Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

We, like all other “higher” animals, are adaptable creatures, we learn.

Learning goes on at different levels and using different mechanisms, but it is commonly agreed that learning involves the creation, change and death of synapses.

We model synapses with a single number, a scalar, which we usually denote by *w*. Learning involves, or in our modelling consists of, changing all *w*’s .

Techniques for changing *w* form a major part of brain and mind modelling.

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

There are three learning paradigms:

Supervised learning
Unsupervised learning or self-organization
Reinforcement learning

Supervised learning was first developed in the 1950's and is widely used today but there is no reason to believe it is used in our brains.

Self-organization became a focus of study in the 1980's. There are good reasons to believe it is used in our brains.

Reinforcement learning is obviously used in our brains, but has not been studied as much as the first two. The reason might be that it is relatively uninteresting to the engineering-physics-mathematics community.

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

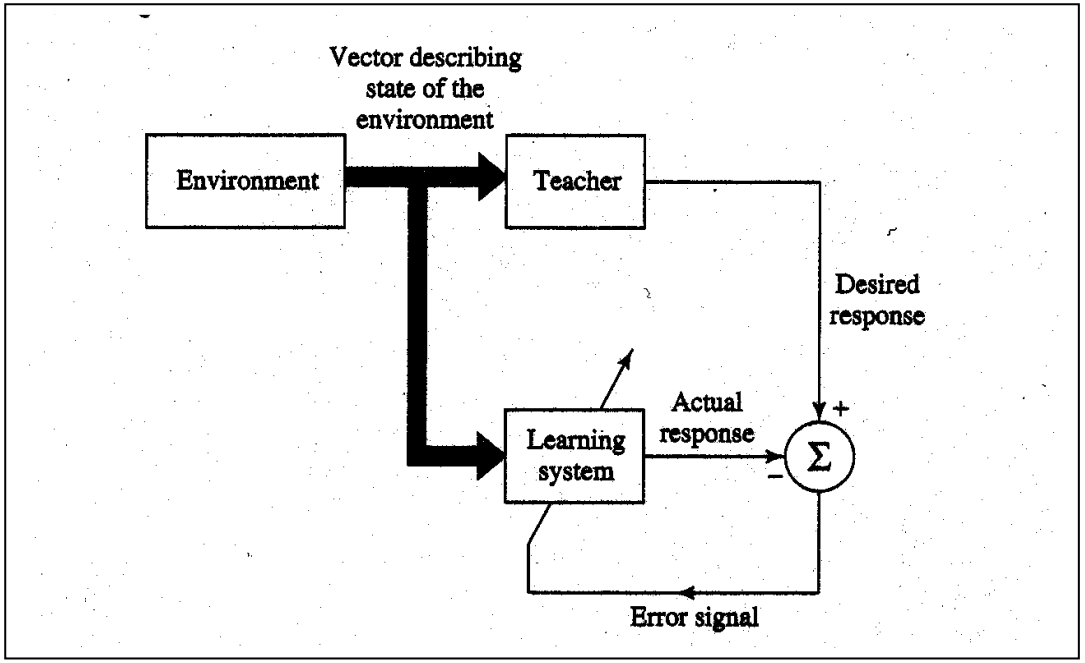
This chapter is fully devoted to supervised learning. This is reasonable for historic reasons – many believed that this was not only an interesting technique but also that it has biological relevance.

We should, however, be aware of the dangers if we try to draw consequences in the biological realm from experiments based on supervised learning. Much work of this kind has been done, it is not clear what lasting value it might have.

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

Then what is supervised learning of a neural network?

It assumes that a teacher (in this case a few lines in a computer program) is present and supervises the response of the neural network to a set of stimuli. Based on the teacher's response, the changes in all weights *w* in the neural network will be calculated so as to make the neural network respond more like the teacher than before.



From Haykin, Neural Networks A Comprehensive Foundation1994, p. 58

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

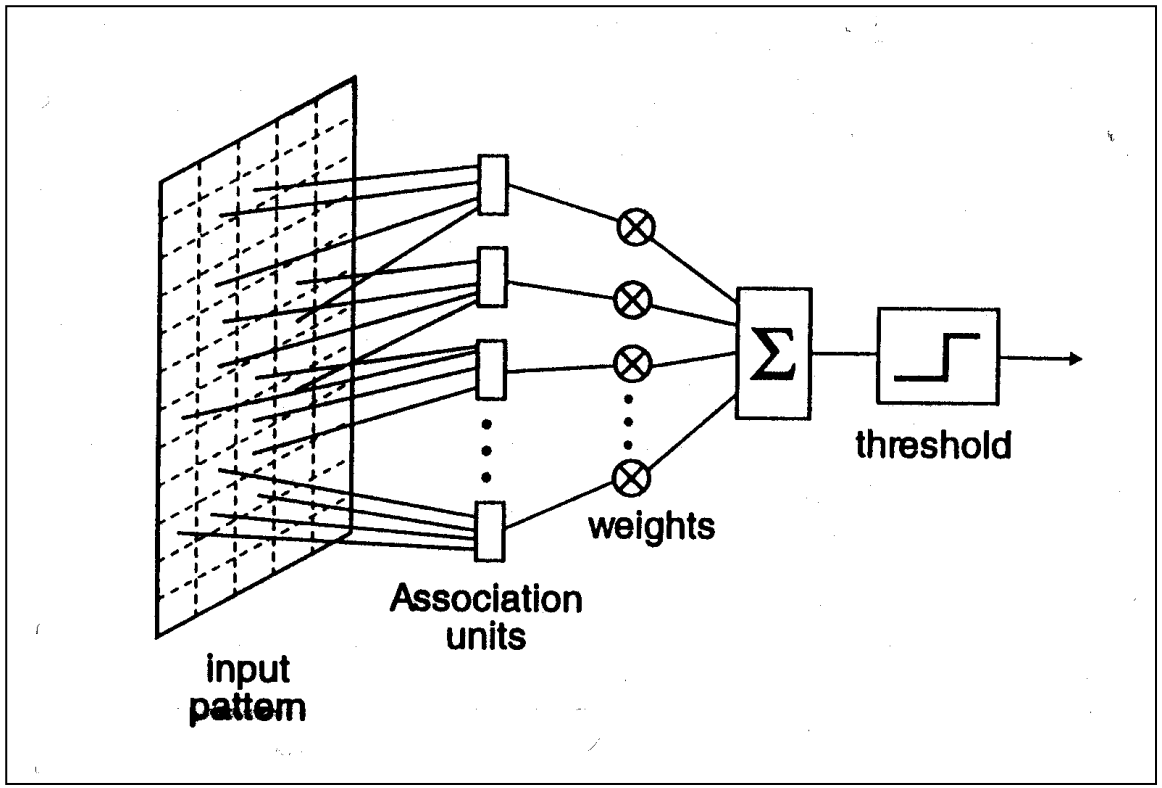
The error of the network's response is the teacher's response minus the networks response. This error is used in one of several error correcting algorithms to change the values of all *w*'s in the network.

The *Delta rule* was developed for the one-layer perceptrons, neural networks popular in the 1950's and 1960's. The idea is easy so we will discuss it in some detail.

The *Back-propagation rule*, or back-prop for short, was developed for multi-layer perceptrons and has been popular since the 1980's. The idea is easy to understand but we leave the details.

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

Here we see a one-layer perceptron with only one neuron:



From Gurney, An Introduction to Neural Networks, 1997, p. 45

The weights should be chosen to make the output high when the input pattern displays an ‘A’, and low otherwise (as an example). How?

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

The input pattern resides on a model of a retina and the predesigned association units represent some processing in the retina, i.e. “preprocessing”.

The model neuron receives its inputs from the association units, the activation is calculated as before and the output is a simple thresholding operation.

The teacher knows if the input pattern is an ‘A’ and if it is the teacher responds with a high output, say 1, otherwise by a 0.

If the input pattern is an ‘A’ then there are two possibilities :

> if the perceptron responds with a 1 then it responded correctly and its weights will not be changed.
>
> if the perceptron responds with a 0 then it responded incorrectly and its weights will be changed

If the input pattern is not an ‘A’ then there are also two possibilities.

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

There are two cases when the weights should be changed. How?

The idea is that the response should be made closer to that of the teacher.

Let us call the vector input from the association units v and the weight vector of the neuron is w. The activation of the neuron is then the dot product wx.

If the teacher responded with a 1 and the network with a 0, then the dot product wx should be increased. If w is made a bit more like x then their dot product will increase. The following rule will take care of that:

$$\Delta w = \alpha(\textit{teacher's response} - \textit{network's response})x^T$$

If the teacher responded with a 0 and the network with a 1, then the dot product wx should be decreased. If w is made a bit less like x then their dot product will decrease. The following rule will take care of that:

$$\Delta w = \alpha(\textit{teacher's response} - \textit{network's response})x^T$$

The rule is the same in the two cases so we have a single rule for changes.

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

What is α in the expression $\Delta w = \alpha(teacher's\ response - network's\ response)x$?

α is called the learning rate. A large α provides large corrective changes in the weights. So the larger the α, the better? No.

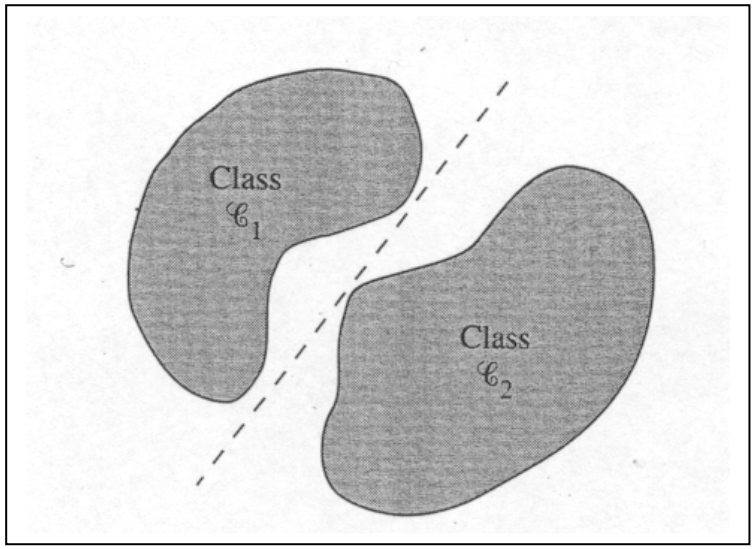
Generally α should be chosen "fairly large" in the beginning of a learning process and then decrease.

Biology certainly knows how to take care of that.

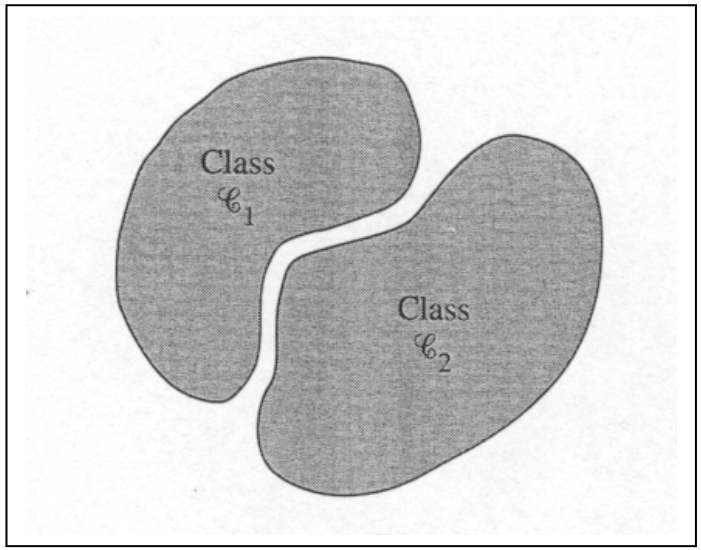
Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

What can you do with a single-layer perceptron and the delta learning rule?

You can decide what class an object belongs to if the classes are “linearly separable”.



Classes are linearly separable



Classes are not linearly separable

From Haykin, Neural Networks A Comprehensive Foundation1994, p. 119

Chapter 9, Supervised Learning: Delta Rule and Back-Propagation

The single-layer perceptron has serious limitations!

These limitations were well exposed in 1969 by Minsky and Papert in their book *Perceptrons*.

Most of the mathematics-physics-engineering community lost interest in neural networks and few worked in the field during the 1970's. The biology-psychology community of course were not worried by the computational limitations of the single-layer perceptrons (it has been suggested that these limitations are similar to those of rats).

The computational limitations of the single-layer perceptrons were overcome by the multi-layer perceptrons and the back-propagation learning rule in the beginning of the 1980's.

These networks work so well that one would like them to be a model of biology. In my opinion this is a futile hope and we therefore leave them to those who are interested in learning machines in general, not specifically the central nervous system.

There is no teacher in the brain and the error correcting learning rules, no matter how powerful, cannot be implemented in the brain, so we must make a fresh start.

A chapter that I miss: Self-organization

With no teacher present the learning situation reduces to:

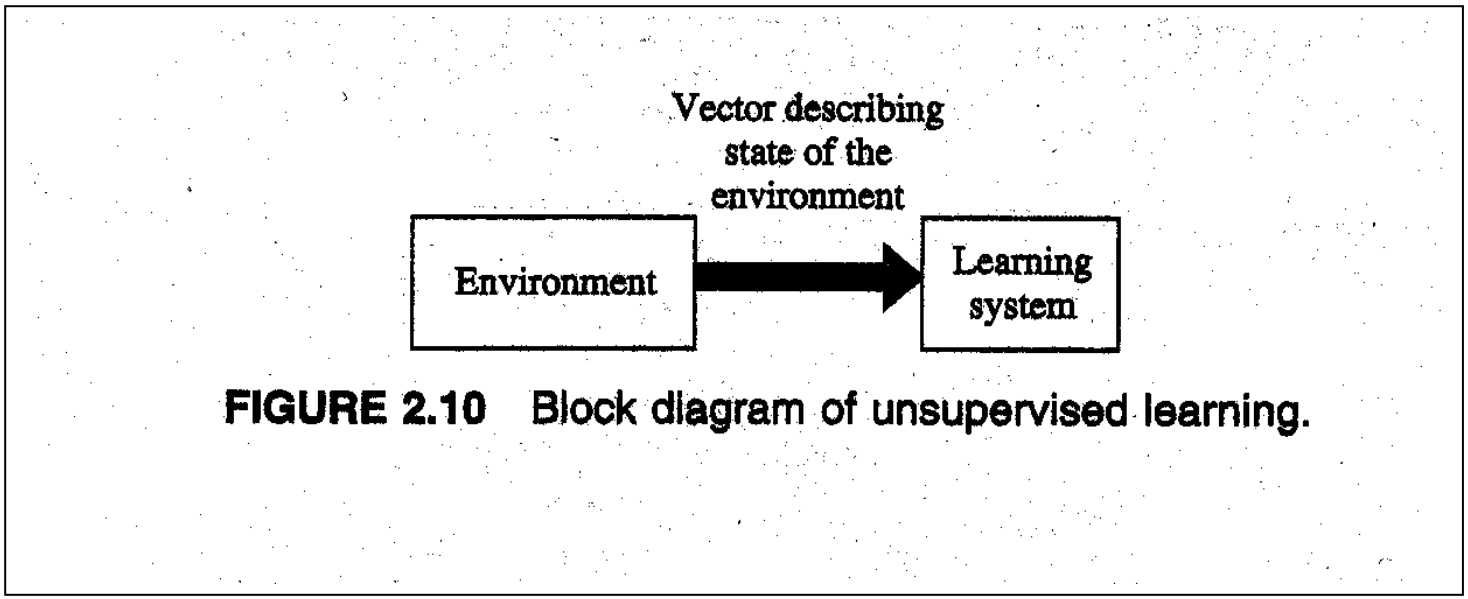


FIGURE 2.10 Block diagram of unsupervised learning.

From Haykin, Neural Networks A Comprehensive Foundation1994, p. 65

There is a seeming magic here. The network does not receive any instructions, yet it learns about the environment!

A chapter that I miss: Self-organization

We may safely assume there is structure in the environment. It is this structure that the network will eventually “pick up”.

It is still seeming magic, just more precisely(?) phrased.

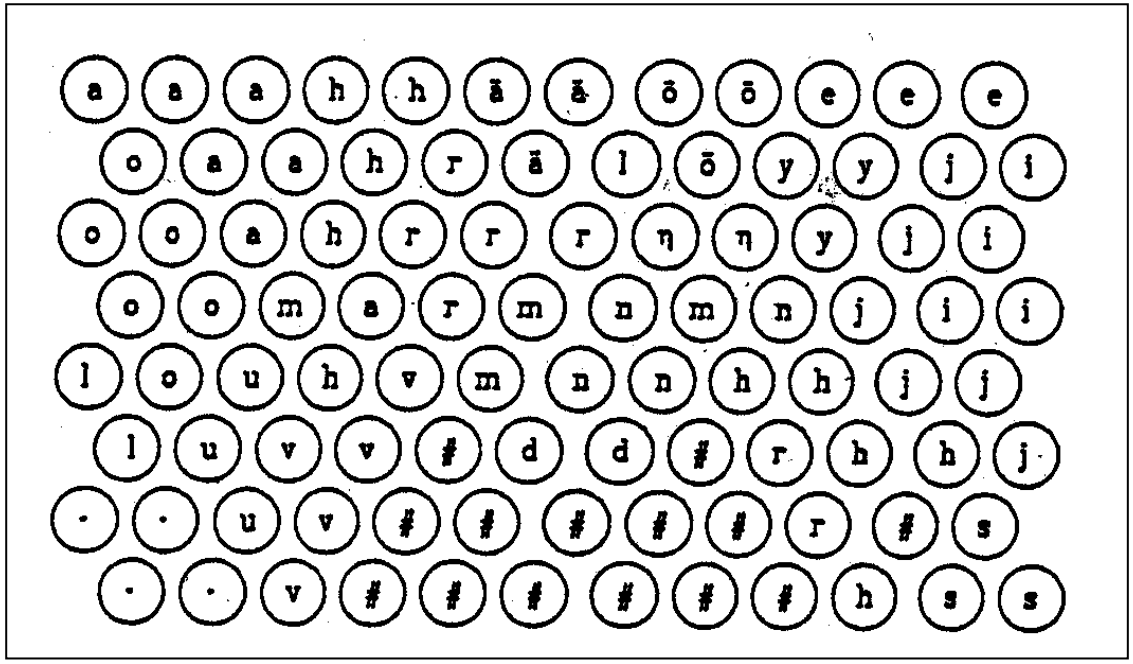
Let us think about sounds. Some of these are sound of speech, phonemes. They are important to us from the very beginning – mothers will speak to their infants in “motherese”, a variant of every language where the phonemes are extended in time to give baby a chance to pick them up.

The first goal of self-organization in auditory cortex is to tune in to the sounds that make up the phonemes of the language in your environment. Other sounds will not be given the same attention, unless of course you are a budding car mechanic.

A chapter that I miss: Self-organization

Kohonen's phonetic typewriter

In the 1980's Kohonen presented the phonetic typewriter, a neural network that received sound signals, picked up by a microphone and bandpass filtered (much like what happens in our ear). The network learned to distinguish Finnish phonemes very quickly (much faster than a human) and could print out speech onto a screen, with something like95% correct spelling.



From Kohonen, Self-Organizing Maps, 1995, p. 95

Self-organization

Kohonen invented a few tricks, speeding up learning incredibly and showing that self-organization is a credible and efficient learning paradigm. We will learn about those tricks in due time. But our primary goal is to understand how learning happens in nature, not to invent a machine that learns faster.

So back to biology!

Self-organization

We have the hypothesis by Hebb from 1949:

> When an axon of cell A is near enough to excite a cell B and repeatedly takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

This is one piece of the puzzle.

Another piece is the architecture of lateral excitatory and inhibitory feedback that we met in connection with the Limulus.

A third and more recently discovered piece is "volume learning" made possible by a "diffusive agent", usually believed to be nitric oxide, NO.

A fourth piece would be the architecture of e.g. neocortex with its six layers and minicolumns. We will leave this fourth piece out of the puzzle for now though – it is difficult enough anyway!

Self-organization

Let us begin with Hebb's law and try to formalize it. It considers the outputs, or efferents, from the two neurons A and B, so those two entities must figure in the formal law. We call the A's efferent x and B's efferent y.

First we try the expression :

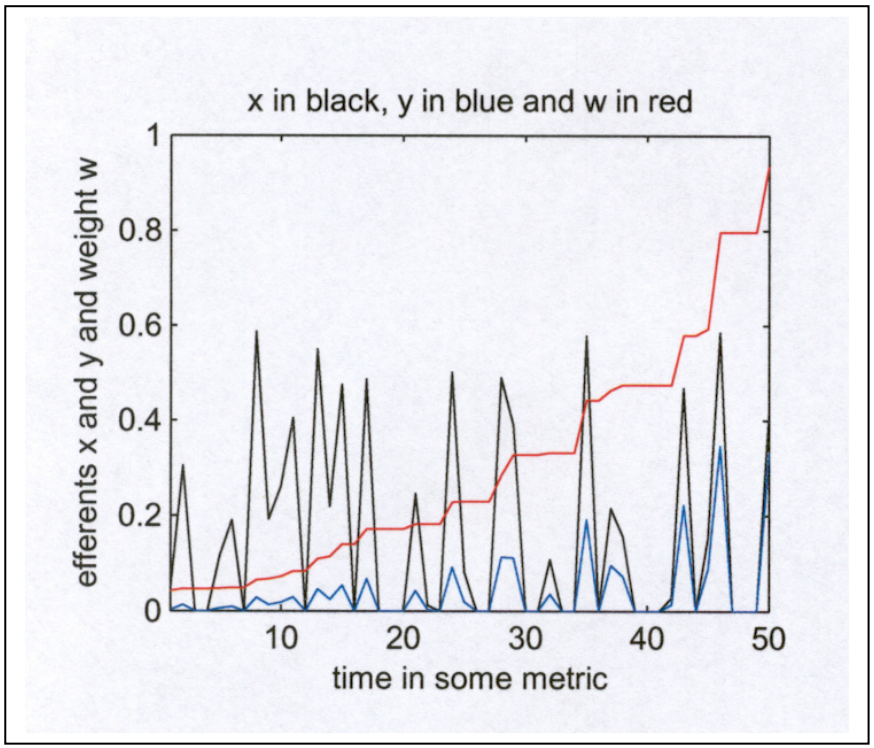
$$\Delta w = \alpha x y$$

for the change in the weight w between A and B.

If A is firing x is large and if B is firing y is large and Δw will be large. This is not in contradiction with Hebb's wording (note that it might be formalized in a number of ways, Hebb was not specific about the mathematics).

Self-organization

So what happens in a concrete case?



There is something fishy here – the weight w (red curve) just grows and grows!

Self-organization

It isn't that Hebb was wrong. He just produced a half-truth. In science that's quite an accomplishment, unlike in testimony in a court of law.

We must make it possible for the weight *w* to decrease also. "Forgetting" is necessary!

It is reasonable to amend Hebb's wording and say that when A is firing and doesn't come close to firing B, then the synapse connecting them isn't doing anything worthwhile and therefore its efficiency, or weight, should decrease. After all, a synapse should, just like a neuron, earn its keep.

We notice that *x* and *y* are both positive or zero. Their product must then in our first formalization always be positive or zero. Therefore *w* will always grow or at best take a temporary rest from growing.

If we were to subtract the means of *x* and *y* we would expect *w* to sometimes grow and sometimes decrease.

Self-organization

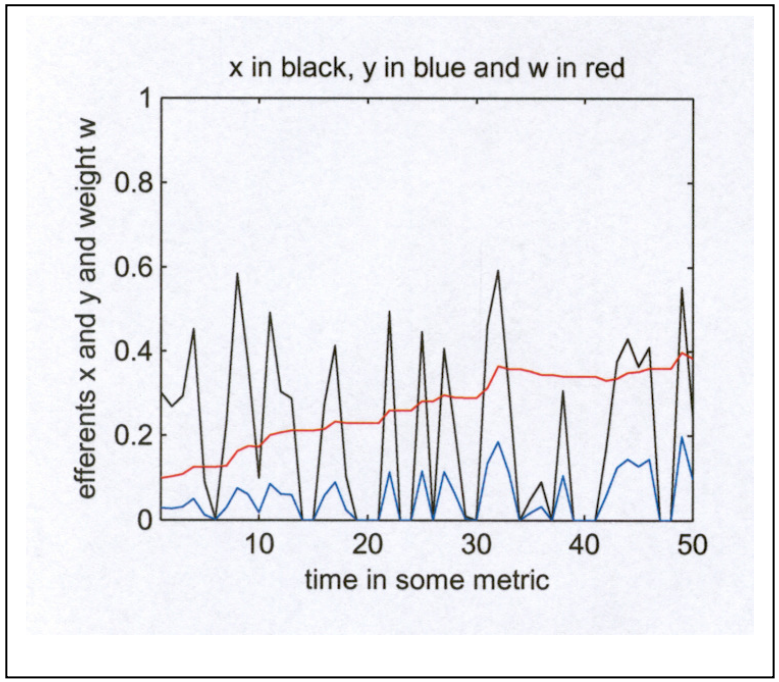
The term $\Delta w = \alpha xy$ in a formalized complete Hebbian learning law is universally accepted. There are, however, a number of different ways of introducing forgetting.

One common variant is the following:

$$\Delta w = \alpha_1 xy - \alpha_2 yw$$

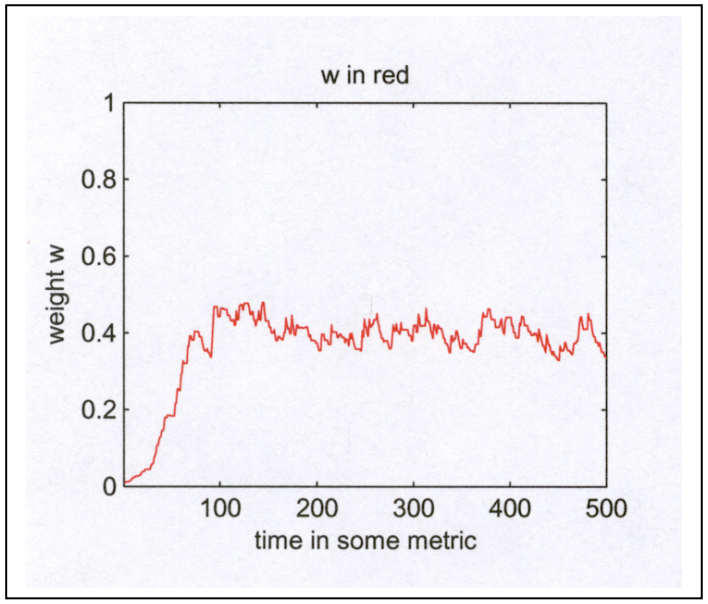
where α_1 is the learning rate and α_2 is the forgetting rate.

Let us see how it works:

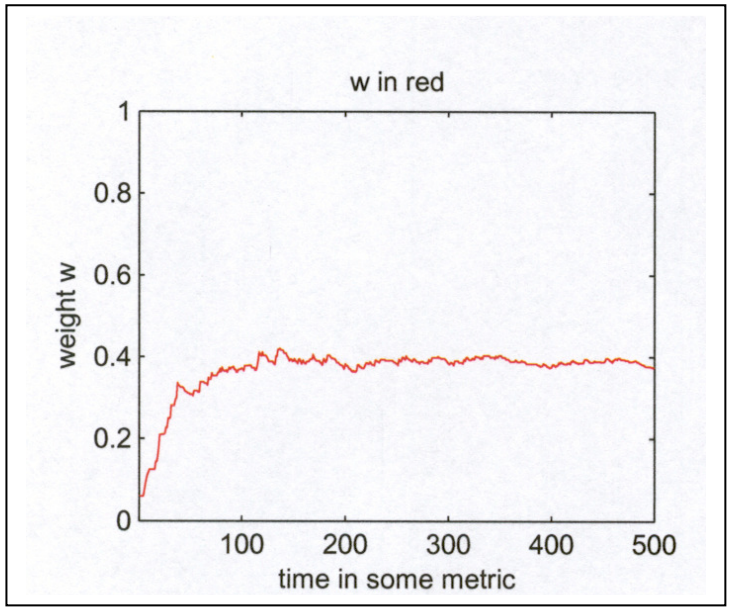


Self-organization

We now have intervals when the weight decreases and it will not grow beyond bounds. If we look at the weight over a longer time we see that it doesn't settle down nicely, however. By letting the learning rate α_1 and the forgetting rate α_2 is decrease over time we will achieve that too.



α_1 and α_2 constant over time



α_1 and α_2 decrease over time

Self-organization

We have shown that we can obtain a weight representing the synapse between neuron A and neuron B in a complete Hebb's learning law.

This is far from self-organizing though.

Any neural network, biological or model, has many neurons, i.e. many efferents y.

Each neuron has many synapses, i.e. many afferents x and weights w.

The many weights should develop over time to reflect structures in the afferents, e.g. the existence of phonemes and their individual characteristics.

Self-organization should develop so that the neural system will make sense of its surroundings!

Self-organization

This might be a suitable time to quote a provocative statement by a Finnish researcher, Heiki Hyötyniemi:

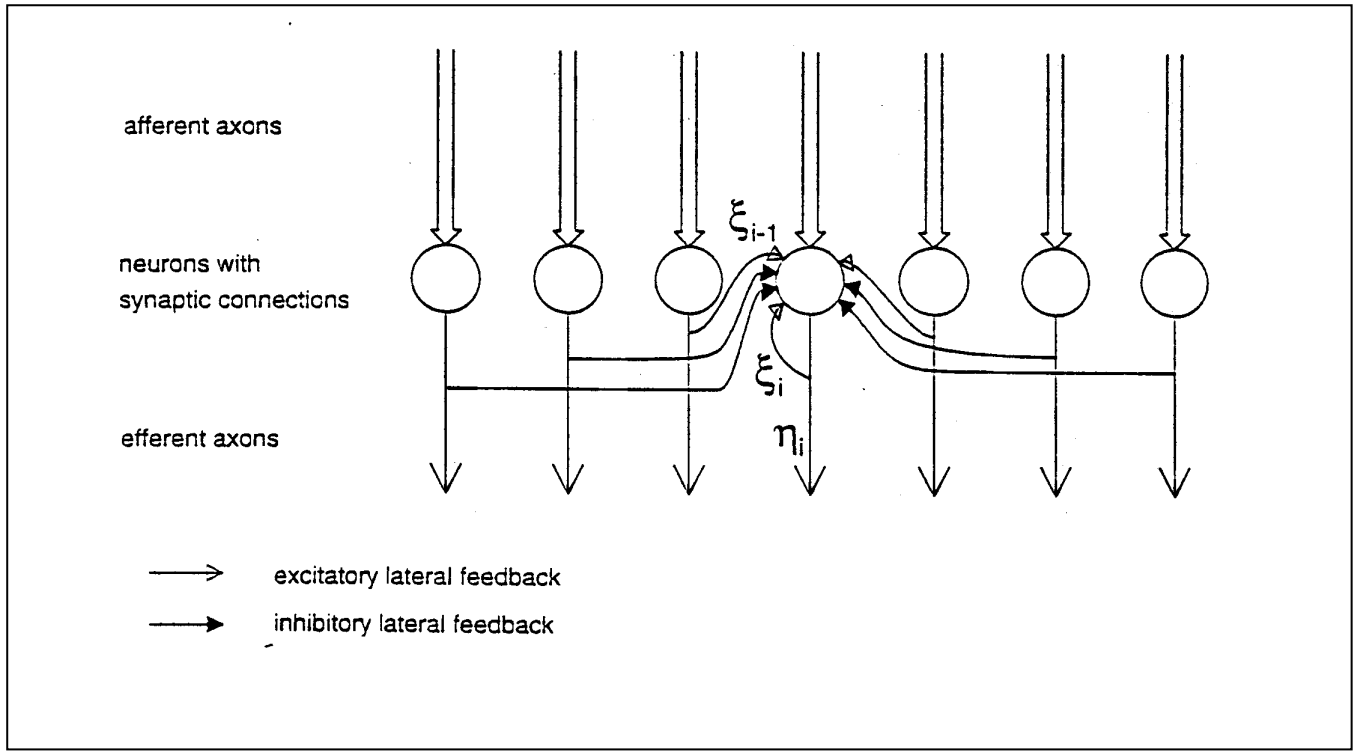
> When the mechanisms are selected in an appropriate way, *self-organization* automatically results in complex structures – to put it strongly, the system cannot help getting intelligent. There is no need of 'motivation', etc., for explaining the 'learning' of the structures.

Note of caution:

> There is a need for some meta-intelligence in the structure – what is worthwhile learning? Motivation will play an important role there. But Hyötyniemi's statement is nevertheless thoughtworthy.

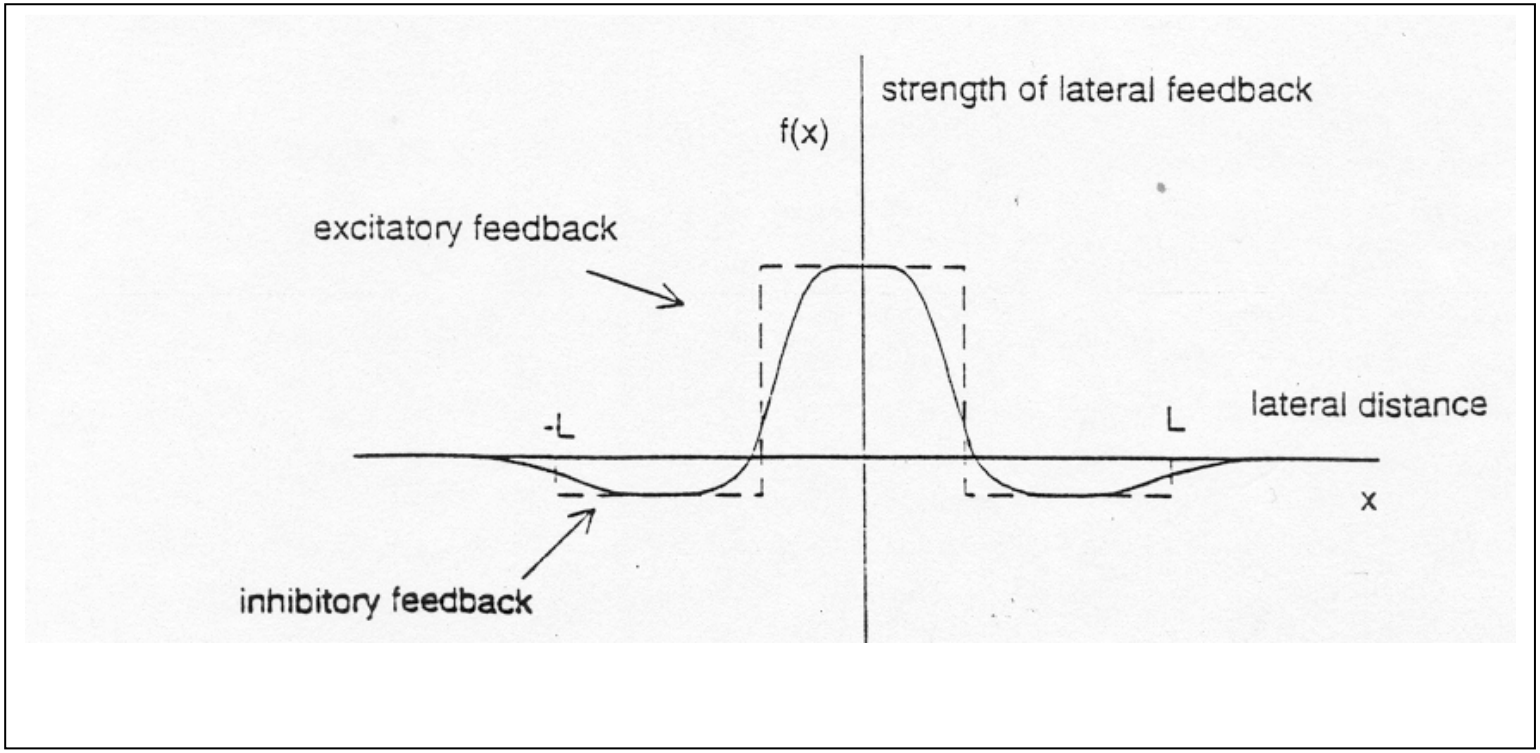
Self-organization
A network architecture

We remember from the Limulus chapter how there was inhibitory lateral feedback and that this architecture could be complemented with excitatory lateral feedback in a close neighbourhood. A linear array of neurons with the feedback connections to the neuron in the middle is shown below.



Self-organization
The Mexican Hat

The strengths of the feedback as a function of the distance between neurons is shown below. When we have a two-dimensional array of neurons this function is known as the "Mexican Hat", for obvious reasons.



Self-organization

How could it all work together?

Assume that we are in the beginning of the self-organization process. An input is presented to the neurons in the array, not to all of them but to a substantial number. Many neurons will fire but since their weights are not “attuned” to the input, the firing will be uneven and spread. Somewhere there is a neuron that fires most. That neuron will cause its close neighbours to fire more because of its excitatory feedback to those close neighbours. At the same time that neuron will cause its more distant neighbours to fire less because of its inhibitory feedback to those more distant neighbours. This effect will evidently have a “spiralling” effect and will cause a column of firing neurons to develop, surrounded by silent neurons.

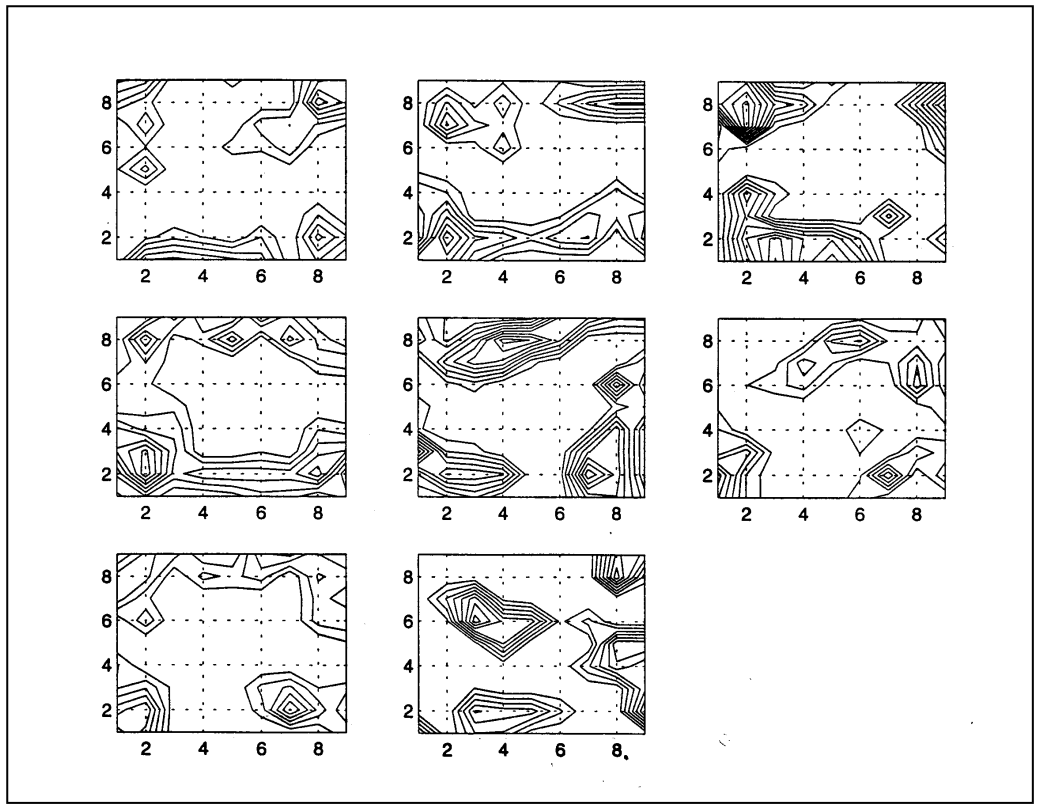
Now the Hebbian learning kicks in. The firing neurons in the column all have high outputs and will have their synapses strengthened which will make them more similar to the input. The next time that same input, or one similar, is presented the neurons in this column will be more adapted to the input than before and will hinder other neurons from firing more effectively.

Did the other neurons loose out? Yes, for that input they did, but they are fresh to adapt to other inputs. And so different groups of neurons, in columnar organization, tend to adapt to different inputs. We have specialists in detecting different inputs.

Self-organization

An example: before self-organization

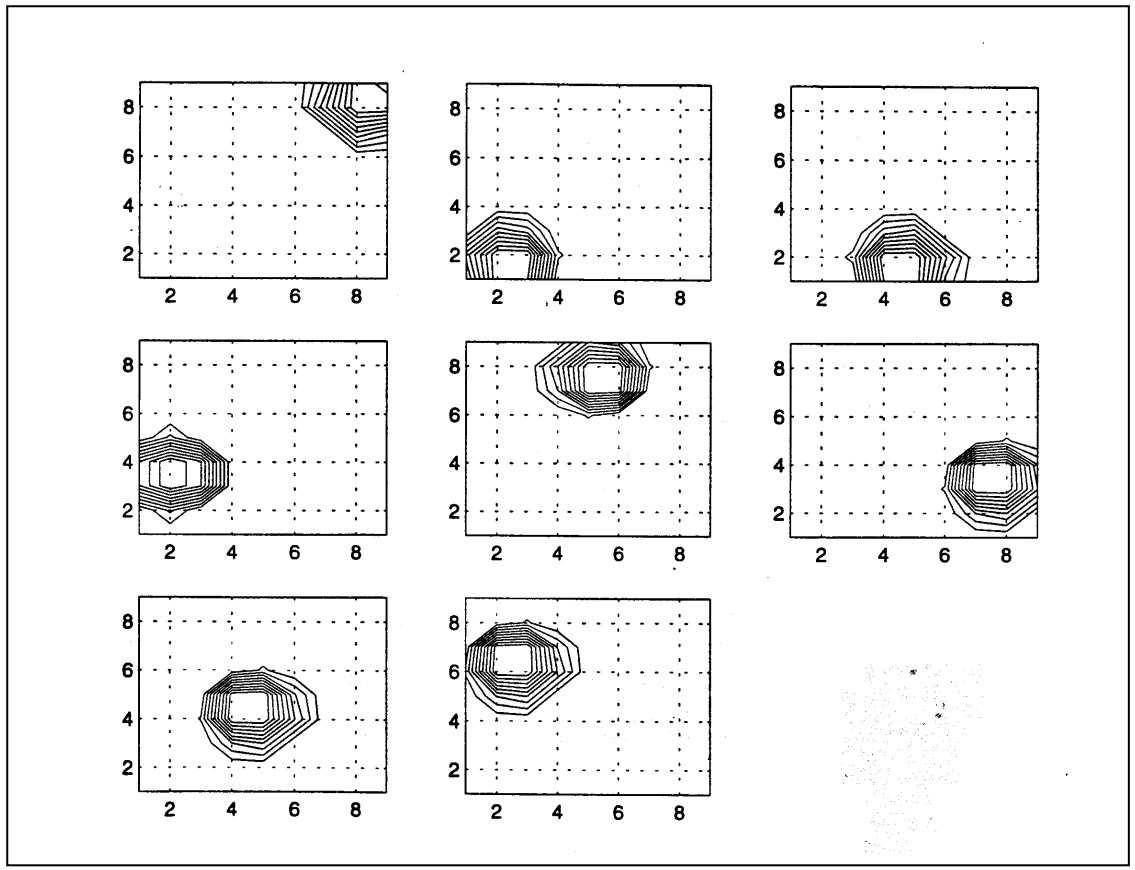
Below is the activity of 81 neurons in a square array. Eight very different inputs are presented to the network. The activity looks rather unorganised.



Self-organization

An example: after self-organization

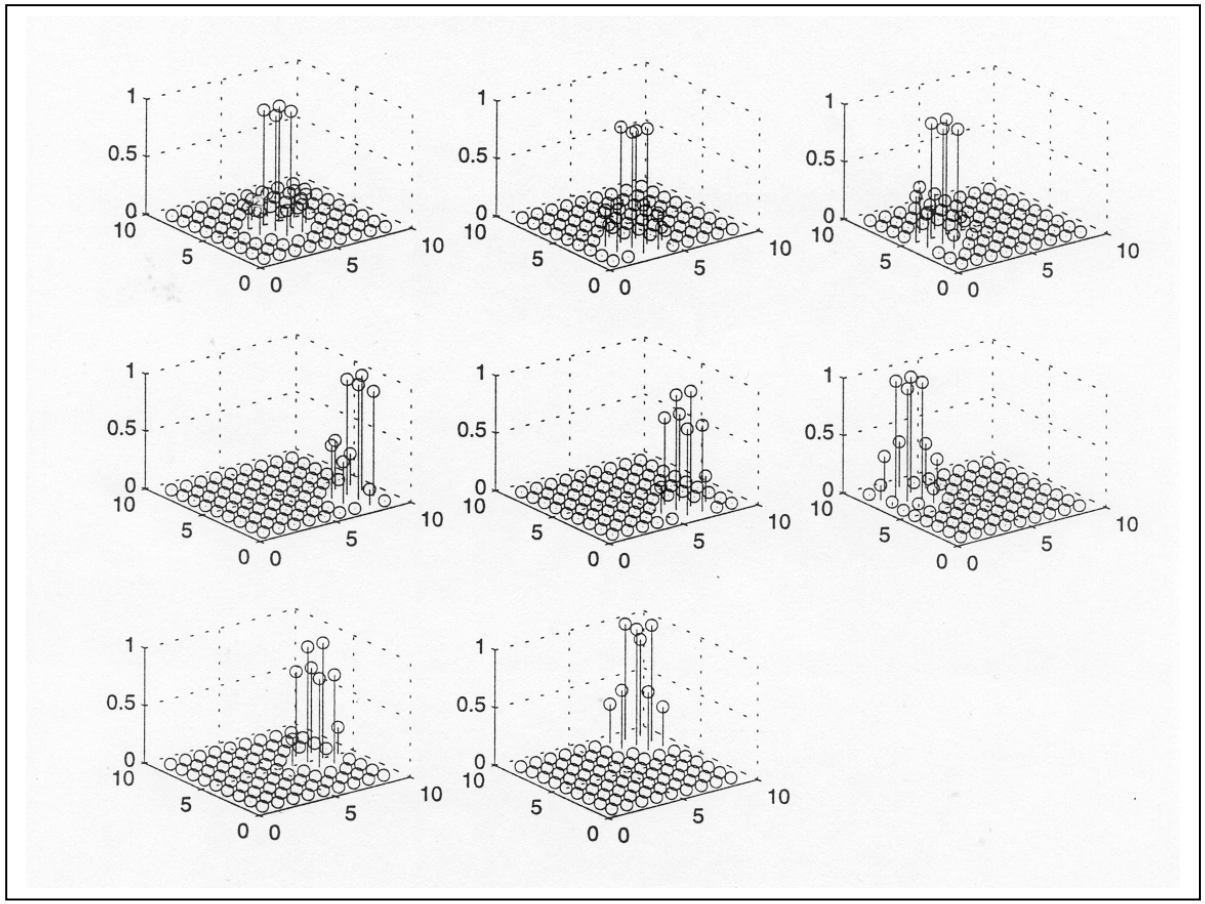
The eight inputs are the same as before. Now there is clearly a column for each of the inputs.



Self-organization

An example: after self-organization

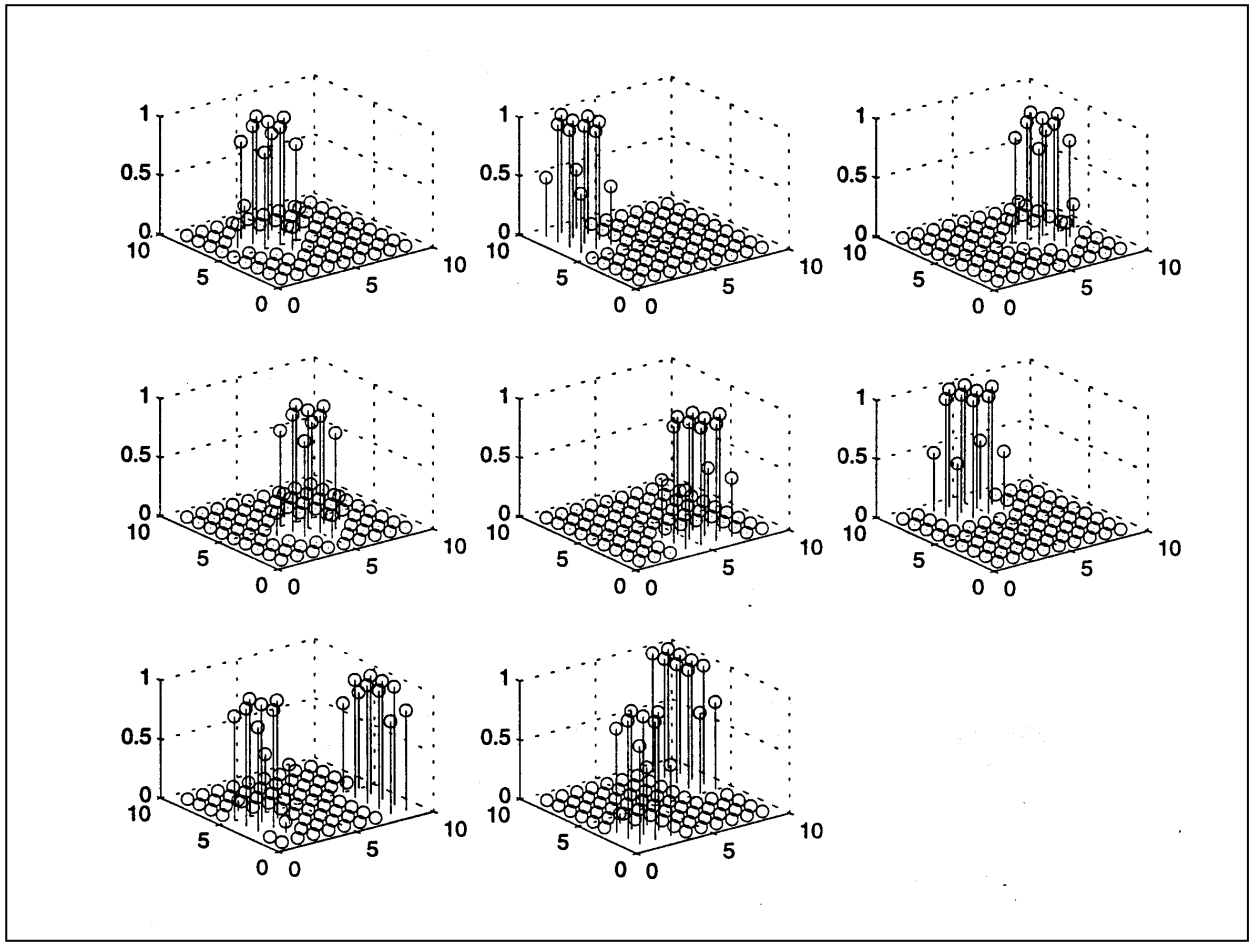
Below are the same columns, visualized in a different way.



Self-organization

An example: after self-organization

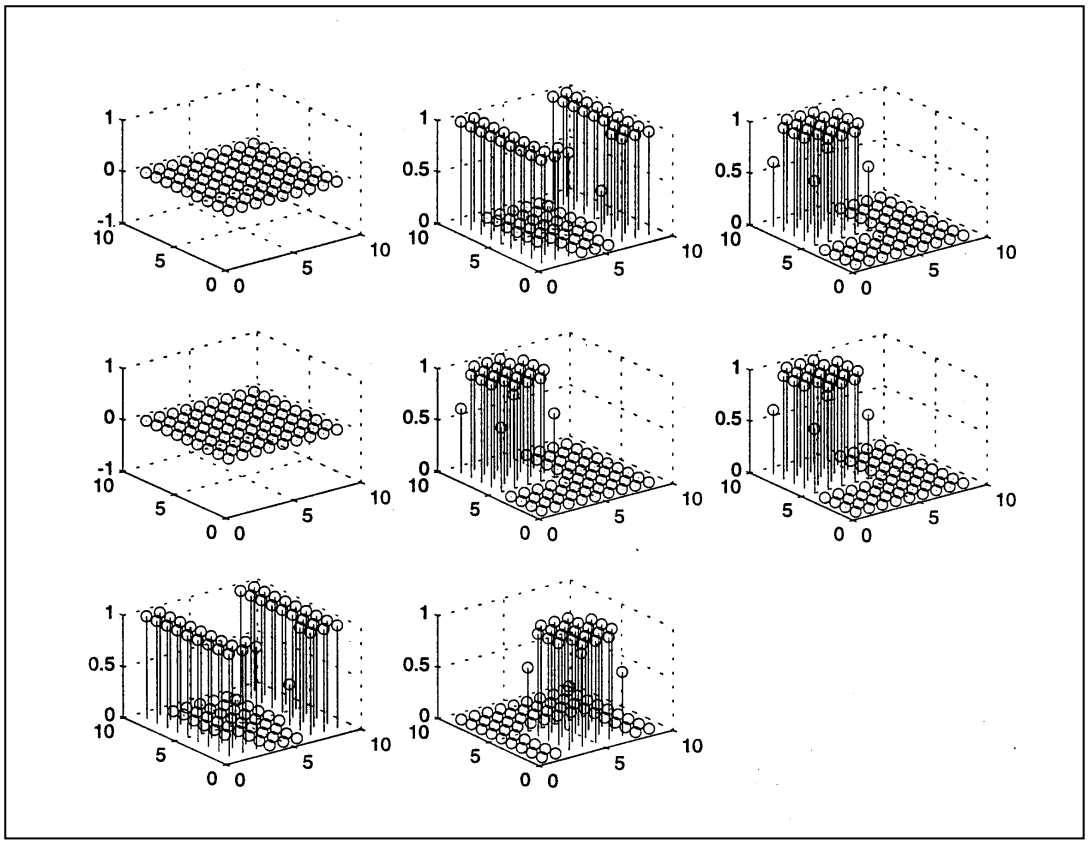
Here we have strengthened the excitatory feedback. The activity starts to be fragmentened.



Self-organization

An example: after self-organization

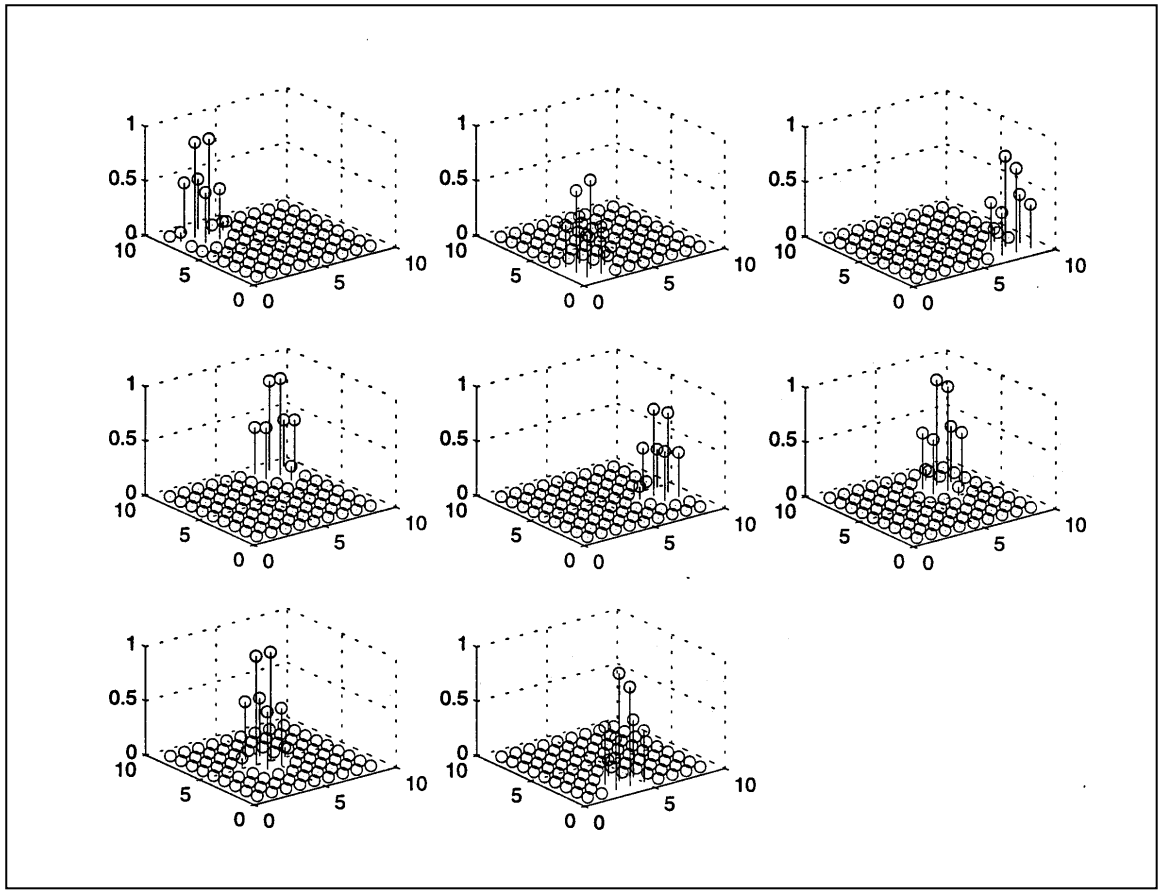
Here we have strengthened the excitatory feedback even more and self-organization no longer works adequately. The network is silent to some inputs and has the same output to several different inputs.



Self-organization

An example: after self-organization

Here we have strengthened the inhibitory feedback. The columns have become narrower. If we further strengthen the inhibitory feedback no columns will develop and there is no self-organization: the neural activity will remain unorganised.



Self-organization

With the right balance between excitatory and inhibitory lateral feedback we achieved a very appealing self-organization in this simple case.

Our case was simple in two ways.

First, the inputs were very different from each other. Technically speaking, they were orthogonal to each other.

Second, the inputs were few.

In cases where we have many different inputs we cannot hope to achieve self-organization with the two mechanisms we have given.

So we need more mechanisms – after all we are able to distinguish phonemes well and should therefore have nice phonotopic maps. More about that later. But first we will acquaint ourselves with one of the most well-known neural networks – Kohonen maps.

Self-organization – Kohonen maps

Fundamentals

During self-organization there develops a clear "winning" column of neurons when an input is presented to the neural network. This process is time-consuming, it consists of enhancing the response of the best adapted neurons and diminishing the response of less well adapted neurons, adapted to the particular input, that is, through the mechanisms of lateral feedback.

In a computer model of a neural network we have a quicker way of identifying the winner – we (or rather our computer) just compare all the outputs of the neuron and identify the largest. We don't need any lateral feedback to do that, and in the network we now study we therefore don't include lateral feedback.

After identifying the winner we change its weights to be even more attuned to the input. That means making it weight vector more like the input vector (their scalar product is the largest when they coincide).

But that's not all. We also change the weights in a suitable neighbourhood of the winning neuron in the same way. That way we gain a characteristic of topological ordering of the map.

Self-organization – Kohonen maps

Let us assume that the i:th neuron was identified as the winner for the input x. Then the weight vector for that neuron, w_i, will be changed by adding a term

$$\Delta w_i = \eta(x^T - w_i)$$

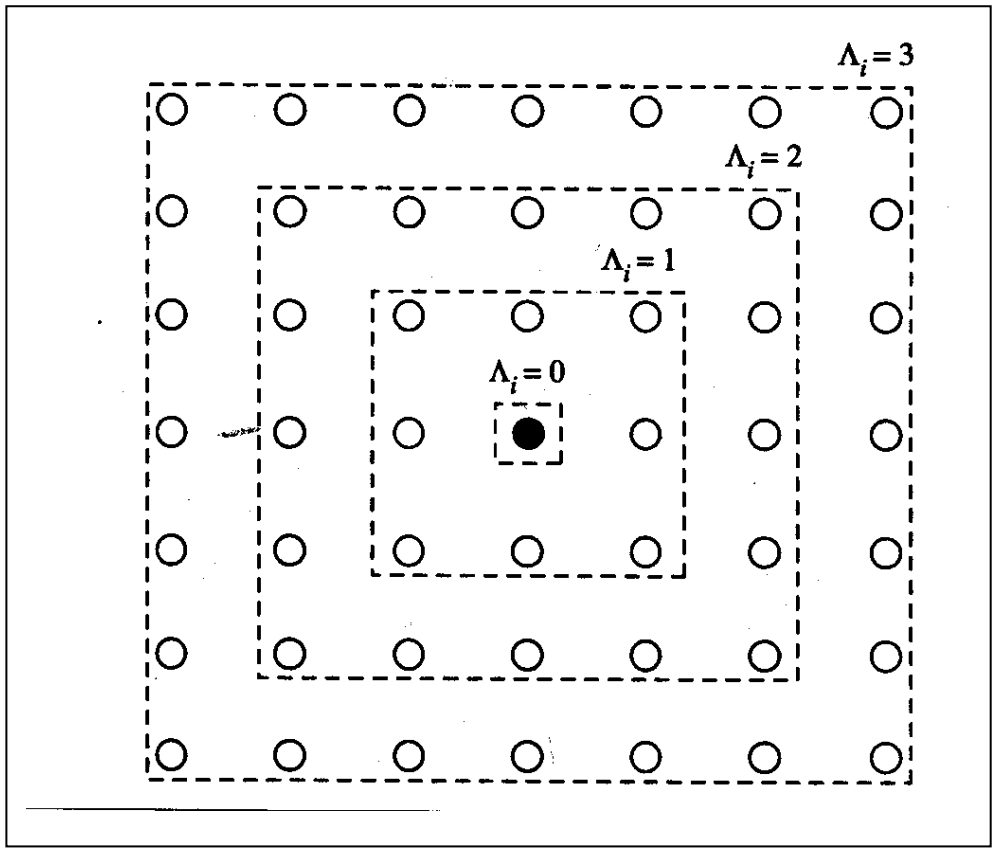
Clearly this takes away a little bit of the "old w_i" and replaces it with something more like x, i.e. x itself. The transpose doesn't change any of that – it's necessary for dimensionality reasons.

But the i:th neuron is not the only one to have its weights changed.

Self-organization – Kohonen maps

We have already stated that a number of neurons in a suitable neighbourhood of the i:th neuron should have their weights changed too. What is a suitable neighbourhood? Let us discuss the figure below. (From Haykin, p. 413)

The neighbourhoods are squares
In this case (that is a common
choice for shape of neighbourhood).
There are four squares, denoted
by $\Lambda_i = 0,\ldots, \Lambda_i = 3$.
The largest square, $\Lambda_i = 3$,
contains 48 neurons around
the winning neuron (the filled circle)
The next largest square contains
24 neurons around the winning neuron.
The next square contains 8 neurons
around the winning neuron.
The smallest square contains no
neuron, except for the winning neuron.



Self-organization – Kohonen maps

One could say that Kohonen improves upon nature here. Instead of the fixed reach of the lateral feedback which determine the width of the column of neurons we found earlier, he lets self-organization proceed in stages.

He first changes the weights for all neurons in the largest neighbourhood. He does that for all different inputs, repeatedly.

In the next stage he changes the weights for all neurons in the second largest neighbourhood and then continues until he changes only the weights of the winning neuron.

What does he gain from all this effort?

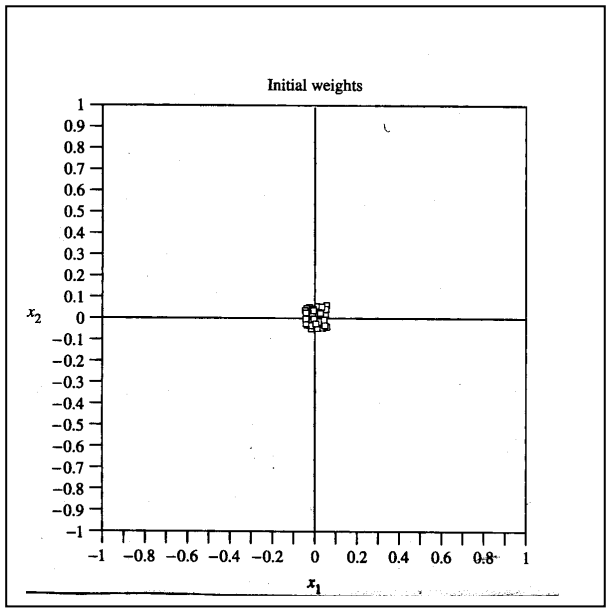
Inputs that are similar will have winners that are close – this is a topological ordering.

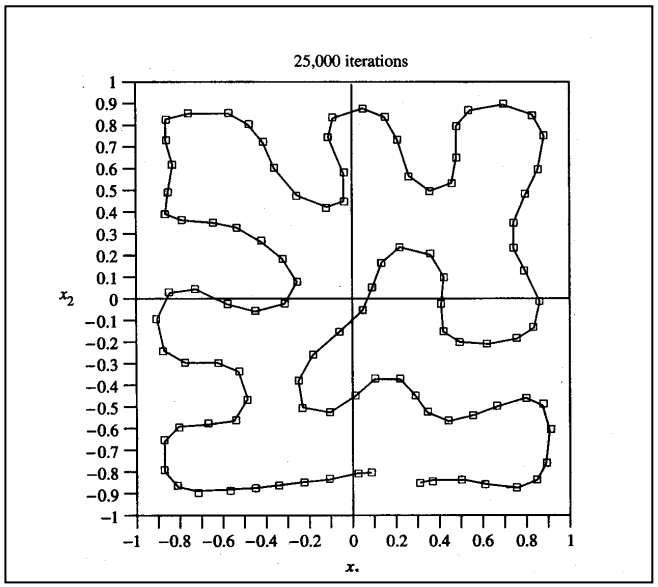
He will not have columns of neurons in the end but single neurons tuned for different inputs. And that's OK – the brain does have columns of neurons but Kohonen doesn't set out to model that.

Let us study some examples.

Kohonen maps – examples

Visualizing Kohonen maps can be done in two "regimes", low-dimensional (dimensions 1, 2 and 3) and high-dimensional. In the low-dimensional regime we may see both inputs and weights in the same plot. Remember that weights of a winning neuron should be close to "its" input. If two-dimensional inputs are many and evenly spread over a square as in the figure below, and a linear array of neurons with small initial weights goes through the process of self-organization we would expect the neurons to have their weights spread to match those of he inputs as well as possible. Here is what it might look like (from Haykin, p.420):

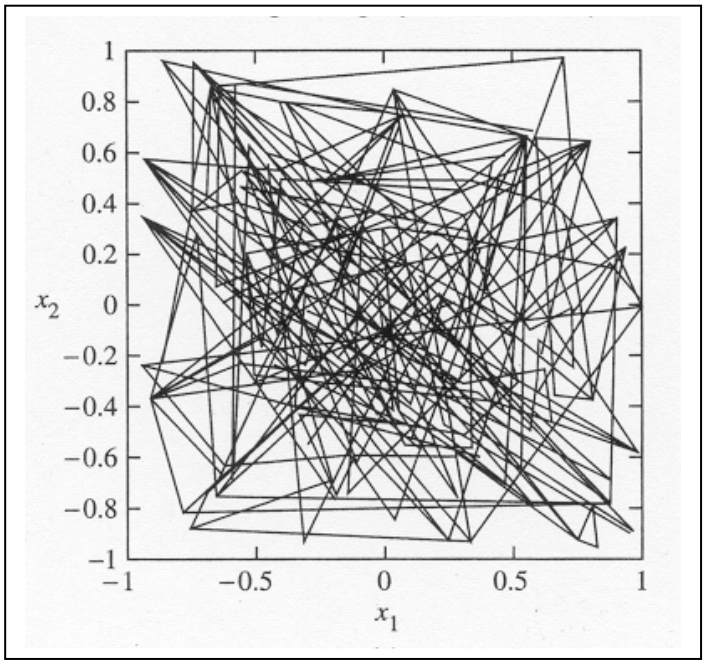




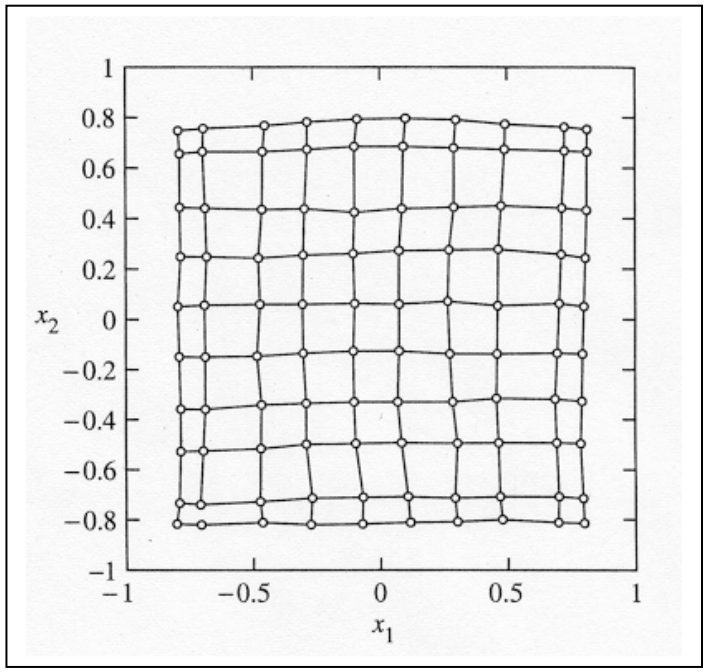
Kohonen maps – examples

The same two-dimensional data can be presented to a square array of neurons. By square we mean that the neurons are located in a square lattice rather than on a line as the example above. What we measure in the coordinate system is the weights of the neurons, not their position in the lattice. The position of the neuron is indicated by the lines connecting the neurons.
From Haykin, p. 414.

Before self-organization

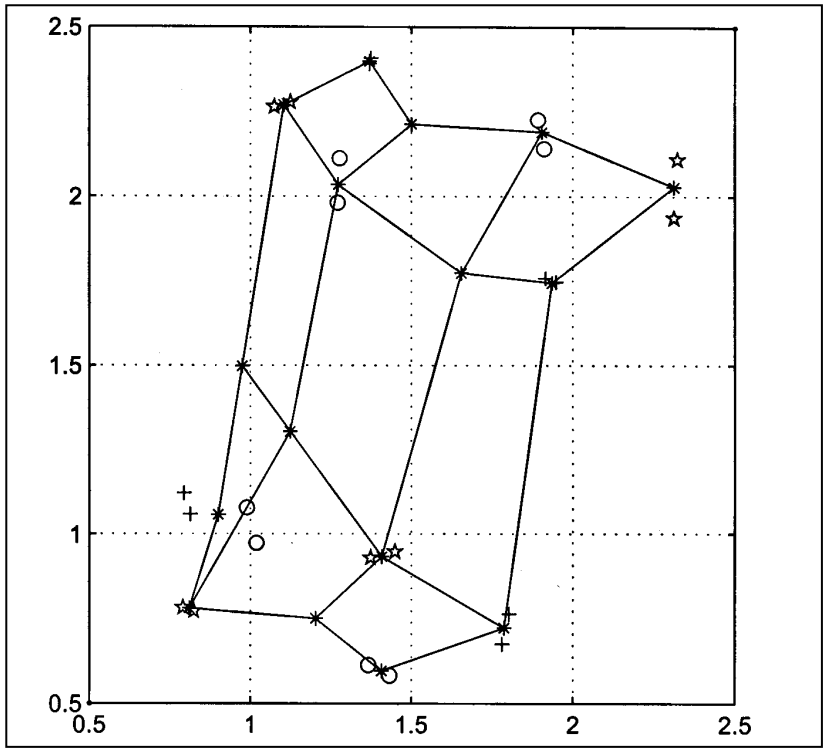


After self-organization



Kohonen maps – examples

Evenly spread data as in the two previous examples are not the most interesting case, maybe. In the example below there are twenty-four data points. These are not evenly spread though, but rather appear in pairs so there are twelve pairs. They have different symbols, o, + and stars. This is not important at the present. The weights of the 4x4 neurons are shown as *. After self-organization we have something like:



Kohonen maps – examples

What might we have expected in the example above? Since there are twelve pairs of data and sixteen neurons we might have expected that there would be one neuron “assigned to” each pair and four neurons might become what is known as “dead neurons”.

This is almost what happens. There are ten pairs of data which each are assigned one neuron. Two pairs of data which are close and look almost like one group of four data are assigned one neuron only. Five neurons become dead neurons.

Let us now go to higher dimensions. We can then not show the data or the weights of the neurons so we will have to show the map in a more abstract way. It might look more to the point and more concrete, but the maps are worthy of some thought.

Higher dimension Kohonen maps – examples

Let us see how a Kohonen map would develop if characteristics of a number of different animals are the inputs. The animals are the following:

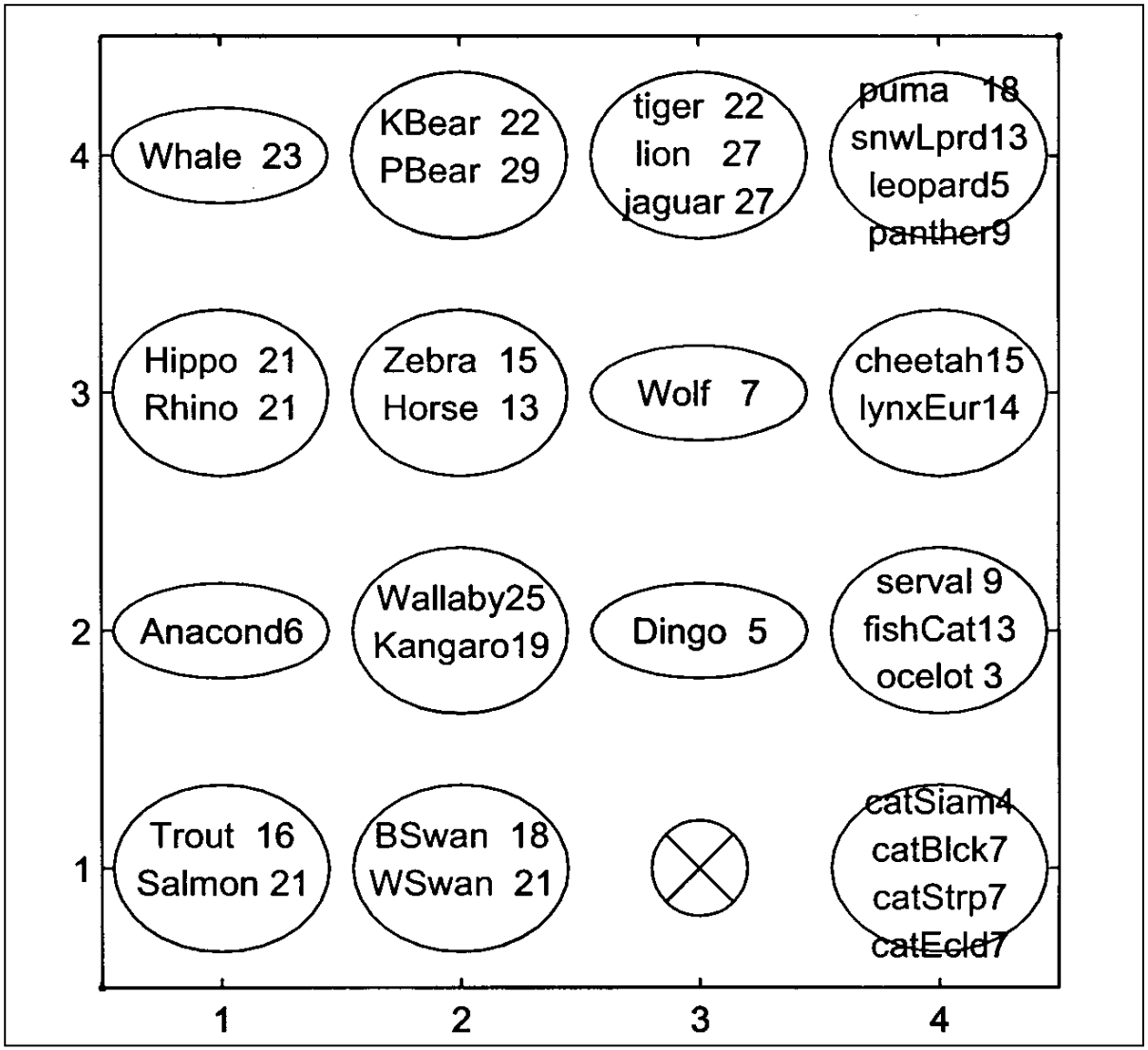
Group A: Przewalski's horse, Grevy's zebra, wolf, dingo, white swan, black swan, Atlantic salmon, rainbow trout, polar bear, Kodiak bear, white rhinoceros, hippopotamus, grey western kangaroo, swamp wallaby, anaconda, grey whale. A total of sixteen animals.

Group B: tiger, lion, jaguar, leopard, snow leopard, black panther, cougar, cheetah, ocelot, Eurasian lynx, serval, fishing cat, black domestic cat, even coloured domestic cat, striped domestic cat, Siamese domestic cat. A total of sixteen felines.

All animals were characterized by their size, whether the are carnivores, omnivores or herbivores, if they have four legs, two or no legs, wings or fins, their sociability (living in groups, pairs, or solitude) , coloration and head shape. Each animal is described by eighteen numbers, i.e. by a vector of dimension eighteen.

There are sixteen neurons in the map, some compromises must be made. Let us see what self-organization achieves.

Higher dimension Kohonen maps – examples



Higher dimension Kohonen maps – examples

What characteristics do we find in this map?

We see the animals listed in ovals. Each oval represents a neuron. There is also a crossed-over neuron, this is a dead neuron.

Each animal is listed at the neuron which has the weight vector which most closely resemble the characteristic vector of the animal.

After each animal wee see a number. This number represents the distance between the neuron weight vector and the animal characteristic vector. A small number is a closer match than a large number.

Higher dimension Kohonen maps – examples

First we see that the really “odd” animals, the whale and the anaconda, each has been assigned one neuron. That seems sensible. If an anaconda is the input we want the map to report this and we don’t want the neuron in effect signalling “an anaconda or …(something similar) is the input”.

We also see a number of pairs of animals who share one neuron, like the white swan and the black swan. This is a good compromise for a map that doesn’t have the capacity to assign one neuron to each animal input.

We further see that the four variants of domestic cats share one neuron. That seems like a good “choice” too, if we want to use our scarce neurons in an economic way.

The tree biggest cats share one neuron. The match is not at all as close as for the domestic cats ut that is to be expected – a black domestic cat and a striped domestic cat have more characteristics in common than a tiger and a lion.

That the wolf and the dingo each has one neuron is an oddity of a kind that is not uncommon.

Higher dimension Kohonen maps – examples

But there is order in the map too!

The felines are represented along the right and part of the upper edge, i.e. they are close to each other.

The felines are represented in size order.

The other carnivorous mammals are represented by neurons close to the cats' neurons.

The bigger animals are in the upper part of the map and the smaller animals are in the lower part of the map.

In the self-organization some characteristics have been found to be of such importance that they are reflected in the ordering of the map. But others have not. So we may ask what is not represented in the map.

The social preferences e.g. Lions live in packs, tigers are solitary and jaguars often live in pairs, if I am correctly informed. But they still share one neuron.