_2 of the hypersphere $h2$ should be chosen so as not to interfere with $\mathbf{x}_1$, i.e. less or equal than (due to inequality in (27), $d(\mathbf{x}_2, \mathbf{r}_1)$ - Fig.29b.

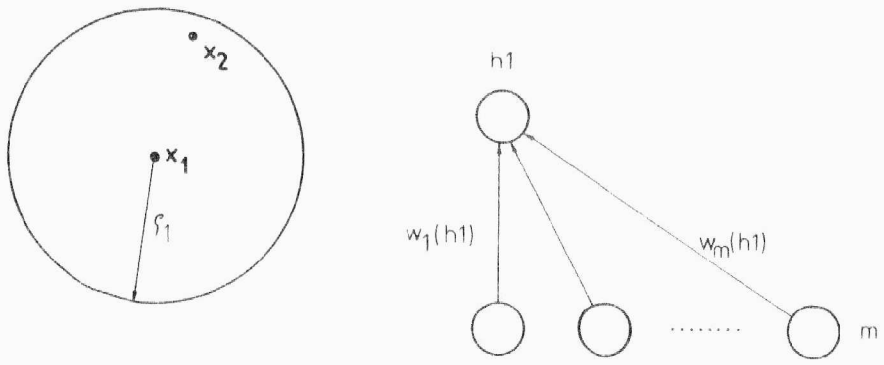


Fig. 29a

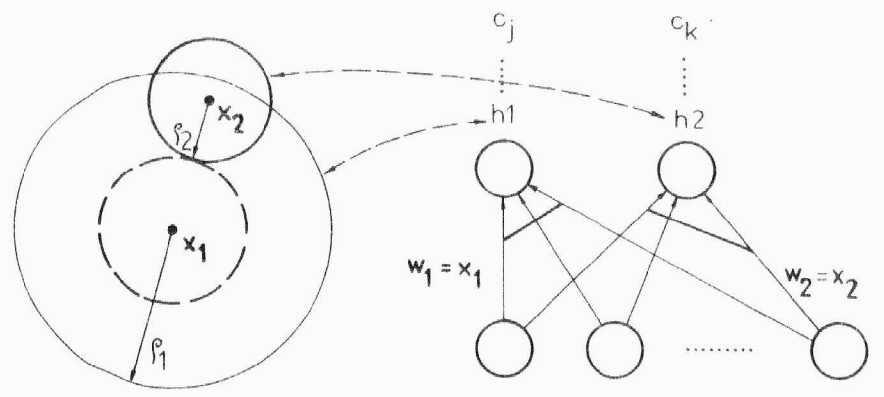


Fig. 29b

-c) $\mathbf{x}_2$ falls outside h1. If the teacher decides that $\mathbf{x}_2$ belongs to the same category as $\mathbf{x}_1$, *i.e.* $C_k = C_j$ (ca), we can copy the procedure known from b), adding next hidden neuron h2; only less care need be devoted to choice of ρ_2, it may well be again 1. And this is the time, when the third layer of the net comes into action. It consists of n neurons (provided that there are n categories, as indicated above), each of which is the ultimate judge telling us, to which category a given input x belongs. Any of these output neurons are connected exactly to all hidden neurons, recognizing the category assigned to it. The new connections, directed from hidden neurons to this output neuron, have the same weight 1 and the output neuron performs just the logical OR, i.e. the output neuron fires if and only if some of hidden neurons, here $h1$ *or* $h2$ which are connected to it, recognized the proper category of the incoming $\mathbf{x}$, here $\mathbf{x}_1, \mathbf{x}_2$. When the teacher however assigns to $\mathbf{x}_2$ another category $C_j \neq C_k$ than he did for $\mathbf{x}_1$ (Fig.29,cb), we have now two neurons, $h1$ and $h2$, with the property that if $h1$ [$h2$] fires, so should do C_j[C_k] and corresponding hyperspheres are disjoint.

Fig.29 illustrates the three cases a), b), c), showing both the structure of the net and hyperspheres, called *fields of influence*, represented by particular hidden neurons.

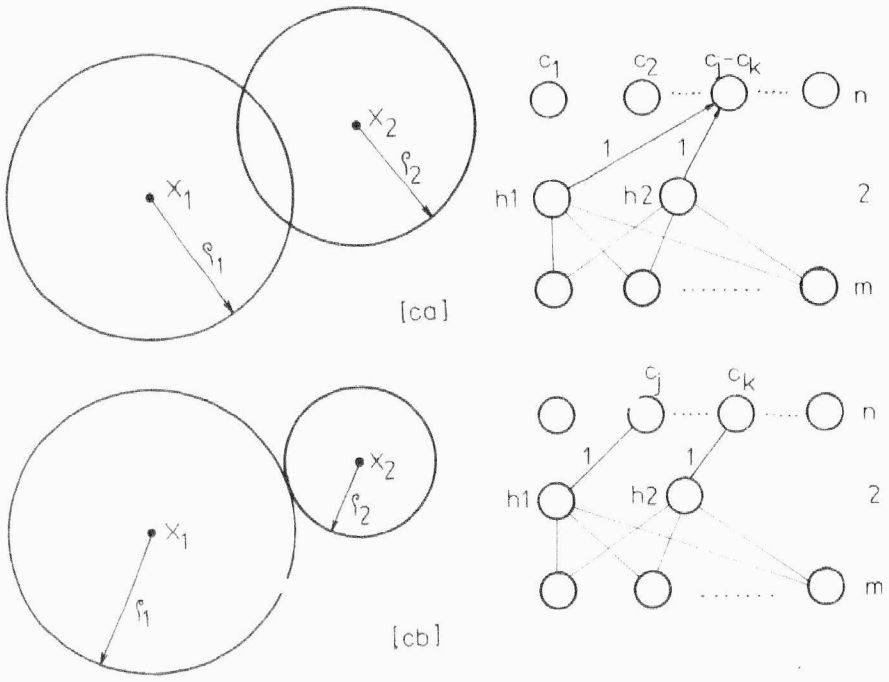


Fig. 29c

Fig. 29 a, b, c Relations among fields of influence

We considered explicitly only two incoming vectors; the idea is however the same if you take into account any number of inputs:

Submitting a new input $\mathbf{x}_h$ being classified as of category C by the teacher, all of its m coordinates come along the weighted connections to all hidden neurons created up to now. Every hidden neuron h calculates the distance of $\mathbf{x}_h$ and its center $\mathbf{r}_h = \mathbf{w}_h$ according to the sum from (27) [which substitutes the dot product **w**.**x** known from previous models] and compares it with its current value of ρ_h [which corresponds - and is sometimes named - after the *threshold* of usual linear neurons]. If the resulting distance is less than the threshold, h fires and so does the output neuron to which h is attached because it represents the desired category C.

The last paragraph serves as a description of *adaptive* [*learning*] dynamics, but also of *activation* [*working*] dynamics. The only difference is in interpretation of the phrase "to which h is attached". During teaching this means that the connection is established, while during activation this means that the connection is used.

The main idea of this RCE network (called after some analogy with "Restricted Coulomb Energy") is to cover different subsets of input space [where any such subset corresponds to different category, or classification class, being continuous or not] by hyperspheres of various radii under the only condition (which will be later relaxed a bit), namely that hyperspheres belonging to different subsets do not intersect.

(Continuation)

Literature Survey

Murray A.F., Corso Dante Del: Pulse-Stream VLSI Neural Networks Mixing Analog and Digital Techniques
IEEE Transactions on Neural Networks Vol.2, 1991 No.2 pp.193-204

Key words: neural networks; digital techniques; pulse-stream.

Abstract: This paper reviews the pulse stream technique, which represents neural states as sequences of pulses.Several general issues are raised, and generic methods appraised, for pulsed encoding, arithmetic, and intercommunication schemes.

Oh S., Marks R.J.: Dispersive Propagation Skew Effects in Iterative Neural Networks
IEEE Trans. on Neural Networks Vol.2, 1991 No.1 pp.160-162

Abstract: During communication between neurons in continuous-time analog neural network, propagation skew typically varies from neuron pair to neuron pair. For no dispersion, we have previously demonstrated that the steady-state performance of an iterative neural network is not affected if the combination of the neural network's weights and neural nonlinearity in contractive. In this letter, this result is extended to the case of dispersive skew. We show that, under nearly the same conditions, the same steady-state result will occur in the neural network in presence of dispersive skew.

Pao Y.H., Sobajic D.J.: Neural Networks and Knowledge Engineering
IEEE Transactions on Knowledge and data Engineering Vol.3, 1991 No.2 pp.185-192

Key words: neural networks; knowledge engineering; associative optimization; heterogeneous nets; knowledge representation; neural-net processing.

Abstract: We propose, instead, that in neural-net computing, the processing is the representation. In other words, the very nature of the processing encodes the knowledge. There is no place and no need for a separate body of global rules to be used by the network for inferencing. If rules exist at all, they are in the nature of local processing steps carried out at individual processors in response to stimuli from other neurons. We develop this theme in this discussion together with a theme which is closely related to it. The second notion is that neural networks may also be thought of and implemented in terms of heterogeneous networks, rather than always or only in terms of massive arrays of identical elemental processors.

Tabatabai A., Troudet T.P.: A Neural Net Based Architecture for the Segmentation of Mixed Gray-Level and Binary Pictures
IEEE Transactions on Circuits and Systems Vol.38, 1991 No.1 pp.66-77

Abstract: A neural net based architecture is proposed to perform segmentation in real time for mixed gray-level and binary pictures.

Instructions to authors

1. Manuscript

Two copies of the manuscript should be submitted to the Editor-in-Chief.

2. Copyright

Original papers (not published or not simultaneously submitted to another journal) will be reviewed. Copyright for published papers will be vested in the publisher.

3. Language

Manuscripts must be submitted in English.

4. Text

Text (articles, notes, questions or replies) double space on one side of the sheet only, with a margin of at least 5 cm, (2") on the left. Any sheet must contain part or all of one article only. Good office duplication copies are acceptable. Titles of chapters and paragraphs should appear clearly distinguished from the text.

Author produced (camera ready) copy is acceptable if typed on special sheets which are available from the Editor, and adherence to the Typing instructions (also available from the Editor) has been taken care of, is emphasized that camera ready text should be typed single space (i.e. with no space between the lines). Complete text records on 5 1/4" floppy discs are also acceptable, if typed according to the instructions available from the Editor.

5. Equations

Mathematical equations inserted in the text must be clearly formulated in such a manner that there can be no possible doubt about meaning of the symbols employed.

6. Figures

The figures, if any, must be clearly numbered and their position in the text marked. They will be drawn in Indian ink on white paper or tracing paper, bearing in mind that they will be reduced to a width of either 7,5 or 15 (3 or 6") for printing. After scaling down, the normal lines ought to have a minimum thickness of 0,1 mm and maximum of 0,3 mm while lines for which emphasis is wanted can reach a maximum thickness of 0,5 mm. Labelling of the figures must be easy legible after reduction. It will be as far as possible placed across the width of the diagram from left to right. The height of the characters after scaling down must not be less than 1mm. Photographs for insertion in the text will be well defined and printed on glossy white paper, and will be scaled down for printing to a width of 7,5 to 15 cm (3 to 6"). All markings on photographs are covered by the same recommendations as for figures. It is recommended that authors of communications accompany each figure or photograph with a descriptive title giving sufficient information on the content of the picture.

7. Tables

Tables of characteristics or values inserted in the text or appended to the article must be prepared in a clear manner, preferably as Camera Ready text. Should a table need several pages these must be kept together by sticking or other appropriate means in such a way as to emphasize the unity of the table.

8. Summaries

A summary of 10 to 20 typed lines written by the author in the English will precede and introduce each article.

9. Required information

Provide title, authors, affiliation, data of dispatch and a 100 to 250 word abstract on a separate sheet. Provide a separate sheet with exact mailing address for correspondence.

10. Reference

References must be listed alphabetically by the surname of the first author. List author(s) (with surname first), title, journal name, volum , year, pages for journal references, and author(s), title, city, publisher, and year for the book references. Examples for article and book respectively:

[1] Dawes, Robyn M. and Corrigan, Bernard: Linear models in decision making, Psychological Bulletin, **81** (1974), 95-106.

[2] Brown, Robert G.: Statistical Forecasting for Inventory Control, New York: McGraw-Hill, 1959.

All references should be indicated in the manuscript by the author's surname followed by the year of publication (e.g., Brown, 1959).

11. Reprints

Each author will receive 25 free reprints of his article.

PD 3818

This is Seagate Technology.

Seagate's line of hard disc drives is packed with high technology. And every one is built to the highest quality and reliability standards in the industry.

And now, Seagate drives are available locally for all your Personal Computer applications.

Only Seagate can offer you full technical support, and a one-year warranty. through our authorised representatives in your country.

Complete technical and interface details are included in the Seagate product brochures, which are free of charge to professional PC buyers and users. Simply use the coupon below to request your copies.

You'll soon see why Seagate has become the world's leading independent manufacturer of disc drives.

To: Seagate Technology Europe,
Seagate House, Fieldhouse Lane,
Globe Park, Marlow SL7 1LW Great Britain.

Please send me technical details of Seagate disc drives

Name _____

Job Title _____

Organisation _____

Address _____

_____

Country _____

Type of business _____

Number of employees _____ Number of PCs _____

☐ I use a PC ☐ I authorise the purchase of PCs

☐ I am a technical support manager

Seagate

Seagate Technology Europe
Seagate House, Fieldhouse Lane, Globe Park, Marlow SL7 1LW Great Britain.
Tel: 0628 890366 Fax: 0628 890660 Telex: 846218 SEAGAT G