# 15

# Kohonen Networks

## 15.1 Self-organization

In this chapter we consider *self-organizing networks*. The main difference between them and conventional models is that the correct output cannot be defined *a priori*, and therefore a numerical measure of the magnitude of the mapping error cannot be used. However, the learning process leads, as before, to the determination of well-defined network parameters for a given application.

### 15.1.1 Charting input space

In Chap. 5 we considered networks with a single layer which learn to identify clusters of vectors. This is also an example of a self-organizing system, since the correct output was not predefined and the mapping of weight vectors to cluster centroids is an automatic process.

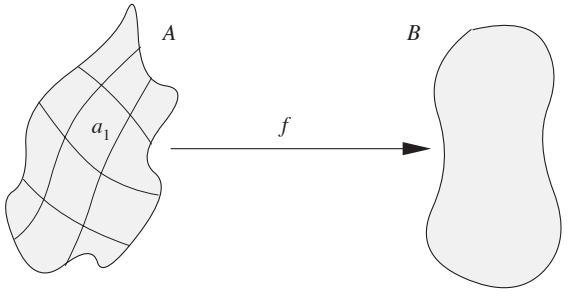


**Fig. 15.1.** A function $f : A \rightarrow B$

When a self-organizing network is used, an input vector is presented at each step. These vectors constitute the "environment" of the network. Each

new input produces an adaptation of the parameters. If such modifications are correctly controlled, the network can build a kind of internal representation of the environment. Since in these networks learning and "production" phases can be overlapped, the representation can be updated continuously.

The best-known and most popular model of self-organizing networks is the topology-preserving map proposed by Teuvo Kohonen [254, 255]. So-called *Kohonen networks* are an embodiment of some of the ideas developed by Rosenblatt, von der Malsburg, and other researchers. If an input space is to be processed by a neural network, the first issue of importance is the structure of this space. A neural network with real inputs computes a function $f$ defined from an input space $A$ to an output space $B$. The region where $f$ is defined can be covered by a Kohonen network in such a way that when, for example, an input vector is selected from the region $a_1$ shown in Figure 15.1, only one unit in the network fires. Such a tiling in which input space is classified in subregions is also called a chart or map of input space. Kohonen networks learn to create maps of the input space in a self-organizing way.

### 15.1.2 Topology preserving maps in the brain

Kohonen's model has a biological and mathematical background. It is well known in neurobiology that many structures in the brain have a linear or planar topology, that is, they extend in one or two dimensions. Sensory experience, on the other hand, is multidimensional. A simple event, such as the perception of color, presupposes interaction between three different kinds of light receptors. The eyes capture additional information about the structure, position, and texture of objects too. The question is: how do the planar structures in the brain manage to process such multidimensional signals? Put another way: how is the multidimensional input projected to the two-dimensional neuronal structures? This important question can be illustrated with two examples.

The visual cortex is a well-studied region in the posterior part of the human brain. Many neurons work together to decode and process visual impressions. It is interesting to know that the visual information is mapped as a two-dimensional projection on the cortex, despite the complexity of the pathway from the retina up to the cortical structures. Figure 15.2 shows a map of the visual cortex very similar to the one already discovered by Gordon Holmes at the beginning of the century [161]. The diagram on the right represents the whole visual field and its different regions. The inner circle represents the center of the visual field, whose contents are captured by the fovea. The diagram on the left uses three different shadings to show the correspondence between regions of the visual cortex and regions of the visual field. Two important phenomena can be observed: firstly, that neighboring regions of the visual field are processed by neighboring regions in the cortex; and secondly, that the surface of the visual cortex reserved for the processing of the information from the fovea is disproportionally large. This means that signals from the center of

the visual field are processed in more detail and with higher resolution than signals from the periphery of the visual field. Visual acuity increases from the periphery to the center.

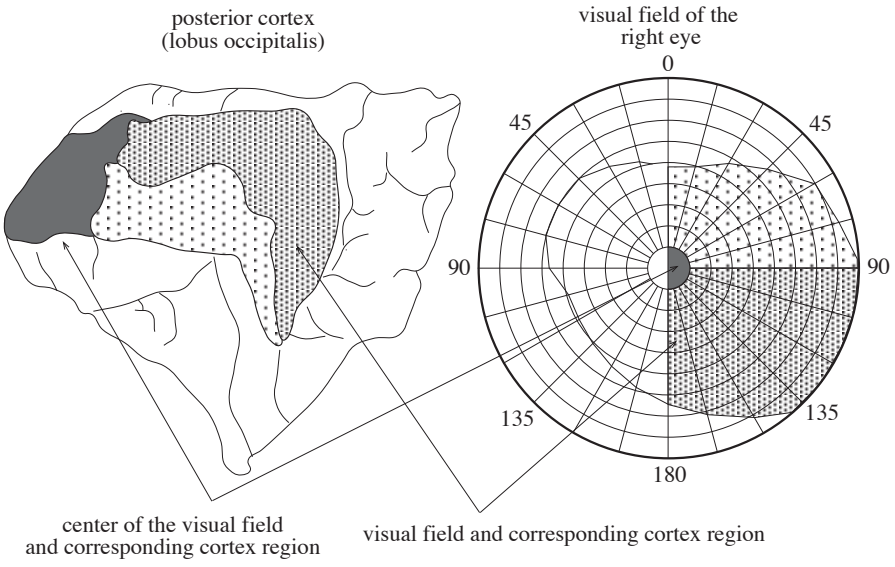


**Fig. 15.2.** Mapping of the visual field on the cortex

The visual cortex is therefore a kind of map of the visual field. Since this is a projection of the visual world onto the spherically arranged light receptors in the retina, a perfect correspondence between retina and cortex would need a spherical configuration of the latter. However, the center of the visual field maps to a proportionally larger region of the cortex. The form of the extended cortex should therefore resemble a deformed sphere. Physiological experiments have confirmed this conjecture [205].

In the human cortex we not only find a topologically ordered representation of the visual field but also of sensations coming from other organs. Figure 15.3 shows a slice of two regions of the brain: to the right the somatosensory cortex, responsible for processing mechanical inputs, and to the left the motor cortex, which controls the voluntary movement of different body parts. Both regions are present in each brain hemisphere and are located contiguous to each other.

Figure 15.3 shows that the areas of the brain responsible for processing body signals are distributed in a topology-preserving way. The region in charge of signals from the arms, for example, is located near to the region responsible for the hand. As can be seen, the spatial relations between the body parts are preserved as much as possible in the sensory cortex. The same phenomenon can be observed in the motor cortex.

**Fig. 15.3.** The somatosensory and motor cortex

Of course, all details of how the cortex processes sensory signals have not yet been elucidated. However, it seems a safe assumption that the first representation of the world built by the brain is a topological one, in which the exterior spatial relations are mapped to similar spatial relations in the cortex. One might think that the mapping of the sensory world onto brains is genetically determined, but experiments with cats that are blind in one eye have shown that those areas of the cortex which in a normal cat process information from the lost eye readapt and can process information from the other eye. This can only happen if the cat loses vision from one eye when the brain is still developing, that is, when it still possesses enough plasticity. This means that the topological correspondence between retina and cortex is not totally genetically determined and that sensory experience is necessary for the development of the correct neural circuitry [124].

Kohonen's model of self-organizing networks goes to the heart of this issue. His model works with elements not very different from the ones used by other researchers. More relevant is the definition of the neighborhood of a computing unit. Kohonen's networks are arrangements of computing nodes in one-, two-, or multi-dimensional lattices. The units have lateral connec-

tions to several neighbors. Examples of this kind of lateral coupling are the inhibitory connections used by von der Malsburg in his self-organizing models [284]. Connections of this type are also present, for example, in the human retina, as we discussed in Chap. 3.

## 15.2 Kohonen's model

In this section we deal with some examples of the ordered structures known as Kohonen networks. The grid of computing elements allows us to identify the immediate neighbors of a unit. This is very important, since during learning the weights of computing units and their neighbors are updated. The objective of such a learning approach is that neighboring units learn to react to closely related signals.

### 15.2.1 Learning algorithm

Consider the problem of charting an $n$-dimensional space using a one-dimensional chain of Kohonen units. The units are all arranged in sequence and are numbered from 1 to $m$ (Figure 15.4). Each unit becomes the $n$-dimensional input $\mathbf{x}$ and computes the corresponding excitation. The $n$-dimensional weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ are used for the computation. The objective of the charting process is that each unit learns to specialize on different regions of input space. When an input from such a region is fed into the network, the corresponding unit should compute the maximum excitation. Kohonen's learning algorithm is used to guarantee that this effect is achieved.

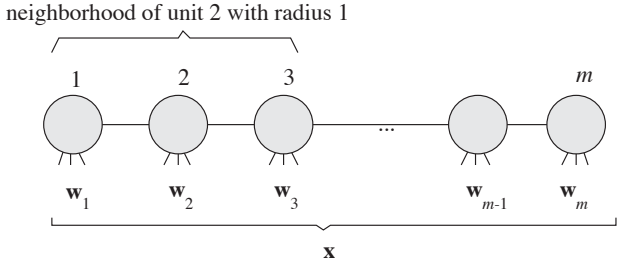


**Fig. 15.4.** A one-dimensional lattice of computing units

A Kohonen unit computes the Euclidian distance between an input $\mathbf{x}$ and its weight vector $\mathbf{w}$. This new definition of excitation is more appropriate for certain applications and also easier to visualize. In the Kohonen one-dimensional network, the neighborhood of radius 1 of a unit at the $k$-th position consists of the units at the positions $k-1$ and $k+1$. Units at both

ends of the chain have asymmetrical neighborhoods. The neighborhood of radius $r$ of unit $k$ consists of all units located up to $r$ positions from $k$ to the left or to the right of the chain.

Kohonen learning uses a neighborhood function $\phi$, whose value $\phi(i,k)$ represents the strength of the coupling between unit $i$ and unit $k$ during the training process. A simple choice is defining $\phi(i,k) = 1$ for all units $i$ in a neighborhood of radius $r$ of unit $k$ and $\phi(i,k) = 0$ for all other units. We will later discuss the problems that can arise when some kinds of neighborhood functions are chosen. The learning algorithm for Kohonen networks is the following:

**Algorithm 15.2.1** *Kohonen learning*

*start*: The $n$-dimensional weight vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ of the $m$ computing units are selected at random. An initial radius $r$, a learning constant $\eta$, and a neighborhood function $\phi$ are selected.

*step 1*: Select an input vector $\xi$ using the desired probability distribution over the input space.

*step 2*: The unit $k$ with the maximum excitation is selected (that is, for which the distance between $\mathbf{w}_i$ and $\xi$ is minimal, $i = 1, \ldots, m$).

*step 3*: The weight vectors are updated using the neighborhood function and the update rule

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta\phi(i,k)(\xi - \mathbf{w}_i), \quad \text{for} \quad i = 1, \ldots, m.$$

*step 4*: Stop if the maximum number of iterations has been reached; otherwise modify $\eta$ and $\phi$ as scheduled and continue with step 1.

The modifications of the weight vectors (step 3) attracts them in the direction of the input $\xi$. By repeating this simple process several times, we expect to arrive at a uniform distribution of weight vectors in input space (if the inputs have also been uniformly selected). The radius of the neighborhood is reduced according to a previous plan, which we call a *schedule*. The effect is that each time a unit is updated, neighboring units are also updated. If the weight vector of a unit is attracted to a region in input space, the neighbors are also attracted, although to a lesser degree. During the learning process both the size of the neighborhood and the value of $\phi$ fall gradually, so that the influence of each unit upon its neighbors is reduced. The learning constant controls the magnitude of the weight updates and is also reduced gradually. The net effect of the selected schedule is to produce larger corrections at the beginning of training than at the end.

Figure 15.5 shows the results of an experiment with a one-dimensional Kohonen network. Each unit is represented by a dot. The input domain is a triangle. At the end of the learning process the weight vectors reach a distribution which transforms each unit into a "representative" of a small region of input space. The unit in the lower corner, for example, is the one

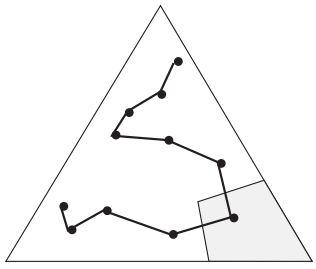

**Fig. 15.5.** Map of a triangular region

which responds with the largest excitation for vectors in the shaded region. If we adopt a "winner-takes-all" activation strategy, then it will be the only one to fire.

The same experiment can be repeated for differently shaped domains. The chain of Kohonen units will adopt the form of a so-called *Peano curve*. Figure 15.6 is a series of snapshots of the learning process from 0 to 25000 iterations [255]. At the beginning, before training starts, the chain is distributed randomly in the domain of definition. The chain unwraps little by little and the units distribute gradually in input space. Finally, when learning approaches its end, only small corrections affect the unit's weights. At that point, the neighborhood radius has been reduced to zero and the learning constant has reached a small value.

**Fig. 15.6.** Mapping a chain to a triangle

Kohonen networks can be arranged in multidimensional grids. An interesting choice is a planar network, as shown in Figure 15.7. The neighborhood of radius $r$ of unit $k$ consists, in this case, of all other units located at most $r$ places to the left or right, up or down in the grid. With this convention, the neighborhood of a unit is a quadratic portion of the network. Of course we can define more sophisticated neighborhoods, but this simple approach is all that is needed in most applications.

Figure 15.7 shows the flattening of a two-dimensional Kohonen network in a quadratic input space. The four diagrams display the state of the network after 100, 1000, 5000, and 10000 iterations. In the second diagram several iterations have been overlapped to give a feeling of the iteration process. Since in this experiment the dimension of the input domain and of the network are the same, the learning process reaches a very satisfactory result.

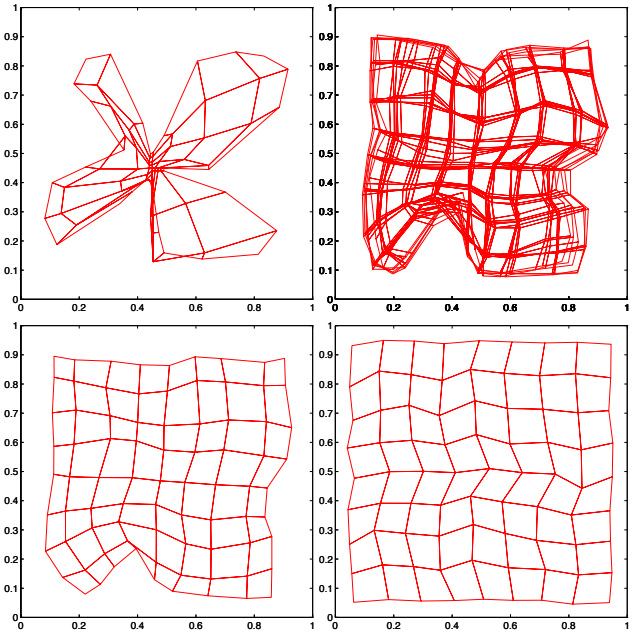

**Fig. 15.7.** Mapping a square with a two-dimensional lattice. The diagram on the upper right shows some overlapped iterations of the learning process. The diagram below it is the final state after 10000 iterations.

Settling on a stable state is not so easy in the case of multidimensional networks. There are many factors which play a role in the convergence process, such as the size of the selected neighborhood, the shape of the neighborhood function and the scheduling selected to modify both. Figure 15.8 shows an example of a network which has reached a state very difficult to correct. A knot has appeared during the training process and, if the plasticity of the network

has reached a low level, the knot will not be undone by further training, as the overlapped iterations in the diagram on the right, in Figure 15.8 show.

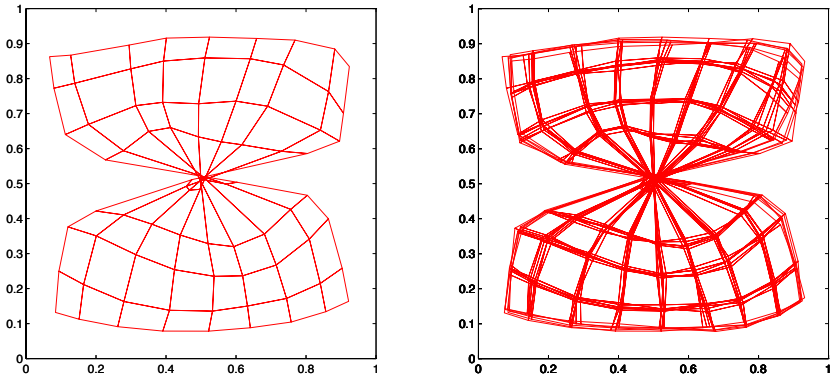


**Fig. 15.8.** Planar network with a knot

Several proofs of convergence have been given for one-dimensional Kohonen networks in one-dimensional domains. There is no general proof of convergence for multidimensional networks.

### 15.2.2 Mapping high-dimensional spaces

Usually, when an empirical data set is selected, we do not know its real dimension. Even if the input vectors are of dimension $n$, it could be that the data concentrates on a manifold of lower dimension. In general it is not obvious which network dimension should be used for a given data set. This general problem led Kohonen to consider what happens when a low-dimensional network is used to map a higher-dimensional space. In this case the network must fold in order to fill the available space. Figure 15.9 shows, in the middle, the result of an experiment in which a two-dimensional network was used to chart a three-dimensional box. As can be seen, the network extends in the $x$ and $y$ dimensions and folds in the $z$ direction. The units in the network try as hard as possible to fill the available space, but their quadratic neighborhood poses some limits to this process. Figure 15.9 shows, on the left and on the right, which portions of the network approach the upper or the lower side of the box. The black and white stripes resemble a zebra pattern.

Remember that earlier we discussed the problem of adapting the planar brain cortex to a multidimensional sensory world. There are some indications that a self-organizing process of the kind shown in Figure 15.9 could also be taking place in the human brain, although, of course, much of the brain structure emerges pre-wired at birth. The experiments show that the foldings of the planar network lead to stripes of alternating colors, that is, stripes which

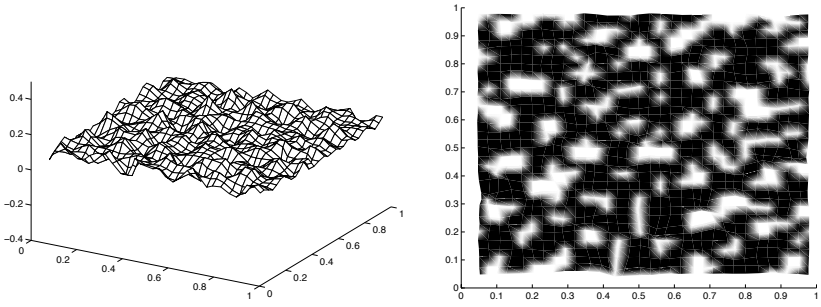

**Fig. 15.9.** Two-dimensional map of a three-dimensional region

map alternately to one side or the other of input space (for the $z$ dimension). A commonly cited example for this kind of structure in the human brain is the visual cortex. The brain actually processes not one but two visual images, one displaced with respect to the other. In this case the input domain consists of two planar regions (the two sides of the box of Figure 15.9). The planar cortex must fold in the same way in order to respond optimally to input from one or other side of the input domain. The result is the appearance of the stripes of ocular dominance studied by neurobiologists in recent years. Figure 15.10 shows a representation of the ocular dominance columns in LeVays' reconstruction [205]. It is interesting to compare these stripes with the ones found in our simple experiment with the Kohonen network.

**Fig. 15.10.** Diagram of eye dominance in the visual cortex. Black stripes represent one eye, white stripes the other.

These modest examples show the kinds of interesting consequence that can be derived from Kohonen's model. In principle Kohonen networks resemble the unsupervised networks we discussed in Chap. 5. Here and there we try to chart an input space distributing computing units to define a vector quantization. The main difference is that Kohonen networks have a *predefined topology*. They

are organized in a grid and the learning problem is to find a way to distribute this grid in input space. One could follow another approach, such as using a $k$-means type of algorithm to distribute $n \times n$ units in a quadratic domain. We could thereafter use these units as the vertices of a planar grid, which we define at will. This is certainly better if we are only interested in obtaining a grid for our quadratic domain. If we are interested in understanding how a given grid (i.e., the cortex) adapts to a complex domain, we have to start training with the folded grid and terminate training with an unfolded grid. In this case what we are investigating is the interaction between "nurture and nature". What initial configurations lead to convergence? How is convergence time affected by a preordered network compared to a fully random network? All these issues are biologically relevant and can only be answered using the full Kohonen model or its equivalent.

## 15.3 Analysis of convergence

We now turn to the difficult question of analyzing the convergence of Kohonen's learning algorithm. We will not go into detail in this respect, since the necessary mathematical machinery is rather complex and would take too much space in this chapter. We will content ourselves with looking at the conditions for stability when the network has arrived at an ordered state, and give some useful references for more detailed analysis.

### 15.3.1 Potential function – the one-dimensional case

Consider the simplest Kohonen network – it consists of just one unit and the input domain is the one-dimensional interval $[a, b]$. The learning algorithm will lead to convergence of the sole weight $x$ in the middle of the interval $[a, b]$.

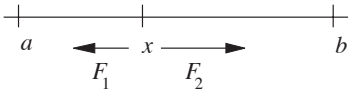


**Fig. 15.11.** The weight $x$ in the interval $[a, b]$

In our example Kohonen learning consists of the update rule

$$x_n = x_{n-1} + \eta(\xi - x_{n-1}),$$

where $x_n$ and $x_{n-1}$ represent the values of the unit's weight in steps $n$ and $n-1$ and $\xi$ is a random number in the interval $[a, b]$. If $0 < \eta \leq 1$, the series $x_1, x_2, \ldots$ cannot leave the interval $[a, b]$, that is, it is bounded. Since the expected value $\langle x \rangle$ of $x$ is also bounded, this means that the expected value of the derivative of $x$ with respect to $t$ is zero,

$$\left\langle \frac{dx}{dt} \right\rangle = 0,$$

otherwise the expected value of $x$ will eventually be lower than $a$ or greater than $b$. Since

$$\left\langle \frac{dx}{dt} \right\rangle = \eta \left( \langle \xi \rangle - \langle x \rangle \right) = \eta \left( \frac{a+b}{2} - \langle x \rangle \right),$$

this means that

$$\langle x \rangle = \frac{a+b}{2}.$$

The same technique can be used to analyze the stable states of the general one-dimensional case. Let $n$ units be arranged in a one-dimensional chain whose respective weights are denoted by $x^1, x^2, \ldots, x^n$. The network is used to chart the one-dimensional interval $[a, b]$. Assume that the weights are arranged monotonically and in ascending order, i.e. $a < x^1 < x^2 < \cdots < x^n < b$. We want to show that the expected values of the weights are given by

$$\left\langle x^i \right\rangle = a + (2i-1)\frac{b-a}{2n}. \tag{15.1}$$

All network weights are distributed as shown in Figure 15.12.

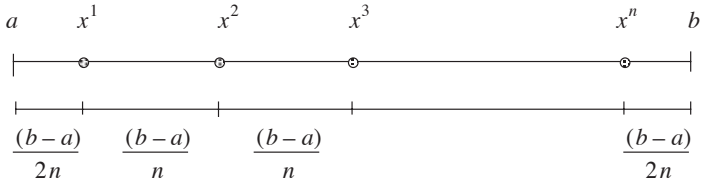


**Fig. 15.12.** Distribution of the network weights in the interval $[a, b]$

The distribution is statistically stable for a one-dimensional Kohonen network because the attraction on each weight of its domain is zero on average. The actual weights oscillate around the expected values $\left\langle x^1 \right\rangle, \ldots, \langle x^n \rangle$. Since, in Kohonen learning, the weights stay bounded, it holds for $i = 1, 2, \ldots, n$ that

$$\left\langle \frac{dx^i}{dt} \right\rangle = 0 \tag{15.2}$$

The learning algorithm does not modify the relative positions of the weights. Equation (15.2) holds if the expected values $\left\langle x^1 \right\rangle, \ldots, \langle x^n \rangle$ are distributed homogeneously, that is, in such a way that attraction from the right balances the attraction from the left. This is only possible with the distribution given by (15.1). Note that we did not make any assumptions on the form of the neighborhood function (we effectively set it to zero). If the neighborhood function

is taken into account, a small difference arises at both ends of the chain, because the attraction from one side of the neighborhood is not balanced by the corresponding attraction from the other. This can be observed during training of Kohonen networks, for example in the lower right diagram in Figure 15.7.

### 15.3.2 The two-dimensional case

Stable states for the two-dimensional case can be analyzed in a similar way. Assume that a two-dimensional Kohonen network with $n \times n$ units is used to map the interval $[a, b] \times [c, d]$. Each unit has four immediate neighbors, with the exception of the units at the borders (Figure 15.13).

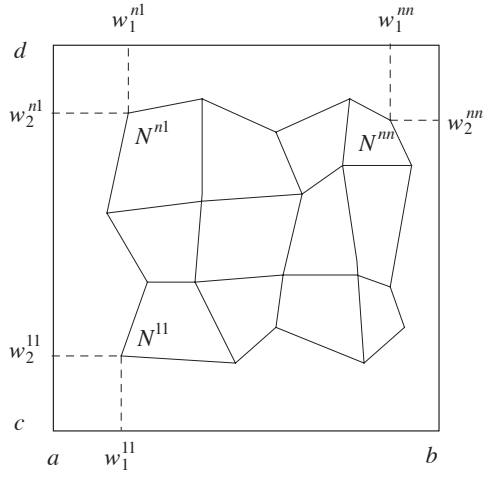


**Fig. 15.13.** Two-dimensional map

Denote the unit in the lower left corner by $N^{11}$ and the unit in the upper right corner by $N^{nn}$. Unit $N^{ij}$ is located at row $i$ and column $j$ of the network. Consider a monotonic ordering of the network weights, i.e.,

$$w_1^{ij} < w_1^{ik} \qquad \text{if } \; j < k \tag{15.3}$$

and

$$w_2^{ij} < w_2^{kj} \qquad \text{if } \; i < k, \tag{15.4}$$

where $w_1^{ij}$ and $w_2^{ij}$ denote the two weights of unit $N^{ij}$.

Kohonen learning is started with an ordered configuration of this type. It is intuitively clear that the weights will distribute homogeneously in the input domain. The two-dimensional problem can be broken down into two one-dimensional problems. Let

$$w_1^j = \frac{1}{n}\sum_{i=1}^{n} w_1^{ij}$$

denote the average value of the weights of all units in the $j$-th column. Since equation (15.4) holds, these average values are monotonically arranged, that is, for the first coordinate we have

$$a < w_1^1 < w_1^2 < \cdots < w_1^n < b.$$

If the average values are monotonically distributed, their expected values $\langle w_1^i \rangle$ will reach a homogeneous distribution in the interval $[a, b]$. Assuming that the neighborhood function has been set to zero, the units' weights $w_1^{11}, w_1^{21}, \ldots, w_1^{n1}$ of the first column will oscillate around the average value $\langle w_1^1 \rangle$. The same considerations can be made for the average values of the weights of each row of units. If the learning constant is small enough, the initial distribution will converge to a stable state.

Unfortunately, to arrive at this ideal state it is first necessary to unfold the randomly initialized Kohonen network. As Figure 15.8 shows, this process can fail at an early stage. The question is, therefore, under which conditions can such a planar network arrive at the unfolded state. Many theoreticians have tried to give an answer, but a general solution remains to be found [66, 91, 367].

### 15.3.3 Effect of a unit's neighborhood

In the previous section we did not consider the effect of a unit's neighborhood on its final stable state. Assume that the neighborhood function of a unit is given by

$$\phi(i,k) = \begin{cases} 1 & \text{if } i = k \\ 1/2 & \text{if } i = k \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

Let a linear Kohonen network be trained with this neighborhood function. The stable ordering that is eventually reached (Figure 15.14) does not divide the input domain in equal segments. The stable mean values for $x_1$ and $x_2$ are $-1/4$ and $1/4$ respectively (see Exercise 2).

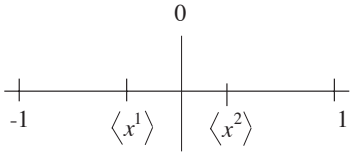


**Fig. 15.14.** Two weights in the interval $[-1, 1]$

The neighborhood function produces a concentration of units around the center of the distribution. The exact pattern depends on the neighborhood

function and the neighborhood radius. Strong coupling of the grid's units attracts them to its center. The correct learning strategy is therefore to start training with a strong coupling, which is reduced gradually as learning progresses.

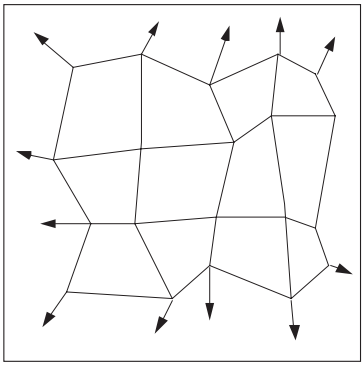

**Fig. 15.15.** Two-dimensional lattice being straightened out

In the case of a two-dimensional network (Figure 15.15), this training strategy concentrates the network towards the center. However, the periphery of the distribution attracts the network, unfolds it, and helps to achieve convergence. A way to help this process is to initialize the network with small weights (when the empirical data is centered at the origin).

### 15.3.4 Metastable states

In 1986 Cotrell and Fort showed that one-dimensional Kohonen networks converge to an ordered state if the input is selected from a uniform distribution [91]. Later Bouton and Pages extended this result for other distributions [66]. This has sometimes been interpreted as meaning that one-dimensional Kohonen networks always converge to a stable ordered state. However, this is only true for the kind of learning rule used in the proof of the convergence results. For a chain of weights $w_1, w_2, \ldots, w_n$, the weight updates are given by

$$w_k = w_k + \gamma(\xi - w_k),$$

where $k$ is equal to the index of the nearest weight to the input $\xi$ and each of its two neighbors (with the obligatory exceptions at both ends) and $\gamma$ is a learning constant. Note that this learning rule implies that the neighborhood function does not decay from one position in the chain to the next. Usually, however, a neighborhood function $\phi(i, k)$ is used to train the network and the question arises whether under some circumstances the one-dimensional Kohonen network could possibly converge to a disordered state. This is indeed the case when the neighborhood function decays too fast.

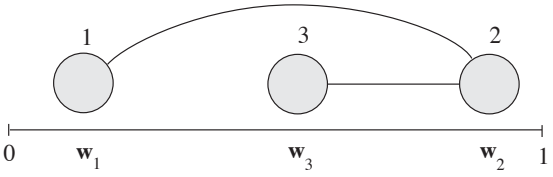


**Fig. 15.16.** A disordered one-dimensional network

To see how these *metastable* states arise, consider a simple chain of three units, which is started in the disordered combination in the interval $[0,1]$, as shown in Figure 15.16. The chain will be in equilibrium if the expected value of the total attraction on each weight is zero. Consider a simple neighborhood function

$$\phi(i,k) = \begin{cases} 1 \text{ if } i = k \\ a \text{ if } i = k \pm 1 \\ 0 \text{ otherwise} \end{cases}$$

and the learning rule

$$w_k = w_k + \gamma\phi(i,k)(\xi - w_k),$$

where all variables have the same meaning as explained before and $a$ is a real positive constant. We will denote the total attractive "force" on each weight by $f_1, f_2, f_3$. In the configuration shown in Figure 15.16 the total attractions are (see Exercise 4):

$$f_1 = (-\frac{3}{4} - a)w_1 + \frac{a}{4}w_2 + (\frac{1}{4} + \frac{a}{4})w_3 + \frac{a}{2} \tag{15.5}$$

$$f_2 = \frac{a}{4}w_1 + (-\frac{3}{4} - a)w_2 + (\frac{1}{4} + \frac{a}{4})w_3 + \frac{1}{2} \tag{15.6}$$

$$f_3 = \frac{1}{4}w_1 + (\frac{1}{4} + \frac{a}{4})w_2 + (-\frac{1}{2} - \frac{3a}{4})w_3 + \frac{a}{2} \tag{15.7}$$

This is a system of linear equations for the three weights $w_1, w_2$ and $w_3$. The associated "force" matrix is:

$$F = \frac{1}{4}\begin{pmatrix} -a-3 & a & 1+a \\ a & -a-3 & a+1 \\ 1 & a+1 & -3a-2 \end{pmatrix}$$

The matrix $F$ can be considered the negative Hesse matrix of a potential function $U(w_1, w_2, w_3)$. The solution is stable if the matrix $-F$ is positive definite. The solution of the system when $a = 0$, that is, when Kohonen units do not attract their neighbors, is $w_1 = 1/6, w_2 = 5/6, w_3 = 1/2$. This is a *stable disordered state*. If $a$ is increased slowly, the matrix of the system of linear equations is still invertible and the solutions comply with the constraints of the problem (each weight lies in the interval $[0,1]$). For $a = 0.01$, for example,

the solutions $w_1 = 0.182989, w_2 = 0.838618, w_3 = 0.517238$ can be found numerically. This is also a stable disordered state as can be proved (Exercise 4) by computing the eigenvalues of the matrix $-F$, which are given by

$$\lambda_1 = \frac{1}{3+2a}$$
$$\lambda_2 = \frac{5+3a+\sqrt{(5+3a)^2-4(4+6a-a^2)}}{2(4+6a-a^2)}$$
$$\lambda_3 = \frac{5+3a-\sqrt{(5+3a)^2-4(4+6a-a^2)}}{2(4+6a-a^2)}.$$

For small values of $a$ all three eigenvalues are real and positive. This means that $-F$ is positive definite.

We can also numerically compute the maximum value of $a$ for which it is still possible to find metastable states. For $a = 0.3$ the stable solutions no longer satisfy the constraints of the problem ($w_2$ becomes larger than 1). We should expect convergence to an *ordered* state for values of $a$ above this threshold.

The example of one-dimensional chains illustrates some of the problems associated with guaranteeing convergence of Kohonen networks. In the multidimensional case care must be taken to ensure that the appropriate neighorhood is chosen, that a good initialization is used and that the cooling scheduling does not freeze the network too soon. And still another problem remains: that of choosing the dimension of the model.

### 15.3.5 What dimension for Kohonen networks?

In many cases we have experimental data which is coded using $n$ real values, but whose effective dimension is much lower. Consider the case in which the data set consists of the points in the surface of a sphere in three-dimensional space. Although the input vectors have three components, a two-dimensional Kohonen network will do a better job of charting this input space than a three-dimensional one. Some researchers have proposed computing the effective dimension of the data before selecting the dimension of the Kohonen network, since this can later provide a smoother approximation to the data.

The dimension of the data set can be computed experimentally by measuring the variation in the number of data points closer to another data point than a given $\varepsilon$, when $\varepsilon$ is gradually increased or decreased. If, for example, the data set lies on a plane in three-dimensional space and we select a data point $\xi$ randomly, we can plot the number of data points $N(\varepsilon)$ not further away from $\xi$ than $\varepsilon$. This number should follow the power law $N(\varepsilon) \approx \varepsilon^2$. If this is the case, we settle for a two-dimensional network. Normally, the way to make this computation is to draw a plot of $\log(N(\varepsilon)$ against $\log(\varepsilon)$. The slope of the regression curve as $\varepsilon$ goes to zero is the fractal dimension of the data set.

Finding the appropriate power law means that in some cases a function has to be fitted to the measurements. Since the dimension of the data can sometimes best be approximated by a fraction, we speak of the *fractal dimension* of the data. It has been shown experimentally that if the dimension of the Kohonen network approaches the fractal dimension of the data, the interpolation error is smaller than for other network sizes [410]. This can be understood as meaning that the units in the network are used optimally to approximate input space. Other methods of measuring the fractal dimension of data are discussed by Barnsley [42].

## 15.4 Applications

Kohonen networks can adapt to domains with the most exotic structures and have been applied in many different fields, as will be shown in the following sections.

### 15.4.1 Approximation of functions

A Kohonen network can be used to approximate the continuous real function $f$ in the domain of definition $[0,1] \times [0,1]$. The set $P = \{(x, y, f(x,y)) | x, y \in [0,1]\}$ is a surface in three-dimensional space. We would like to adapt a planar grid to this surface. In this case the set $P$ is the domain which we try to map with the Kohonen network.

After the learning algorithm is started, the planar network moves in the direction of $P$ and distributes itself to cover the domain. Figure 15.17 shows the result of an experiment in which the function $z = 5 \sin x + y$ had to be learned. The combinations of $x$ and $y$ were generated in a small domain. At the end of the training process the network has "learned" the function $f$ in the region of interest.

The function used in the example above is the one which guarantees optimal control of a *pole balancing system*. Such a system consists of a pole (of mass 1) attached to a moving car. The pole can rotate at the point of attachment and the car can only move to the left or to the right. The pole should be kept in equilibrium by moving the car in one direction or the other. The necessary force to keep the pole in equilibrium is given by $f(\theta) = \alpha \sin\theta + \beta \, d\theta/dt$ [368], where $\theta$ represents the angle between the pole and the vertical, and $\alpha$ and $\beta$ are constants. (Figure 15.18). For small values of $\theta$ a linear approximation can be used. Since a Kohonen network can learn the function $f$, it can also be used to provide the automatic control for the pole balancing system.

When a combination of $x$ and $y$ is given (in this case $\theta$ and $d\theta/dt$), the unit $(i,j)$ is found for which the Euclidian distance between its associated weights $w_1^{(i,j)}$ and $w_2^{(i,j)}$ and $(\theta, d\theta/dt)$ is minimal. The value of the function at this point is the learned $w_3^{(i,j)}$, that is, an approximation to the value of the

**Fig. 15.17.** Control surface of the balancing pole [Ritter et al. 1990]

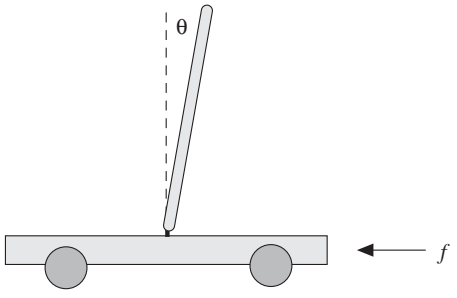


**Fig. 15.18.** A balancing pole

function $f$. The network is therefore a kind of look-up table of the values of $f$. The table can be made as sparse or as dense as needed for the application at hand. Using a table is in general more efficient than computing the function each time from scratch. If the function is not analytically given, but has been learned using some input-output examples, Kohonen networks resemble backpropagation networks. The Kohonen network can continue adapting and, in this way, if some parameters of the system change (because some parts begin to wear), such a modification is automatically taken into account. The Kohonen network behaves like an adaptive table, which can be built using a minimal amount of hardware. This has made Kohonen networks an interesting choice in robotic applications.

### 15.4.2 Inverse kinematics

A second application of Kohonen networks is mapping the configuration space of a mechanical arm using a two-dimensional grid. Assume that a robot arm

with two joints, as shown in Figure 15.19, is used to reach to all points in a table. The robot arm has two degrees of freedom, i.e., the angles $\alpha$ and $\beta$. The network is trained using a feedback signal from the manipulator. The two-dimensional Kohonen network is distributed homogenously in the quadratic domain. Now the tip of the arm is positioned manually at randomly selected points of the working surface. The weights of the Kohonen unit which is nearest to this point are updated (and also the weights of the neighbors), but the inputs for the update are the joints' angles. The network therefore charts the space of degrees of freedom. The parameters stored as weights at each node represent the combination of angles needed to reach a certain point, as shown in Figure 15.19. In this case the Kohonen network can again be considered a parameter table. Figure 15.19 shows that if the robot arm must be positioned at the selected point, it is only necessary to read the parameters from the grid. Note that in this case we assumed that the learning process avoids reaching the same point with a different parameter combination.

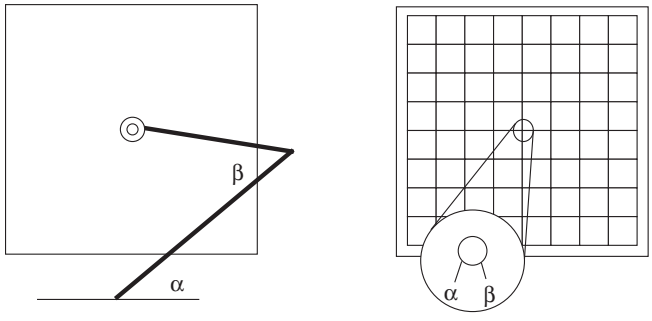


**Fig. 15.19.** Robot arm (left) and trained network (right)

More interesting is the case where we want to displace the tip of the manipulator from point $A$ to point $B$. It is only necessary to find a path from $A$ to $B$ on the grid. All units in this path provide the necessary parameters for the movement. Positions in between can be interpolated from the data.

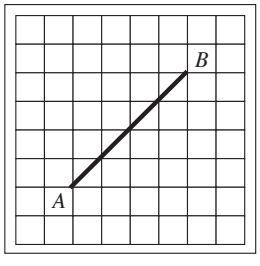


**Fig. 15.20.** Optimal path from $A$ to $B$

The problem can be made more difficult if obstacles are allowed in the working surface. Figure 15.21 shows such a hurdle. If a Kohonen network is trained, it charts the configuration space of the manipulator in such a way as to avoid the forbidden zones. If we distribute the network in the working area according to the stored parameters, the result is the one shown to the left of Figure 15.21. The network itself is still a two-dimensional grid. If we want to move the manipulator from $A$ to $B$, we can plan the movement on the grid in the way we discussed before. The actual path selection is the one shown on the right of Figure 15.21. The manipulator moves to avoid the obstacle.

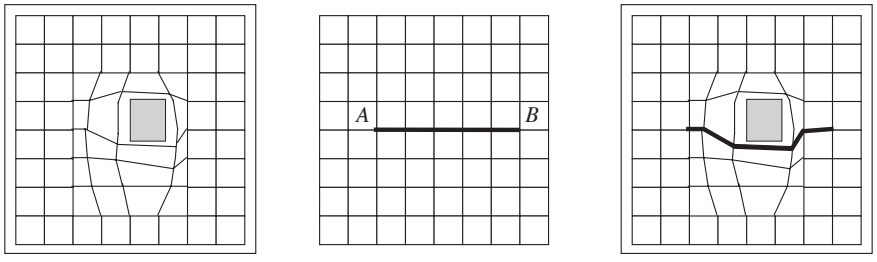


**Fig. 15.21.** Obstacles in the work area

This method can only be employed if we have previously verified that the edges of the network provide valid interpolations. It could be that two adjacent nodes contain valid information, but the path between them leads to a collision. Before using the network in the way discussed above, a previous validation step should guarantee that the edges of the grid are collision-free.

The Kohonen network can be used only if the obstacles do not change their positions. It is conceivable that if the changes occur very slowly, a fast learning algorithm could adapt the network to the changing circumstances. Kohonen networks behave in this kind of application as a generalized coordinate system. Finding the shortest path in the network corresponds to finding the shortest path in the domain of the problem, even when the movement performed by the manipulator is rather complex and nonlinear.

## 15.5 Historical and bibliographical remarks

Many applications have been developed since Kohonen first proposed the kind of networks which now bear his name. It is fascinating to see how such a simple model can offer plausible explanations for certain biological phenomena. Kohonen networks have even been used in the field of combinatorics, for example, to solve the Traveling Salesman Problem with the so-called *elastic net* algorithm [117]. A ring of units is used to encircle the cities to be visited in the Euclidian plane and Kohonen learning wraps this ring around them, so that a round trip of nearly minimal length is found.

The first applications in the field of robotics were developed and perfected in the 1980s. Kohonen has carried out some research on the application of topology-preserving networks in such diverse fields as speech recognition and structuring of semantic networks. The *phonetic typewriter* is a system based on a Kohonen map of the space of phonemes. A word can be recognized by observing the path in phoneme space reconstructed by the network [431].

Even if the original Kohonen model requires some non-biological assumptions (for example, the exchange of non-local information) there are ways to circumvent this problem. Its most important features are its self-organizing properties which arise from a very simple adaptation rule. Understanding the mathematics of such self-organizing processes is a very active field of research [222].

## Exercises

1. Implement a Kohonen one-dimensional network and map a two-dimensional square. You should get a Peano curve of the type discussed in this chapter.
2. Prove that the neighborhood function of the linear network in Sect. 15.3.3 has a stable state at $x_1 = -1/4$ and $x_2 = 1/4$. Derive a general expression for a linear chain of $n$ elements and the same neighborhood function.
3. Is it possible for a two-dimensional Kohonen network in an ordered state to become disordered after some iterations of the learning algorithm? Consider the case of a two-dimensional domain.
4. Derive the expected values of $f_1, f_2$ and $f_3$ given in equations (15.5), (15.6), and (15.7).
5. Give an example of an obstacle in the working field of the robot arm discussed in the text that makes an edge of the Kohonen network unusable, although the obstacle is far away from the edge.