ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

Growing Self-organizing Networks - Why?

Bernd Fritzke

Systembiophysik
Institut für Neuroinformatik
Ruhr-Universität Bochum, Germany
http://www.neuroinformatik.ruhr-uni-bochum.de

Abstract. The reasons to use growing self-organizing networks are investigated. First an overview of several models of this kind is given are they are related to other approaches. Then two examples are presented to illustrate the specific properties and advantages of incremental networks. In each case a non-incremental model is used for comparison purposes. The first example is pattern classification and compares the supervised growing neural gas model to a conventional radial basis function approach. The second example is data visualization and contrasts the growing grid model and the self-organizing feature map.

1. Introduction

Growing (or incremental) network models have no pre-defined structure. Rather, they are generated by successive addition (and possibly occasional deletion) of elements. At first sight this makes them a lot more complicated than networks with static structure such as normal multi-layer-perceptrons or self-organizing maps the topology of which is chosen a priori and does not change during parameter adaptation. For growing networks, however, suitable insertion strategies have to be defined as well as criteria how to eventually stop the growth.

Is there any "pay-off" for this added complexity? Are there any practical or even principal advantages of using a growth process for network generation? Why should anyone use incremental self-organizing networks?

We like to address the questions posed by contrasting incremental networks and their corresponding non-incremental counterparts for a few examples. The examples are taken both from unsupervised and supervised learning and should illustrate that there are in fact some rather good reasons for using growing self-organizing networks.

2. Overview of growing self-organizing networks

In this section we will give a short overview of different incremental models we have developed during the last few years. From an architectural point of

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

view these models are in fact not very different at all. On the other hand, they address quite distinct problems ranging from combinatorial optimization over data visualization to pattern classification and function approximation.

2.1. Common properties

We first like to state the properties shared by all models described below. This "factorization" will then allow for a very concise description of the specific features each model has.

Common to the models in this section is the following:

- The network structure is a graph consisting of a number of nodes (also denoted as units) and a number of edges connecting the nodes.
- Each unit c has an associated position (or reference vector) w_c in input space.
- Adaptation of the reference vectors is done by generating an input signal ξ and moving the reference vector of the nearest (or "winning") units s_1 and its direct topological neighbors in the graph towards the input signal:

$$\begin{aligned} \Delta w_{s_1} &= \epsilon_b(\xi - w_{s_1}) \\ \Delta w_i &= \epsilon_n(\xi - w_i) \qquad (\forall i \in N_{s_1}) \end{aligned}$$

  Thereby N_{s_1} denotes the set of direct topological neighbors of s_1, i.e. those units sharing an edge with s_1. The symbols ϵ_b and ϵ_n are adaptation constants with $\epsilon_b \gg \epsilon_n$.

- At each adaptation step local error information is accumulated at the winning unit s_1:

$$\Delta \mathrm{E}_{s_1} = (\text{error term})$$

  The particular choice of the above error term depends on the application. For vector quantization one would, e.g., choose $\Delta \mathrm{E}_{s_1} = \|w_{s_1} - \xi\|^2$ whereas for entropy maximization an appropriate term is $\Delta \mathrm{E}_{s_1} = 1$. In the case of supervised learning one can use, e.g., the classification error whereas for a robotics application a positioning error might be used. Abstractly speaking, the error term should be a measure which is to be reduced and which is likely to be reduced in a particular area of the input space by insertion of new units in exactly this area.

- The accumulated error information is used to determine (after a fixed number of adaptation steps) where to insert new units in the network.
- All model parameters are constant over time.

When an insertion is done the error information is locally re-distributed which increases the probability that the next insertion will be somewhere else. The local error variables act as a kind of memory which lasts over several adaptation/insertion cycles and indicates where much error has occurred. The fact that the error information is kept even after insertions allows to have a constant number of adaptation steps per insertion leading to a very fast build-up of the network[1]. An exponential decay of all error variables stresses the influence of the more recently accumulated error information.

All models can in principle be used also for supervised learning by associating output values to the units either of radial basis function (RBF) type or of local linear mapping (LLM) type. This makes, however, most sense for the growing neural gas method since it reflects the inherent structure of the data most accurately [5, 8]. The other two methods due to their fixed dimensionality skew the input onto a certain number of dimensions and are for this reason more suited for (unsupervised) data visualization (see the example given below).

The respective differences of the models lie only in the constraints imposed on the topology and will be described below.

2.2. Growing cell structures

The growing cell structures (GCS) model has a structure consisting of *hypertetrahedrons* (or simplices) of a dimensionality chosen in advance [6]. A k-dimensional hypertetrahedron is special among all k-dimensional polyhedrons since it is the most simple one, having only $k+1$ vertices. Examples of hypertetrahedrons for $k \in \{1, 2, 3\}$ are lines, triangles, and tetrahedrons.

The model is initialized with exactly one hypertetrahedron. Adaptation steps as described above are performed. Always after a number λ of these adaptation steps the unit q with the maximum accumulated error is determined and a new unit is inserted by splitting the longest of those edges emanating from q. Moreover, additional edges are inserted to re-build the structure in such a way that it consists only of k-dimensional hypertetrahedrons (again). The exact re-connection procedure is actually simple enough to be described in one sentence: Let the edge which is split lead from q to a unit f then the new unit should be connected with q and with f and with all *common* neighbors of q and f. This is valid for an arbitrary dimension k.

Since the GCS model has a fixed dimensionality it effectively realizes a dimensionality-reducing mapping from the (possibly very high-dimensional) input space into a k-dimensional space. This can be used for data visualization and as a special case also for combinatorial optimization [9].

[1] We currently investigate a variant of the described method where all error information is discarded after an insertion. This eliminates completely the need to redistribute or decay error information but requires to make a number of adaptation steps per insertion which is proportional to the network size.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

2.3. Growing neural gas

The growing neural gas (GNG) model imposes no explicit constraints on the graph. Rather, the graph is generated and continuously updated by competitive Hebbian learning[2], a technique proposed by Martinetz [11]. The core of the competitive Hebbian learning method is simply to create an edge between the winning and the second winning unit at each adaptation step (if such an edge does not already exist). The graph generated is a subgraph of the Delaunay triangulation corresponding to the reference vectors. The Delaunay triangulation, however, is special among all possible triangulations of a point set since it has been shown to be optimal for function approximation by Omohundro [14].

After a fixed number λ of adaptation steps the unit q with the maximum error is determined and a new unit is created between q and one of its neighbors in the graph. Error variables are locally re-distributed and another λ adaptation steps are performed.

The topology of a GNG network reflects the topology of the input signal distribution and can have different dimensionalities in different parts of the input space. For this reason a visualization is only possible for low-dimensional input data.

2.4. Growing grid

The growing grid (GG) method [7] enforces a hyper-rectangular structure on the graph. Stated otherwise, the graph is a rectangular grid of a certain dimensionality k. The starting configuration is a k-dimensional hypercube (e.g., a 2×2-grid for $k = 2$ and a $2 \times 2 \times 2$-grid for $k = 3$). To keep this structure consistent, it is necessary to always insert complete (hyper)-rows or (hyper)-columns.

Apart from this the growing grid is very similar to the methods described above. Also the adaptation is done in exactly the same way. The accumulated error information is used to identify after λ adaptation steps the unit q with maximum error. The longest edge emanating from q is determined and a new complete hyper-row or -column is inserted such that this edge is split.

2.5. Relation to other models

What is the relation of the described incremental models to other approaches? Both, the GCS model and the GG model have some relations to Kohonen's self-organizing feature map (SOFM) [10] since a network structure of a fixed dimensionality is used. In the case of GG the relation is even stronger since a hyper-rectangular grid is produced which is also the most common network structure for SOFM. Different are the parameters which are constant in the case of GCS and GG and time-varying for SOFM.

[2]Interestingly, there exists a seemingly unrelated technique also called competitive Hebbian learning described in Ray H. White (1992), Competitive Hebbian Learning: Algorithms and Demonstrations, Neural Networks 5, pp. 261–275

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

Recently Bauer and Villmann have proposed an incremental self-organizing network (GSOM) generating also a hyper-rectangular structure by inserting complete rows or columns [2]. Instead of using accumulated error info to determine where to insert new units, they always insert near the center of the current topology. The network parameters are time-varying (e.g., the neighborhood adaptation is cooled down according to saw-tooth shaped function). In contrast to our GG method GSOM automatically chooses a dimensionality during the growth process.

Bruske and Sommer have independently from us developed a method called dynamic cell structures (DCS) [3] combining the principle of insertion based on accumulated error (first proposed in [4]) with competitive Hebbian learning as introduced by Martinetz. They have applied this to supervised learning and it is rather similar to GNG.

The GNG method is somewhat related to the neural gas (NG) method proposed by Martinetz. NG however, is a pure vector quantization method not defining a topology among the units. Rather the adaptations are done based on the distance in input space. Decaying parameters make it necessary to pre-define the total number of adaptation steps. The number of units is constant.

NG, however, may be combined with competitive Hebbian learning to build up a topology during self-organization. The resulting method has been called "topology-representing networks" (which may be a term a little too general for being limited to this particular model) [12] and delivers results comparable to the (unsupervised variant of) GNG. It should be noted, however, that the topology created by competitive Hebbian learning does not influence in any way the underlying NG method whereas in the case of GNG the topology is exploited to position new units between existing ones.

There are a couple of other related models and the discussion here is not meant to be complete overview.

3. Examples

In this section we like to give some examples contrasting incremental and non-incremental networks.

3.1. Pattern classification

A central property of the incremental networks described above is the possibility to choose an arbitrary error measure as the basis for insertion. In the case of pattern classification an obvious choice is the classification error. In figure 1 an RBF network is shown which was constructed with the supervised GNG method. The problem is a two-class classification problem and the data consists of two large well-separated clusters and four smaller ones which are rather close to each other and of less compact shape. The error-based growth process leads in this case to a distribution of centers which differs considerably from

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

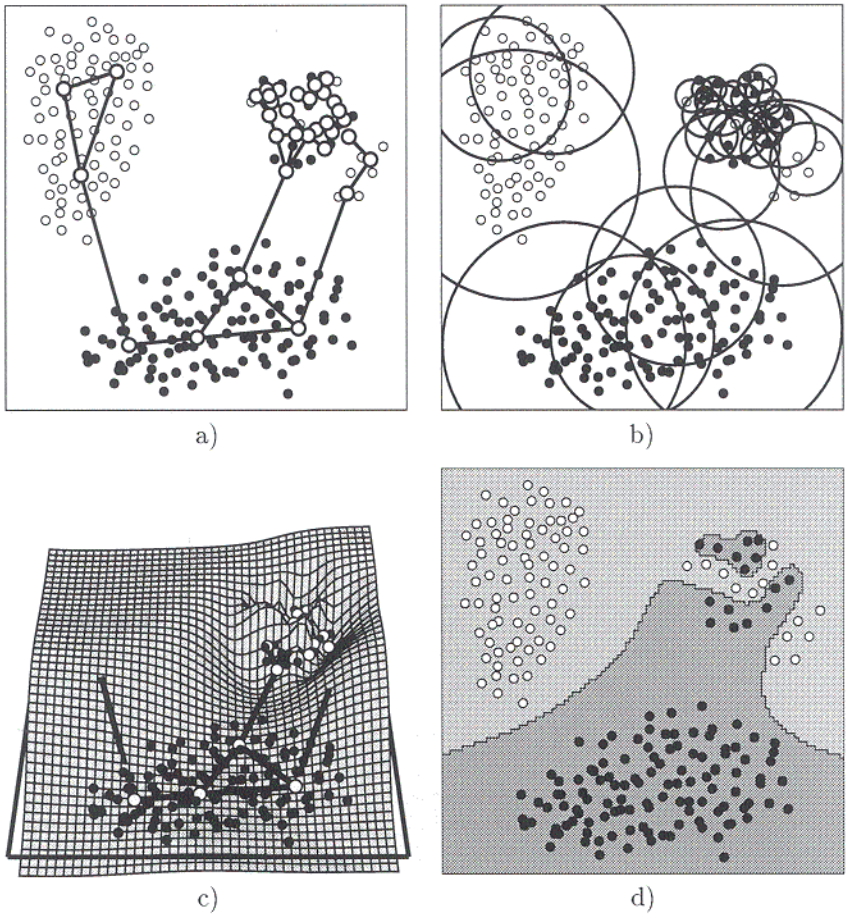


Figure 1: Classification with supervised growing neural gas. Shown is a radial basis function network generated by the supervised GNG method using the classification error as insertion criterion. a) generated nodes and edges b) standard deviations of Gaussians (computed from mean edge length) c) raw output of the networks d) decision regions obtained by thresholding the raw output.

the distribution of the data. In the final network the large clusters are handled by only a few rather large Gaussians whereas the majority of the Gaussians is located in the region of the smaller clusters which are much harder to separate.

As contrast a more conventional RBF approach following largely the work of Moody and Darken [13] is shown in figure 2. This network was generated by first distributing a fixed number of centers in input vector space with a clustering method (NG plus competitive Hebbian learning in this case). Then center positions were fixed, Gaussian activation functions were associated with

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

the centers with a width chosen equal to the mean edge length of the respective center. The delta rule was then used to train the weights to the single linear output unit which was needed to differentiate between two classes. Although the size of the network is the same as that of the previous network in this case the training patterns are not learned completely. The rather obvious reason is that more units than necessary are used for handling the two large, but "simple" clusters so that not enough units remain for handling the more difficult areas.

Why are the results of supervised GNG and a conventional RBF network so different? The clustering method used in the conventional RBF network uses only the input part of the training data (it disregards the class labels). Therefore, it has no possibility of taking into account how difficult it is to classify in a certain area of the input space. Since all data looks alike the best a clustering method can do is to distribute the available centers such that each center gets a similar share of the input data. In the second phase the centers are not moved anymore and the weights leading from the Gaussian units to the linear output unit are trained using the available labelled data. Data points of different classes lying in the region of the same Gaussian evoke a very similar activation vector and the linear output unit may not be able to separate them even in the 33-dimensional space of Gaussian unit activations (33 is the size of the networks in both examples). This is presumably what happened with the network shown in figure 2.

The supervised GNG network however starts with only a few units and immediately begins training the weights to the output unit(s) while at the same time the center positions are adapted (only slightly, however, since the adaptation parameters are very small). If a classification error occurs it is accumulated at the nearest Gaussian unit which is in a sense responsible for this area of the input space. After a while it becomes evident which Gaussian unit q has most difficulties handling "its" data and a new unit is inserted nearby, taking over part of the data of q. This is iterated until the network performs sufficiently well. If in a region of the input space no misclassifications occur then no units are added in this area even if there is a high density of data points. The network concentrates its resources on the more difficult areas.

A remark should be made concerning over-fitting. This is a potential problem for both approaches. If we increase the number of centers used for the conventional RBF approach we eventually will reach a number where the training data can be learned. Since the unsupervised clustering distributes the centers regardless of the labels we will also have more centers in those areas of the input space where the data is easy to classify potentially leading to over-fitting there. For the supervised GNG network, on the other hand, we must define a criterion when the growth should be stopped or it will continue to generate new units indefinitely and over-fit to the data. A simple method to avoid this is to observe the performance of the network on a separate validation set which is not used for training. If the performance on the validation set does not improve anymore the training should be stopped.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

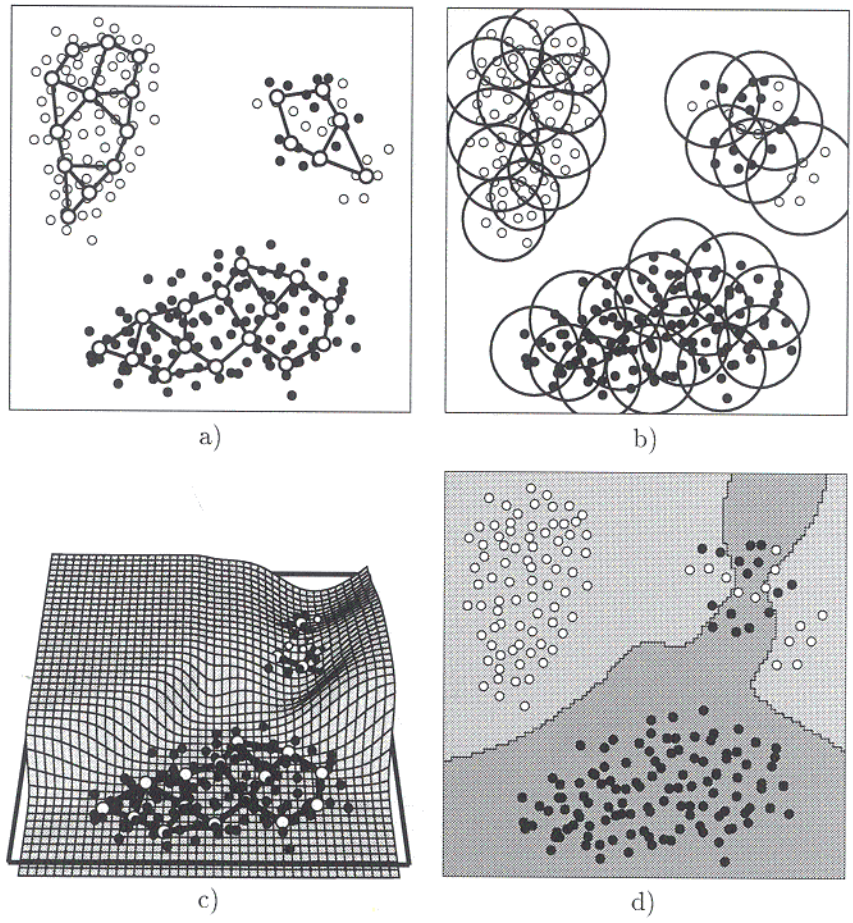


Figure 2: Classification with a conventional radial basis function network. Shown is an RBF-network consisting of 33 units generated in two phases as proposed by Moody and Darken. First the centers are distributed in an unsupervised fashion. In this case we used Martinetz' neural gas method with competitive Hebbian learning to generate connections but another clustering technique such as k-means would have produced similar results. In a second phase the center positions are frozen, Gaussian activation functions are associated with the units and weights to a linear output unit are trained with the delta rule. a) nodes and edges b) standard deviations of Gaussians (computed from mean edge length) c) raw output of the networks d) decision regions showing how the network generalizes over unseen data. The network is not able to classify all training examples correctly although it has the same size as the the net in the previous figure which had been generated by supervised GNG

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

3.2. Data visualization

Visualizing complex and high-dimensional data is of increasing relevance in may industrial and research areas. Self-organizing feature maps are often proposed for this task since they generate a mapping from the possibly high-dimensional input space to the low-dimensional structure used as network topology. In many cases this is a rectangular, often square, two-dimensional grid of units. There is no question that SOFMs generate those dimensionality-reducing mappings. The question is how well do they preserve topological relationships in the original data? For a long time this issue was handled mostly by visual inspection. Recently, however, an exact definition of topology preservation has been given by Villmann et al., the so-called topographic function [16] which makes it possible to actually do objective comparisons among different architectures. In contrast to earlier work by Bauer and Pawelzik [1] the topology of the given data is taken into account for this measure which we regard as an important contribution. We do not use this measure in the present article but we plan to do this in the future and like to mention this work.

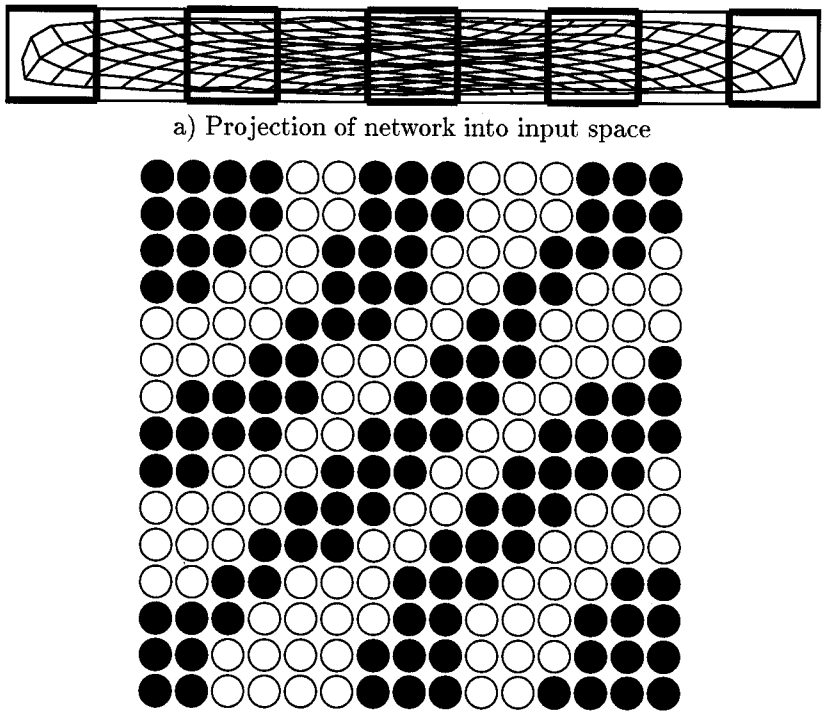
In the following we like to investigate the representation of a rather simple data set realized by a) the self-organizing feature map and b) our growing grid method. The data comes from a 9×1 rectangular area and has been "colored" black and white for easier visualization

For the SOFM we chose an array of size 15×15. We then performed a self-organizing process with a parameters and decay functions as proposed by Ritter et al. [15]. The result of the self-organization can be seen in figure 3. The projection of the network in the input space is shown in figure 3 a). The representation of the data onto the maps is depicted in figure 3 b). It is rather obvious that the map represents the data in a rather skewed way. This is not surprising since the shape of the manifold containing the data (an elongated rectangle) and the shape of the map (a square) are rather different. There is a mapping from one to the other but the topology is only partially preserved (which would certainly be confirmed by the mentioned topographic function which relates the distances in input space to those in output space).

If we now use the same data to let a growing grid network develop the result looks quite different as can be seen in figure 4. There is a much better correspondence between representation on the map and actual distances in the input space.

Another propoperty of the growing grid method may be of particular importance for data visualization. We could continue the simulation and let the network add more rows and columns if we decided after inspecting the result obtained so far that we needed a higher resolution. Since all model parameters are constant this is easily possible. The self-organizing feature map in figure 3, however, is finished in a certain sense. We have cooled down the parameters (neighborhood range and adaptation strength) according to a specific schedule and there is no obvious way of increasing the size of the network and letting it re-organize without loosing the order achieved so far. We could of course interpolate new units from existing ones but we would be limited to the convex

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72



a) Projection of network into input space

b) Projection of data to network

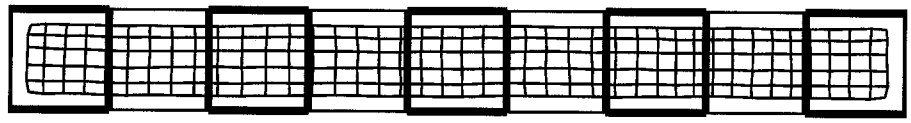
Figure 3: A self-organizing feature map has adapted to a uniform distribution in a 9×1 two-dimensional area. The network consists of 15×15 units. Due to the mismatch between network structure and shape of the data manifold a skewed representation results.

hull of the present set of units in doing this. The only general way to increase the resolution is to repeat the whole process with a larger network. Even then, however, we can not be sure that the size is sufficient and it might be necessary to try even larger networks. With the growing grid, on the other hand, we can completely re-use the results of previous simulations.

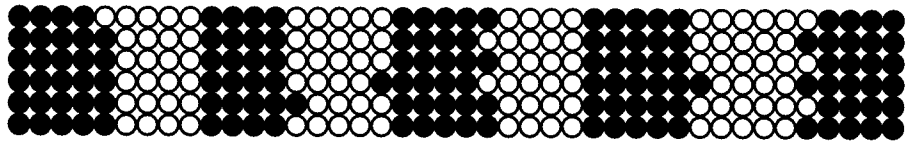
4. Discussion

In this paper we tried to clarify the motivation behind growing self-organizing networks. We did this by first giving an overview of different incremental models which all are based on the principle of *insertion based on locally accumulated error information* and which differ mainly in the constraints imposed on their topology. We related those incremental models to other approaches. We presented two examples, one from supervised and one from unsupervised learning, to point out specific advantages of incremental approaches.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72



a) Projection of network into input space



b) Projection of data to network

Figure 4: A growing grid self-organizing network has adapted to the same distribution as in the previous figure. Starting from a 2×2 grid a 43×6 network has developed automatically. The width/height ratio of the network is 7.17 and matches fairly well the corresponding ratio of the data manifold (9).

Summarizing, these advantages are:

- The possibility to use problem dependent error measures to determine where new units are inserted (insertion where necessary)
- The possibility to interrupt the self-organization process or to continue a previously interrupted one: due to the constant parameters there are no different phases in the self-organization
- Fewer magic numbers to define: the network size need not be defined in advance but can be defined indirectly by giving a performance criterion which must be met. For each parameter only its value must defined and not starting value, end value as well as its function over time as in many other approaches.

Another goal of the paper was to shed a little light on the sometimes confusing multitude of self-organizing models and to indicate some relations among them. This could not at all be complete but perhaps we were able to answer one or the other question with our remarks.

References

[1] H.-U. Bauer and K. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Transactions on Neural Networks*, 3(4):570–579, 1992.

[2] H.-U. Bauer and Th. Villmann. Growing a hypercubical output space in a self-organizing feature map. Tr-95-030, International Computer Science Institute, Berkeley, 1995.

ESANN'1996 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-25-26 April 1996, D-Facto public., ISBN 2-9600049-6-5, pp. 61-72

[3] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 7(4):845–865, 1995.

[4] B. Fritzke. Let it grow - self-organizing feature maps with problem dependent cell structure. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 403–408. North-Holland, Amsterdam, 1991.

[5] B. Fritzke. Fast learning with incremental RBF networks. *Neural Processing Letters*, 1(1):2–5, 1994.

[6] B. Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[7] B. Fritzke. Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995.

[8] B. Fritzke. Incremental learning of local linear mappings. In F. Fogelman, editor, *ICANN'95: International Conference on Artificial Neural Networks*, pages 217–222, Paris, France, 1995. EC2 & Cie.

[9] B. Fritzke and P. Wilke. FLEXMAP - A neural network with linear time and space complexity for the traveling salesman problem. In *Proc. of IJCNN-91*, pages 929–934, Singapore, 1991.

[10] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[11] T. M. Martinetz. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *ICANN'93: International Conference on Artificial Neural Networks*, pages 427–434, Amsterdam, 1993. Springer.

[12] T. M. Martinetz and K. J. Schulten. Topology representing networks. *Neural Networks*, 7(3):507–522, 1994.

[13] J. E. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[14] S. M. Omohundro. The Delaunay triangulation and function learning. Tr-90-001, International Computer Science Institute, Berkeley, 1990.

[15] H. Ritter, T. Martinetz, and K. Schulten. *Neural Computation and Self-Organizing Maps: An Introduction.* Addison-Wesley, New York, 1992.

[16] Th. Villman, R. Der, M. Herrmann, and Th. Martinetz. Topology preservation in self-organizing feature maps: exact definition and measurement. CSE Department preprint 11/94, Univ. Leipzig, 1994. (to appear in IEEE TNN).