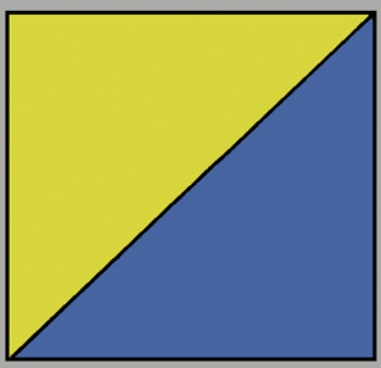
R. Eckmiller, Editor

ADVANCED NEURAL COMPUTERS



North-Holland

ADVANCED NEURAL COMPUTERS

This page intentionally left blank

ADVANCED NEURAL COMPUTERS

Edited by

Rolf ECKMILLER
Division of Biocybernetics
Department of Biophysics
Heinrich Heine University Düsseldorf
F.R.G.

1990

NORTH-HOLLAND
AMSTERDAM • NEW YORK • OXFORD • TOKYO

ELSEVIER SCIENCE PUBLISHERS B.V.
Sara Burgerhartstraat 25
P.O. Box 211, 1000 AE Amsterdam, The Netherlands

Distributors for the United States and Canada:

ELSEVIER SCIENCE PUBLISHING COMPANY INC.
655 Avenue of the Americas
New York, N.Y. 10010, U.S.A.

Library of Congress Cataloging-in-Publication Data

Advanced neural computers / edited by Rolf Eckmiller.
p. cm.
Includes bibliographical references and index.
ISBN 0-444-88400-9 (U.S.)
1. Neural computers--Congresses. 2. Neural networks (Computer science)--Congresses. I. Eckmiller, Rolf, 1942- .
QA76.87.A37 1990
006.3--dc20 90-39685
CIP

ISBN: 0 444 88400 9

Printed in The Netherlands

PREFACE

This book is the outcome of the International Symposium on Neural Networks for Sensory and Motor Systems (NSMS) held in Neuss (near Düsseldorf (FRG)) from 22 to 24 March, 1990.

The NSMS symposium assembled 45 invited experts from Europe, America, and Japan representing the fields of Neuroinformatics, Computer Science, Computational Neuroscience, and Neuroscience.
More than 150 additional scientists from various countries representing a number of research institutes and companies with interest in neural computing participated in the discussions and made poster presentations.

The 45 invited contributions in this book are arranged in six sections ranging from Biological Sensory and Motor Systems via Theory of Artificial Neural Networks and Neural Network Simulators to Pattern Recognition and Motor Control with Artificial Neural Networks.

The readibility of this book was enhanced by a number of measures:
* The invited papers are arranged in six sections.
* The collection of References from all Contributions provides an alphabetical list of all references quoted in the individual contributions.
* A separate List of General References serves newcomers in this field to find other recent books.
* Separate Author and Subject Indices facilitate access to various details.

It is hoped that this book as a rapidly published report on the 'State of the Art in Neural Computing' and as a reference book for future research in the highly interdisciplinary field of 'Theory and Applications of Neural Computers' will provide useful in the endeavour to: *Transfer Concepts of Brain Function and Structure to Novel Neural Computers with Adaptive, Dynamical Neural Net Topologies.*

The editor wishes to thank Claudia Berge, Miriam Buck, and Sandra Winter for their efficient and thorough technical and managerial assistance, which was essential to meet the tight deadline for preparation of the final book manuscript.
The subsequent expert work of the publisher made it possible to make this book available in high quality within less than four months after the NSMS symposium.

Düsseldorf, June 1990 The Editor

ACKNOWLEDGEMENT OF SPONSORSHIP

The generous sponsorship of several American and German Governmental Ministries and Agencies, which made the International Symposium on Neural Networks for Sensory and Motor Systems (NSMS) and thereby this book possible, is gratefully acknowledged:

Air Force Office of Scientific Research (AFSC) / Washington (USA)

Defence Advanced Research Project Agency (DARPA) / Arlington (USA)

Minister für Wissenschaft und Forschung des Landes Nordrhein-Westfalen / Düsseldorf (FRG)

Office of Naval Research (ONR) / Arlington (USA)

Office of Naval Research (ONR), European Office / London (UK)

TABLE OF CONTENTS

Section 1: General Introduction

Section 2: Biological Sensory and Motor Systems

Section 3: Theory of Artificial Neural Networks

Section 4: Neural Network Simulators

Section 5: Pattern Recognition with Neural Networks

Section 6: Motor Control with Neural Networks

Section 1
General Introduction

This page intentionally left blank

Advanced Neural Computers / R. Eckmiller (Editor)

Prolegomena to an Analysis of Form and Structure

Eduardo R. Caianiello

Dipartimento Fisica Teorica
University of Salerno
Salerno, Italy

I. INTRODUCTION

1. - This title is meant as an expression of realistic humility, in front of problems whose extension and depth I deem to be much beyond our present grasp. As an example, the task of understanding in full the structure of English or Japanese, and of making an acceptable translation from one into the other, appears of a magnitude comparable to that of the study of cerebral activity. This was in fact the reason that prompted our first researches on the subject in the early 60's [1,2]: our purpose was to learn something about the latter by studying the "produce" of the brain, conceived as a biological machine (almost a heresy at that time) whose inner workings we had just started modeling with mathematical equations describing Neural Nets and their learning mechanisms.

Our methodological approach provided several insights, which were, when possible, subjected to computer analyses (of texts in several languages) that well corresponded to our expectations, thus stimulating further researches, still under way. Implementation with Neural Nets is being also considered, but will not be discussed here. I present it, in sketchy outline but with several improvements with respect to earlier studies, because I think it paradigmatic for many problems of interest to us. E.g., a robot may be described by specifying the "translation" it performs from a "sensory" to a "motor" language: a problem for which the appellative "prolegomena" given to these considerations seems not inappropriate. They may be taken, in a generalized sense, to belong to "mathematical linguistics". The major difference between most approaches in this area and ours is that we consider the language which we happen to study as our "universe", which we treat as a physicist does with his: he cannot change it at will and is forbidden from making a priori assumptions on its laws. Likewise, we demand that what structures, grammars, etc. may exist in a "language" must be found by applying a systematic methodology, not postulated as already known.

The major conceptual difference from earlier works is given by our refounding the whole approach on properties of the Kullback-Leibler entropy. Basic notions become more perspicuous and rigorous, some statements proposed before as heuristic become mathematical corrolaries.
We name this improved version "Procrustes II", since our previous work was called

"Procrustes" (after the name of the first theorist in our history)

2.- The "form of an object is understood as a quality bestowed upon it by the "observer" (a typical Kantian phenomenon). It includes "pattern", "image", etc. . "Form" is thus a "set property"; the "observer" is defined in turn through the set of elements stipulated by him to have the same form (e.g. according to weight, colour, shape, nutritional value...) . It is well known that a "set consisting of a single element" is a quite different object from that single element alone: the difference is the observer! This raises profound system-theoretical questions (think also of Quantum Mechanics), which will not concern us here. The operation performed by the observer when defining a form coincides with "abstraction", typical, in our approach, of learning neural nets [3]. Iteration of this operation creates hierarchies and heterarchies of sets, or forms, or "features" (as such partial sets are often called when seen from a higher level). The study of this global process we call "Structural Analysis". Considerations which we do not report here [4] show that it involves usually a "Quantification" into the discrete (and finite) domain (think of our "seven" notes or colours as contrasted with the underlying physical continuum). We shall assume, as starting point, that the objects we deal with are quantified, either naturally or because of some preliminary pre-processing.

We propose to analyze structures as "nested sequences of substructures". The analogy with a typed linguistic text is cogent; we shall use, proceeding "bottom-up", the words "letter", "syllable", word",...
This may be misleading, because it implies an underlying "linear dimension" as for a typwritten text, i.e. the existence of "complete ordering" relations, which may well not be"natural" in the analysis, say, of pictures. In such cases we assume, for the purpose of the present discussion, such relations to be the result of some prescription, as always possible in the discrete: think of TV scanning, or of the saccadic eye movements of Yarbus. By so doing, we deal in fact with the most complex situation, that of "strings" of elements in which "order" is essential; suppression of order would simplify our treatment into that of "clusters", an easier task not treated here (it would imply a reconsideration of the "codes" discussed next).

3.- We shall call henceforth "language" a given (extensively or potentially, finite or infinite) set of "texts" written in terms of "letters" of an "alphabet" A (e.g. English letters, or Kanji and Kana). Our attention will be concentrated on a single text T (the addition of more texts will enrich later our knowledge). A "code" is a collection of strings (code words) of letters of A. We defined [5] a code "closed" if "left cancellation" of a code word gives again a code word: thus, if $a_1a_2a_3$ is a code word , a_2a_3 and a_3 are also code words .

"Natural codes" are defined as those which contain "terminal letters": in our analysis a subset of them, "closed natural codes" (CNC), will play a relevant role.

"Instantaneous codes" are the subclass of "uniquely decipherable codes" for which the end of a code word is recognizable without exploration of the next letter. We defined "closed instantaneous codes" (CIC) the subclass of instantaneous codes which is closed under left cancellation, except that it does not contain as words suffixes which have code words as prefixes: thus $a_1a_2a_3a_4$; a_3a_4; a_3a_2; a_2; a_4 is a CIC (the suffix $a_2a_3a_4$ is forbidden). CIC's contain terminal symbols also within their code words, and are thus wider than CNC's, in terms of which they can be further analyzed.

The interest of CIC and CNC lies for us in the fact that our algorithm was proved to converge always to one of these two codes (applications to texts written with other types of codes prove this assertion; for a discussion, see ref. [5])

II - Preliminary Remarks

1.- Two main topics are relevant for our structural analysis:

- frequency count;

- search for individual structures, no matter how rare.

We have restricted our attention only to the second, which presented the greater challenge and is a necessary pre-requisite for the first; a study of texts in various languages, especially Italian, substantiated our conclusions. There is of course no doubt that both topics are important in a complete analysis of the type we propose.

2.- Our search for structural levels has to meet two basic demands:

- finding, proceeding bottom-up, a hierarchy of nested substructures, none excluded, in terms of which our texts may be fully described;

- destroying the evident influence of higher structures on the distribution of lower ones. (One may read thus most of Italian literature without ever meeting the word "soqquadro", which denotes a situation of "total upsetting wantonly created", and is the only Italian word to have the sequence "qqua"; yet this string cannot be ignored

by our search!)

The second is easily answered at the first two steps:

- transition from the study of "language" to that of a single "text" T .

- transition from the text T to the corresponding "reduced text", or vocabolary, W , which contains each word of T only once.

The latter is paradigmatic for our procedure; the reader may compare our approach, for analogies and differences, with C. Shannon's celebrated analysis of English. Assume level "1" to be given by the alphabet letters (because given, or as a higher level from some binary, or Morse, code); call level "K" that given by all the words of the reduced text W . Think now of two sets of B. Russell's monkeys, one with letter-keyboards, the other with word-keyboards: the first will reproduce all that we want and can of structure at level "1", the second at level "K". How the destruction of the damaging influence of higher levels comes about is quite clear in both cases.

3.- Our aim is to obtain, at whatever levels may exist between "letters" and "words", this same result. It suffices to consider only one intermediate level between the two just named, that of "syllables". It is the old problem of "parsing", when no indications are elicitable from the text on how to divide words into syllables. We can replace monkeys with semigroup theory: we demand that our "syllables", if written as strings of alphabet letters, generate a "submonoid" (that covers W) which is a "free submonoid" of the monoid generated by the letters. Our algorithms must therefore automatically, on reading the text, retrieve the generators of the free submonoid that is smaller than the monoid generated by letters and larger than that generated by words. (We have assumed just one intermediate level, but the extension to whole hierarchies should be evident).

III - Information as Kullback-Leibler Entropy

1.- Our purpose is not to repeat statements and proofs already available in the references, but rather to "distill" out of them, with greater mathematical clarity than was then possible, the crucial points that we deem of value for further investigation; the result ought to be a vast improvement, because the use of "conjectures" then made, turned now into legitimate mathematical statements, will allow great gains in computational speed.

We only discuss here, as an illustration, the paradigm "letter - syllable - word" just quoted. If we denote A* the monoid generated by the alphabet A,
we propose to find a set of syllables S such that:

$$A^* \supset S^* \supset W^*$$ (in the strong sense, or else there is no intermediate level between A and W).

2.- We are interested in structures, not in frequencies. We build therefore at each stage probability schemes of "microcanonical" type, as follows. Our search for S must start and stop automatically (words are assumed to have a finite maximal length); it proceeds, by iteration, through the construction of "intermediate provisional alphabets" containing letters, digrams, trigrams..., until the previous alphabet is identically reproduced. It constitutes the wanted S : our procedure, explained next, guaranties that S is a CNC or a CIC.

Call X_α a string of (one or more) letters, y_h the letter next to it at right in the word. Call $p(X_\alpha\, y_h)$ the probability (to be specified later) that the string $X_\alpha\, y_h$ exists in W. Consider then the marginal probabilities

$$p(X_\alpha) = \Sigma_h\ p(X_\alpha\, y_h), \qquad p(y_h) = \Sigma_\alpha\ p(X_\alpha\, y_h)$$

and the conditional probabilities

$$p(y_h / X_\alpha) = p(X_\alpha\, y_h) / p(X_\alpha);$$
$$p(X_\alpha / y_h) = p(X_\alpha\, y_h) / p(y_h);$$

form the entropies

$$H(Y) = -\Sigma_h\ p(y_h) \log p(y_h)$$

$$H(Y / X) = -\Sigma_h\ p(y_h / X_\alpha) \log p(y_h / X_\alpha)$$

$$H_X(Y) = \Sigma_\alpha\ p(X_\alpha)\, H(Y / X\alpha)$$ (equivocation)

and from them the "mutual information", or "divergence", or "Kullback-Leibler entropy" (all synonims; the last concept is the most interesting for us, as it implies a "measure" of how near we are to our aim):

$$H(X;Y) = H(Y) - H_X(Y) = H(X) + H(Y) - H(X,Y) =$$

$$= \Sigma_{\alpha,h}\ p(X_\alpha\, y_h)\ \log\{p(X_\alpha\, y_h) / p(X_\alpha)\, p(y_h)\}\,.$$

The gain in information with respect to the average information on y when X_α is known is:

$$\mathfrak{F}(X_\alpha) = H(Y) - H(Y / X_\alpha)$$

3. - Proceed now as follows.

Step 1):

a) Construct a histogram having as abscissae and ordinates the letters of A; set in it a "1" whenever the digram $X_\alpha\, y_h$ exists in W, no matter how many times, a "0" otherwise (here, X_α is a single letter of A);

b) Call D the total number of 1's in histogram (we treat only the case $D < A^2$ here, denoting with A also the number of letters of A); set

$$p(X_\alpha\, y_h) = \begin{cases} 1 / D & \text{if } X_\alpha\, y_h \in D \sqsubset A^2 \\ 0 & \text{if } X_\alpha\, y_h \notin D \end{cases}$$

c) Compute $\mathfrak{F}(X_\alpha)$ for all values on abscissa; if

$\mathfrak{F}(X_\alpha) \leq H(X;Y)$ take all monograms X_α as letters of first provisional alphabet;

if

$\mathfrak{F}(X_\alpha) > H(X;Y)$ remove X_α from letteral alphabet A, complete first provisional alphabet with all such digrams.

Step 2):

Set as abscissae X_α all elements of first provisional alphabet, as ordinates all letters of A. Iterate step 1).

Step 3) etc.: Iterate.

End: When iteration reproduces previous alphabet. This is taken as that of wanted syllables. Wanted structure is found.

The main point is now easily understood. Looking at the Kullback-Leibler entropy, we see that it vanishes when $p(X_\alpha\ y_h) = p(X_\alpha)p(y_h)$: the wanted "syllabic" alphabet is reached when none of the symbols that have been constructed with our procedure yields information on the next letter, as expressed by the factorization of the probability. This situation is "ideal", because any text we examine can only be finite; one can however (thus destroying the constraints posed by word-structure on syllable-finding) easily replace the "word level" with one in which "all" syllables are assumed freely adjoinable. This, again, acts like a thermodynamic limit, and simplifies computation greatly. It also implies that $p(y_h)$ is taken not as the marginal probability of y_h ("last" letter of a string), but of y_h as any letter of the alphabet (assumed equiprobable). In earlier works we found, empirically, this to be indeed the best choice.

3.- Discussion.

We have thus built a sequence of "vocabularies", each listing the structures of a level.. This is only a first start, which however indicates several problems that can be studied next with in ways not only heuristic. They are mentioned in ref.[5-8]. Our specific aim demanded the exclusion of direct "frequency counts" (but information on the frequency of a symbol among those of higher levels might be usefully obtained). Such counts will be of course of major importance for other parts of the structural analysis we propose, of which only the mere beginnings are here reported.

References

[1] Caianiello, E.R. and Crocchiolo, Programma Procuste per l'Analisi di Linguaggi Naturali, Calcolo, 2, 83, (1965).

[2] Caianiello, E.R., On the analysis of natural languages, Odessa, Proceedings of the 3rd All Union SSR Congress on Cybernetics (1965).

[3] Caianiello, E.R., Outline of a theory of thought processes and thinking machine, J. Theor. Biol., 1, p. 209 (1961).

[4] Caianiello, E.R., Some remarks on organization and structure, Biological Cybernetics,26,p. 151, (1977).
[5] Caianiello, E. R. and Capocelli, R.M., On form and language: the Procuste's algorithms for feature extraction, Kybernetik, 8,p. 223, (1971).
[6] Caianiello, E.R. and Capocelli, R.M., Structural analysis of hierarchical systems, Proc. of 3rd joint conference Pattern Recognition, (1976).
[7] Caianiello , P., Neural models, structure and learning, Proc. of 1st Italian workshop on parallel architecture and neural networks, (Word Scientific, Singapore, 1988).
[8] Caianiello , P., Learning by data compression in neural nets, Proc. of2nd Italian workshop on parallel architecture and neural networks, (Word Scientific, Singapore, 1988).

Advanced Neural Computers / R. Eckmiller (Editor)

The Truck Backer-Upper: An Example of Self-Learning in Neural Networks

Derrick Nguyen and Bernard Widrow

Information Systems Laboratory
Department of Electrical Engineering
Stanford University
Stanford, CA 94305

Neural networks can be used to solve highly nonlinear control problems. A two-layer neural network containing 26 adaptive neural elements has learned to back up a computer simulated trailer truck to a loading dock, even when initially "jackknifed." It is not yet known how to design a controller to perform this steering task. Nevertheless, the neural net was able to learn of its own accord to do this, regardless of initial conditions. Experience gained with the truck backer upper should be applicable to a wide variety of nonlinear control problems that appear in power systems.

1 Introduction

The control of severely nonlinear systems has for the most part escaped the attention of control theorists and practitioners. This paper addresses the issue from the point of view of utilizing self-learning techniques to achieve nonlinear controller design. The methodology shows promise for applications to control problems that are so complex that analytical design techniques either do not exist or will not exist for some time to come. Neural networks can be used to implement highly nonlinear controllers whose weights or internal parameters can be chosen or determined by a self-learning process.

Backing a trailer truck to a loading dock is a difficult exercise for all but the most skilled truck drivers. Anyone who has tried to back up a house trailer or a boat trailer will realize this. Normal driving instincts lead to erroneous movements. A great deal of practice is required to develop the requisite skills.

When watching a truck driver backing toward a loading dock, one often observes the driver backing, going forward, backing again, going forward, etc., and finally backing to the desired position along the dock. The forward and backward movements help to position the trailer for successful backing up to the dock. A more difficult backing up sequence would only allow backing, with no forward movements permitted. The specific problem treated in this paper is that of the design by self-learning of a nonlinear controller to control the steering of a trailer truck while backing up to a loading dock from an arbitrary initial position. Only backing up is allowed. Computer simulation of the truck and its controller has demonstrated workability, although no mathematical proof yet exists. The experimental controller contains 26 adaptive ADALINE units [1] and exhibits exquisite backing up control. The trailer truck can be initially "jackknifed" and aimed in many different directions, toward and away from the dock, but as long as there is sufficient clearance, the controller appears to be capable of finding a solution.

Figure 1 shows a computer-screen image of the truck, the trailer, and the loading dock. The critical state variables representing the position of the truck and that of the loading dock are θ_{cab}, the angle

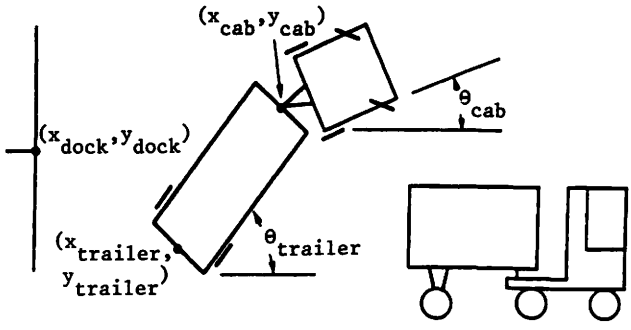


Figure 1: The truck, the trailer, and the loading dock

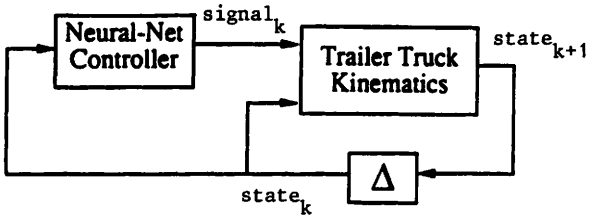


Figure 2: Overview Diagram

of the truck, x_{cab} and y_{cab}, the cartesian position of the yoke, $x_{trailer}$ and $y_{trailer}$, the cartesian position of the rear of the center of the trailer, and x_{dock} and y_{dock}, the cartesian position of the center of the loading dock. Definition of the state variables is illustrated in Figure 1.

The truck backs up until it hits the dock, then stops. The goal is to cause the back of the trailer to be parallel to the loading dock, and to have the point $(x_{trailer}, y_{trailer})$ be aligned as closely as possible with point (x_{dock}, y_{dock}). The controller will learn to achieve this objective.

2 Training

The approach to self-learning control that has been successfully used with the truck backer-upper involves a two-stage learning process. The first stage involves the training of a neural network to be an emulator of the truck and trailer kinematics. The second stage involves the training of a neural-network controller to control the emulator. A similar approach has been used by Widrow [2, 3] and by Jordan [4]. Once the controller knows how to control the emulator, it is then able to control the actual trailer truck. Figure 2 gives an overview, showing how the present state vector $state_k$ is fed to the controller which in turn provides a *steering signal*$_k$ between -1 (hard right) and $+1$ (hard left) to the truck. The time index is k. Each time cycle, the truck backs up by a fixed small distance. The next state is determined by the present state and the steering signal, which is fixed during the cycle.

Figure 3 shows a block diagram of the process used to train the emulator. The truck backs up randomly, going through many cycles with randomly selected steering signals. By this process, the emulator "gets the feel" of how the trailer and truck behave. The emulator, chosen as a two layer

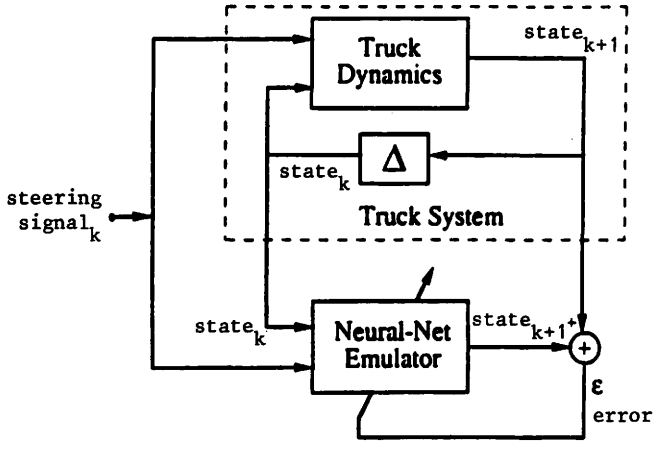


Figure 3: Training the neural-net truck emulator

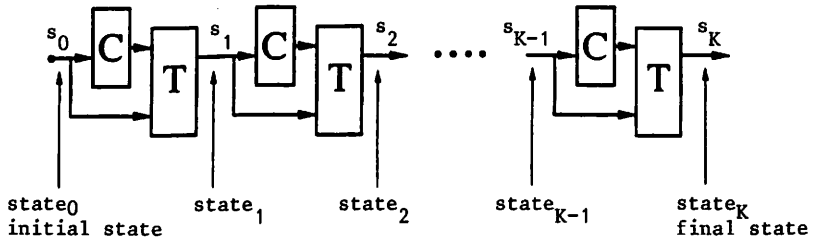


Figure 4: State transition flow diagram

neural network, learns to generate the next positional state vector when given the present state vector and the steering signal. This is done for a wide variety of positional states and steering angles. The two-layer emulator is adapted by means of the back-propagation algorithm [5, 6, 7]. The first layer had six present state inputs plus the present steering signal input. This layer contained forty five hidden adaptive ADALINE units producing six next-state predictions. Once the emulator is trained, it can then be used to train the controller.

Refer to Figure 4. The identical blocks labeled C represent the controller. The identical blocks labeled T represent the truck and trailer emulator. Suppose that the truck is engaged in backing up. Let C be chosen randomly and be initially fixed. The initial state vector s_0 is fed to C, which produces the steering signal output which sets the steering angle of the truck. The backing up cycle proceeds with the truck and trailer soon arriving at the next state s_1. With C remaining fixed, the backing up process continues from cycle to cycle until the truck hits something and stops. The final state s_K is compared with the desired final state (the rear of the trailer parallel to the dock with proper positional alignment) to obtain the final state error vector ϵ_K. This error vector contains three elements (which are the errors of interest), $x_{trailer}$, $y_{trailer}$ and $\theta_{trailer}$, and is used to adapt the controller C.

The method of adapting the controller C is illustrated in Figure 5. The final state error vector ϵ_K is used to adapt the blocks labeled C, which are maintained identical to each other throughout the

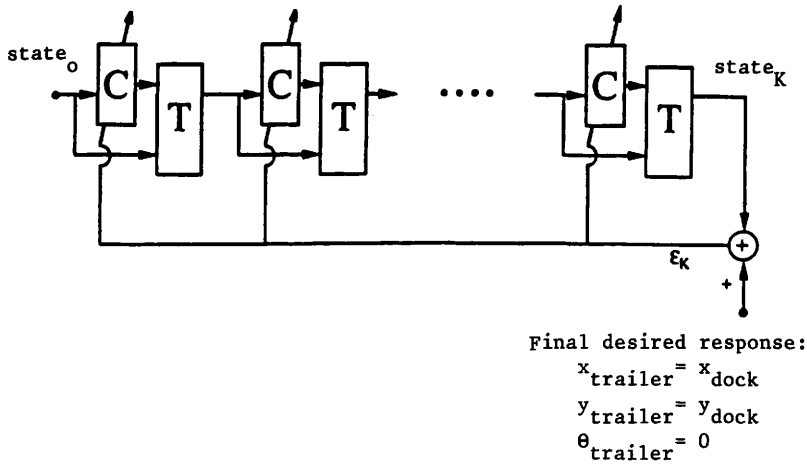


Figure 5: Training the controller with back-propagation

adaptive process. The controller C is a two-layer neural network. The first layer has the six state variables as inputs, and this layer contains twenty five adaptive ADALINE units. The second or output layer has one adaptive ADALINE unit and produces the steering signal as its output.

The procedure for adapting C goes as follows. The weights of C are chosen initially at random. The initial position of the truck is chosen at random. The truck backs up, undergoing many individual back up cycles, until it stops. The final error is used by back-propagation to adapt the controller. Each of the C blocks could be tentatively adapted by back-propagation if they were independent of each other, but the actual weight changes in C are taken as the sum of the tentative changes. In this way, the C blocks are maintained identical to each other. The weights are changed by this constrained back-propagation algorithm to reduce the sum of the squares of the components of the final state error ϵ_K by following the negative of the gradient, using the method of steepest descent. The entire process is repeated by placing the truck and trailer in another initial position, and allowing it to back up until it stops. Once again, the controller weights are adapted. And so on.

Figure 6 shows details of one of the state transition stages of Figure 5. One can see the structure of controller C and of emulator T, and how they are interconnected. Each stage of Figure 5 amounts to a four-layer neural network. The entire process of going from an initial state to the final state can be seen from Figures 4 and 5 to be analogous to a neural network having a number of layers equal to four times the number of backing up steps when going from the initial state to the final state. The number of steps varies of course with initial position of the truck and trailer.

The diagram of Figure 5 was simplified for clarity of presentation. The output error does not go directly to the C-blocks as shown, but back-propagates through the T-blocks and C-blocks. Thus, the error used to adapt each of the C-blocks does originate from the output error ϵ_K, but travels through the proper back-propagation paths. For purposes of back-propagation of the error, the T-blocks are the truck emulator. But the actual truck kinematics are used when sensing the error ϵ_K itself.

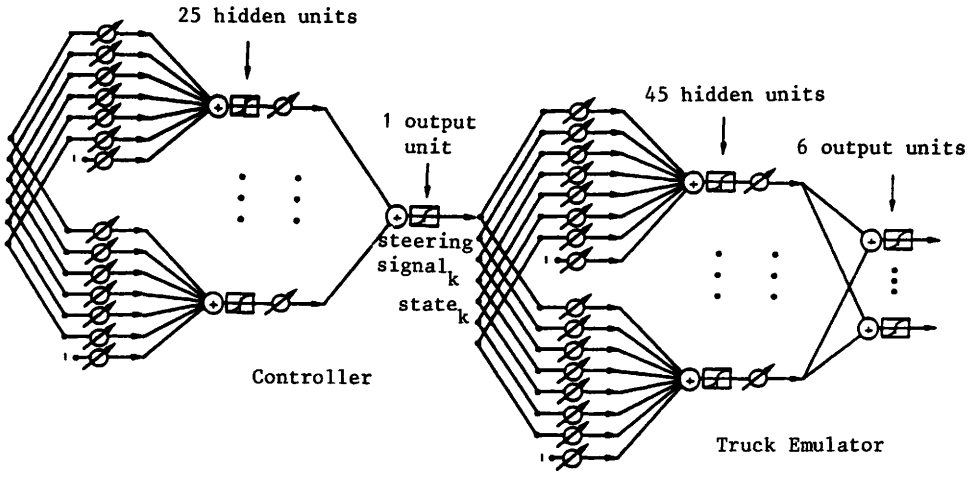


Figure 6: Details of emulator and controller

3 Summary and Results

The truck emulator was able to represent the trailer and truck when jackknifed, in line, or in any condition in between. Nonlinearity in the emulator was essential to represent the truck and trailer. The angle between truck and trailer were not small. *Sin* θ could not be represented approximately as θ. Nonlinearity in the controller was also essential. Self-learning processes were used to determine the parameters of both the emulator and the controller. Thousands of back ups were required to train these networks. Without the learning process however, substantial amounts of human effort and design time would have been required to devise the controller.

The training of the controller was divided into several "lessons". In the beginning, the controller was trained with the truck initially set to points very near the dock and the trailer pointing at the dock. Once the controller was proficient at working with these initial positions, the problem was made harder by starting the truck farther away from the dock and at increasingly difficult angles. This way, the controller learned to do easy problems first and more difficult problems after it mastered the easy ones. There were 16 lessons in all. In the easiest lesson the trailer was set about half a truck length from the dock in the x direction pointing at the dock, and the cab at a random angle between ± 30 degrees. In the last and most difficult lesson the rear of the trailer was set randomly between 1 and 2 truck lengths from the dock in the x direction and ± 1 truck length from the dock in the y direction. The cab and trailer angle was set to be the same, at a random angle between ± 90 degrees. The controller was trained for about 1000 truck backups per lesson during the early lessons, and 2000 truck backups per lesson during the last few. It took altogether about 20000 backups to train the controller.

The controller learned to control the truck very well with the the above training process. Near the end of the last lesson, the root mean square error of $y_{trailer}$ was about 3 percent of a truck length. The root mean square error of $\theta_{trailer}$ was about 7 degrees. There is no error in $x_{trailer}$ since a truck backup is stopped when $x_{trailer} = x_{dock}$. One may, of course, trade off the error in $y_{trailer}$ with the error in $\theta_{trailer}$ by giving them different weightings during training.

Results with the adaptive controller are illustrated in Figures 7, 8, 9, and 10. The controller has already been trained and its weights remained fixed for all the experiments. The truck and trailer

were placed in a variety of initial conditions, and backing up was effected in each case. Initial and final states are shown in the computer screen displays, and the dynamics of backing up is illustrated by the time-lapse plots.

The truck backer-upper learns to solve sequential decision problems. The control decisions made early in the backing up process have substantial effects upon final results. Early moves may not always be in a direction to reduce error, but they position the truck and trailer for ultimate success. In many respects, the truck backer-upper learns a control strategy that is like a dynamic programming problem solution. The learning is done in a layered neural network. Connecting signals from one layer to another corresponds to idea that the final state of a backing up cycle is the same as the initial state of the next backing up cycle.

Future research will be concerned with:

- Determination of complexity of emulator as related to complexity of the system being controlled.
- Determination of complexity of controller as related to complexity of emulator.
- Determination of convergence and rate of learning for emulator and controller.
- Proof of robustness of control scheme.
- Analytic derivation of non-linear controller for truck backer-upper, and comparison with self learned controller.
- Re-learning in the presence of movable obstacles.
- Application of self-learning neural networks to control of power generators, power system apparatus, and to power distribution systems.

This research was sponsored by SDIO Innovative Science and Technology Office and managed by ONR under contract #N00014-86-K-0718, by the Department of the Army Belvoir R D & E Center under Contract #DAAK70-89-K-0001, and by grants from the Thomson CSF Company and the Lockheed Missiles and Space Company.

This material is based on work supported under a National Science Foundation Graduate Fellowship. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] B. Widrow and M. E. Hoff, Jr. Adaptive switching circuits. In *1960 IRE WESTCON Conv. Record, Part 4*, pages 96–104, 1960.

[2] Bernard Widrow and Samuel D. Stearns. *Adaptive Signal Processing.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

[3] B. Widrow. Adaptive inverse control. In *Adaptive Systems in Control and Signal Processing 1986.* International Federation of Automatic Control, July 1986.

[4] M. I. Jordan. Supervised learning and systems with excess degrees of freedom. COINS 88-27, Massachusetts Institute of Technology, 1988.

[5] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, Cambridge, MA, August 1974.

[6] D.B. Parker. Learning logic. Technical Report TR-47, Center for Comput. Res. Econ. and Manage., Mass. Inst. Technol., 1985.

[7] David E. Rumelhart and James L. McClelland, editors. *Parallel Distributed Processing*, volume 1, chapter 8. The MIT Press, Cambridge, Mass., 1986.

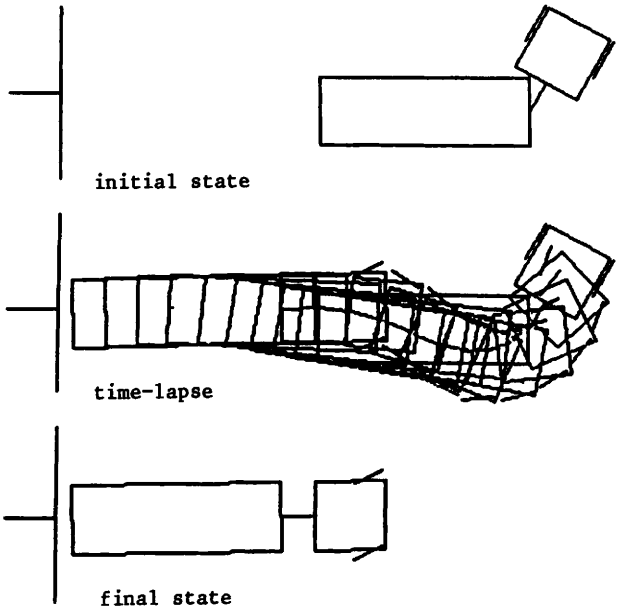


Figure 7

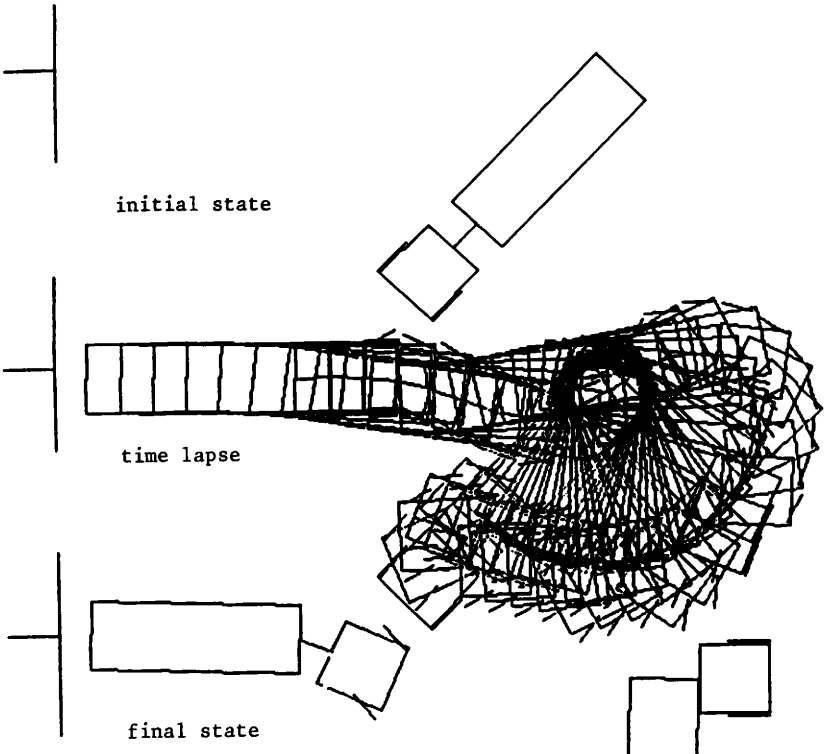


Figure 8

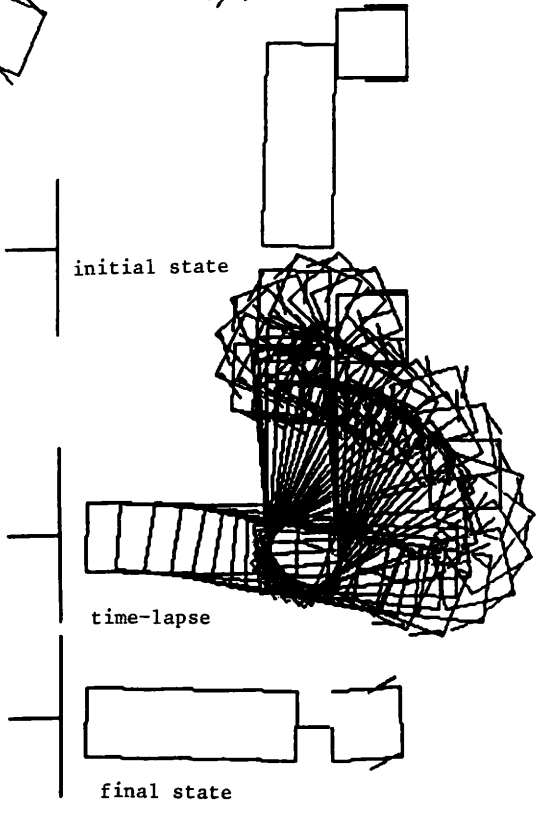


Figure 9

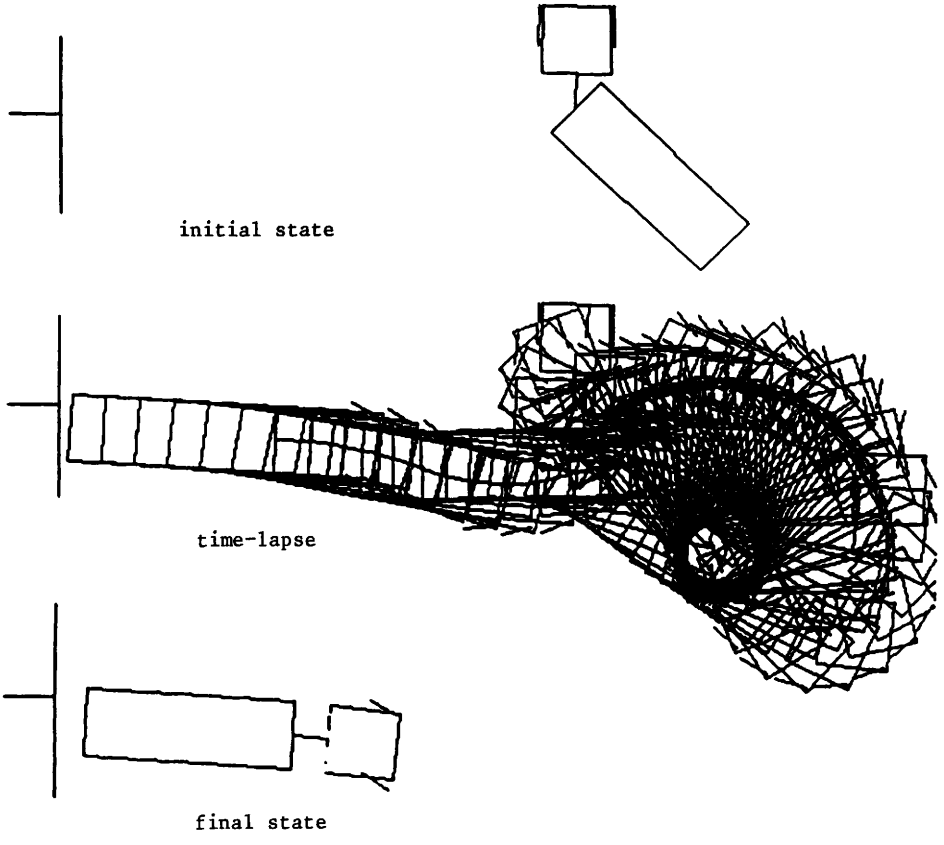


Figure 10

This page intentionally left blank

Advanced Neural Computers / R. Eckmiller (Editor)

DIE LERNMATRIX - THE BEGINNING OF ASSOCIATIVE MEMORIES

Karl STEINBUCH

Formerly University of Karlsruhe, Germany (Now retired)

1. INTRODUCTION

Comparison of information processing by man and by computers (of John von Neumann's structure) shows:
+ Computers can simulate all - or most - information processing modes of man.
+ But for very simple modes of man's information processing, computers are less appropriate (for example for pattern recognition).

This was the starting point of considerations about 30 years ago - and I rapidly came to the "conditioned reflexes", which the famous russian physiologist I.P. Pavlov had discovered at the beginning of our century.

In Pavlov's well known experiment, a dog was conditioned to salivate upon the signal of a bell - a signal, which has been without any effect before.

Conditioned reflexes suppose circuits, whose rules of operation are being changed as a function of input information. Apparently they provide for storage of variable logical operation.

I proposed several simple technical realizations of conditioned reflexes, for example formation of threads of metallic silver in an environment of silver bromide or ferro-magnetic realizations which are useful for reversible connections.

Basic idea hereby was not to simulate organisms, but rather to find simple technical prinziples for learning processes.

In order not to confuse terminology, the expression "conditioned reflex" will not be used, but "conditioned connection".

Basically, two kinds of circuitry could be conceived, dependend on whether binary or non-binary patterns are to be processed.

2. BINARY LEARNING MATRIX

The structure of a learning matrix is indicated in Figure 1.

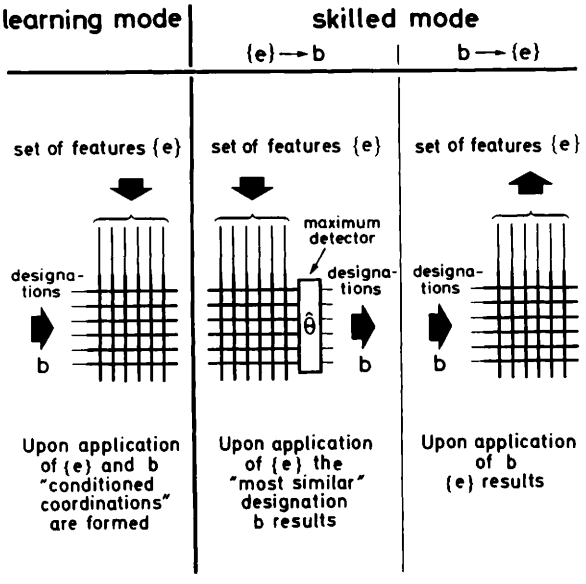


FIGURE 1

The Learning matrix, principle

Two sets of conductors intersect forming a matrix like structure. The cross-points of the matrix behave as" conditioned connections". The vertical conductors carry a set of features or a pattern $\{e\}$ consisting of binary signals, e.g. black and white.

The horizontal set of conductors carries statements of designations b . In this set only one at a time will be marked by a signal, the statements of designation are offered in a "1-out-of-m-code".

Two modes of operation can be distinguished:

1) a learning state, during which both a pattern $\{e\}$ and an associated designation b are applied to the learning matrix and therefore the conditioned connections may be built up, and
2) a skilled state or knowing phase during which alternatively
   + upon application of an arbitrary set of features $\{e\}$ the most similar designation b or
   + upon application of designation b the associated pattern $\{e\}$ will be signalized.

In some cases it is useful and by special arrangements of realization as well possible to gradually change the learned coordination between patterns e and particular designation b simply by continuous operation (feature of readjustment). The conditioned connections mentioned above are in this sense functional elements whose logical operation changes with experience or former performance.

How learning may be achieved, depends on the special technical realization of the matrix.

In the case of an electro-chemical realization, no conductivity exists at all at the beginning of the learning state.

In the case of a ferro-magnetic realization, the magnetic elements representing intersecting points are at the beginning in the zero point of the hysteresis loop.

Learning is performed by applying the set of features mentioned before to the column inputs and at the same time a certain signal to the special row into which the pattern is to be stored.

In fig. 2 it is assumed, that four different patterns have been learned by the matrix in the sense discussed.

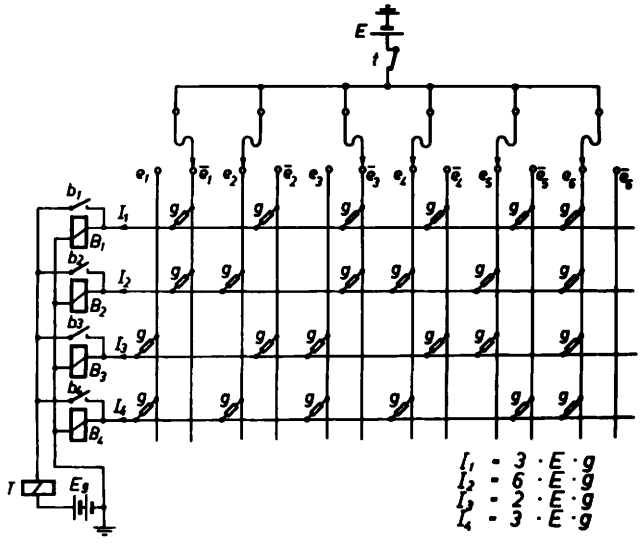


FIGURE 2

Learning matrix, knowing state

As fig. 2 shows, for each input element one double column has been provided. This is necessary in order to make use of the information e = 0 , otherwise the pattern {e} = 11111 would suppress all the other ones.

When learning is completed, skilled or knowing mode operation may start. The principle of it, too, is indicated in fig. 2. Again a pattern is presented to the column heads, let us say in form of a set of voltages, which are now coupled to the left hand column input if the particular feature **e** is 1 or to the right hand one if it is 0 . In fig. 2 {e} = 010111 is applied as an input. On the rows leads the coincidences with the stored patterns are summed up.

Apparently the one having the greatest similarity to the pattern applied will carry the largest current.

A coincidence in all positions causes a current contribution by all double columns. Hence the current on the row corresponding to the pattern applied (provided it has been learned before by the matrix) will have a maximum value. Lack of coincidence in one or more positions will result in smaller currents. Actually one can state that the row current will be smaller the larger the Hamming distance is between the pattern applied and the one stored in the particular row. The largest current will operate the correspondent relay which upon operation switches the fast relay T disconnecting the inputs. This is to demonstrate the behaviour of the maximum detection circuit, which in reality consists of transistors, diodes a.s.o.

In summary, a learning matrix determines the most similar pattern to a given input set of events, this is a maximum likelihood detection.

So far only conductivities at the intersection points have been discussed. Probably of greater importance are ferro-magnetic storage elements providing for the additional features of reversibility of the matrix and its learning processes and for relearning. By making use of half-current pulses of short duration (as compared with the switching time of the magnetic material) a core representing a storage element can be changed stepwise like a multiple state magnetic counter.

Hereby the stored value corresponds to a whole series of applied patterns each changing the row elements by just a small amount. Reading has to be nondestructive, the signals of all rows are fed into a maximum detection circuit selecting again the row containing the most similar pattern to the one applied.

3. NON-BINARY LEARNING MATRIX

Besides binary learning matrices discussed so far another type can be conceived dealing with analogue signals.

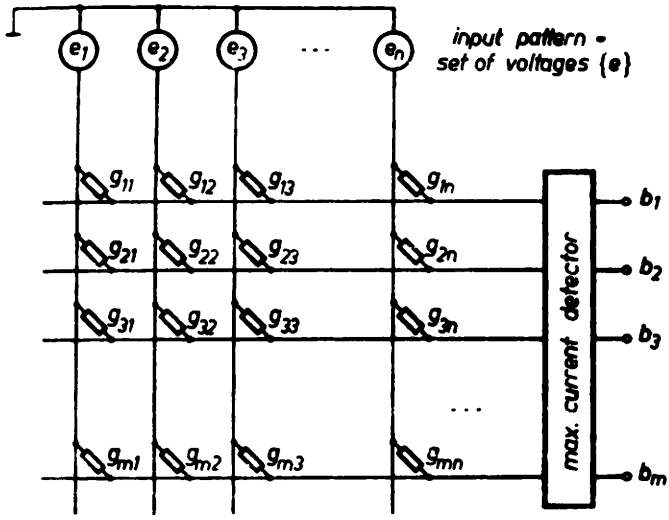


FIGURE 3

Non - binary Learning matrix

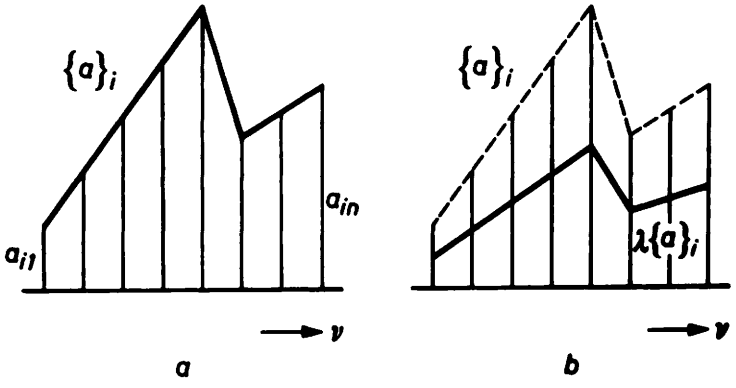


FIGURE 4
Sampled Patterns

Fig. 4 a shows a typical pattern or set of features which can be processed by the matrix. Into the columns a signal is fed, e.g. a voltage e corresponding in magnitude to the particular sample value of the analogue pattern. By an appropriate learning process the elements along a certain row likewise have been brought to a proportional value a_{iv}.

A simple calculations shows:

+ the learned patterns must be normalized with respect to their energy. The learned (stored) patterns are called "unity patterns" or "perceptual forms".
+ the recognition by the learning matrix is invariant to affine transformed patterns. Affine transformation fo a pattern means multiplying each one of its attributes by the same positive constant (Fig. 5).

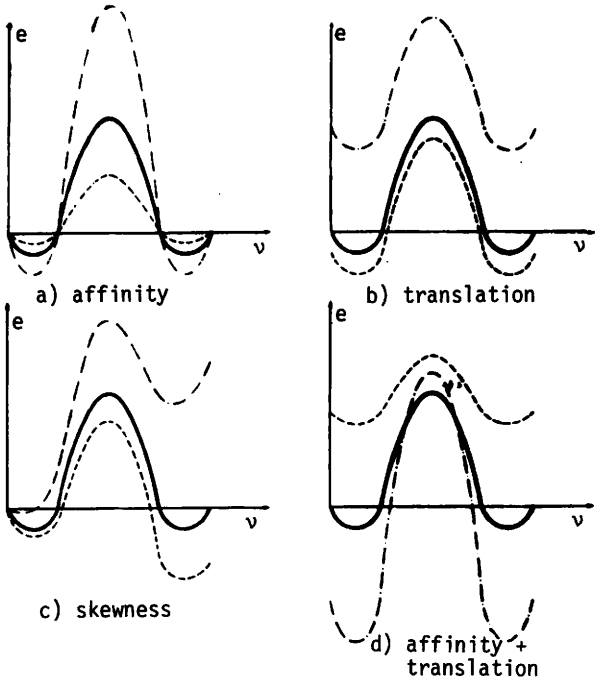


FIGURE 5
Some invariants

Still one problem remains: Such a matrix using resistors is not able to deal with patterns containing negative attributes. But an arrangement using "contradictory" columns for each input solves this problem.

Arrangements similar to the non-binary matrix are known (Chow 1957). But it has been assumed, so far, that the magnitudes of the connecting elements are set "a priori" proportional to the attributes of the learned patterns, for instance by soldering in ordinary resistors.

A learning non-binary matrix could be realized with almost all physical means which have already been used in binary learning matrices, such as chemical cells, the resistance of which can be altered by currents fed into the cell. However, the most versatile elements are magnetic devices. With irreversible connecting elements some difficulties could arise, for instance the impossibility of relearning.

4. CIRCUITS WITH SEVERAL LEARNING MATRICES

Interesting aspects arise from combined circuits of learning matrices. They easily could be arranged by means of intermediate storage to use a set of meanings coming out in sequence of a first learning matrix as an input set to a second matrix (see Fig. 6).

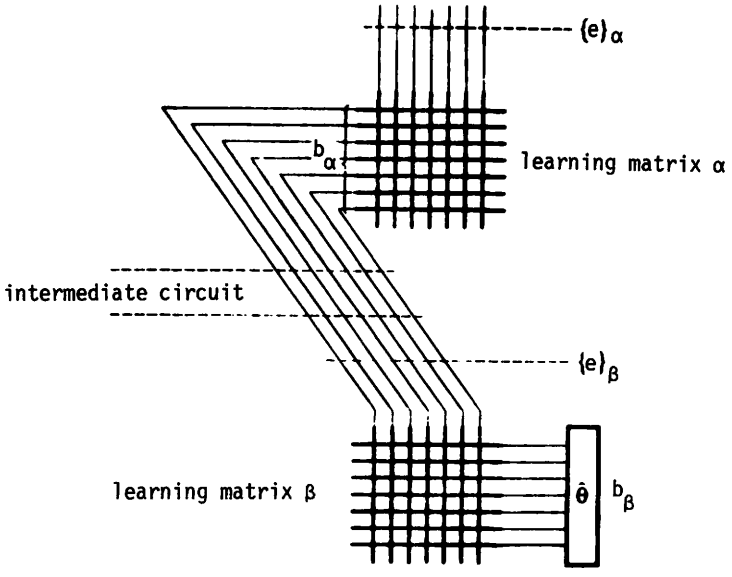


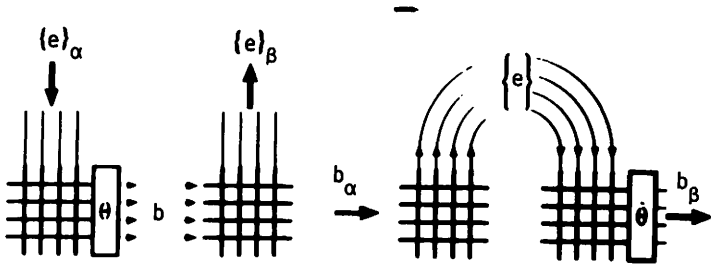
FIGURE 6

Layers of Learning matrices

By that combination e.g. words could be recognized of the single letters being the meanings of the first matrix. This could be extended to recognition of sentences using another intermediate circuit. This method resembles to the ability of organic systems to restore disturbed code words, e.g. wrongly spelled words.

Another way of making use of several learning matrices is the parallel circuit of two of them: Coupled learning matrices, "learning matrix - dipoles".

Two kinds of coupling are possible.



ebe coupling
a)

beb coupling
b)

FIGURE 7
LM - Dipoles

+ ebe - coupling: transforming a set of characteristics into another one having meaning b as an invariant. This may be a model for translation of the meaning of one language into another.
+ beb - coupling: transforming the meaning, keeping a set of features {e} invariant.

5. APPLICATIONS

When I had found and realized the learning matrix some 30 years ago, I was very optimistic about practical applications.
For example:

a) Automatic Pattern Recognition
So far the types of characters to be recognized had to be known a priori in order to construct and wire the logical network. A learning matrix could shift from one learned alphabet (e.g. latin typewriter) to another (certain types of printing, Greek letters, Kyrillic letters a.s.o.). Naturally one must not use black/white information as considered above but rather symptoms of shape, topological symptoms a.s.o.

b) Automatic Speech Recognition
Learning here can bridge the gap between individual differences in speaking.

c) Machine Translation could probably make good use of learning matrices adjusting the semantics to a certain text.

d) Information retrieval: Here new aspects are given by the fact, that the maximum likelihood decision is merely based on the information itself - no adress is needed. Thus a publication easily may be located by simply calling up the key-words.

e) For chemical analysis by means of spectrogramms a circuit consisting of a non-binary learning matrix may be conceived, where rather than a maximum detector the output row signals indicate the percentage of different compounds.

It may be too useful for automatic control, weather prediction, medical diagnosis, expert-systems a.s.o.

These where hopes 30 years ago.

But meanwhile my optimism concerning practical application of the learning matrix has been reduced.

Nevertheless, I estimate that the learning matrix has some exciting possibilities: It permits the design of information processing networks
+ which must not be programmed - and
+ which may perhaps surpass the limitations, which Goedel has stated for "normal" computers.

I am grateful, that a hungarian physiologist (J. Szentagothai) remarked, that he observed in the brain structures similar to the learning matrix.

References:

Steinbuch, K. "Die Lernmatrix"
Kybernetik 1 (1961) H. 1, S 36-45

Hönerloh, H.J.
Kraft, H. "Technische Verwirklichung der Lernmatrix"
In Billing, H. (Ed.) "Lernende Automaten"
R. Oldenbourg, München 1961

Görke, W.
Kazmierczak, H.
Wagner, S.W. "Anwendungen der Lernmatrix"
in Billing, H. (Ed.) "Lernende Automaten"
R. Oldenbourg, München 1961

Steinbuch, K.
Frank, H. "Nichtdigitale Lernmatrizen als Perzeptoren"
Kybernetik 1 (1961) S. 117-124

Chow, C.L. "An Optimum Character Recognition System Using Decision Function"
Trans. IRE EC - 6 H. 4

This page intentionally left blank

Section 2
Biological Sensory and Motor Systems

This page intentionally left blank

Advanced Neural Computers / R. Eckmiller (Editor)

MODEL OF VISUO-MOTOR TRANSFORMATIONS PERFORMED BY THE CEREBRAL CORTEX TO COMMAND ARM MOVEMENTS AT VISUAL TARGETS IN THE 3-D SPACE

Y.Burnod[1], R.Caminiti[2], P.Johnson[2], Ph.Granguillaume[1], I.Otto[1]
[1] Institut des Neurosciences, Univ. Paris VI
[2] Istituto di Fisiologia, Univ. La Sapienza.

We propose a biologically realistic neural network to perform arm reaching movements in the 3-D space, and which is consistent with anatomical and physiological data on the cortical areas involved in the command of these movements. Studies of the neuronal activities in the motor and premotor cortices of behaving monkeys have shown that individual neurons, related to motor synergies, are broadly tuned to preferred directions in the 3-D space [1]. Recent data demonstrate that these cell preferred directions rotate with the initial position of the arm [2]. Furthermore, the rotation of the arm in space predicts the rotation of the whole population of preferred directions. These results support the hypothesis that cortical neural circuits compute the appropriate motor command by projecting visual information on a reference frame rotating with the arm.
This computation can be performed by two processing layers made of units which model the properties of the cortical column, as the basic neuronal circuits common to all cortical areas. During spontaneous movements of the arm, the learning rules can memorize a relation, between somatic and visual information, which can be approximated by a bilinear form. Such bilinear combination explains the projection of visual information on a coordinate system rotating with the arm, and the invariant properties of cellular activities in the motor area. Neural circuits converging toward a single neuron in the motor cortex can easily learn and generalize such invariant properties in a 2-D subspace, but not in the whole 3-D space. However, the uniform distribution of cell preferred directions can explain the computation of the correct solution by a population of motor cortical neurons. These two predictions of the model are consistent with the experimental results.

I. BIOLOGICAL CONSTRAINTS

In order to propose a biologically realistic neural network that can learn visually guided movements of the arm in the 3-D space, it is necessary to meet at least three essential requirements, at three levels of the biological organization :

1. The global processing of the whole population of units performs a continuous transformation from visual into motor coordinates and results in a reaching movement. Parallel processing by all these units produce the correct command which orients the movement toward the target whatever the initial position of the hand as given in retinal and body coordinates. Several models have been proposed for the learning of arm kinematics and dynamics (review in [3]), and for the computation of invariant properties of sensorimotor transformations [4].

2. In the cortical areas involved in visuomotor tasks (parietal, motor, premotor), the activities of neurons recorded during a 3-D reaching movement should be explained by the model. These activities can be interpreted in relation with sensory inputs, motor outputs and learning conditions [5].

3. At the local level, the processing units should model the operations and learning properties of the cortical column which is the basic neuronal circuit of the cerebral cortex [6]. The operations performed by this unit have been modelled in relation with the behavioral adaptations performed by different cortical areas [7].

II. INVARIANT PROPERTIES OF MOTOR CORTICAL NEURONS.

Experiments are designed in order to quantify the cortical processes which result in the appropriate command of a 3-D arm reaching movement. Neuronal activity is recorded in the shoulder zone of the motor cortex of monkeys performing a 3-D movement of the hand between two points determined at each trial by a conditioning program. These trajectories follow eight possible directions which can be performed in three different parts of the work space, due to three initial positions of the arm.

1. Directional properties of neuronal activities.

Within each part of the workspace, it has been shown in previous studies that there is a particular movement direction for which the cell discharge frequency is maximal [1]. For a constant initial position of the arm, it is thus possible to define, for each recorded neuron, a preferential direction.

2. Population code.

Although single cells were broadly tuned to a particular direction in the 3-D space, the direction of the effective movement of the arm can be predicted with a population of cortical neurons by summing their preferred direction vectors weighted by their activities [1]. The set of preferred directions of the recorded cells covers the entire 3D-continuum, and their distribution is uniform in all the 3 parts of the work space.

3. Shift of preferred directions with the initial position of the arm

When the initial position is changed, the orientation of the preferred direction for each cell rotates in space [2]. In order to quantify the rotation of the whole set of preferred direction in the shoulder zone of the motor cortex, a spherical regression analysis has been performed between the different initial conditions. This statistical analysis shows that the rotation of the whole set of preferred directions exactly follows the rotation of the shoulder joint between the different initial positions of the arm.

All the results obtained in recording cells of the motor cortex are consistent with the following assessments : (1) the motor cortical cells command muscle synergies, which can be represented by a vector corresponding to the sum of the individual muscles actions ; (2) the orientation of such a synergy vector does not remain constant with respect to an extrapersonal coordinate system, but rotates with the position of the arm in space and has invariant property in an arm centered reference system ; (3) for both shoulder and elbow joints, a cortical command sent to the muscles will change the arm position in a constant way within this intrinsic coordinate system (Ref in [2]).

In the task which is experimentally analyzed, information about the desired trajectory of the hand (visual, oculomotor and proprioceptive) is projected on a body coordinate system, since the body does not move, and the eyes fixate the target before the movement. Consequently, the cortical neural network has to compute the projection of this information in a reference system rotating with the arm in order to produce the appropriate motor command.

The shift of cells preferred directions was evident even in the early phase of the reaction time, suggesting that the computation is performed before the initiation of the movement, and then without the sensory feedback loop due to the movement.

III. SENSORIMOTOR COMBINATIONS IN CORTICAL AREAS.

The shift of cell preferred directions with the initial position of the arm reflects the computation of the appropriate motor output by the combination between two different sources of inputs : (1) an information about the position of the arm with respect to the body (for example angles of the shoulder and elbow joints) ; (2) an information about the trajectory to be performed as defined by the visual system (image of the hand when the target is centered on the fovea), and the oculomotor system (angle of gaze).

Computations from experimental data give a vectorial interpretation to cell properties (cell preferred directions), and allow to delimit the "combinatory domain" of cortical circuits converging on a cell , with at least two parameters :

- The dimension (Dc) of the combinatory domain: vectorial inputs from the two sensory sources may be matched in the whole space (3D), or within a plane (2D) or in only one specific direction.
- The combinatory order (Oc): only pairs of vectors leading to a common output are matched (Oc=1), or all the possible combinations are effected (Oc=2).

The cell's preferred direction does not rotate if Dc=1, and rotates in a plane if Dc=2. The rotation of the cell's preferred direction can follow the rotation of the arm if Oc=1. Experimental results are consistent with a computation of the motor command by two processing layers, as shown in Figure 1 :

- a **matching layer** models bimodal associative cortical maps (somatic+visual) in the parietal lobe (Dc=1 and Oc=1).
- an **output layer** model motor and premotor cortex which receives information from parietal neurons; it combines them in a subspace (Dc=2) and in an ordered way (Oc=1)

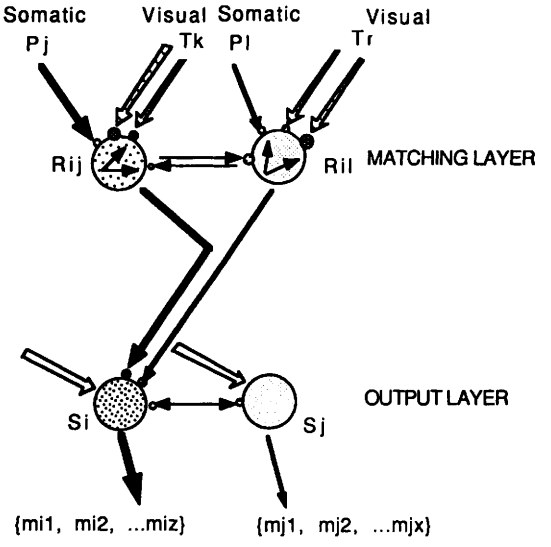


figure 1 :

Furthermore, it is important to take into account the direct internal relation from the output layer to the matching layer ("efferent copy" of the motor command) This architecture is consistent with anatomical data on the relations between premotor and parietal regions.

The properties of the cortical combinations are due both to anatomical connections, and reinforcement of specific pathways by learning.

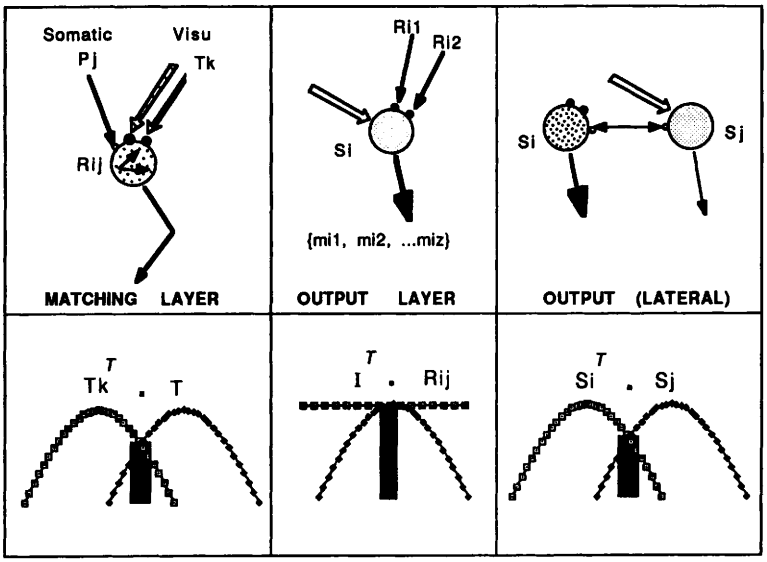


figure 2

In the model, three types of relations between units are learned (in Figure 2) A representation of the 3-D external space is learned by the adjustment of lateral connections between output units (right column). A global transformation of visual inputs by somatic signals is learned in two steps (left and middle columns), thanks to the visual feedback signals produced by spontaneous movements of the arm.

IV. PROCESSING UNITS: MODEL OF THE CORTICAL COLUMN.

Activation and learning properties of the processing units in these two layers should not be chosen arbitrary, but should be designed as close as possible to the main known features of the real intracortical network [7]. Such units do not correspond to single neurons but to cortical columns, which are repetitive multicellular neuronal circuits, with common properties whatever the adaptation they produce within each specific cortical region.

Two properties of cortical columns are important for the command of visually-guided hand movements :

1. **Activation rules** : the coactivation of two inputs from independent sources (for example proprioceptive and visual) can generate a strong non-linear increase of activity (gating and amplification). An increasing activity can result first in an activation, and after in an inhibition of related units with lower activities (conditional inhibition).

2. **Learning rules** correspond to operant conditioning. If a column, strongly active, participates to an action outside the cortex, and if this action results in a reactivation of a cortical input of the column, this input will gain by learning a new influence : after learning, columns selectively activate other columns whose actions were previously efficient to result in their reactivation. These rules allow to build multilayered networks with feedback, feedforward, and local connections, which