Topology learning solved by extended objects: a neural network model

Csaba Szepesvári[1], **László Balázs**[1] **and András Lőrincz**
Department of Photophysics, Institute of Isotopes of the Hungarian Academy of Sciences
Budapest, P.O. Box 77, Hungary, H-1525

Abstract

It is shown that local, extended objects of a metrical topological space shape the receptive fields of competitive neurons to local filters. Self-organized topology learning is then solved with the help of Hebbian learning together with extended objects that provide unique information about neighborhood relations. A topographical map is deduced and is used to speed up further adaptation in a changing environment with the help of Kohonen type learning that teaches the neighbors of winning neurons as well.

Introduction

Self-organized learning is a most attractive feature of certain artificial neural network paradigms (Grossberg, 1976, Kohonen, 1984, Carpenter and Grossberg, 1987, Földiák 1990). It is considered as a means of solving problems in an unknown environment. The architecture of the neural network, however, is in general 'inherited', in other words is prewired and may severely limit the adaptation possibilities of the net. An example is the Kohonen-type topographical map that has a built-in neighborhood relation. Various attempts have been made to resolve this problem such as the self-building model of Fritzke (Fritzke, 1991) and the neural gas network of Martinetz and Schulten (Martinetz and Schulten, 1991). The closest to the present work is the neural gas network. It is based on the idea that topology can be learnt on the basis of joint similarity. It determines adjacent neurons based on the distance measured in the metric of the input vector space. Consider, however, the example of a maze embedded in a two-dimensional Euclidean space. The two sides of one of the walls in the maze have very close input vectors in the metric of the Euclidean space though they may be very far from each other in the metric space brought about by the topology of the maze. The question then arises if and how and under what conditions a given network is capable of determining the topology the external world.

The model we present here relies on the extended nature of objects in the external world. This method can take advantage of the same idea of joint similarity and provides a

[1]Permanent address: János Bolyai Institute of Mathematics, University of Szeged, Szeged, Hungary, H-6720. E-mails: szepes@inf.jate.u-szeged.hu, lorincz@obelix.iki.kfki.hu.

simple route for exploring the neighborhood relations as the objects themselves bring the information. The receptive fields of the neurons are local filters and the neural network may be considered as a dimensional reducing system that provides position information and neglects object details.

In order to take advantage of neighboring relations and the possibility of Kohonen-type neighbor training a distance function may be established with the help of Hebbian learning: extended objects may overlap with receptive fields of different neurons and may excite more than one neuron at the same time. These neurons then assume activities different from zero, and Hebbian learning may be used to set up connectivity that matches the topology. The strength of the connection may be related to the distance in further processing. The greater the strength, the smaller the distance. In this way a topographical map is established and Kohonen-type training becomes a feasible means of speeding up further adaptation in a changing environment.

Dimensionality reduction with spatial filters

First, we define local, extended objects. Let us assume that the external world is a metrical topological space equipped with a measure and is embedded in a bounded region of Euclidean space. Let us then consider a mapping from the subsets of the bounded region of the Euclidean space into a finite dimensional vector space. This mapping could, for example, be defined by choosing two vectors of the subsets randomly. Another type of mapping may, for example, spatially digitize the external world, and form a digital image. Hereinafter we shall use this mapping and call the elements of the digitized image as pixels. A vector of the vector space will be considered a local, extended object if *(i)* there exists a connected open set of the metrical topological space that after mapping is identical with the said vector, *(ii)* if the measure of that open set is not zero, and *(iii)* if the open set's convex hull taken in the vector space is in the topological space.

Let us further assume that our inputs are local, extended objects and our task is to provide the approximate position of the corresponding real object with no reference to its form. In order to be more concrete, let us take the example of a three dimensional object mapped onto two two-dimensional retinas, i.e. to a many dimensional vector space. The vector, that describes the digitized image on the retinas is the extended object. The task is to determine the position of the original, real object, with no reference to its form and with no a priori knowledge of the dimensionality, nor even of the topology of the external world. This problem will not be considered here, however these tools are general enough to solve it. In the following we restrict our investigations to the case of a single retina.

We may say, for example, that an object is 'in the middle' or 'in the upper left corner' or that it is 'down and right'. This task may be considered as a dimensionality reduction problem since if the image is given in the form of $n \times n$ pixels, having continuous grey-scale intensities, then one maps an $n \times n$ input matrix having elements on the $[0, 1]$ real interval into the world of m expressions that denote the possible different positions.

Let us assume that the spatial filters that correspond to our position expressions already exist and let us list the expressions and organize the filters in a way, that the i^{th} filter corresponds to the i^{th} expression. For example, in the case of a two-dimensional image the expression 'middle' would correspond to a spatial filter that transforms the image by causing no change in pixel intensities around the middle of the image but the farther the pixels are from the center the more the filter decreases the pixel intensities. As a demonstration Fig. 1 shows spatial filters of a nine-expression set. These are Gaussian filters but other filters could serve just as well. The filters are digitized by replacing the center value of every pixel by the closest digitized grey scale value. Let $\mathbf{G}^{(i)}$ denote the i^{th} digitized Gaussian spatial filter, and let $\mathbf{S}$ denote an input (image) vector. Let $g_{pq}^{(i)}$, $s_{pq}(1 \leq p, q \leq n)$ denote the values in pixels (p, q) of the digitized i^{th} Gauss filter and the input vector, respectively.

Now, the searched position estimation may be given in the following fashion: First, let the input vector pass all of the digitized Gauss filters. Let us denote the output of the i^{th} filter by G_i — examples will be given later — and denote the mapping by d:

$$G_i = d(\mathbf{G^{(i)}}, \mathbf{S}), (\mathbf{i = 1, 2, ..., m}) \tag{1}$$

The mapping d is to be engineered in such a way that it can be considered as a 'distance' function $\mathbf{R}^{n \times n} \longrightarrow \mathbf{R}^{+} \cup \{0\}$ providing distance-like quantities between the pattern inputting the network and the Gaussian filters. With such a d function one might choose the smallest G_i value. If that has index j, then we say: the position of the object is the j^{th} expression. There are various possibilities for function d; here we list three of them:

- Conventional filtering is defined by multiplying the input values by the filter values and then integrating over the input space. In order to fulfil our requirements for the 'distance' function d, let us define it in the following fashion:

$$d_1(\mathbf{X}, \mathbf{Y}) = 1 - (1/n^2) \sum_{i,j=1}^{n} x_{ij} y_{ij} \tag{2}$$

  from here onwards it is assumed that $0 \leq x_{ij}, y_{ij} \leq 1$. This 'distance' function has the form $1 - (1/n^2)\mathbf{X} \cdot \mathbf{Y}$ where $\mathbf{X} \cdot \mathbf{Y}$ is the inner product or spherical distance. Since this 'distance' definition is normalized, we might define an 'input-to-filter-similarity-measure', or measure of similarity, S in short, as $S = 1 - d$, where d is the 'distance'. The smaller the distance, the larger the similarity between input and filter vectors.

- One might try to use the usual Euclidean distance in $\mathbf{R}^{n \times n}$, that is

$$d_2(\mathbf{X}, \mathbf{Y}) = (1/n^2) \sqrt{\sum_{i,j=1}^{n} (x_{ij} - y_{ij})^2} \tag{3}$$

- Another form that is not a metric but may be used here is

$$d_3(\mathbf{X}, \mathbf{Y}) = (1/n^2) \sqrt{\sum_{i,j=1}^{n} x_{ij} (x_{ij} - y_{ij})^2} \tag{4}$$

Forming spatial filters by competitive learning

Special competitive networks may form equiprobabilistic digitization of the input space according to the density distribution of input vectors (Desieno 1988). Networks of pure competitivity are known to solve the problem of equiprobabilistic digitization for uniform distributions (Kohonen 1984).

Let us define a competitive neural network of m neurons. The external world provides inputs to input nodes. Every input node is connected to every neuron of the network (see Fig.2). Every neuron stores a vector of the input space. The stored vector of the i^{th} neuron is denoted by $\mathbf{w}_i$. Training modifies the stored vectors. The procedure of modification is as follows:

- An input is presented to the network in accordance with the density distribution. Input nodes forward inputs to the neurons.
- Neurons process their inputs and develop activities in accordance with the equation
$$D_i = d(\mathbf{x}, \mathbf{w}_i) = 1 - S_i \tag{5}$$
where $\mathbf{x}$ is the forwarded input vector, d is a 'distance' function and S_i is the measure of similarity.
- Competition starts. The winner of the competition is the neuron whose stored vector is the 'closest' to the input vector, i.e. the neuron having the smallest 'distance' (or largest similarity).
- The stored vector of the winning neuron i is then modified with the help of the update rule:
$$\Delta \mathbf{w}^{(i)} = \alpha(\mathbf{x} - \mathbf{w}^{(i)}) \tag{6}$$
where α is the so called learning rate; $0 < \alpha \leq 1$. Here we apply a constant learning rate during the whole training. It was found that the time dependent learning rate did not improve training results. Time-independent learning rate has the advantage that it keeps adaptivity.

In the numerical simulations, we presented two-dimensional objects of a two-dimensional space to the network. The input space and one of the input vectors that was presented are illustrated in Fig.3. Input vectors were derived by computing the overlap of the local, extended, randomly positioned objects and the pixels of digitization. Two different objects were used in these runs, an X shaped object (shown in Fig.3) and an 0 like object (not shown). In the first set of runs a single object was presented to the network at random positions. In other runs two or three objects were presented simultaneously to the network at random positions.

The training procedure resulted in spatial filters for 'distance' functions d_1, d_2, and d_3 defined in the previous section. We tried single objects for 'distance' functions d_2 and d_3. For spherical distance d_1 up to three objects were presented simultaneously.

First, we tried the Euclidean distance function d_2. Judging from our experience the network was able to learn only if the stored vectors were set close to zero prior to training. The noise resistance of the network equipped with the Euclidean distance was rather small. The heuristic reasoning for this finding is given in the Appendix.

The term
$$\sum_{(k,l)\in I} (w_{kl}^{(i)})^2$$
of Eq. (12) in the Appendix — the indices correspond to the digitization of the two dimensional space and I denotes the set of zero elements of input vector $\mathbf{x}$ of a given extended object — is responsible for the poor performance of the Euclidean distance

function. This term manifests itself in large distance values for inputs with a fair amount of noise. In this way one single neuron that has small $w_{kl}^{(i)}$ components can always win the competition. Training — as the analysis in the Appendix shows — tends to lead to this attractor.

The simplest way of eliminating that term is to modify the Euclidean distance function to d_3 of (4). 'Distance' function d_3 has a strong resemblance to the spherical distance functions d_1. Both of these functions solve the problem. There are other possible solutions to this problem, such as trying to decrease the mean value of the initial noise, or the learning rate, or start the learning rate from 1 and changing it in an appropriate fashion (Desieno, 1988). Analysis shows, however, that the spherical distance works better under more demanding conditions.

Single-object training results are shown for the spherical distance function in Fig.4. The results we present from now on were produced with this distance function.

One of the results of the competition is that if one increases the size of the local, extended objects one or more neurons may lose their receptive fields; in other words may have near zero stored vectors. Neurons lose their receptive fields by first approaching a corner of the two dimensional region. The number of neurons having non-zero receptive fields depends on the ratio of the bounded Euclidean region and the average area of the local, extended objects. The bounded Euclidean region is shared by the neurons: it is divided into nearly non-overlapping regions that correspond to the average object size.

In another set of training runs when two or three (more than one) randomly positioned objects were simultaneously presented to the network the results were very similar: filters were formed just like before, however, the rest of the filter vectors of the neurons were noisy. In other words the filters (the receptive fields) were surrounded by a low noise homogeneous background showing that winning neurons learnt of the presence of other objects as well and represented those as a random background. This is an attractive property of the algorithm; our strong competition forces the neurons to learn the most important correlations that being the locality of single objects and thus the neurons can neglect the correlations between two or more randomly positioned objects. To improve the winning chances neurons develop a random like background if more than one object is inputted to the network simultaneously. The background was considerably larger for the three-object case than for the two-object case. There was no noise for the single-object case. The two- and three-object filters are shown in Fig.5. In the following only the single-object case will be studied.

It is worth noting that in the general case some neurons may be sentenced to have very small — but nonzero — receptive fields. As the receptive field of these neurons never becomes exactly zero one may hope that these neurons are only 'sleeping' or 'not needed at present' or 'of small role' but not dead neurons. As it is shown in the paper these 'small role neurons' may recover and assume an equal role if adaptivity is kept and the external world changes.

Topology by Hebbian learning

It is a relatively easy task to build up the internal representation of the topology of the external world when the spatial filters are given. Let us introduce connections between neurons. These connections can represent the topology of the given metrical topogical space in the following fashion: The closer the stored vectors of two neurons are in the metric of the topological space, the stronger should be the connection between the two neurons and vice versa: if the connecting weight between two neurons is larger than zero then the vectors of the two neurons should be close in the metric of the topological space.

To form these connections one needs to note that a local, extended object may overlap with two spatial filters and may excite two neurons simultaneously. This means that the closer two spatial filters are, the more often the neurons representing them fire simultaneously. In our notation it means that their D_i values are small. One may use this fact to develop connections between the neurons. Let us set the strength of the connections to zero at the beginning and use the following Hebbian update rule in a parallel fashion during the whole training procedure with training rule (6):

$$\Delta q_{ij} = \beta \, (S_i^{(N)} \, S_j^{(N)} - q_{ij}) \tag{7}$$

where $S_i^{(N)} = 1 - D_i^{(N)}$ is the measure of similarity for the i^{th} filter for a given input, q_{ij} denotes the strength of the connections, β is the learning rate, and N denotes that both the S_i and the D_i values are normalized to the $[0, 1]$ interval. Connection strength q_{ij} is constrained to interval $[0, 1]$. The best results were achieved when only the winning neuron could update its connections:

$$\Delta q_{ij} = \beta \, (y_i + y_j) \, (S_i^{(N)} \, S_j^{(N)} - q_{ij}) \tag{8}$$

where y_i is the output of the ith neuron after competition: the output is 1 for the winning neuron and 0 for the others. In this way $y_i + y_j$ is not zero if and only if either the ith or the jth neuron was winning. Connection strengths are shown in the left hand side of Fig.6. Connection strengths are depicted by the thicknesses of the connecting lines between neurons. The position and the size of the circles represent the position and the size of the spatial filters, respectively. A non-connected topology was also produced by showing local, extended objects along three horizontal strips only (see the right hand side of Fig.6). Figure 6 shows well developed connections between neighboring filters in both the one dimensional and the two dimensional topologies. It is worth noting that connections between neurons that are farther, i.e. connections that would represent medium-range topology properties, did not develop in this model. In the present training examples filters are formed according to the object size and thus the object may excite only neighboring neurons. It is reasonable to expect, however, that if they have a distribution of object or feature sizes, filters will develop according to the average size. The larger-than-average object would develop connections between non-neighboring neurons as well if topology allows it.

The neural gas model of Martinetz & Schulten (1991) could not build up the correct topology for this case as it is not based on the neighborhood relations of topological space provided by our local, extended objects, but is based on a closeness relation in the metric of Euclidean space into which the topology is embedded. As an example assume, that the input is such that the closest neuron has the top-left receptive field. Second closest neuron is then either the middle-left or the top-middle neuron or both according to the exact position of the object. That is the neural gas model would develop connections between the top-left and the middle-left neurons too and the connection structure would become two a dimensional grid.

The neural gas algorithm in its present form is capable of representing the topology of only those worlds in which the closeness relation belonging to the topology and the closeness relation belonging to the Euclidean distance are identical. Slight modifications

— modification of inputs and modification of distance function — can make the neural gas model work for all cases too.

It has been shown for the case of single-object training that a lateral weight q_{ij} is non-zero if and only if the presented local object series has an infinite subseries in a way that the objects of this subseries overlap with the outer inverse image of both the $\mathbf{w}_i$ and the $\mathbf{w}_j$ vectors (Szepesvári, 1992). Since the presented objects are local objects one may conclude that the sets represented by the $\mathbf{w}_i$ and $\mathbf{w}_j$ vectors are locally connected, i.e. the non-zero lateral weights represent the topology. The necessary and sufficient condition of the proof is that both the digitization of the topological space and the distance function of the neural network should satisfy a separability condition (Szepesvári, 1992). The separability condition generalizes the view (naive in mathematical terms) that filter response should be zero if and only if the filter does not overlap with the input. This is the very point where the Euclidean distance fails.

Figure 7 shows the connection strengths as a function of 'distance' d_3 of the spatial filters.

Topographical map and Kohonen training

The Hebbian connections allow us to utilize the Kohonen type neighbor training, i.e. to introduce a cooperative learning scheme and to speed up the adaptivity of the network in a changing environment. The closeness or connection strengths of the neurons in the Kohonen map are predefined. Here we develop connection strengths in a dynamic fashion and that leads to a new problem when introducing Kohonen type neighbor training: Cooperative training may win over filter forming competition. The original Kohonen neighbor training may be written as

$$\Delta \mathbf{w}^{(i)} = \begin{cases} \alpha\, H(q_{ik})(\mathbf{x} - \mathbf{w}^{(i)}), & \text{if } i \neq k; \\ \alpha\, (\mathbf{x} - \mathbf{w}^{(i)}), & \text{if } i = k. \end{cases} \tag{9}$$

where index k denotes the winning neuron, and the connection strengths q_{ik} may be considered as predefined time dependent functions. Their initial values determine an inherited closeness between neurons. The said closeness is a slowly decreasing function of time whereas function H is a strictly decreasing monotonic function of distance; in other words, it is a that is a strictly increasing monotonic function of connection strength q_{ik} with: $H(0) = 0$ and $H(1) = 1$.

If one tries to establish an adaptive cooperative neighbor training then first the inherited rule of closeness should be replaced by rule (7) in order ot determine the closeness relations. However, this simple replacement and the usual function H lead to the loss of competition between neurons: the receptive fields of different neurons grossly overlap and become identical asymptotically. This is due to the fact that if the distance of the weights of two neurons is small then they efficiently teach each other and the connection strength between them further increases as they are often active simultaneously.

A solution to this problem is to choose another function $\hat{H}$. Such a function should have the following properties: *(i)* it should be positive, *(ii)* start from zero, *(iii)* increase towards a maximum, and *(iv)* decrease for larger arguments down to zero. Condition *(iv)* ensures that neurons cannot share learning if they are too close to each other. The other conditions ensure that neurons far from each other cannot learn the same input.

A function of the following form $\hat{H}(x) = \zeta(1-x)^3(\mathrm{e}^{-\gamma x} - \mathrm{e}^{-\gamma})$ satisfies these conditions if its parameters are appropriately chosen and competition persists. Parameters for our case were chosen to be $\zeta = 100, \gamma = 10$. In these runs, just as in the other experiments to be discussed later, lateral connection development and the neighbor training through these connections were applied from the very beginning, i.e. the development of lateral connections and the development of feed-forward connections were both on from the very beginning.

It is quite surprising that there is a large family of training rules that does not utilize the arbitrary function $\hat{H}$ and keeps the cooperative properties: The idea is that one may try to use the activity of the neurons in the learning rule. The point to remember here is that in the fully developed neural network we hope that 'far away neurons' will have disjunct receptive fields and a given input will give rise to no activity of most of the neurons. The learning rule may now be expressed as:

$$\Delta \mathbf{w}^{(i)} = \begin{cases} \alpha\,(1-q_{ik})^a\,(S_i^{(N)})^b\,(\mathbf{x}-\mathbf{w}^{(i)}), & \text{if } i \neq k; \\ \alpha\,(\mathbf{x}-\mathbf{w}^{(i)}), & \text{if } i = k. \end{cases} \tag{10}$$

where k denotes the index of the winning neuron, a and b are fixed positive powers. In this learning rule it is the dependence on the activities that results in no simultaneous learning for remote neurons. Factor $(1-q_{ik})$ decreases cooperativity for neurons coming too close to each other. In this way dynamic balance is ensured for cooperative learning. Based on our numerical experiments powers a and b should both be larger than 2 to keep competitivity. Integers between 2 and 4 were tried and all of them succeeded. In the limit of a and b go to infinity the neighbor training of (10) disappears and one is left with a simple competitive network. It is then expected that the training rule (10) is stable for a and b values both larger than two. This family of learning rules seems appropriate as a means of setting up adaptive cooperative Kohonen type neighbor training. The advantages of such training are dealt with below.

It may be expected that cooperative neighbor training helps adaptivity. Since in our model the distinct learning rules may be compared in a relatively unambigous fashion we tried separate runs so that we could compare the adaptivity of a competitive network and the network that utilized neighbor training (10), by applying a sudden change in the average object size. Networks respond to the change by changing filter sizes and creating or destroying filters. The time evolutions of filter sizes are shown in Fig.8. In the numerical experiment object size was decreased to one half of its original size. The competitive network (left side) responded with a sudden decrease in the size of active filters and developed a new filter much later. The network that utilized neighbor training did not allow the activity of any of its neurons to decrease to very low levels and both the decrease in the size of the active filters and the increase in the small activity filters took place at a high rate. This was followed by a slow decrease of the activity of one neuron, the only one that could not play a role in the new situation and was sentenced to remain silent. The comparison clearly shows that adaptivity increases with cooperative learning.

Conclusions

Competitive neural networks having local, extended objects as inputs can be used to form spatial filters, are able to discover the topology of the external world, and offer a means of designing neighbor training, which significantly improves adaptivity. The use of local, extended objects helps in reducing the necessary a priori information about the external world built into self-organizing neural networks.

Acknowledgement is made to the referees for their constructive criticism.

Appendix: Problems with the robustness of the Euclidean distance function

The activity of the i^{th} neuron may be expressed as:

$$D_i = \frac{1}{n^2}\sqrt{\sum_{k,l=1}^{n}(x_{kl} - w_{kl}^{(i)})^2} \tag{11}$$

where the indices correspond to the digitization of the two-dimensional space. Let us denote the set of zero elements of the input vector $\mathbf{x}$ of a given extended object by I. The number of elements of set I are denoted by $\mid I \mid$. The sum of Eq. (11) may be divided into two parts:

$$(n^2 D_i)^2 = \sum_{(k,l)\in I}(w_{kl}^{(i)})^2 + \sum_{(k,l)\notin I}(x_{kl} - w_{kl}^{(i)})^2 \tag{12}$$

Let us set the components of the initial stored vectors around $\overline{w}_{(0)}$ with a small noise content. Without loss of generality one may assume that the first neuron wins for the first presented input vector $\mathbf{x}$. Let us assume as well that $\mid I \mid$ is typically large, i.e. the extended objects are small. Now, we may approximate the average updating as:

$$\begin{aligned} \overline{w}^{(1)} &= (1-\alpha)\overline{w}_{(0)}, \text{ and} \\ \overline{w}^{(i)} &= \overline{w}_{(0)}, \text{ if } i \neq 1, \end{aligned} \tag{13}$$

where the bar denotes averaging over the components of the stored vector of a neuron. Let us examine the case that a neuron has won and a new randomly positioned object is shown to the neural network. We are interested in the probability that the same neuron shall win. To this end let us give upper and lower estimates for the activities of the previously winning and the other neurons, respectively, in the new presentation:

$$\begin{aligned} (n^2 D_1)^2 &< (1-\alpha)^2\overline{w}_{(0)}^2 \mid I \mid + (n^2 - \mid I \mid) \\ (n^2 D_i)^2 &> \overline{w}_{(0)}^2 \mid I \mid \end{aligned} \tag{14}$$

bearing in mind that the weights always fall into the [0,1] interval. If

$$D_1 < D_i, i \neq 1 \tag{15}$$

then in the next training step it is the first neuron that wins again. This inequality, however, is easily fulfilled. The inequalities (14), and (15) lead to

$$\mid I \mid /n^2 > 1/(1 + (1 - (1-\alpha)^2)\overline{w}_{(0)}^2) \tag{16}$$

The larger $\overline{w}_{(0)}$ and α, the easier it is to fulfil this condition. Let us assume that the the first neuron won t times in a row. If inequality

$$\mid I \mid /n^2 > 1/(1 + (1 - (1-\alpha)^{2t})\overline{w}_{(0)}^2) \tag{17}$$

is fulfilled, then it wins again. Inequality (17) shows that the first neuron's chance of winning keeps growing. Taking the limit of $t \to \infty$ we have

$$\mid I \mid /n^2 > 1/(1 + \overline{w}_{(0)}^2) \tag{18}$$

and this expression is independent of α. This gives rise to an upper limit of $\overline{w}_{(0)}$. Above that limit, i.e. for small objects, the Euclidean distance function cannot solve the problem or, at least, one may say that the probability of having only one winning neuron is larger than zero: according to our experience it is close to 1. Having more than one neuron, however, does not mean that more than one spatial filter will be formed. The question is how to form separate filters. As it is shown in the paper the 'spherical distance' is an appropriate solution of this problem.

References

Carpenter, G.A., and Grossberg, S. 1987. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Comp. Vision, Graphics and Image Proc.* **37** 54-115.

Desieno, D. 1988. Adding conscience to competitive learning. *Proc. Int. Conf. on Neural Networks*, IEEE Press, New York, pp. 117–124.

Földiák, P. 1990. Forming sparse representation by local anti-Hebbian learning. *Biol. Cybern.* **64** 165–170.

Fritzke, B. 1991. Let it grow — self organizing feature maps with problem dependent cell structure. *Artificial Neural Networks*, T. Kohonen, M. Mäkisara, O. Simula and J. Kangas, eds., Elsevier Science Publishers B.V., Amsterdam **Vol. 1** pp. 403–408.

Grossberg, S. 1976. Adaptive pattern classification and universal recoding. I & II. *Biol. Cybern.* **23** 121–134.

Kohonen, T. 1984. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin

Martinetz, T., and Schulten, K. 1991. A "neural-gas" network learns topologies. *Artificial Neural Networks*, T. Kohonen, M. Mäkisara, O. Simula and J. Kangas, eds., Elsevier Science Publishers B.V., Amsterdam **Vol. 1** pp. 397–402.

Szepesvári, Cs. 1992. Competitive neural networks with lateral connections: collective learning and topology representation. *TDK Thesis,* in Hungarian.

Figure captions

- Figure 1. *Digitized spatial Gaussian filters*

  Gaussian filters of a two dimensional box corresponding to expressions: 'upper left', 'upper middle' 'upper right', etc. The figure was drawn by generating pixels randomly with probabilities that correspond to the gray scale values.

- Figure 2. *Architecture of the artificial neural network*

  The ANN has a set of input nodes. Inputs connected to the network are denoted by $x_i, i = 1, 2, ..., n$. Every input node is connected to every neuron of the network. Every neuron stores a vector of the input space; neuron j stores $(w_1^{(j)}, w_2^{(j)}, \ldots w_n^{(j)})$. Neurons develop another set of connections, the (q_{kl}) topology connections, an internal representation of the topology of the external world.

- Figure 3. *Typical input*

  The box on the left hand side shows an object at a given position that was transmitted to the input nodes. The middle box shows the outputs of the input nodes developed. Input nodes developed activities according to their overlaps with the inputted object. The upper and the lower boxes on the right hand side show the outputs $D_i^{(N)}$ of the neurons and the stored vector of the winning neuron, respectively.

- Figure 4. *Training results on self-organized filter formation during training* The numbers show the training steps. Filters are formed during the first 5000 steps. At later steps the configuration undergoes minor modifications in accordance with the random object generation, but stays stable. The figure was drawn by generating pixels randomly with probabilities that correspond to the gray scale values.

- Figure 6. *Learnt one and two dimensional topologies*

  Connection thicknesses show the strengths of topology connections q_{kl}. In the left hand side figure objects were generated everywhere in the two dimensional box. No line means approximately zero strength connections. In the right hand side figure objects were generated along three horizontal strips in the two dimensional box with arbitrary ordinates. No line means zero strength connections.

- Figure 7. *Monotonicity of topological connection strengths*

  Strength of topology connection q_{kl} as a function of overlap of filters. The overlap of the k^{th} and l^{th} filters is defined as $\sum_{i,j} w_{ij}^{(k)} w_{ij}^{(l)}$.

- Figure 8. *Adaptivity of ANN's*

  After a sudden change in the external world or, here, the average object size, networks try to adapt. Adaptation means a change of filter size and creation or death of filters. The graph on the left hand side shows the evolution of filter sizes for the competitive network. The graph on the right hand side shows the evolution of filter sizes for a competitive network that developed topology connections and Kohonen type neighbor training. The graphs depict points from 100,000 learning steps prior to and 100,000 learning steps after the sudden change. Step number zero is the time of the sudden change. The region of the first 20,000 steps after the change is enlarged and enclosed with dashed lines. Size is defined as $\sum_{i,j}(w_{ij}^{(k)})^2$

- Figure 5. *Training results of the three-object case*

  Receptive filters are formed in the case of training with by showing three randomly positioned objects simultaneously. The noisy background is the result of the presence of more than one object at the same time. The noise increases the avarage filter size resulting in the strong decrease of the receptive field of one neuron in the three object case.

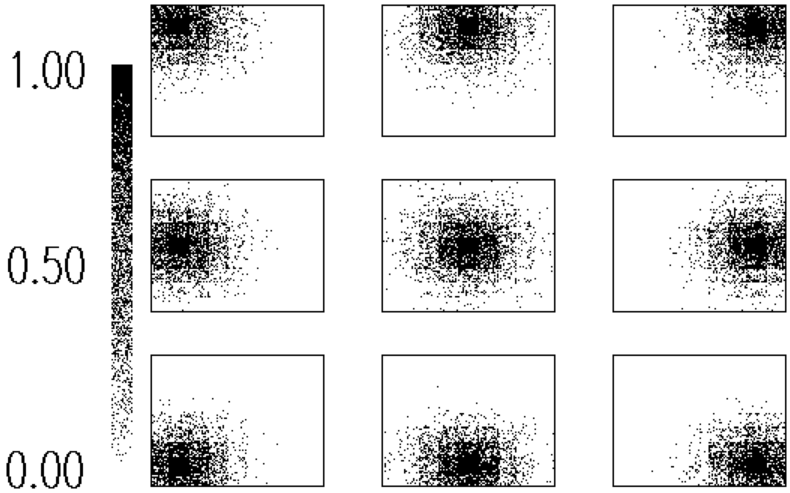

Figure 1: Digitized spatial Gaussian filters

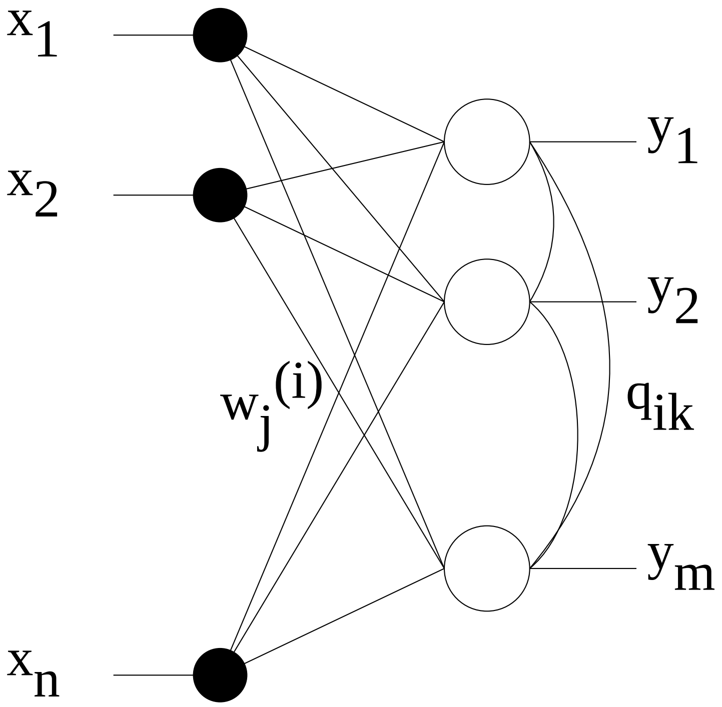

Figure 2: Architecture of the artificial neural network

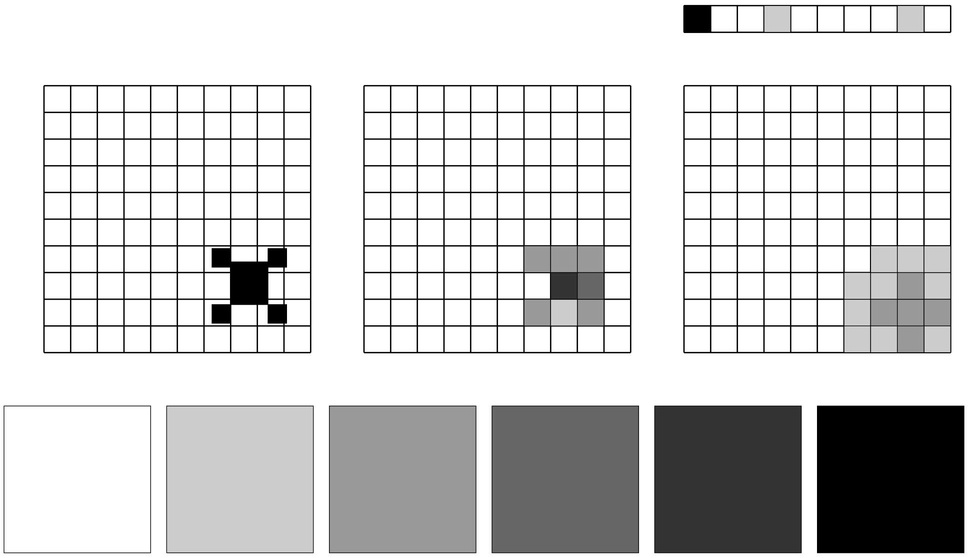

Figure 3: A captured screen

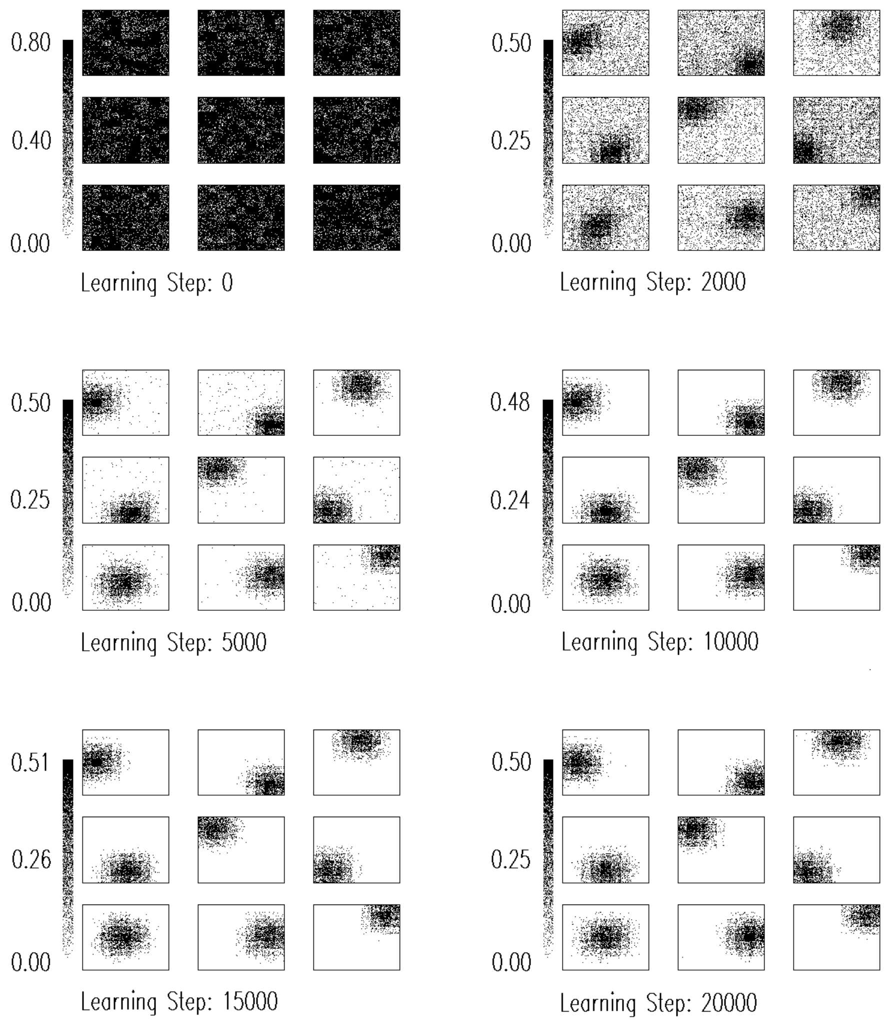

Figure 4: Training results on self-organized filter formation during training

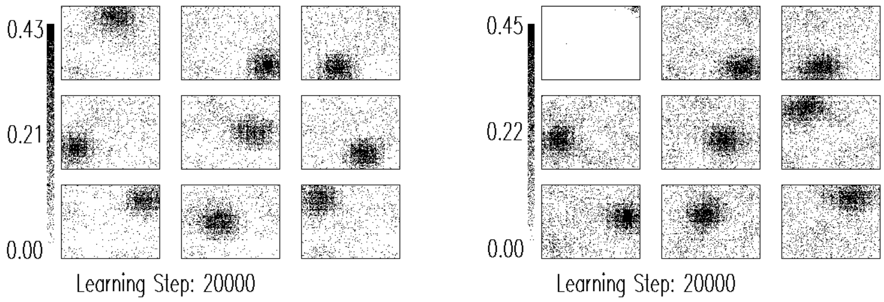

Figure 5: Training results of the two- and three-object cases

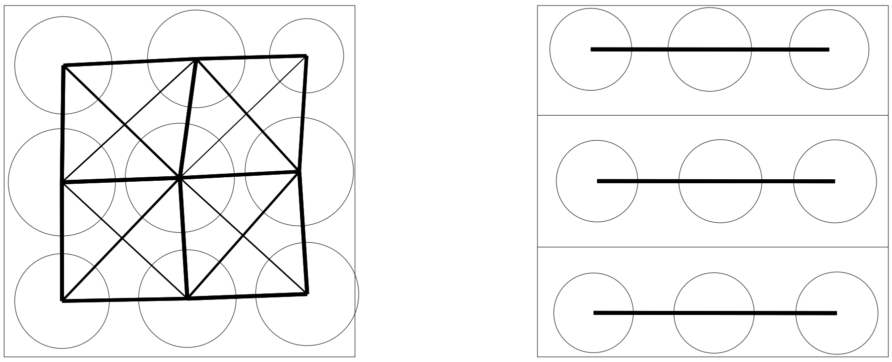

Figure 6: Learnt one and two dimensional topologies

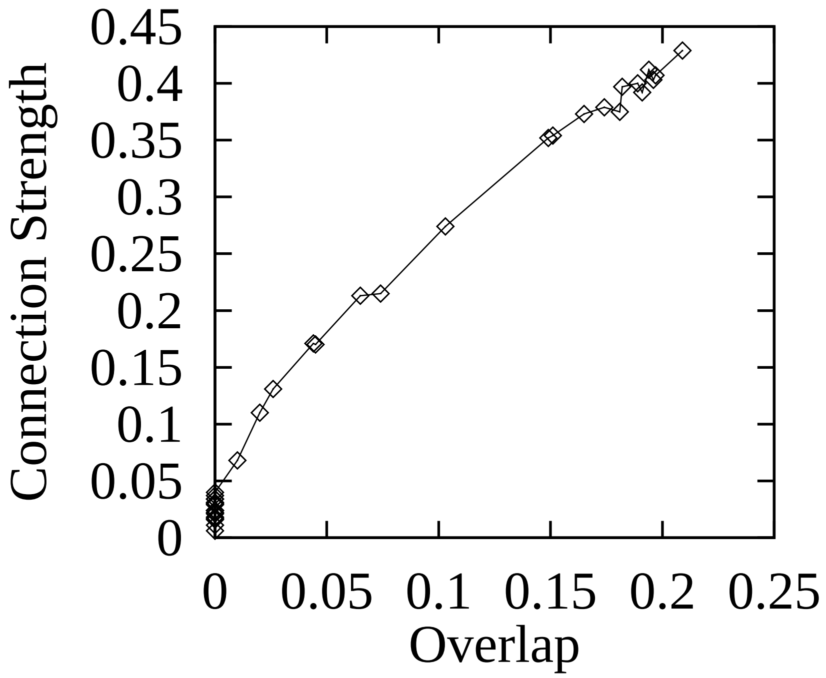

Figure 7: Monotonity of topological connection strengths

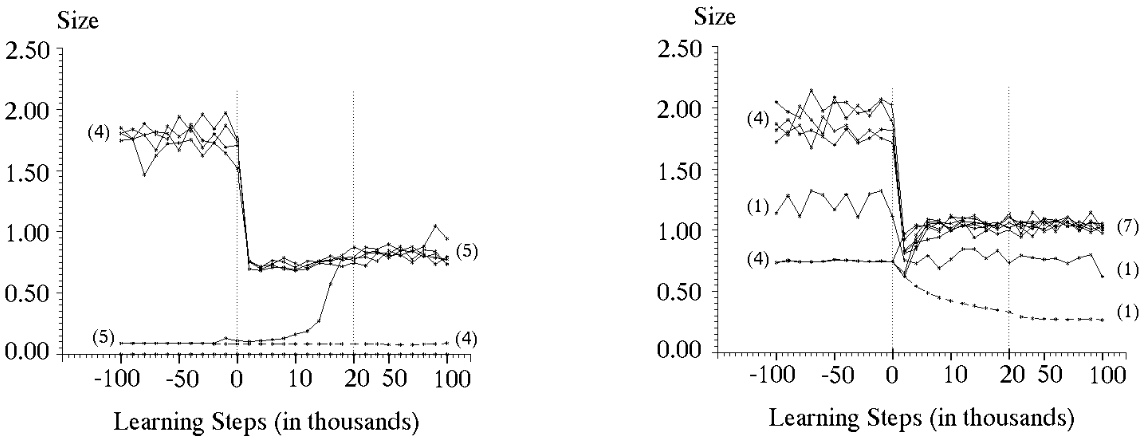

Figure 8: Adaptivity of the network