The Self-Organizing Map

TEUVO KOHONEN, SENIOR MEMBER, IEEE

Invited Paper

Among the architectures and algorithms suggested for artificial neural networks, the Self-Organizing Map has the special property of effectively creating spatially organized "internal representations" of various features of input signals and their abstractions. One novel result is that the self-organization process can also discover semantic relationships in sentences. In this respect the resulting maps very closely resemble the topographically organized maps found in the cortices of the more developed animal brains. After supervised fine tuning of its weight vectors, the Self-Organizing Map has been particularly successful in various pattern recognition tasks involving very noisy signals. In particular, these maps have been used in practical speech recognition, and work is in progress on their application to robotics, process control, telecommunications, etc. This paper contains a survey of several basic facts and results.

I. Introduction

A. On the Role of the Self-Organizing Map Among Neural Network Models

The network architectures and signal processes used to model nervous systems can roughly be divided into three categories, each based on a different philosophy. *Feedforward networks* [94] transform sets of input signals into sets of output signals. The desired input-output transformation is usually determined by external, supervised adjustment of the system parameters. In *feedback networks* [27], the input information defines the initial activity state of a feedback system, and after state transitions the asymptotic final state is identified as the outcome of the computation. In the third category, neighboring cells in a neural network compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of different signal patterns. In this category learning is called *competitive, unsupervised,* or *self-organizing.*

The Self-Organizing Map discussed in this paper belongs to the last category. It is a sheet-like artificial neural network, the cells of which become specifically tuned to various input signal patterns or classes of patterns through an unsupervised learning process. In the basic version, only one cell or local group of cells at a time gives the active response to the current input. The locations of the responses tend to become ordered as if some meaningful coordinate system for different input features were being created over the network. The spatial location or coordinates of a cell in the network then correspond to a particular domain of input signal patterns. Each cell or local cell group acts like a separate *decoder* for the same input. It is thus the presence or absence of an active response at that location, and not so much the exact input–output signal transformation or magnitude of the response, that provides an interpretation of the input information.

The Self-Organizing Map was intended as a viable alternative to more traditional neural network architectures. It is possible to ask just how "neural" the map is. Its analytical description has already been developed further in the technical than in the biological direction. But the learning results achieved seem very natural, at least indicating that the adaptive processes themselves at work in the map may be similar to those encountered in the brain. There may therefore be sufficient justification for calling these maps "neural networks" in the same sense as their traditional rivals.

Self-Organizing Maps, or systems consisting of several map modules, have been used for tasks similar to those to which other more traditional neural networks have been applied: pattern recognition, robotics, process control, and even processing of semantic information. The spatial segregation of different responses and their organization into topologically related subsets results in a high degree of efficiency in typical neural network operations.

Although the largest map we have used in practical applications has only contained about 1000 cells, its learning speed, especially when using computational shortcuts, can be increased to orders of magnitude greater than that of many other neural networks. Thus much larger maps than those used so far are quite feasible, although it also seems that practical applications favor hierarchical systems made up of many smaller maps.

It may be appropriate to observe here that if the maps are used for pattern recognition, their classification accuracy can be multiplied if the cells are fine-tuned using supervised learning principles (cf. Sec. III).

Although the Self-Organizing Map principle was introduced in early 1981, no complete review has appeared in compact form, except perhaps in [44], which does not contain the latest results. I have therefore tried to collect a variety of basic material in the present paper.

Manuscript received August 18, 1989; revised March 15, 1990.

The author is with the Department of Computer Science, Helsinki University of Technology, 02150 Espoo, and the Academy of Finland, 00550 Helsinki, Finland.

IEEE Log Number 9038826.

B. Brain Maps

As much as a hundred years ago, a quite detailed topographical organization of the brain, and especially of the cerebral cortex, could be deduced from functional deficits and behavioral impairments induced by various kinds of lesion, or by hemorrhages, tumors, or malformations. Different regions in the brain thereby seemed to be dedicated to specific tasks. One modern systematic technique for causing controllable, reversible simulated lesions is to stimulate a particular site with small electric currents, thereby eventually inducing both excitatory and inhibitory effects and disturbing the assumed local function [75]. If such a spatially confined stimulus then disrupts a specific cognitive ability such as naming of objects, it gives at least some indication that this site is essential to that task.

One straightforward method for locating a response is to record the electric potential or train of neural impulses associated with it. Many detailed mappings, especially from the primary sensory and associative areas of the brain, have been made using various electrophysiological recording techniques.

Direct evidence for any localization of brain functions can also be obtained using modern imaging techniques that display the strength and spatial distribution of neural responses simultaneously over a large area, with a spatial resolution of a few millimeters. The two principal methods which use radioactive tracers are positron emission tomography (PET) [80] and autoradiography of the brain through very narrow collimators (gamma camera). PET reveals changes in oxygen uptake and phosphate metabolism. The gamma camera method directly detects changes in cerebral blood flow. Both phenomena correlate with local neural activity, but they are unable to monitor rapid phenomena. In magnetoencephalography (MEG), the low magnetic field caused by electrical neural responses is detected, and by computing its sources, quite rapid neural responses can be directly analyzed, with a spatial resolution of a few millimeters. The main drawback of MEG is that only current dipoles parallel to the surface of the skull are detectable; and since the dipoles are oriented perpendicular to the cortex, only the sulci can be studied with this method. A review of experimental techniques and results relating to these studies can be found in [32].

After a large number of such observations, a fairly detailed organizational view of the brain has evolved [32]. Especially in higher animals, the various cortices in the cell mass seem to contain many kinds of "map" [33], such that a particular location of the neural response in the map often directly corresponds to a specific modality and quality of sensory signal. The field of vision is mapped "quasiconformally" onto the primary visual cortex. Some of the maps, especially those in the primary sensory areas, are ordered according to some feature dimensions of the sensory signals; for instance, in the visual areas, there are line orientation and color maps [11], [116], and in the auditory cortex there are the so-called tonotopic maps [91], [103], [104], which represent pitches of tones in terms of the cortical distance, or other auditory maps [98]. One of the sensory maps is the somatotopic map [29], [30] which contains a representation of the body, i.e., the skin surface. Adjacent to it is a motor map [70] that is topographically almost identically organized. Its cells mediate voluntary control actions on muscles. Similar maps exist in other parts of the brain [97]. On the higher levels the maps are usually unordered, or at most the order is a kind of ultrametric topological order that is not easily interpretable. There are also singular cells that respond to rather complex patterns, such as the human face [9], [93].

Some maps represent quite abstract qualities of sensory and other experiences. For instance, in the word-processing areas, neural responses seem to be organized according to categories and semantic values of words ([6], [15], [65], [67], [106–109], and [114]; cf. also Sec. V).

It thus seems as if the *internal representations* of information in the brain are generally organized *spatially*. Although there is only partial biological evidence for this, enough data are already available to justify further theoretical studies of this principle. Artificial self-organizing maps and brain maps thus have many features in common, and what is even more intriguing, we now fully understand the processes by which such artificial maps can be formed adaptively and completely automatically.

C. Early Work on Competitive Learning

The basic idea underlying what is called competitive learning is roughly as follows: Assume a sequence of statistical samples of a vectorial observable $\boldsymbol{x} = \boldsymbol{x}(t) \in \mathbb{R}^n$, where t is the time coordinate, and a set of variable reference vectors $\{\boldsymbol{m}_i(t): \boldsymbol{m}_i \in \mathbb{R}^n, i = 1, 2, \cdots, k\}$. Assume that the $\boldsymbol{m}_i(0)$ have been initialized in some proper way; random selection will often suffice. If $\boldsymbol{x}(t)$ can somehow be simultaneously compared with each $\boldsymbol{m}_i(t)$ at each successive instant of time, taken here to be an integer $t = 1, 2, 3, \cdots$, then the best-matching $\boldsymbol{m}_i(t)$ is to be updated to match even more closely the current $\boldsymbol{x}(t)$. If the comparison is based on some distance measure $d(\boldsymbol{x}, \boldsymbol{m}_i)$, altering $\boldsymbol{m}_i$ must be such that, if $i = c$ is the index of the best-matching reference vector, then $d(\boldsymbol{x}, \boldsymbol{m}_c)$ is decreased, and all the other reference vectors $\boldsymbol{m}_i$, with $i \neq c$, are left intact. In this way the different reference vectors tend to become specifically "tuned" to different domains of the input variable $\boldsymbol{x}$. It will be shown below that if p is the probability density function of the samples $\boldsymbol{x}$, then the $\boldsymbol{m}_i$ tend to be located in the input space $\mathbb{R}^n$ in such a way that they approximate to p in the sense of some minimal residual error.

Vector Quantization (*VQ*) (cf., e.g., [19], [54], [58]) is a classical method, that produces an approximation to a continuous probability density function $p(\boldsymbol{x})$ of the vectorial input variable $\boldsymbol{x}$ using a finite number of codebook vectors $\boldsymbol{m}_i$, $i = 1, 2, \cdots, k$. Once the "codebook" is chosen, the approximation of $\boldsymbol{x}$ involves finding the reference vector $\boldsymbol{m}_c$ closest to $\boldsymbol{x}$. One kind of optimal placement of the $\boldsymbol{m}_i$ minimizes E, the expected rth power of the reconstruction error:

$$E = \int \|\boldsymbol{x} - \boldsymbol{m}_c\|^r p(\boldsymbol{x})\, d\boldsymbol{x} \tag{1}$$

where $d\boldsymbol{x}$ is the volume differential in the $\boldsymbol{x}$ space, and the index $c = c(\boldsymbol{x})$ of the best-matching codebook vector ("winner") is a function of the input vector $\boldsymbol{x}$:

$$\|\boldsymbol{x} - \boldsymbol{m}_c\| = \min_i \{\|\boldsymbol{x} - \boldsymbol{m}_i\|\}. \tag{2}$$

In general, no closed-form solution for the optimal placement of the $\boldsymbol{m}_i$ is possible, and iterative approximation schemes must be used.

It has been pointed out in [14], [64], and [115] that (1)

defines a placement of the codebook vectors into the signal space such that their point density function is an approximation to $[p(\boldsymbol{x})]^{r/(r+n)}$, where n is the dimensionality of $\boldsymbol{x}$ and $\boldsymbol{m}_i$. We usually consider the case $r = 2$. In most practical applications $n \gg r$, and then the optimal VQ can be shown to approximate $p(\boldsymbol{x})$.

Using the square-error criterion ($r = 2$), it can also be shown that the following stepwise "delta rule," in the discrete-time formalism ($t = 0, 1, 2, \cdots$), defines the optimal values asymptotically. Let $\boldsymbol{m}_c = \boldsymbol{m}_c(t)$ be the closest codebook vector to $\boldsymbol{x} = \boldsymbol{x}(t)$ in the Euclidean metric. The steepest-descent gradient-step optimization of E in the $\boldsymbol{m}_c$ space yields the sequence

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)],$$
$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \qquad \text{for } i \neq c \tag{3}$$

with $\alpha(t)$ being a suitable, monotonically decreasing sequence of scalar-valued gain coefficients, $0 < \alpha(t) < 1$. This is then the simplest analytical description of competitive learning.

In general, if we express the dissimilarity of $\boldsymbol{x}$ and $\boldsymbol{m}_i$ in terms of a general distance function $d(\boldsymbol{x}, \boldsymbol{m}_i)$, we have first to identify the "winner" $\boldsymbol{m}_c$ such that

$$d(\boldsymbol{x}, \boldsymbol{m}_c) = \min_i \{d(\boldsymbol{x}, \boldsymbol{m}_i)\}. \tag{4}$$

After that an updating rule should be used such that d decreases monotonically: the correction $\delta\boldsymbol{m}_i$ of $\boldsymbol{m}_i$ must be such that

$$[\text{grad}_{\boldsymbol{m}_i}\, d(\boldsymbol{x}, \boldsymbol{m}_i)]^T \cdot \delta\boldsymbol{m}_i < 0. \tag{5}$$

If (1) is used for signal approximation, it often turns out to be more economical to first observe a number of training samples $\boldsymbol{x}(t)$, which are "classified" (labeled) on the basis of (2) according to the closest codebook vectors $\boldsymbol{m}_i$, and then to perform the updating operation in a single step. For the new codebook vector $\boldsymbol{m}_i$, the average is taken of those $\boldsymbol{x}(t)$ that were identified with codebook vector i. This algorithm, termed the *k-means algorithm* is widely used in digital telecommunications engineering [58].

The $\boldsymbol{m}_i(t)$, in the above processes, actually develop into a set of *feature-sensitive detectors*. Feature-sensitive cells are also known to be common in the brain. Neural modelers like Nass and Cooper [72], Pérez *et al.* [79], and Grossberg [21] have been able to suggest how such feature-sensitive cells can emerge from simplified membrane equations of model neurons.

In the above process and its biophysical counterparts, all the cells act independently. Therefore the *order* in which they are assigned to the different domains of input signals is more or less haphazard, most strongly depending on the initial values of the $\boldsymbol{m}_i(0)$. In fact, in 1973, v.d. Malsburg [59] had already published a computer simulation in which he demonstrated *local* ordering of feature-sensitive cells, such that in small subsets of cells roughly corresponding to the so-called columns of the cortex, the cells were tuned more closely than were more remote cells. Later, Amari [1] formulated and analyzed the corresponding system of differential equations, relating them to spatially continuous two-dimensional media. Such continuous layers interacted in the lateral direction; the arrangement was called a *nerve field*. The above studies are of great theoretical importance because they involve a self-organizing tendency. The ordering power they demonstrated was, however, still weak as nerve-field type equations only describe this tendency as a marginal effect. In spite of numerous attempts, no "maps" of practical importance could be produced; ordering was either restricted to a one-dimensional case, or confined to small parcelled areas of the network [60], [77], [78], [99], [100], [110], [111].

Indeed it later transpired that system equations have to involve much stronger, idealized self-organizing effects, and that the organizing effect has to be maximized in every possible way before useful global maps can be created. The present author, in early 1981, was experimenting with various architectures and system equations, and found a process description [34]–[36] that seemed generally to produce globally well-organized maps. Because all the other system models known at that time only yielded results that were significantly more "brittle" with respect to the selection of parameters and to success in achieving the desired results, we may skip them here and concentrate on the computationally optimized algorithm known as the *Self-Organizing Map* algorithm.

II. An Algorithm that Orders Responses Spatially

Readers who are not yet familiar with the Self-Organizing Maps may benefit from a quick look at Figs. 5 and 6 or Fig. 9 to find out what spatial ordering of output responses means.

The Self-Organizing Map algorithm that I shall now describe has evolved during a long series of computer experiments. The background to this research has been expounded in [44]. While the purpose of each detail of the final equations may be clear in concrete simulations, it has proved extremely difficult, in spite of numerous attempts, to express the dynamic properties of this process in mathematical theorems. Strict mathematical analysis only exists for simplified cases. And even they are too lengthy to be reviewed here: cf. [7], [8], [24], [57], [83], [86], [89], [90]. It is therefore hoped that the simulation experiments and practical applications reported below in Secs. II-C, II-E, IV, V, and VI will suffice to convince the reader about the utility of this algorithm.

It may also be necessary to emphasize again that for practical purposes we are trying to extract or explain the self-organizing function in its purest, most effective form, whereas in genuine biological networks this tendency may be more or less disguised by other functions. It may thus be conceivable, as has been verified by numerous simulation experiments, that the two essential effects leading to spatially organized maps are: 1) *spatial concentration* of the network activity on the cell (or its neighborhood) that is *best tuned* to the present input, and 2) further *sensitization or tuning* of the best-matching cell *and its topological neighbors* to the present input.

A. Selection of the Best-Matching Cell

Consider the two-dimensional network of cells depicted in Fig. 1. Their arrangement can be hexagonal, rectangular, etc. Let (in matrix notation) $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R}^n$ be the *input vector* that, for simplicity and computational efficiency, is assumed to be connected in parallel to all the neurons i in this network. (We have also shown that subsets of the same input signals can be connected at random to the

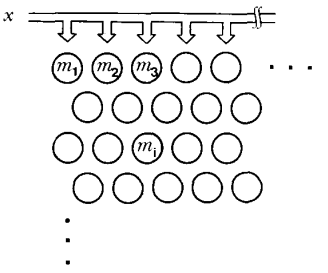

Fig. 1. Cell arrangement for the map and definition of variables.

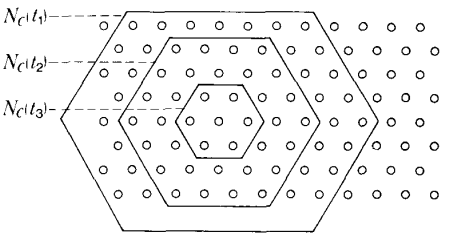

Fig. 2. Examples of topological neighborhood $N_c(t)$, where $t_1 < t_2 < t_3$.

cells; cf. the "tonotopic map" discussed in [35] and [44]. Experiments are in progress in which the input connections to the cells can be made in a cascade.) The *weight vector* of cell i shall henceforth be denoted by $\boldsymbol{m}_i = [\boldsymbol{m}_{i1}, \boldsymbol{m}_{i2}, \cdots, \boldsymbol{m}_{in}]^T \in \mathbb{R}^n$.

The simplest analytical measure for the match of $\boldsymbol{x}$ with the $\boldsymbol{m}_i$ may be the inner product $\boldsymbol{x}^T\boldsymbol{m}_i$. If, however, the self-organizing algorithm is to be used for, say, natural signal patterns relating to metric vector spaces, a better and more convenient (cf. the adaptation law below) matching criterion may be used, based on the *Euclidean distances* between $\boldsymbol{x}$ and $\boldsymbol{m}_i$. The minimum distance defines the "winner" $\boldsymbol{m}_c$ (cf. (2)). A shortcut algorithm to find $\boldsymbol{m}_c$ has been presented in [49].

Comment. Definition of the input vector, $\boldsymbol{x}$, as an ordered set of signal values is only possible if the interrelation between the signals is simple. In many practical problems, such as image analysis (cf. Discussion, Sec. VII) it will generally be necessary to use some kind of preprocessing to extract a set of invariant features for the components of $\boldsymbol{x}$.

B. Adaptation (Updating) of the Weight Vectors

It is crucial to the formation of ordered maps that the cells doing the learning are not affected independently of each other (cf. competitive learning in Sec. I-C), but as *topologically related subsets*, on each of which a similar kind of correction is imposed. During the process, such selected subsets will then encompass different cells. The net corrections at each cell will thus tend to be smoothed out in the long run. An even more intriguing result from this sort of spatially correlated learning is that *the weight vectors tend to attain values that are ordered along the axes of the network.*

In biophysically inspired neural network models, correlated learning by spatially neighboring cells can be implemented using various kinds of lateral feedback connection and other lateral interactions. In the present process we want to enforce lateral interaction directly in a general form, for arbitrary underlying network structures, by defining a *neighborhood set* N_c around cell c. At each learning step, all the cells within N_c are updated, whereas cells outside N_c are left intact. This neighborhood is centered around that cell for which the best match with input $\boldsymbol{x}$ is found:

$$\|\boldsymbol{x} - \boldsymbol{m}_c\| = \min_i \{\|\boldsymbol{x} - \boldsymbol{m}_i\|\}. \tag{2'}$$

The width or radius of N_c can be time-variable; in fact, for good global ordering, it has experimentally turned out to be advantageous to let N_c be very wide in the beginning and shrink monotonically with time (Fig. 2). The explanation for this may be that a wide initial N_c, corresponding to a coarse spatial resolution in the learning process, first induces a rough global order in the $\boldsymbol{m}_i$ values, after which narrowing the N_c improves the spatial resolution of the map; the acquired global order, however, is not destroyed later on. It is even possible to end the process with $N_c = \{c\}$, that is, finally updating the best-matching unit ("winner") only, in which case the process is reduced to simple competitive learning. Before this, however, the "topological order" of the map would have to be formed.

The updating process (in discrete-time notation) may read

$$\boldsymbol{m}_i(t+1) = \begin{cases} \boldsymbol{m}_i(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] & \text{if } i \in N_c(t), \\ \boldsymbol{m}_i(t) & \text{if } i \notin N_c(t), \end{cases} \tag{6}$$

where $\alpha(t)$ is a scalar-valued "adaptation gain" $0 < \alpha(t) < 1$. It is related to a similar gain used in the stochastic approximation processes [49], [92], and as in these methods, $\alpha(t)$ should decrease with time.

An alternative notation is to introduce a scalar "kernel" function $h_{ci} = h_{ci}(t)$,

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] \tag{7}$$

whereby, above, $h_{ci}(t) = \alpha(t)$ within N_c, and $h_{ci}(t) = 0$ outside N_c. On the other hand, the definition of h_{ci} can also be more general; a biological lateral interaction often has the shape of a "bell curve". Denoting the coordinates of cells c and i by the vectors $\boldsymbol{r}_c$ and $\boldsymbol{r}_i$, respectively, a proper form for h_{ci} might be

$$h_{ci} = h_0 \exp(-\|\boldsymbol{r}_i - \boldsymbol{r}_c\|^2/\sigma^2), \tag{8}$$

with $h_0 = h_0(t)$ and $\sigma = \sigma(t)$ as suitable decreasing functions of time.

C. Demonstrations of the Ordering Process

The first computer simulations presented here are intended to illustrate the effect that the weight vectors tend to approximate to the density function of the input vectors in an orderly fashion. In these examples, the input vectors were chosen to be two-dimensional for visual display purposes, and their probability density function was arbitrarily selected to be uniform over the area demarcated by the borderlines (square or triangle). Outside the frame the density was zero. The vectors $\boldsymbol{x}(t)$ were drawn from this density function independently and at random, after which they caused adaptive changes in the weight vectors $\boldsymbol{m}_i$.

The $\boldsymbol{m}_i$ vectors appear as points in the same coordinate system as that in which the $\boldsymbol{x}(t)$ are represented; in order to indicate to which unit each $\boldsymbol{m}_i$ value belongs, the points

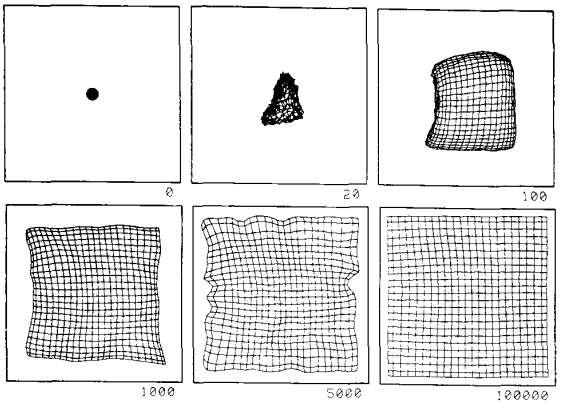

Fig. 3. Weight vectors during the ordering process, two-dimensional array.

corresponding to the $\boldsymbol{m}_i$ vectors have been connected by a lattice of lines conforming to the topology of the processing unit array. In other words, a line connecting two weight vectors $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ is only used to indicate that the corresponding units i and j are adjacent in the array. In Fig. 3 the arrangement of the cells is rectangular (square), whereas in Fig. 4 the cells are interconnected in a linear chain.

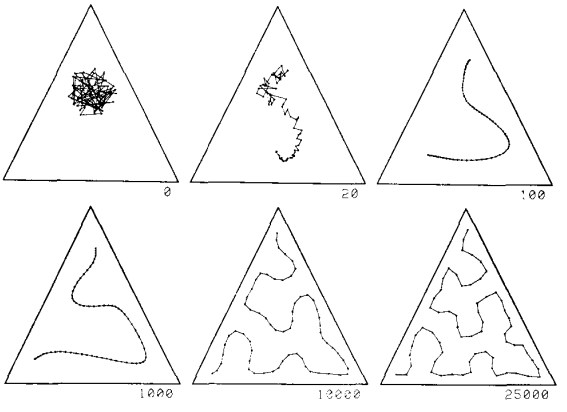

Fig. 4. Weight vectors during the ordering process, one-dimensional array.

Examples of intermediate phases during the self-organizing process are given in Figs. 3 and 4. The initial values $\boldsymbol{m}_i(0)$ were selected at random from a certain domain of values.

As stated above, in Fig. 3 the array was two-dimensional. The results, however, are particularly interesting if the distribution and the array have different dimensionalities: Fig. 4 illustrates a case in which the distribution of $\boldsymbol{x}$ is two-dimensional, but the array is one-dimensional (linear row of cells). The weight vectors of linear arrays tend to approximate to higher-dimensional distributions by Peano curves. A two-dimensional network representing three-dimensional "bodies" (uniform-density function) is shown in Fig. 5.

In practical applications, the input and output weight vectors are usually high-dimensional; e.g., in speech recognition, the dimensionality n may be 15 to 100.

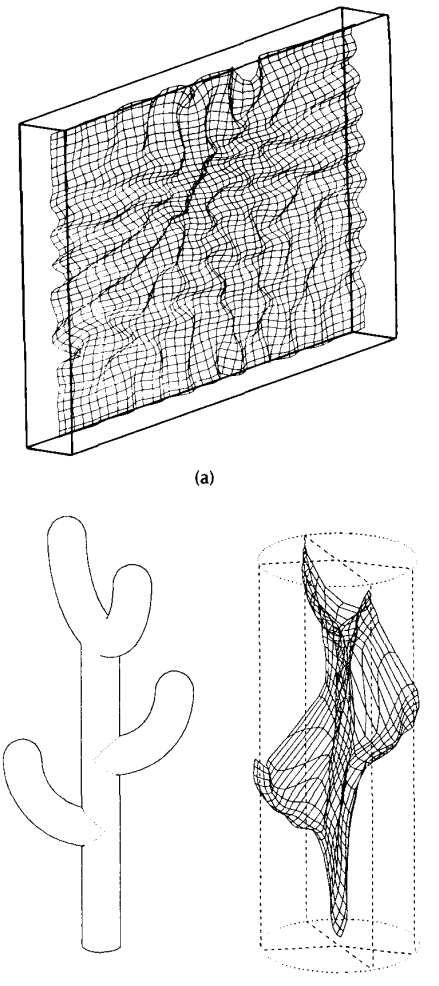

Fig. 5. Representation of three-dimensional (uniform) density functions by two-dimensional maps.

Since no factor present defines a particular orientation in the output map, the latter can be realized in the process in any mirror or point-symmetric inversion, mainly depending on the initial values $\boldsymbol{m}_i(0)$. If a particular orientation is to be favored, the easiest way to achieve this result is by asymmetric choice of the initial values $\boldsymbol{m}_i(0)$.

D. Some Practical Hints for the Application of the Algorithm

When applying the map algorithm, (2) or (2′) and (6) alternate. Input $\boldsymbol{x}$ is usually a random variable with a density function $p(\boldsymbol{x})$, from which the successive values $\boldsymbol{x}(t)$ are drawn. In real-world observations, such as speech recognition, the $\boldsymbol{x}(t)$ can simply be successive samples of the input observables in their natural order of occurrence.

The process may be started by choosing arbitrary, even random, initial values for the $\boldsymbol{m}_i(0)$, the only restriction being that they should be different.

We shall give numerical examples of efficient process parameters with the simulation examples. It may also be helpful to emphasize the following general conditions.

1) Since learning is a stochastic process, the final statistical accuracy of the mapping depends on the number of steps, which must be reasonably large; there is no way to circumvent this requirement. A "rule of thumb" is that, for good statistical accuracy, the number of steps must be at least 500 times the number of network units. On the other hand, the number of components in $\boldsymbol{x}$ has no effect on the number of iteration steps, and if hardware neural computers are used, a very high dimensionality of input is allowed. Typically we have used up to 100 000 steps in our simulations, but for "fast learning," e.g., in speech recognition, 10 000 steps and even less may sometimes be enough. Note that the algorithm is computationally extremely light. If only a small number of samples are available, they must be recycled for the desired number of steps.

2) For approximately the first 1000 steps, $\alpha(t)$ should start with a value that is close to unity, thereafter decreasing monotonically. An accurate rule is not important: $\alpha = \alpha(t)$ can be linear, exponential, or inversely proportional to t. For instance, $\alpha(t) = 0.9(1 - t/1000)$ may be a reasonable choice. The *ordering* of the $\boldsymbol{m}_i$ occurs during this initial period, while the remaining steps are only needed for the fine adjustment of the map. After the ordering phase, $\alpha = \alpha(t)$ should attain small values (e.g., of the order of or less than .01) over a long period. Neither is it crucial whether the law for $\alpha(t)$ decreases linearly or exponentially during the final phase.

3) Special caution is required in the choice of $N_c = N_c(t)$. If the neighborhood is too small to start with, the map will not be ordered globally. Instead various kinds of mosaic-like parcellations of the map are seen, between which the ordering direction changes discontinuously. This phenomenon can be avoided by starting with a fairly wide $N_c = N_c(0)$ and letting it shrink with time. The initial radius of N_c can even be more than half the diameter of the network! During the first 1000 steps or so, when the proper ordering takes place, and $\alpha = \alpha(t)$ is fairly large, the radius of N_c can shrink linearly to, say, one unit; during the fine-adjustment phase, N_c can still contain the nearest neighbors of cell c.

Parallel implementations of the algorithm have been discussed in [25] and [61].

E. Example: Taxonomy (Hierarchical Clustering) of Abstract Data

Although the more practical applications of the Self-Organizing Maps are available, for example, in pattern recognition and robotics, it may be interesting to apply this principle first to abstract data vectors consisting of hypothetical *attributes* or *characteristics*. We will look at an example with implicitly defined (*hierarchical*) structures in the primary data, which the map algorithm is then able to reveal. Although this system is a single-level network, it can produce a hierarchical representation of the relations between the primary data.

The central result in self-organization is that if the input signals have a well-defined probability density function, then the weight vectors of the cells try to imitate it, however complex its form. It is even possible to perform a kind of numerical taxonomy on this model. Because there are no restrictions on the semantic content of the input signals, they can be regarded as arbitrary attributes, with discrete or continuous values. In Table 1, 32 items, each with five hypothetical attributes, are recorded in a data matrix. (This example is completely artificial.) Each of the columns represents one item, and for later inspection the items are labeled "A" through "6", although these labels were not referred to during the learning.

The attribute values $(a_1, a_2, \cdots, a_5)$ constitute the pattern vector $\boldsymbol{x}$ which acts as a set of signal values at the inputs to the network of the type shown in Fig. 1. During training, the vectors $\boldsymbol{x}$ were selected from Table 1 at random. Sampling and adaptation were continued iteratively until the asymptotic state could be considered stationary. Such a "learned" network was then calibrated using the items from Table 1 and labeling the best-matching map cells according to the different calibration items. Such a labeled map is shown in Fig. 6. It can be seen that the "images" of different items are related according to a taxonomic graph where the different branches are visible. For comparison, Fig. 7 illus-

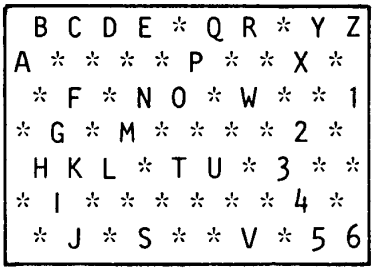

Fig. 6. Self-organized map of the data matrix of Table 1.

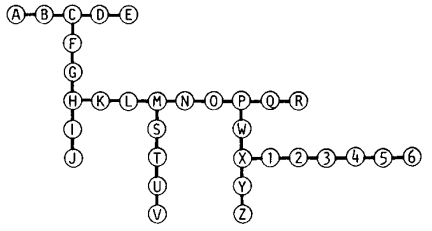

Fig. 7. Minimal spanning tree corresponding to Table 1.

Table 1 Input Data Matrix

	Item																													
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	1	2	3	4	5	6
Attribute																																
a_1	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
a_2	0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
a_3	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	6	6	6	6	6	6	6	6	6	6
a_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4	2	2	2	2	2	2
a_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6

trates the minimal spanning tree (where the most closely similar pairs of items are linked) describing the similarity relations of the items in Table 1. The system parameters in this process were:

$\alpha = \alpha(t)$: During the first 1000 steps, α decreased linearly with time from .5 to .04 (the initial value could have been closer to unity, say, .9). During the subsequent 10 000 steps, α decreased from .04 to zero linearly with time.

$N_c = N_c(t)$: The lattice was hexagonal, 7 by 10 units, and during the first 1000 steps, the *radius* of N_c decreased from the value six (encompassing the majority of cells in the network) to one (encompassing neuron c and its six neighbors) linearly with time, thereafter keeping the value one.

F. Another Variant of the Algorithm

One further remark may be necessary. It has sometimes been suggested that $\boldsymbol{x}$ be normalized before it is used in the algorithm. Normalization is not necessary in principle, but it may improve numerical accuracy because the resulting reference vectors then tend to have the same dynamic range.

Another aspect, as mentioned above, is that it is also possible to apply a general distance measure in the matching; then, however, the matching and updating laws should be mutually compatible with respect to the same metric. For instance, if the inner-product measure of similarity were applied, the learning equations should read:

$$\boldsymbol{x}^T(t)\boldsymbol{m}_c(t) = \max_i \{\boldsymbol{x}^T(t)\boldsymbol{m}_i(t)\}, \tag{9}$$

$$\boldsymbol{m}_i(t+1) = \begin{cases} \dfrac{\boldsymbol{m}_i(t) + \alpha'(t)\boldsymbol{x}(t)}{\|\boldsymbol{m}_i(t) + \alpha'(t)\boldsymbol{x}(t)\|} & \text{if } i \in N_c(t), \\ \boldsymbol{m}_i(t) & \text{if } i \notin N_c(t), \end{cases} \tag{10}$$

and $0 < \alpha'(t) < \infty$; for instance, $\alpha'(t) = 100/t$. This process normalizes the reference vectors at each step. The normalization computations slow down the training algorithm significantly. On the other hand, the linear matching criterion applied during recognition is very simple and fast, and amenable to many kinds of simple analog computation, both electronic and optical.

III. Fine Tuning of the Map by Learning Vector Quantization (LVQ) Methods

If the Self-Organizing Map is to be used as a *pattern classifier* in which the cells or their responses are grouped into subsets, each of which corresponds to a discrete class of patterns, then the problem becomes a decision process and must be handled differently. The original Map, like any classical Vector Quantization (VQ) method (cf. Sec. I-D) is mainly intended to approximate input signal values, or their probability density function, by quantized "codebook" vectors that are localized in the input space to minimize a quantization error functional (cf. Sec. III-A below). On the other hand, if the signal sets are to be *classified* into a finite number of categories, then several codebook vectors are usually made to represent each class, and their identity *within* the classes is no longer important. In fact, only decisions made at class borders count. It is then possible, as shown below, to define *effective* values for the codebook vectors such that they directly define *near-optimal decision borders* between the classes, even in the sense of classical Bayesian decision theory. These strategies and learning algorithms were introduced by the present author [38], [43], [45] and called *Learning Vector Quantization (LVQ)*.

A. Type One Learning Vector Quantization (LVQ1)

If several codebook vectors $\boldsymbol{m}_i$ are assigned to each class, and each of them is labeled with the corresponding class symbol, the class regions in the $\boldsymbol{x}$ space are defined by simple nearest-neighbor comparison of $\boldsymbol{x}$ with the $\boldsymbol{m}_i$; the label of the closest $\boldsymbol{m}_i$ defines the classification of $\boldsymbol{x}$.

To define the optimal placement of $\boldsymbol{m}_i$ in an iterative learning process, initial values for them must first be set using any classical VQ method or by the Self-Organizing Map algorithm. The initial values in both cases roughly correspond to the overall statistical density function $p(\boldsymbol{x})$ of the input. The next phase is to determine the labels of the codebook vectors, by presenting a number of input vectors with known classification, and assigning the cells to different classes by majority voting, according to the frequency with which each $\boldsymbol{m}_i$ is closest to the calibration vectors of a particular class.

The classification accuracy is improved if the $\boldsymbol{m}_i$ are updated according to the following algorithm [41], [43]–[45]. The idea is to pull codebook vectors away from the decision surfaces to demarcate the class borders more accurately. Let $\boldsymbol{m}_c$ be the codebook vector closest to $\boldsymbol{x}$ in the Euclidean metric (cf. (2), (2')); this then also defines the classification of $\boldsymbol{x}$. Apply training vectors $\boldsymbol{x}$ the classification of which is known. Update the $\boldsymbol{m}_i = \boldsymbol{m}_i(t)$ as follows:

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)]$$

if x is classified correctly,

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)]$$

if the classification of x is incorrect,

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \text{ for } i \neq c. \tag{11}$$

Here $\alpha(t)$ is a scalar gain ($0 < \alpha(t) < 1$), which is decreasing monotonically in time, as in earlier formulas. Since this is a fine-tuning method, one should start with a fairly small value, say $\alpha(0) = 0.01$ or 0.02 and let it decrease to zero, say, in 100 000 steps.

This algorithm tends to reduce the point density of the $\boldsymbol{m}_i$ around the Bayesian decision surfaces. This can be deduced as follows. The minus sign in the second equation may be interpreted as *defining corrections in the same direction as if (10) were used for the class to which $\boldsymbol{m}_c$ belongs, but with the probability density function of the neighboring (overlapping) class subtracted from that of $\boldsymbol{m}_c$.* In other words, we would perform a classical Vector Quantization of the function $|p(\boldsymbol{x}|C_i)P(C_i) - p(\boldsymbol{x}|C_j)P(C_j)|$ where C_i and C_j are the neighboring classes, $p(\boldsymbol{x}|C_i)$ is the conditional probability density function of samples $\boldsymbol{x}$ belonging to class C_i, and $P(C_i)$ is the *a priori* probability of occurrences of the class C_i samples. The difference between the density functions of the neighboring classes, by definition, drops to zero at the Bayes border, inducing the above "depletion layer" of the codebook vectors.

After training, the $\boldsymbol{m}_i$ will have acquired values such that classification using the "nearest neighbor" principle, by

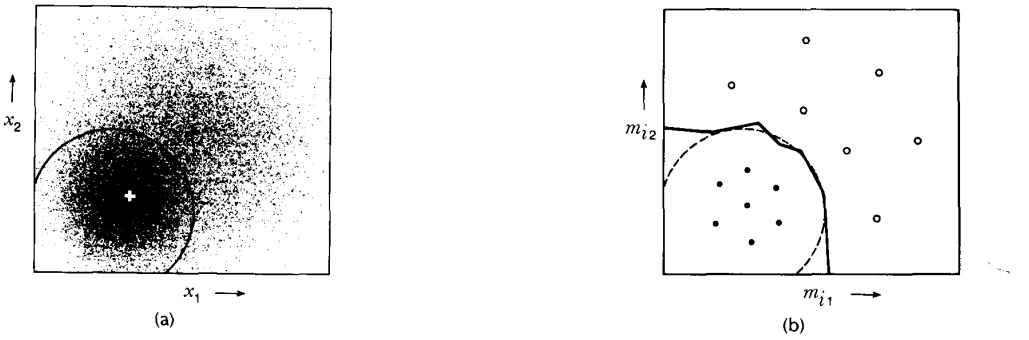

Fig. 8. (a) An illustrative example in which $\boldsymbol{x}$ is two-dimensional and the probability density functions of the classes substantially overlap. (a) The probability density function of $\boldsymbol{x} = [x_1, x_2]^T$ is represented here by its samples, the small dots. The superposition of two symmetric Gaussian density functions corresponding to two different classes C_1 and C_2, with their centroids shown by the white and the dark cross, respectively, is shown. Solid curve: the theoretically optimal Bayes decision surface. (b) Large black dots: reference vectors of class C_1. Open circles: reference vectors of class C_2. Solid curve: decision surface in the Learning Vector Quantization. Broken curve: Bayes decision surface.

comparing of $\boldsymbol{x}$ with the $\boldsymbol{m}_i$, already rather closely coincides with that of the Bayes classifier. Figure 8 represents an illustrative example in which $\boldsymbol{x}$ is two-dimensional, and the probability density functions of the classes substantially overlap. The decision surface defined by this classifier seems to be near-optimal, although piecewise linear, and the classification accuracy in this rather difficult example is within a fraction of a percent of that achieved with the Bayes classifier. For practical applications of the LVQ1, cf. [12], [76]. A rigorous mathematical discussion of the LVQ1, and suggestions to improve its stability, have been represented in [51].

B. Type Two Learning Vector Quantization (LVQ2)

The previous algorithm can easily be modified to comply even better with Bayes' decisionmaking philosophy [43]–[45]. Assume that two codebook vectors $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ that belong to different classes and are closest neighbors in the vector space are initially in a wrong position. The (incorrect) discrimination surface, however, is always defined as the midplane of $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$. Let us define a symmetric *window* of nonzero width around the midplane, and stipulate that *corrections to* $\boldsymbol{m}_i$ *and* $\boldsymbol{m}_j$ *shall only be made if* $\boldsymbol{x}$ *falls into the window on the wrong side of the midplane* (cf. Fig. 9).

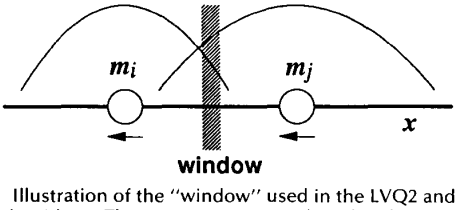

Fig. 9. Illustration of the "window" used in the LVQ2 and LVQ3 algorithms. The curves represent class distributions of $\boldsymbol{x}$ samples.

If the corrections are made according to (12), it is easy to see that for vectors falling into the window, the corrections of both $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$, on average, have such a direction that the midplane moves towards the crossing surface of the class distributions, and thus asymptotically coincides with the Bayes decision border.

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)],$$

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)],$$

if C_i is the nearest class, but $\boldsymbol{x}$ belongs to $C_j \neq C_i$ where C_j is the next-to-nearest class ("runner-up"); furthermore $\boldsymbol{x}$ must fall into the "window". In all the other cases,

$$\boldsymbol{m}_k(t+1) = \boldsymbol{m}_k(t). \tag{12}$$

The optimal width of the window must be determined experimentally, and it depends on the number of available samples. With a relatively small number of training samples, a width 10 to 20% of the difference between $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ seems to be proper.

One question concerns the practical definition of the "window". If we are working in a high-dimensional signal space, it seems reasonable to define the "window" in terms of relative distances d_i and d_j from $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$, respectively, having constant ratio s. In this way the borders of the "window" are Apollonian hyperspheres. The vector $\boldsymbol{x}$ is defined to lie in the "window" if

$$\min(d_i/d_j,\ d_j/d_i) > s. \tag{13}$$

If w is the relative width of the window in its narrowest point, then $s = (1 - w)/(1 + w)$. The optimal size of the window depends on the number of available training samples. If we had a large number of samples, a narrow window would guarantee the most accurate location of the border; but for good statistical accuracy the number of samples falling into the window must be sufficient, too, so a 20% window seems a good compromise, at least in the experiments reported below.

For reasons explained in the next section, the classification accuracy of the LVQ2 is first improved when the decision surface is shifted towards the Bayes limit; after that, however, the $\boldsymbol{m}_i$ continue "drifting away". Therefore this algorithm ought to be applied for a relatively short time only, say, starting with $\alpha = 0.02$ and letting it to decrease to zero in at most 10 000 steps.

C. Type Three Learning Vector Quantization (LVQ3)

The LVQ2 algorithm was based on the idea of *differentially* shifting the decision borders towards the Bayes limits, while no attention was paid to what might happen to the location of the $\boldsymbol{m}_i$ in the long run if this process were continued. Thus, although researchers have reported good results, some have had problems, too. It turns out that at least two different kinds of detrimental effect must be taken into account. First, because corrections are proportional to the difference of $\boldsymbol{x}$ and $\boldsymbol{m}_i$, or $\boldsymbol{x}$ and $\boldsymbol{m}_j$, the correction on $\boldsymbol{m}_j$ (correct class) is of larger magnitude than that on $\boldsymbol{m}_i$ (wrong class); this results in monotonically decreasing distances $\|\boldsymbol{m}_i - \boldsymbol{m}_j\|$. One remedy is to compensate for this effect, approximately at least, by accepting *all the training vectors from the "window,"* and the only condition is that *one* of $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ must belong to the *correct* class, and *the other* to the *incorrect* class. The second problem arises from the fact that if the process in (12) is continued, it may lead to another asymptotic equilibrium of $\boldsymbol{m}_i$ that is no longer optimal. Therefore it seems necessary to include corrections that ensure that the $\boldsymbol{m}_i$ continue approximating the class distributions, at least roughly. Combining these ideas, we now obtain an improved algorithm that may be called LVQ3:

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) - \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)],$$

$$\boldsymbol{m}_j(t+1) = \boldsymbol{m}_j(t) + \alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_j(t)],$$

where $\boldsymbol{m}_i$ and $\boldsymbol{m}_j$ are the two closest codebook vectors to $\boldsymbol{x}$, and $\boldsymbol{x}$ and $\boldsymbol{m}_j$ belong to the same class, while $\boldsymbol{x}$ and $\boldsymbol{m}_i$ belong to different classes; furthermore $\boldsymbol{x}$ must fall into the "window";

$$\boldsymbol{m}_k(t+1) = \boldsymbol{m}_k(t) + \epsilon\alpha(t)[\boldsymbol{x}(t) - \boldsymbol{m}_k(t)]$$

for $k \in \{i, j\}$, if $\boldsymbol{x}$, $\boldsymbol{m}_i$, and $\boldsymbol{m}_j$ belong to the same class. (14)

In a series of experiments, applicable values for ϵ between 0.1 and 0.5 were found. The optimal value of ϵ seems to depend on the size of the window, being smaller for narrower windows. This algorithm seems to be self-stabilizing, i.e., the optimal placement of the $\boldsymbol{m}_i$ does not change in continued learning.

Notice that whereas in LVQ1 only one of the $\boldsymbol{m}_i$ values was changed at each step, LVQ2 and LVQ3 change two codebook vectors simultaneously.

IV. Application of the Map to Speech Recognition

When artificial neural networks are to be used for a practical pattern recognition application such as speech recognition, the first task is to make clear whether it is desirable to perform the complete chain of processing operations starting, e.g., from the pre-analysis of the microphone signal and leading on to some form of linguistic encoding of speech using "all-neural" operations, or whether "neural networks" should only be applied at the most critical stage, whereby the rest of the processing operations can be implemented on standard computing equipment. This choice mainly depends on whether the objective is commercial or academic.

Another issue is whether the aim is to demonstrate the ultimate capabilities of "neural networks" in the analysis of dynamical speech information, or whether it is only to replace some of the traditional "spectral" and "vector space" pattern recognition algorithms by highly adaptive, learning "neural network" principles.

In a speech recognizer, a proper place for artificial neural networks is in the *phonemic recognition stage* where exacting statistical analysis is needed. It should be remembered that if phonemes, i.e., classes of different phonological realizations of vowels and consonants, are selected for the basic phonetic units, then account has to be taken of their transformations due to *coarticulation effects*. In other words, the spectral properties of the phonemes are changed in the context or frame of other phonemes. In an "all-neural" speech recognizer it may not be necessary to distinguish or consider phonemes at all, because interpretation of speech is then regarded as an integral, implicit process. Introduction of the phoneme concept already implies that the system must be able to automatically identify them in one form or another and to label the corresponding time interval. Correction of coarticulation effects may then already be implemented in the acoustic analysis itself, by regarding the speech states as Markov processes, and analyzing the state transitions statistically [53]. A different approach altogether is first to apply some vector quantization classification, whereby the speech waveform is only labeled by class symbols of stationary phonemes, as if no coarticulation effects were being taken into account. Corrections can then be made afterwards in a separate postprocessing stage, in symbolic form. We have used the latter approach.

We have implemented a practical "phonetic typewriter" for unlimited speech input using the Self-Organizing Map to spot and recognize phonemes in continuous speech (Finnish and Japanese) [42], [46], [48]. The "network" was fine-tuned for optimal decision accuracy by the Learning Vector Quantization. After that, in the postprocessing stage we applied a self-learning grammar that corrects the majority of coarticulation errors and derives its numerous transformation rules automatically from given examples. This principle, termed *"Dynamically Expanding Context"* [37], [40], actually belongs to the category of learning Artificial Intelligence methods, and thus falls outside the scope of this article (cf. Sec. IV-D below).

A. Acoustic Preprocessing of the Speech Signal

It is known that biological sensory organs such as the inner ear are usually able to adapt to signal transients in a fast, nonlinear way. Nonetheless, we decided to apply conventional frequency analysis to the preprocessing of speech. The main reason for this was that digital Fourier analysis is both accurate and fast, and the fundamentals of digital filtering are well understood. Deviations from physiological reality are not essential since the self-organizing neural network can accept many alternative kinds of preprocessing and can compensate for minor imperfections.

The technical details of the acoustic preprocessing stage are briefly as follows: 1) 5.3-kHz low-pass switched-capacitor filter, 2) 12-bit A/D-converter with 13.02-kHz sampling rate, 3) 256-point FFT formed every 9.83 ms using a Hamming window, 4) logarithmization and smoothing of the power spectrum, 5) combination of spectral channels from the frequency range 200 Hz–5 kHz into a 15-component pattern vector, 6) subtraction of the average from the com-

ponents, 7) normalization of the pattern vectors. Except for steps 1) and 2), an integrated-circuit signal processor, TMS32010, is used for the computation.

B. Phoneme Map

The simplest type of speech maps formed by self-organization is the static *phoneme map*. There are 21 phonemes in Finnish: /u, o, a, œ, ø, y, e, i, s, m, n, ŋ, l, r, j, v, h, d, k, p, t/. For their representation we used *short-time spectra* as the input patterns $x(t)$. The spectra were evaluated every 9.83 ms. They were computed by the 256-point FFT, from which a 15-component spectral vector was formed by grouping of the channels. In the present study all the spectral samples, even those from the transitory regions, were employed and presented to the algorithm in the natural order of their utterance. During learning, the spectra were not segmented or labeled in any way: any features present in the speech waveform contributed to the self-organized map. After adaptation, the map was calibrated using known stationary phonemes (Fig. 10). The map resembles the well-known *formant maps* used in phonetics; the main difference is that in our maps *complete spectra*, not just two of their resonant frequencies as in formant maps, are used to define the mapping.

Recognition of discrete phonemes is a decision-making process in which the final accuracy only depends on the rate of misclassification errors. It is therefore necessary to try to minimize them using a decision-controlled (supervised) learning scheme, using a training set of speech spectra with known classification.

In practice, for a new speaker, it will be sufficient to dictate 200 to 300 words which are then analyzed by an automatic segmentation method. The latter picks up the training spectra that are applied in the supervised learning algorithm. The finite set of training spectra (of the order of 2000) must be repeated in the algorithm either cyclically or in a random permutation. LVQ1, LVQ2, or LVQ3 can be used as the fine tuning algorithm. A map created for a typical (standard) speaker can then be modified for a new speaker very quickly, using 100 more dictated words, and LVQ fine tuning only.

C. Specific Problems with Transient Phonemes

Generally, the spectral properties of consonants behave more dynamically than those of vowels. Especially in the case of stop consonants, it seems to be better to pay attention to the plosive burst and transient region between the consonant and the subsequent vowel in order to identify the consonant. In our system transient information is coded in additional "satellite" maps (called *transient maps*) and they are trained, using transient spectral samples alone, to describe the dynamic features with higher resolution [48]. Our system was in fact developed in two versions: one for Finnish and one for Japanese. In the Japanese version, four transient maps have been constructed to distinguish the following cases:

1) voiceless stops /k, p, t/ and glottal stop (vowel at the beginning of utterance),
2) voiceless stops /k, p, t/ without comparison with the glottal stop,
3) voiced stops /b, d, g/,
4) nasals /m, n, ŋ/.

Only one transient map has been adopted for the Finnish version, making the distinction between /k, p, t/ and the glottal stop. (/b/ and /g/ do not exist in original Finnish.)

D. Compensation for Coarticulation Effects using the "Dynamically Expanding Context"

Because of coarticulation effects, i.e., transformation of the speech spectra due to neighboring phonemes, systematic errors appear in phonemic transcriptions. For instance, the Finnish word "hauki" (meaning pike) is almost invariably recognized as the phoneme string /haouki/ by our acoustic processor. It may then be suggested that if a transformation rule /aou/ → /au/ is introduced, this error will be corrected. It might also be imagined that it is possible to list and take into account all such variations. However, there may be hundreds of different frames or contexts of neighboring phonemes in which a particular phoneme may occur, and in many cases such empirical rules are contradictory; they are only statistically correct. The frames may

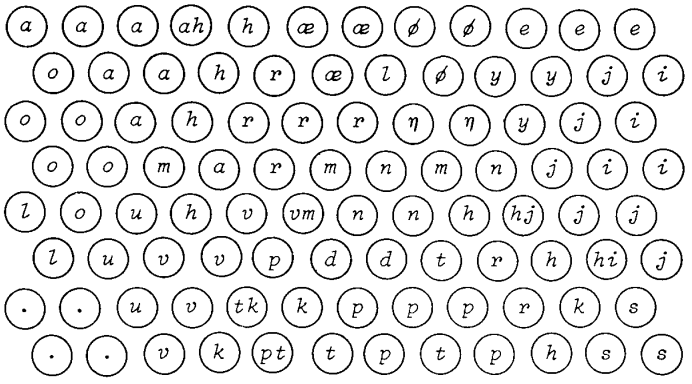

Fig. 10. An example of a *phoneme map*. Natural Finnish speech was processed by a model of the inner ear which performs its frequency analysis. The resulting signals were then connected to an artificial neural network, the cells of which are shown in this picture as circles. The cells were tuned automatically, without any supervision or extra information given, to the acoustic units of speech known as *phonemes*. The cells are labeled by the symbols of those phonemes to which they "learned" to give responses; most cells give a unique answer, whereas the double labels show which cells respond to two phonemes.

also be erroneous. In order to find an optimal and very large system of rules, the *Dynamically Expanding Context* grammar mentioned above was applied [37], [40]. Its rules or productions can be used to transform erroneous symbol strings into correct ones, and even into orthographic text.

Because the correction rules are made accessible in memory using a software content-addressing method (hash coding), they can be applied very quickly, such that the overall operation of the grammar, even with 15 000 rules, is almost in real time. This algorithm is able to correct up to 70% of the errors left by the phoneme map recognizer.

E. Performance of the "Phonetic Typewriter"

In order to get some idea of the accuracy of the map algorithm, we first show a comparative benchmarking of five different methods, namely, classification of manually selected phonemic spectra by the classical parametric Bayes classification, the well-known k-Nearest-Neighbor method (kNN), and LVQ1, LVQ2 and LVQ3.

In this partial experiment, the spectral samples were of Finnish phonemes (divided into 19 classes and using 15 frequency channels for the spectral decomposition). There were 1550 training vectors, and 1550 statistically independent vectors that were only used for testing. The error percentages are given in Table 2.

Table 2 Speech Recognition Experiments with Error Percentages for Independent Test Data

	Parametric Bayes	kNN	LVQ1	LVQ2	LVQ3
Test 1	12.1	12.0	10.2	9.8	9.6
Test 2	13.8	12.1	13.2	12.0	11.5

Note that the parametric Bayes classifier is not even theoretically the best because it assumes that the class samples are normally distributed. We have not been able to find any method, theoretical or heuristic, that classifies speech spectra better than LVQ1, LVQ2 or LVQ3.

In its complete form, the "Phonetic Typewriter" has been tested on several Finnish and Japanese speakers over a long period. To someone familiar with practical speech recognizers it will be clear that it would be meaningless to evaluate and compare different test runs statistically; the results obtained in each isolated test depend so much on the experimental situation and the content of text, the status and tuning of the equipment, as well as on the physical condition of the speaker. The number of tests performed over many years is also too large to be discussed fully here. Let it suffice to mention that the accuracy of spotting and recognizing phonemes in arbitrary continuous speech typically varies between 80 and 90% (depending on the automatic segmentation and recognition of any phoneme), and this figure depends on the speaker and the text. After compensation for coarticulation effects and editing the text into orthographic form, the accuracy, in terms of correctness of any letter, is of the order of 92 to 97%, again depending on the speaker and the text.

The Phonetic Typewriter has already been implemented in several hardware versions using signal processor chips. The latest versions operate in genuine real time with continuous dictation.

It may be of interest here to mention other results, independent of ours. McDermott and Katagiri [28], [66] have carried out experiments on all the Japanese phonemes, and report that LVQ2 gave consistently higher accuracies than Backpropagation Time Delay Neural Networks [105] and was faster in learning.

V. Semantic Map

Demonstrations such as those reported above have indicated that the Self-Organizing Map is indeed able to extract abstract information from multidimensional primary signals, and to represent it as a location, say, in a two-dimensional network. Although this is already a step towards generalization and symbolism, it must be admitted that the extraction of features from geometrically or physically relatable data elements is still a very concrete task, in principle at least.

The operation of the brain at the higher levels relies heavily on abstract concepts, symbolism, and language. It is an old notion that the deepest semantic elements of any language should also be physiologically represented in the neural realms. There is now new physiological evidence for linguistic units being locatable in the human brain [6], [15].

In attempting to devise Neural Network models for linguistic representations, the first difficulty is encountered when trying to find metric distance relations between symbolic items. Unlike with primary sensory signal patterns for which similarity is easily derivable from their mutual distances in the vector spaces in which they are represented, it can not be assumed that encodings of symbols in general have any relationship with the observable characteristics of the corresponding items. How could it then be possible to represent the "logical similarity" of pairs of items, and to map such items topographically? The answer lies in the fact that *the symbol, during the learning process, is presented in context*, i.e., in conjunction with the encodings of a set of other concurrent items. In linguistic representations context might mean a few adjacent words. Similarity between items would then be reflected through the *similarity of the contexts*. Note that for ordered sets of arbitrary encodings, invariant similarity can be expressed, e.g., in terms of the number of items they have in common. On the other hand, it may be evident that the meaning (semantics) of a symbolic encoding is only derivable from the conditional probabilities of its occurrences with other encodings, independent of the type of encoding [68].

However, in the learning process, the literal encodings of the symbols must be memorized, too. Let vector $\boldsymbol{x}_s$ represent the symbolic expression of an item, and $\boldsymbol{x}_c$ the representation of its context. The simplest neural model then assumes that $\boldsymbol{x}_s$ and $\boldsymbol{x}_c$ are connected to the same neural units, i.e., the representation (pattern) vector $\boldsymbol{x}$ of the item is formed as a concatenation of $\boldsymbol{x}_s$ and $\boldsymbol{x}_c$:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_s \\ \boldsymbol{x}_c \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_s \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{x}_c \end{bmatrix}. \tag{15}$$

In other words, the symbol part and the context part form a vectorial sum of two orthogonal components.

The core idea underlying symbol maps is that the two parts are weighted properly such that *the norm of the context part predominates over that of the symbol part during the self-organizing process*; the topographical mapping

then mainly reflects the metric relationships of the sets of associated encodings. But since the inputs for symbolic signals are also active all the time, memory traces of them are formed in the corresponding inputs of those cells in the map that have been selected (or actually enforced) by the context part. *If then, during recognition of input information, the context signals are missing or are weaker, the (same) map units are selected solely on the basis of the symbol part. In this way the symbols become encoded into a spatial order reflecting their logical (or semantic) similarities.*

In the following, I shall demonstrate this idea, which was originated by H. Ritter, using a simple language [84]. The simplest definition of the context of a word is to take all those words (together with their serial order) that occur in a certain "window" around the selected word. For simplicity, we shall imagine that the content of each "window" can somehow be presented to the x_c input ports of the neural system. We are not interested here in any particular means for the conversion of, say, temporal signal patterns into parallel ones (this task could be done using paths with different delays, eigenstates that depend on sequences, or any other mechanisms implementable in short-term memory).

The vocabulary used in this experiment is listed in Fig. 11(a) and comprises nouns, verbs, and adverbs. Each word

Bob/Jim/Mary	1
horse/dog/cat	2
beer/water	3
meat/bread	4
runs/walks	5
works/speaks	6
visits/phones	7
buys/sells	8
likes/hates	9
drinks/eats	10/11
much/little	12
fast/slowly	13
often/seldom	14
well/poorly	15

(a)

Sentence Patterns:		
1-5-12	1-9-2	2-5-14
1-5-13	1-9-3	2-9-1
1-5-14	1-9-4	2-9-2
1-6-12	1-10-3	2-9-3
1-6-13	1-11-4	2-9-4
1-6-14	1-10-12	2-10-3
1-6-15	1-10-13	2-10-12
1-7-14	1-10-14	2-10-13
1-8-12	1-11-12	2-10-14
1-8-2	1-11-13	2-11-4
1-8-3	1-11-14	2-11-12
1-8-4	2-5-12	2-11-13
1-9-1	2-5-13	2-11-14

(b)

Mary likes meat
Jim speaks well
Mary likes Jim
Jim eats often
Mary buys meat
dog drinks fast
horse hates meat
Jim eats seldom
Bob buys meat
cat walks slowly
Jim eats bread
cat hates Jim
Bob sells beer
(etc.)

(c)

Fig. 11. Outline of vocabulary used in this experiment. (a) List of used words (nouns, verbs, and adverbs), (b) sentence patterns, and (c) some examples of generated three-word-sentences.

class has further categorial subdivisions, such as names of persons, animals, and inanimate objects. To study semantic relationships in their purest form, it must be stipulated that the semantic meaning be not inferable from any patterns used for the encoding of the individual words, but only from the context in which the words occur (i.e., combinations of words). To this end each word was encoded by a random vector of unit length (here, seven-dimensional).

A sequence of randomly generated meaningful three-word sentences was used as the input data to the self-organizing process. Meaningful sentence patterns had therefore first to be constructed on the basis of word categories (Fig. 11(b)). Each explicit sentence was then constructed by randomly substituting the numbers in a randomly selected sentence pattern from Fig. 11(b) by words with compatible numbering in Fig. 11(a). A total of 498 different three-word sentences are possible, a few of which are exemplified in Fig. 11(c). These sentences were concatenated into a single continuous string, S.

The context of a word in this string was restricted to the pair of words formed by its immediate predecessor and successor in S (ignoring any sentence borders; i.e., words from adjacent sentences in S are uncorrelated, and act like random noise in that field). The code vectors of the predecessor/successor-pair forming the context to a word were concatenated into a single 14-dimensional code vector x_c. In this simple demonstration we thus only took into account the context provided by the immediately adjacent textual environment of each word occurrence. Even this restricted context already contains interesting semantic relationships.

In our computer experiments it turned out that instead of presenting each phrase separately to the algorithm, a much more efficient learning strategy is first to consider each word *in its average context* over a set of possible "windows". The (mean) context of a word was thus first defined as *the average over 10 000 sentences of all code vectors of predecessor/successor-pairs surrounding that particular word*. The resulting thirty 14-dimensional "average word contexts", normalized to unit length, assumed the role of the "context fields" x_c in (14). Each "context field" was combined with a 7-dimensional "symbol field" x_s, consisting of the code vector for the word itself, but scaled to length a. The parameter a determines the relative influence of the symbol part x_s in comparison to the context part x_c and was set to 0.2.

For the simulation, a planar, rectangular lattice of 10 by 15 cells was used. The initial weight vectors of the cells were chosen randomly, so that no initial order was present. Updating was based on (7) and (8). The learning step size was $h_0 = 0.8$ and the radius $\sigma(t)$ of the adjustment zone (cf. (8)) was gradually decreased from an initial value $\sigma_i = 4$ to a final value $\sigma_f = 0.5$ according to the law $\sigma(t) = \sigma_i(\sigma_f/\sigma_i)^{t/t_{max}}$ Here t counts the number of adaptation steps.

After $t_{max} = 2000$ input presentations the responses of the neurons to presentation of the symbol parts alone were tested. In Fig. 12, the symbolic label is written to that site at which the symbol signal $x = [x_s, 0]^T$ gave the maximum response. We clearly see that *the contexts "channel" the word items to memory positions whose arrangements reflects both grammatical and semantic relationships.* Words of same type, i.e., nouns, verbs, and adverbs, are segregated into separate, large domains. Each of these domains is further organized according to similarities on the semantic level. Adverbs with opposite meaning tend to be close to each other, because sentences differing in one word only are regarded as semantically correlated, and the words that are different then usually have the opposite meaning. The groupings of the verbs correspond to differences in the ways they can co-occur with adverbs, persons, animals, and nonanimate objects such as, e.g., food.

It could be argued that the structures resulting in the map were artificially created by a preplanned choice of the sentence patterns allowed as input. This is not the case, however, since it is easy to check that the categorial sentence patterns in Fig. 11(b) almost completely exhaust the possibilities for forming semantically meaningful three-word sentences.

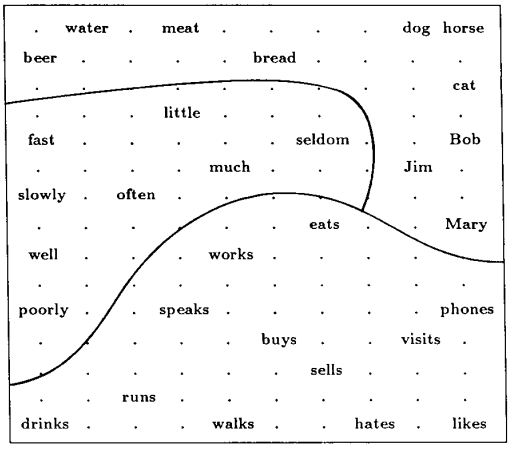

Fig. 12. "Semantic map" obtained on a network of 10 × 15 cells after 2000 presentations of word-context-pairs derived from 10 000 random sentences of the kind shown in Fig. 10(c). Nouns, verbs, and adverbs are segregated into different domains. Within each domain a further grouping according to aspects of meaning is discernible.

VI. Survey of Practical Applications of the Map

In addition to numerous more abstract simulations, theoretical developments, and "toy examples," the following practical problem areas have been suggested for the Self-Organizing Map or the LVQ algorithms. In some of them concrete work is already in progress.

- Statistical pattern recognition, especially recognition of speech [42], [48];
- control of robot arms, and other problems in robotics [17], [18], [63], [85], [87], [88];
- control of industrial processes, especially diffusion processes in the production of semiconductor substrates [62], [102];
- automatic synthesis of digital systems [23];
- adaptive devices for various telecommunications tasks [2], [4], [47];
- image compression [71];
- radar classification of sea-ice [76];
- optimization problems [2];
- sentence understanding [95];
- application of expertise in conceptual domains [96]; and even
- classification of insect courtship songs [73].

Of these, the application to speech recognition has the longest tradition in demonstrating the power of the map method when dealing with difficult stochastic signals. My personal expectations are, however, that the greatest industrial potential of this method may lie in process control and telecommunications.

On the other hand, it is a little surprising that so few applications of the maps to computer vision are being studied. This does not mean that the problems of vision are not important. It is rather that automatic analysis and extraction of visual features, without heuristic or analytical approach, has turned out to be an extremely difficult problem. Biological and artificial vision probably require very complicated hierarchical systems using many stages (e.g., several different maps) [55], [56]. One unclarified problem is how the maps should be interconnected, e.g., whether special nonlinear interfaces are needed [82]; and in hierarchical systems, adaptive normalization of input (cf. [31]) also seems necessary. Only a few isolated problems, such as texture analysis that is under study in our laboratory, might be amenable to the basic method as such.

VII. Discussion

It was stated in Secs. I and III that it is not advisable to use the Self-Organizing Map for classification problems because decision accuracy can be significantly increased if fine tuning such as LVQ is used. Another important notion, that not only concerns the maps but most of the other neural network models as well, is that it would often be absurd to use primary signal elements, such as temporal samples of speech waveform or pixels of an image, for the components of $\boldsymbol{x}$ directly. This is especially true if the input patterns are fine-structured, like line drawings. It is not possible to achieve any *invariances* in perception unless the primary information is first *transformed*, using, e.g., various convolutions with, say, Gabor functions [13], or other, possibly nonlinear functionals of the image field [81], as components of $\boldsymbol{x}$. Which particular choice of functionals should be used for preprocessing in a particular task is a very difficult and delicate question, and cannot be discussed here.

One question concerns the maximum capacity achievable in the maps. Is it possible to increase their size, to eventually use them for data storage in large knowledge data bases? It can at least be stated that the brain's maps are not particularly extensive; they mainly seem to provide for efficient encoding of a particular subset of signals to enhance the operation and capacity of associative memory [44]. If more extensive systems are required, it might be more efficient to develop hierarchical structures of abstract representations.

The hardware used for the maps has so far only consisted of co-processor boards (cf., e.g., [42], [48]). If the simple algorithm is to be directly built into special hardware, one of its essential operations will be the global extremum selector, for which conventional parallel computing hardware is available [39]. Analog "winner-take-all" circuits can also be used [20], [21], [52]. Another question is whether the learning operations ought to be performed on the board, or whether fixed values for weights could be loaded into the cells. Note that in the latter case the function of the cells can be very simple, like that of the conventional formal neurons. One beneficial property of the maps is that their parameters usually stabilize out into a narrow dynamic range, and the accuracy requirements are then modest. In this case even integer arithmetic operations can provide for sufficient accuracy.

What is the most significant difference between the Self-Organizing Map and other contemporary neural-model approaches? Most of the latter strongly emphasize the aspect of distributed processing, and only consider spatial organization of the processing units as a secondary aspect. The map principle, on the other hand, is in some ways complementary to this idea. The intrinsic potential of this particular self-organizing process for creating a localized, structured arrangement of representations in the basic network module is emphasized.

Actually, we should not talk of the localization of a "function": it is only the response that is localized. I am

thus not opposed to the view that neural networks are distributed systems. The massive interconnects that underlie all neural processing are certainly spread over the network; their *effects*, on the other hand, may be "focused" on local sites.

It seems inevitable, however, that any complex processing task requires *organization of information into separate parts*. Distributed processing models in general underrate this issue. Consequently, many models that process features of input data without structuring exhibit slow convergence and poor generalization ability, usually ensuing from ignorance of the localization of the adaptive processes.

On the lower perceptual levels, localization of responses in topographically organized maps has already been demonstrated long time ago, and it is known that such maps need not be prespecified in detail, but can instead organize themselves on the basis of the statistics of the incoming signals. Such maps have already been applied with success in many complex pattern recognition and robot control tasks.

On the higher levels of representation, relationships between items seem to be based on more subtle roles in their occurrence, and are less apparent from their immediate intrinsic properties. Nonetheless it has also been shown recently that even with a simple modeling assumption of semantic roles, topological self-organization of semantic data will take place. To describe the role of an item, it is sufficient that the input data are presented together with a sufficient amount of context. This then controls the adaptation process.

In the practical application that we have studied most carefully, viz. speech recognition, a statistical accuracy of phonemic recognition has been achieved that is clearly equal to or better than the results produced by more conventional methods, even when the latter are based on analysis of signal dynamics [28], [66].

It should be emphasized that the map method is not restricted to using of any particular form of preprocessing, such as amplitude spectra in speech recognition, or even to phonemes as basic phonological units. For instance, analogous maps may be formed for diphones, syllables, or demisyllables, and other spectral representations such as linear prediction coding (LPC) coefficients or cepstra may be used as the input information to the maps.

Although the basic one-level map, as demonstrated in Sec. II-E, has already been shown to be capable of creating hierarchical (ultrametric) representations of structured data distributions, it might be expected that the real potential of the map lies in a genuine hierarchical or otherwise structured system that consists of several interconnected map modules. In a more natural system, such modules might also correspond to contiguous areas in a single large sheet, where each area receives a different kind of external input, as in the different areas in the cortex. In that case, the borders between the modules might be diffuse. The problem of hierarchical maps, however, has turned out to be very difficult. One of the particular difficulties arises if the inputs to a cell come from very different sources; it then seems inevitable that for the comparison of input patterns, an asymmetrical distance function, in which the signal components are provided with adaptive tensorial weights, must be applied [31]. Another aspect concerns the interfaces of modules in a hierarchical map system: the signals merging from different modules may have to be combined nonlinearly [82]. On the other hand, it has already been demonstrated that the map, or the LVQ algorithms, can be used as a preprocessing stage for other models [26], [69], [101]. In the Counterpropagation Network of Hecht-Nielsen [22], competitive learning is neatly integrated into a hierarchical system as a special layer. Combinations of maps have also been studied in [74], [112], and [113].

It should be noted that slightly different self-organization ideas have recently been suggested [5], [10], [16]. They, however, fall outside the scope of this article.

One of the strongest original motives for starting the development of (artificial) neural networks was their use as learning systems that might effectively be able to utilize the vast capacities of active circuits that can be manufactured using semiconductor or optical technologies. It is therefore a little surprising that most of the theoretical research on and simulations of neural networks have been restricted to relatively small networks containing only a few tens to a few thousands of nodes (let alone parallel networks for preprocessing images). The main problem with most circuits seems to be slow convergence of learning, which again indicates that the best learning mechanisms are yet to be found.

References

[1] S.-I. Amari, "Topographic organization of nerve fields," *Bull. Math. Biology*, vol. 42, pp. 339-364, 1980.

[2] B. Angéniol, G. de la Croix Vaubois, and J.-Y. Le Texier, "Self-organizing feature maps and the travelling salesman problem," *Neural Networks*, vol. 1, pp. 289-293, 1988.

[3] N. Ansari and Y. Chen, "Dynamic digital satellite communication network management by self-organization," *Proc. Int. Joint Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-567-II-570.

[4] D. S. Bradburn, "Reducing transmission error effects using a self-organizing network," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, D.C., 1989) pp. II-531-537.

[5] D. J. Burr, "An improved elastic net method for the traveling salesman problem," *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88* (San Diego, Cal., 1988) pp. I-69-I-76.

[6] A. Caramazza, "Some aspects of language processing revealed through the analysis of acquired aphasia: The lexical system," *Ann. Rev. Neurosci.*, vol. 11, pp. 395-421, 1988.

[7] M. Cottrell and J.-C. Fort, "A stochastic model of retinotopy: A self-organizing process," *Biol. Cybern.*, vol. 53, pp. 405-411, 1986.

[8] ——, "Étude d'un processus d'auto-organisation," *Ann. Inst. Henri Poincaré*, vol. 23, pp. 1-20, 1987.

[9] A. R. Damasio, H. Damasio, and G. W. Van Hoesen, "Prosopagnosia: Anatomic basis and behavioral mechanisms," *Neurology*, vol. 24, pp. 89-93, 1975.

[10] R. Durbin and D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method," *Nature*, vol. 326, pp. 689-691, 1987.

[11] D. Essen, "Functional organization of primate visual cortex," in *Cerebral Cortex*, vol. 3, A. Peters, E. G. Jones (Eds.). New York: Plenum Press, 1985, pp. 259-329.

[12] J. Fuller and A. Farsaie, "Invariant target recognition using feature extraction," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-595-II-598.

[13] D. Gabor, "Theory of communication," *J.I.E.E.*, vol. 93, pp. 429-459, 1946.

[14] A. Gersho, "On the structure of vector quantizers," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 4, pp. 373-380, July 1979.

[15] H. Goodglass, A. Wingfield, M. R. Hyde, and J. C. Theurkauf, "Category specific dissociations in naming and recognition by aphasic patients," *Cortex*, vol. 22, pp. 87-102, 1986.

[16] I. Grabec, "Self-organization based on the second maximum entropy principle," *First IEE Int. Conf. on Artificial Neural Networks*, Conference Publication No. 313. (London, 1989) pp. 12–16.
[17] D. H. Graf and W. R. LaLonde, "A neural controller for collision-free movement of general robot manipulators," *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88* (San Diego, Cal., 1988) pp. I-77–I-84.
[18] D. H. Graf and W. R. LaLonde, "Neuroplanners for hand/eye coordination," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, DC, 1989) pp. II-543–II-548.
[19] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, pp. 4–29, 1984.
[20] S. Grossberg, "On the development of feature detectors in the visual cortex with applications to learning and reaction-diffusion systems," *Biol. Cybern.*, vol. 21, pp. 145–159, 1976.
[21] ——, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors; II. Feedback, expectation, olfaction, illusions," *Biol. Cybern.*, vol. 23, pp. 121–134 and 187–202, 1976.
[22] R. Hecht-Nielsen, "Applications of counterpropagation network," *Neural Networks*, vol. 1, pp. 131–139, 1988.
[23] A. Hemani and A. Postula, "Scheduling by self organisation," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-543–II-546.
[24] R. E. Hodges and C.-H. Wu, "A method to establish an autonomous self-organizing feature map," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. I-517–I-520.
[25] R. E. Hodges, C.-H. Wu, and C.-J. Wang, "Parallelizing the self-organizing feature map on multi-processor systems," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-141–II-144.
[26] R. M. Holdaway, "Enhancing supervised learning algorithms via self-organization," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, D.C., 1989) pp. II-523–II-529.
[27] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.
[28] H. Iwamida, S. Katagiri, E. McDermott, and Y. Tohkura, "A hybrid speech recognition system using HMMs with an LVQ-trained codebook," ATR Technical Report TR-A-0061, ATR Auditory and Visual Perception Research Laboratories, 1989.
[29] J. H. Kaas, R. J. Nelson, M. Sur, C. S. Lin, and M. M. Merzenich, "Multiple representations of the body within the primary somatosensory cortex of primates," *Science*, vol. 204, pp. 521–523, 1979.
[30] J. H. Kaas, M. M. Merzenich, and H. P. Killackey, "The reorganization of somatosensory cortex following peripheral nerve damage in adult and developing mammals," *Annual Rev. Neurosci.*, vol. 6, pp. 325–356, 1983.
[31] J. Kangas, T. Kohonen, and J. Laaksonen, "Variants of self-organizing maps," *IEEE Trans. Neural Networks*, vol. 1, pp. 93–99, 1990.
[32] A. Kertesz, Ed., *Localization in Neuropsychology*. New York, N.Y.: Academic Press, 1983.
[33] E. I. Knudsen, S. du Lac, and S. D. Esterly, "Computational maps in the brain," *Ann. Rev. Neurosci.*, vol. 10, pp. 41–65, 1987.
[34] T. Kohonen, "Automatic formation of topological maps of patterns in a self-organizing system," *Proc. 2nd Scandinavian Conf. on Image Analysis* (Espoo, Finland, 1981) pp. 214–220.
[35] ——, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, pp. 59–69, 1982.
[36] ——, "Clustering, taxonomy, and topological maps of patterns," *Proc. Sixth Int. Conf. on Pattern Recognition* (Munich, Germany, 1982) pp. 114–128.
[37] ——, "Dynamically expanding context, with application to the correction of symbol strings in the recognition of continuous speech," *Proc. Eighth Int. Conf. on Pattern Recognition* (Paris, France, 1986) pp. 1148–1151.
[38] ——, "Learning Vector Quantization," Helsinki University of Technology, Laboratory of Computer and Information Science, Report TKK-F-A-601, 1986.
[39] ——, *Content-Addressable Memories*, 2nd ed. Berlin, Heidelberg, Germany: Springer-Verlag, 1987.
[40] ——, "Self-learning inference rules by dynamically expanding context," *Proc. IEEE First Ann. Int. Conf. on Neural Networks* (San Diego, CA, 1987) pp. II-3–II-9.
[41] ——, "An introduction to neural networks," *Neural Networks*, vol. 1, pp. 3–16, 1988.
[42] ——, "The 'neural' phonetic typewriter," *Computer*, vol. 21, pp. 11–22, March 1988.
[43] ——, "Learning vector quantization," *Neural Networks*, vol. 1, suppl. 1, p. 303, 1988.
[44] ——, *Self-Organization and Associative Memory*, 3rd ed. Berlin, Heidelberg, Germany: Springer-Verlag, 1989.
[45] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with neural networks: Benchmarking studies," *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88* (San Diego, Cal., 1988) pp. I-61–I-68.
[46] T. Kohonen, K. Mäkisara, and T. Saramäki, "Phonotopic maps—insightful representation of phonological features for speech recognition," *Proc. Seventh Int. Conf. on Pattern Recognition* (Montreal, Canada, 1984) pp. 182–185.
[47] T. Kohonen, K. Raivio, O. Simula, O. Ventä, and J. Henriksson, "An adaptive discrete-signal detector based on self-organizing maps," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-249–II-252.
[48] T. Kohonen, K. Torkkola, M. Shozakai, J. Kangas, and O. Ventä, "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.
[49] H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York, Berlin: Springer-Verlag, 1978.
[50] J. Lampinen and E. Oja, "Fast self-organization by the probing algorithm," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, DC, 1989) pp. II-503–II-507.
[51] A. LaVigna, "Nonparametric classification using learning vector quantization," Ph.D. Thesis, University of Maryland, 1989.
[52] J. Lazarro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all network of O(N) complexity," in *Advances in Neural Information Processing Systems I*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann Publishers, 1989.
[53] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, pp. 1035–1073, Apr. 1983.
[54] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantization," *IEEE Trans. Communication*, vol. COM-28, pp. 84–95, 1980.
[55] S. P. Luttrell, "Self-organizing multilayer topographic mappings," *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88* (San Diego, CA, 1988) pp. I-93–I-100.
[56] ——, "Hierarchical self-organizing networks," *First IEE Int. Conf. on Artificial Neural Networks*, Conference Publication No. 313. (London, 1989) pp. 2–6.
[57] ——, "Self-organization: A derivation from first principles of a class of learning algorithms," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, DC, 1989) pp. II-495–II-498.
[58] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, pp. 1551–1588, 1985.
[59] Ch. v.d Malsburg, "Self-organization of orientation sensitive cells in the striate cortex," *Kybernetik*, vol. 14, pp. 85–100, 1973.
[60] Ch. v.d. Malsburg and D. J. Willshaw, "How to label nerve cells so that they can interconnect in an ordered fashion," *Proc. Natl. Acad. Sci. USA*, vol. 74, pp. 5176–5178, 1977.
[61] R. Mann and S. Haykin, "A parallel implementation of Kohonen feature maps on the Warp systolic computer," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-84–II-87.
[62] K. M. Marks and K. F. Goser, "Analysis of VLSI process data based on self-organizing feature maps," *Proc. Neuro-Nîmes'88* (Nîmes, France, 1988) pp. 337–347.

[63] J. Martinetz, H. J. Ritter, and K. J. Schulten, "Three-dimensional neural net for learning visuomotor coordination of a robot arm," *IEEE Trans. Neural Networks*, vol. 1, pp. 131-136, 1990.
[64] J. Max, "Quantizing for minimum distortion," *IRE Trans. Inform. Theory*, vol. IT-6, no. 2, pp. 7-12, Mar. 1960.
[65] R. A. McCarthy and E. K. Warrington, "Evidence for modality specific meaning systems in the brain," *Nature*, vol. 334, pp. 428-430, 1988.
[66] E. McDermott and S. Katagiri, "Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2," *Proc. Int. Conf. on Acoustics, Signals, and Speech, ICASSP 89* (Glasgow, Scotland) pp. 81-84.
[67] P. McKenna and E. K. Warrington, "Category-specific naming preservation: A single case study," *J. Neurol. Neurosurg. Psychiatry*, vol. 41, pp. 571-574, 1978.
[68] R. Miikkulainen and M.-G. Dyer, "Forming global representations with extended backpropagation," *Proc. IEEE Int. Conf. on Neural Networks, ICNN 88* (San Diego, CA, 1988) pp. 285-292.
[69] P. Morasso, "Neural models of cursive script handwriting," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, DC, 1989) pp. II-539-II-542.
[70] J. T. Murphy, H. C. Kwan, W. A. MacKay, and Y. C. Wong, "Spatial organization of precentral cortex in awake primates, III, Input-output coupling," *J. Neurophysiol.*, vol. 41, pp. 1132-1139, 1977.
[71] N. M. Nasrabadi and Y. Feng, "Vector quantization of images based upon the Kohonen Self-Organizing Feature Maps," *Proc. IEEE Int. Conf. on Neural Networks, ICNN-88* (San Diego, CA, 1988) pp. I-101-I-108.
[72] M. M. Nass and L. N. Cooper, "A theory for the development of feature detecting cells in visual cortex," *Biol. Cybern.*, vol. 19, pp. 1-18, 1975.
[73] E. K. Neumann, D. A. Wheeler, J. W. Burnside, A. S. Bernstein, and J. C. Hall, "A technique for the classification and analysis of insect courtship song," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC*, (Washington, DC, 1990) pp. II-257-262.
[74] Y. Nishikawa, H. Kita, and A. Kawamura, "NN/I: a neural network which divides and learns environments," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC*, (Washington, DC, 1990) pp. I-684-I-687.
[75] G. A. Ojemann, "Brain organization for language from the perspective of electrical stimulation mapping," *Behav. Brain Sci.*, vol. 2, pp. 189-230, 1983.
[76] J. Orlando, R. Mann, and S. Haykin, "Radar classification of sea-ice using traditional and neural classifiers," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC*, (Washington, DC, 1990) pp. II-263-II-266.
[77] K. J. Overton and M. A. Arbib, "The branch arrow model of the formation of retinotectal connections," *Biol. Cybern.*, vol. 45, pp. 157-175, 1982.
[78] K. J. Pearson, L. H. Finkel, and G. M. Edelman, "Plasticity in the organization of adult cerebral maps: a computer simulation based on neuronal group selection," *J. Neurosci.*, vol. 12, pp. 4209-4223, 1987.
[79] R. Pérez, L. Glass, and R. J. Shlaer, "Development of specificity in cat visual cortex," *J. Math. Biol.*, vol. 1, pp. 275-288, 1975.
[80] S. E. Petersen, P. T. Fox, M. I. Ponsner, M. Mintun, and M. E. Raichle, "Positron emission tomographic studies of the cortical anatomy of single-word processing," *Nature*, vol. 331, pp. 585-589, 1988.
[81] M. Porat and Y. Y. Zeevi, "The generalized Gabor scheme of image representation in biological and machine vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-10, pp. 452-468, 1988.
[82] H. Ritter, "Combining self-organizing maps," *Proc. Int. Joint Conf. on Neural Networks, IJCNN 89* (Washington, DC, 1989) pp. II-499-II-502.
[83] ——, "Asymptotic level density for a class of vector quantization processes," Helsinki University of Technology, Lab. of Computer and Information Science, Report A9, 1989.
[84] H. Ritter and T. Kohonen, "Self-organizing semantic maps," *Biol. Cybern.*, vol. 61, pp. 241-254, 1989.
[85] H. J. Ritter, T. M. Martinetz, and K. J. Schulten, "Topology conserving maps for learning visuo-motor-coordination," *Neural Networks*, vol. 2, pp. 159-168, 1989.
[86] H. Ritter and K. Schulten, "On the stationary state of Kohonen's self-organizing sensory mapping," *Biol. Cybern.*, vol. 54, pp. 99-106, 1986.
[87] ——, "Topology conserving mappings for learning motor tasks," *Proc. Neural Networks for Computing*, AIP Conference (Snowbird, Utah, 1986) pp. 376-380.
[88] ——, "Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements," *NATO ASI Series*, vol. F41, pp. 393-406, 1988.
[89] ——, "Kohonen's self-organizing maps: exploring their computational capabilities," *Proc. IEEE Int. Conf. on Neural Networks, ICNNN 88* (San Diego, CA, 1988) pp. I-109-I-116.
[90] ——, "Convergency properties of Kohonen's topology conserving maps: Fluctuations, stability and dimension selection," *Biol. Cybern.*, vol. 60, pp. 59-71, 1989.
[91] R. A. Reale and T. J. Imig, "Tonotopic organization in auditory cortex of the cat," *J. Comp. Neurol.*, vol. 192, pp. 265-291, 1980.
[92] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400-407, 1951.
[93] E. T. Rolls, "Neurons in the cortex of the temporal lobe and in the amygdala of the monkey with responses selective for faces," *Hum. Neurobiol.*, vol. 3, pp. 209-222, 1984.
[94] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1.: Foundations*, D. E. Rumelhart, J. L. McClelland and the PDP research group, Eds. Cambridge, Mass.: MIT Press, 1986, pp. 318-362.
[95] J. K. Samarabandu and O. E. Jakubowicz, "Principles of sequential feature maps in multi-level problems," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-683-II-686.
[96] P. G. Schyns, "Expertise acquisition through concepts refinement in a self-organizing architecture," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. I-236-I-239.
[97] D. L. Sparks and J. S. Nelson, "Sensory and motor maps in the mammalian superior colliculus," *TINS*, vol. 10, pp. 312-317, 1987.
[98] N. Suga and W. E. O'Neill, "Neural axis representing target range in the auditory cortex of the mustache bat," *Science*, vol. 206, pp. 351-353, 1979.
[99] N. V. Swindale, "A model for the formation of ocular dominance stripes," *Proc. R. Soc.*, vol. B.208, pp. 243-264, 1980.
[100] A. Takeuchi and S. Amari, "Formation of topographic maps and columnar microstructures," *Biol. Cybern.*, vol. 35, pp. 63-72, 1979.
[101] T. Tanaka, M. Naka, and K. Yoshida, "Improved back-propagation combined with LVQ," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. I-731-734.
[102] V. Tryba, K. M. Marks, U. Rückert, and K. Goser, "Selbstorganisierende Karten als lernende klassifizierende Speicher," *ITG Fachbericht*, vol. 102, pp. 407-419, 1988.
[103] A. R. Tunturi, "Physiological determination of the arrangement of the afferent connections to the middle ectosylvian auditory area in the dog," *Am. J. Physiol.*, vol. 162, pp. 489-502, 1950.
[104] ——, "The auditory cortex of the dog," *Am. J. Physiol.*, vol. 168, pp. 712-717, 1952.
[105] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech and Signal Processing*, vol. ASSP-37, pp. 382-339, 1989.
[106] E. K. Warrington, "The selective impairment of semantic memory," *Q. J. Exp. Psychol.*, vol. 27, pp. 635-657, 1975.
[107] E. K. Warrington and R. A. McGarthy, "Category specific access dysphasia," *Brain*, vol. 106, pp. 859-878, 1983.
[108] ——, "Categories of knowledge," *Brain*, vol. 110, pp. 1273-1296, 1987.
[109] E. K. Warrington and T. Shallice, "Category-specific impairments," *Brain*, vol. 107, pp. 829-854, 1984.

[110] D. J. Willshaw and Ch. v.d. Malsburg, "How patterned neural connections can be set up by self-organization," *Proc. R. Soc. London*, vol. B 194, pp. 431–445, 1976.
[111] ——, "A marker induction mechanism for the establishment of ordered neural mappings: its application to the retinotectal problem," *Proc. R. Soc. London*, vol. B 287, pp. 203–243, 1979.
[112] L. Xu and E. Oja, "Vector pair correspondence by a simplified counter-propagation model: a twin topographic map," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. II-531–534.
[113] ——, "Adding top-down expectation into the learning procedure of self-organizing maps," *Proc. Int. Joint. Conf. on Neural Networks, IJCNN-90-WASH-DC* (Washington, DC, 1990) pp. I-735–738.
[114] A. Yamadori and M. L. Albert, "Word category aphasia," *Cortex*, vol. 9, pp. 112–125, 1973.
[115] P. L. Zador, "Asymptotic quantization error of continuous signals and the quantization dimension," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 139–149, March 1982.
[116] S. Zeki, "The representation of colours in the cerebral cortex," *Nature*, vol. 284, pp. 412–418, 1980.

Teuvo Kohonen (Senior Member, IEEE) received the D.Eng. degree in physics from Helsinki University of Technology, Espoo, Finland, in 1962.

He is a professor on the Faculty of Information Sciences of Helsinki University of Technology, and is also a research professor of the Academy of Finland. His scientific interests are neural computers and pattern recognition. He is the author of four textbooks, of which *Content-Addressable Memories* (Springer, 1987) and *Self-Organization and Associative Memory* (Springer, 1989) are best known.

Dr. Kohonen is a member of the Finnish Academy of Sciences, and of the Finnish Academy of Engineering Sciences. He has served as the first vice chairman of the International Association for Pattern Recognition, and as a Governing Board Member of the International Neural Network Society. He was awarded the Eemil Aaltonen Prize in 1983 and the Cultural Prize of Finnish Commercial Television in 1984.