An Incremental Unsupervised Learning Scheme for Function Approximation

Christian-A. Bohn
German National Research Center for Information Technology
Sankt Augustin, Germany
bohn@gmd.de

Abstract

A new algorithm for general robust function approximation by an artificial neural network is presented. The basis for this work is Fritzke's supervised growing cell structures approach which combines supervised and unsupervised learning. It is extended by the capability of resampling the function under examination automatically, and by the definition of a new error measure which enables an accurate approximation of arbitrary goal functions.

1. Introduction

In many applications, instead of calculating a function's exact value, it is sufficient to accept a rough, fast, and cheap approximation, at a first glance, and then to decide whether to increase accuracy by investing more computational resources. In most of these cases, a functional *model* is trained by single test samples calculated from the function under examination — the goal function.

The approximation model should be efficient in terms of accounting for coherency in the goal function, i.e., its internal representation should focus on highly varying locations. The same holds for the selection of training samples. Instead of drawing these arbitrarily, a resampling strategy should be able to direct attention to locations of low approximation accuracy.

Fritzke's *supervised growing cell structures* algorithm (SGCS) [2] can be seen as an alternative to the classical *Kohonen self-organizing map* (SOM) [4]. Both are able to adapt their internal structure to the coherency in the input space (*clustering*). SGCS are additionally suitable for being trained by supervision. Due to the combination of supervised and unsupervised learning, it delivers the basis for the development of the *incremental supervised growing cell structures* (ISGCS) presented in this work.

In the following, we roughly explain the standard SGCS approach. Then, we discuss our extensions: first, we modify the error measure (*resource term*) to enable general approximation of arbitrary goal functions, second, a method for incremental resampling of the function space according to the coherency and to the approximation accuracy of the function values is presented. We prove the results by means of several example experiments.

2. Supervised growing cell structures

2.1. Overview

An SGCS network contains two layers. The first is instantiated by a set of n-dimensional *radial basis functions* (RBF) [5], called *cells*, the second accumulates each of the output *activations* of the RBFs to form the m-dimensional output vector of the network (see figure 1). It realizes a function $\tilde{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ which serves as an approximation of a goal function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Samples are drawn from f, which the network is trained with. At each state of the learning process the network generalizes the function f over its input space to a certain accuracy.

An important feature of SGCS is the combination of supervised and unsupervised learning through different learning strategies for the first and the second layers. The network is trained by presenting input/output pairs $(\xi, \zeta) \in (\mathbb{R}^n \times \mathbb{R}^m)$. The unsupervised part is accomplished by moving the cells of the first layer according to the input ξ to find centers of clusters in the input data. Concurrently, the second layer is adapted to deliver the intended output ζ.

Moving the RBFs accounts for an additional neighborhood relation between the single cells. It creates a topological structure of predefined dimensionality k on the training data. In the two-dimensional case, the final network of RBFs depicts a *map* of reference cells, similar to Kohonen's self-organizing feature map. Generally, the network consists of a mesh of predefined basic elements, such as triangles, quadrilaterals, tetrahedrons, etc., according to the

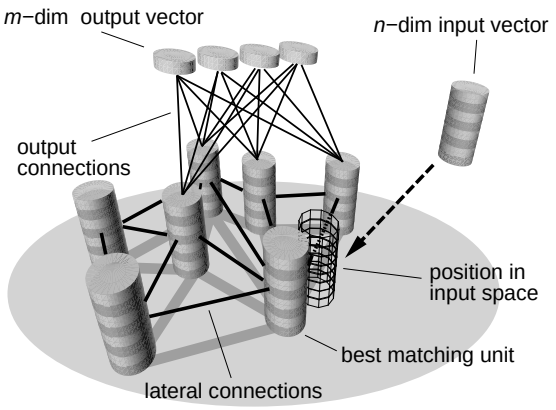

Figure 1. An example SGCS network.

underlying k.

Each cell of the network contains a *resource term* which holds particular information about the samples presented to the network. It delivers local information about the training accuracy and is utilized to insert or to delete cells from the network. The SGCS grows at locations with a high resource term and shrinks at places where the resource term signalizes a sufficient approximation accuracy.

In the following we formally explain the general algorithm limited to the features crucial for this work. For more information see [2].

2.2. Network definition

The initial topology of the network is a set of cells, A, connected in a k-dimensional structure by lateral connections. The only basic element is a k-dimensional simplex, i.e., for $k = 2$, this is a triangle. During a self-organizing process, new cells are added to A in a way that, at each time step, this basic structure is maintained.

Every cell c has attached an n-dimensional synaptic vector $\vec{w}_c$ which can be seen as the position of c in the input vector space V. A *mapping* $\phi_{\vec{w}} : V \to A$ is defined as

$$\phi_{\vec{w}} : V \to A, (\xi \in V) \mapsto (\phi_{\vec{w}}(\xi) \in A), \\ ||\vec{w}_{\phi_{\vec{w}}(\xi)} - \xi|| = \min_{r \in A} ||\vec{w}_r - \xi||, \qquad (1)$$

with $\vec{w}$ the set of all synaptic vectors $\vec{w}_c, c \in A$. $\phi_{\vec{w}}(\xi)$ is called the *winning* or the *best matching unit* (BMU) for an input vector ξ. The BMU is the cell with the smallest distance to ξ.

By equation (1) V is partitioned into a number of regions F_c $(c \in A)$, each consisting of locations with the common nearest synaptic vector $\vec{w}_c$. This is known as *Voronoi tessellation* and the regions are denoted by *Voronoi regions* [2]. We define f_c the k-dimensional Voronoi volume of a cell and approximate it by $f_c = \overline{l_c}^l \approx |F_c|$, with $\overline{l_c}$ is the mean length of the l *edges* (the lateral connections) emanating from a cell c. The length of an edge between two cells i and j is defined as the Euclidian distance between their synaptic vectors, $l_{i,j} = ||\vec{w}_i - \vec{w}_j||$.

The supervised layer of the network is defined by one output weight vector for each cell, $\vec{v}_c \in \mathbb{R}^m, c \in A$. The output of the network, $\kappa : \mathbb{R}^n \to \mathbb{R}^m$, according to an input ξ is calculated as

$$\kappa(\xi) = \sum_{c \in A} \vec{v}_c M_c(\xi),$$

with

$$M_c(\xi) = \exp(-\frac{||\xi - \vec{w}_c||^2}{\delta_c^2}).$$

M_c is the output of a cell c (the *activation*). δ_c is assigned to f_c.

We define the neighborhood N_c of a cell c as the set of cells which are directly connected by lateral connections determining the k-dimensional topology.

2.3. Training

If an I/O pair (ξ, ζ) is presented to the network for training, the sets $\vec{w}$ and $\vec{v}$ with $\vec{v} = \{\vec{v}_c, c \in A\}$ are modified such that $\vec{w}$ adapts to the input distribution by moving cells in n-space, and the vectors $\vec{v}$ are modified in order to approach the intended output value ζ. With the notation $X^{\text{new}} = X^{\text{old}} + \Delta X$, and considering s as a specific BMU, b as one of the direct neighbors of s, and c as one arbitrary cell of the network A, the adaption of the cell weights for each iteration cycle is done as follows.

$$\begin{aligned} \Delta\vec{w}_s &= \epsilon_b(\xi - \vec{w}_s), && \text{for the BMU } s, \\ \Delta\vec{w}_b &= \epsilon_n(\xi - \vec{w}_b), && \forall b \in N_s, \\ \Delta\vec{v}_c &= \eta(\zeta - \kappa(\xi)) \cdot M_c, && \forall c \in A, \end{aligned}$$

with ϵ_b, ϵ_n, and η the learning parameters for the input weights of the BMU, for its neighbors, and for the output weights, respectively.

Every cell contains a *resource term* τ_c which tracks a value for the actual approximation accuracy. For the classical GCS approach, the resource term of a specific cell c is proposed in two fashions. First, counter-like as an approximation of the local sample distribution presented during a certain time, second, as some mean value of the error which is generated by the succeeding samples. The first type is used for realizing pure clustering, the second for supervised learning.

The resource term for supervised training is defined as the difference between the output of the network $\tilde{\zeta} = \kappa(\xi)$

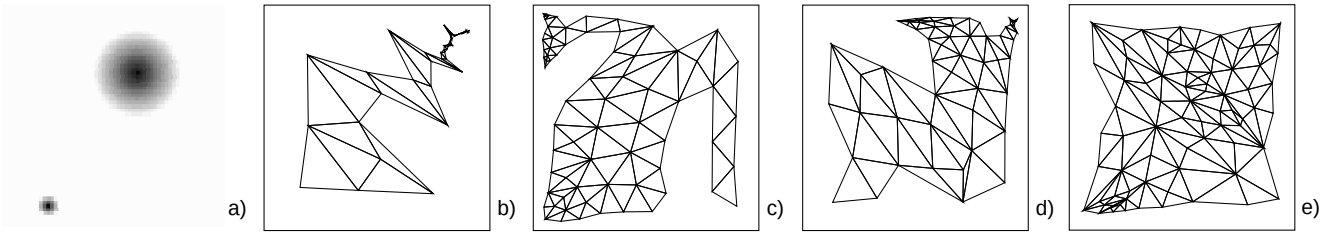

Figure 2. (a) is the goal function to be approximated, (b-d) example network topologies trained with the classical definition of the resource term, (e) the result from the new resource term definition.

and the intended output ζ. For one iteration step, τ_c of each cell of the whole network is modified as follows.

$$\Delta\tau_s = \|\zeta - \tilde{\zeta}\|^2, \quad \text{for the BMU } s, \tag{2}$$

$$\Delta\tau_c = -\alpha \cdot \tau_c^{(old)}, \quad \forall c \in A. \tag{3}$$

α is a predefined "forgetting" parameter which weighs the influence of more recent iteration steps more strongly.

After a certain number λ of iteration cycles, the cell c with the largest τ_c is selected from A, and insertion of a new cell is done in the middle of the longest edge originating from c.

In case of insertion or deletion of cells, the network structure must be kept homogeneous (only simplices of dimension k), and the resource terms and weights must be modified in order to account for the modified Voronoi regions. For implementation details see [2].

So far we have described the standard SGCS approach. Our extensions are explained in the following. For clarification, the next section is accompanied by practical examples, and, to ease visualization, we concentrate on two-dimensional functions, although the presented algorithms are not bounded to $n, k = 2$. If necessary, we give hints for the adaption to higher dimensions.

3. Incremental supervised growing cell structures

3.1. A new resource term

Several experiments indicated that the definitions (2, 3) are not suitable for approximating arbitrary functions, since the error mainly depends on the sample distribution, i.e., the number of hits inside a specific Voronoi region. Even our resampling strategy proposed in the next section will run into problems with this resource term.

The Voronoi volume of the cells must additionally be taken into account leading to a definition which looks similar to the L_n error measure. In our experiments the L_2 error has been proven as to be very robust and we will refer to it, only.

We redefine the resource term by two components $\tau_{cnt,c}$ and $\tau_{err,c}$. Consider the cell s as BMU. $\tau_{err,s}$ is incremented by the error which the network delivers compared to the actual ζ. $\tau_{cnt,s}$ counts the samples which fall into that Voronoi region,

$$\begin{aligned}\Delta\tau_{err,s} &= \|\zeta - \tilde{\zeta}\|^2,\\ \Delta\tau_{cnt,s} &= 1.\end{aligned}$$

Concurrently for all cells $c \in A$, the terms

$$\begin{aligned}\Delta\tau_{cnt,c} &= -\alpha \cdot \tau_{cnt,c},\\ \Delta\tau_{err,c} &= -\alpha \cdot \tau_{err,c}\end{aligned}$$

are added, which provide the weighted mean value over a certain time period. The final resource value is calculated according to

$$\tau_c = \frac{\tau_{err,c}}{\tau_{cnt,c}} \cdot |F_c|, \ \forall c \in A. \tag{4}$$

Examples. Figures (2b-d) show simple tests with the original definition of the resource term from equations (2, 3). The goal function is shown in figure (2a), drawn as grey-scale picture. The input space is projected on the x- and y-axis of the image. The brightness of a pixel is proportional to the function value.

It can be observed (figures (2b-d)) that the generation of cells is directed into regions of varying function values in order to increase approximation accuracy. Using the classical error definition which does not consider the Voronoi area, a high resource term is not essentially decreased, and, in the worst case, the place of generation of new cells will never leave these locations. Even in less critical cases, the network structure is not well suited to represent the goal function adequately.

The different results from figures (2b-d) are generated by slightly modifying the learning parameters. This also shows

the low robustness of the classical error definition. The algorithm totally collapses, if the proposed resampling algorithm (presented in the next section) enforces a high sample distribution in regions with low approximation accuracy.

In contrast to that, figure (2e) shows an example network trained with the proposed error measure from equation (4).

3.2. Resampling

As depicted in the introduction, resampling should avoid selecting too many initial samples from the goal function f in order to circumvent undersampling which happens in two cases: first, if the shape of the goal function is more complicated than the number of local samples can expose (*uncertainty principle*), second, the goal function generally contains sharp boundaries which separate areas where the input is defined from those where no function values exist. Due to the fact that cells representing a lower sample distribution tend to be pulled into regions of a higher distribution, these 'exterior' locations in general are not sufficiently represented by the final network topology. In the following, we call the first kind *critical regions* (CR), the second *exterior regions* (XR).

The proposed resampling algorithm selects new samples from the goal function to directedly increase the sample distribution at the CRs and the XRs.

Assuming that CRs can be identified through a high cell resource value, and that XRs are equivalent to the regions of the function space which lie "outside" of the actual network expansion, we define the resampling strategy as follows.

Consider a relation $\sigma_\tau : (c \in A) \rightarrow \{T, F\}$, which determines, if a cell c is a *critical cell* (CC), i.e., if it is located in a critical region of the function input space,

$$\sigma_\tau(c) = (\tau_c > \omega \cdot \overline{\tau}),\ c \in A,$$

with ω a threshold for being a high-resource cell and $\overline{\tau}$ the mean value of the resource terms of all cells. Further, we define the *sub-network* of A which consists only of critical cells by the term A_{CC},

$$A_{CC} = \{c, c \in A | \sigma_\tau(c)\},$$

and the relation "inside of the range of a set of cells", $\rho : \mathbb{R}^n \rightarrow \{T, F\}$,

$$\rho_A(\xi) = (D_A(\xi) > \varphi), \tag{5}$$

with

$$D_A : \mathbb{R}^n \rightarrow \mathbb{R}, D_A(\xi) = \sum_{c \in A} M_c(\xi), \tag{6}$$

with a threshold φ, and the overall *network activation* D_A, the sum of the activations of all cells of network A.

Definitions (5, 6) enable two essential predicates. First, an input ξ lies within a critical region of a particular network, A, if $\rho_{A_{CC}}(\xi)$ returns true. Second, ξ lies outside of the range of the whole network if the term $\rho_A(\xi)$ returns false.

If a test sample ξ fed into the network exposes one of these cases — if the function θ with $\theta : \mathbb{R}^n \rightarrow \{T, F\}, \theta(\xi) = \rho_{A_{CC}}(\xi) \vee \neg\rho_A(\xi)$ is true — then the input space around ξ is assumed being not sufficiently represented by the network topology, and resampling will take place there.

The resampling step scans the whole input space by evaluating θ. At places where θ succeeds, new samples are created and added to the actual training sets.

The algorithm successively generates sets of sample I/O pairs, $\mathcal{S}_j \in \mathbb{R}^n \times \mathbb{R}^m : \{(\xi_i, \zeta_i)\} |\ \zeta_i := f(\xi_i), i = 1, ..., |S_j|,\ j = \{0, ..., p-1\}$. p is the actual number of sample sets in the set of all samples $\mathcal{S} = \{\mathcal{S}_j\}, j = 0, ..., p-1$. The initial set $\mathcal{S}_0$ is taken from V completely at random. All further sets are calculated as described. For training the network, samples are selected from all sets $\mathcal{S}$ with equal probability.

Finally, we need to know, *when* a new resample step has to be triggered. Comparable to the uncertainty principle, we define the ratio of the number of cells and the number of training samples as criterion for an undersampling through the actual amount of cells. If $\psi = |A|/|\mathcal{S}|$, with $|\mathcal{S}| = \sum_{j=0}^{p-1} |\mathcal{S}_j|$, rises above a certain threshold $\psi_{A,\mathcal{S}}$ a new set of samples is generated. The complete algorithm is outlined in figure 3.

Create initial sample set $\mathcal{S}_0$ randomly, set # sample sets $p = 1$.
Set the counter $\lambda_c := \lambda$.
  Select i, j randomly, present $(\xi_i, \zeta_i) \in \mathcal{S}_j$ to the net.
  Adapt the cell weights according to (ξ_i, ζ_i).
  if $\lambda_c = 0$ insert a cell, set $\lambda_c := \lambda$, else decrement λ_c.
  repeat until #cells/#samples exceeds $\psi_{A,\mathcal{S}}$.
  add a new I/O set $\mathcal{S}_{p+1}$ to $\mathcal{S}$, increment p.
repeat

Figure 3. The extended ISGCS algorithm.

Examples. Figure 4 shows training results for the goal function from figure (2a), with $n = 2, m = 1$, created using the proposed resampling scheme. Figure (4a) is the approximated function delivered by the network. A difference to the goal function is hardly noticeable. The approximation error is below 5% (L_2).

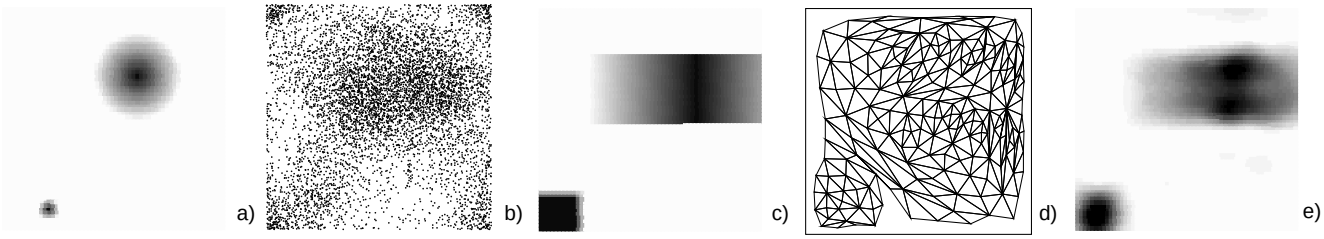

Figure 4. (a) is the approximation of the function from figure (2a), (b) the sample distribution generated during training by 5-10 resampling steps. Figure (d) is the network structure, and (e) the approximation of the goal function from (c). (b) and (d) expose the sample and the cell distributions according to the coherency in the goal function.

Figure (4b) presents the corresponding sample distribution which arose after several (less than 10) resampling steps. It clearly marks regions in the goal function input space which are more difficult to approximate with a bigger amount of samples. Even the exterior zones are resampled several times in order to avoid a network growing without considering the function boundaries.

Figure (4c) shows a different goal function. Its final network structure can be seen in figure (4d) and the approximation in figure (4e). It can be observed that the distribution of cells behaves the same way as the distribution of samples. Sharp boundaries with low coherency of the function values are represented by more cells than "smooth" locations.

For completeness, we list some example parameters used in the presented training tests: $\epsilon_b = 0.01$, $\epsilon_n = 0.001$, $\eta = 0.1$, $\alpha = 0.05$, $\lambda = 300$, $\omega = 0.6$, $\varphi = 1$, $\psi_{A,\mathcal{S}} = 0.05$.

3.3. Attaching cells at function boundaries

In the preceeding section, we presented a counteragent to avoid an insufficient representation by the network approximation in exterior regions of the goal function. This section deals with another way to account for this, based on the user's knowledge of the function boundaries.

In contrast to defining an initial network (one simplex) at random, boundary cells are selected which have a fixed position, and which are connected to one central cell. Thus, the iteration starts with a network which spans the whole function space.

For the presented two-dimensional functions, the initial cells are placed at the corners of the rectangle defined by the goal function expanse. The training process begins with four triangles.

The boundary cells take part only in the unsupervised training of the output weight vectors. That means, they are marked as fixed and do not move according to the input samples. Cells which are inserted at boundary edges must be signed to behave the same way. Boundary edges are clearly denoted by the fact that the two attached cells are marked as fixed.

Generally, this scheme is valid for any dimension k of the ISGCS by initially connecting groups of k neighboring boundary cells with a center cell forming a k-dimensional simplex. It should be mentioned that problems may arise if the initial boundaries have a complicated shape. This leads to inconsistent simplex shapes, and thus, to a lower robustness of the training process.

Examples. Figure 5 shows two example networks which are calculated by applying the complete ISGCS algorithm. It can clearly be observed that the networks span the whole function range.

Figure (5a) and (5b) are the network structure and the approximation of the example function from figure (2a). Figures (5c-e) show the results from learning the function from figure (4c).

Consider the accurate representation of high varying function locations by a higher amount of samples and cells. Difficulties in the representation of boundary regions are now completely avoided (see the representation of the small square at the bottom of figures (4e) and (5e)). Even the increased sample distributions at the boundaries vanish (figure (5d)) since the network is forced to take care of these regions through the fixed boundary cells.

4. Summary

We have described an extension to the classical growing cell structures approach by Fritzke [2], which enables the approximation of arbitrary goal functions by an iterative learning scheme.

The presented incremental supervised growing cell structures (ISGCS) deliver a method which is capable of resampling the goal function under examination automatically. This essentially reduces the amount of samples needed for an accurate training, since the method selects samples only

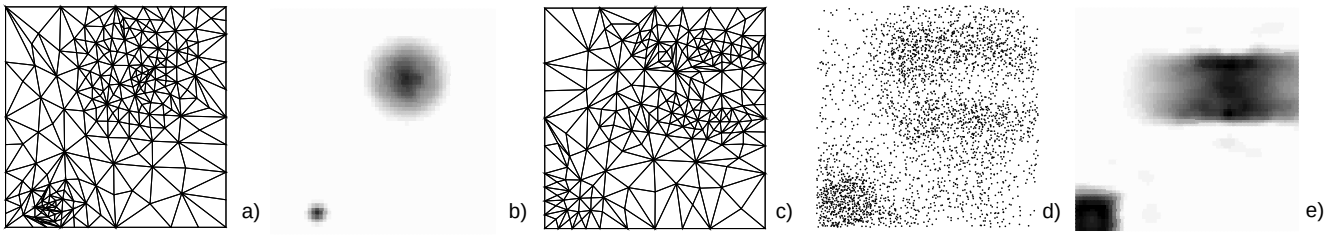

Figure 5. (a-e) show results from the complete ISGCS algorithm. (a) is the network topology for the goal function from figure (2a), (b) its approximation. (c) is the final network structure for the approximation of the function from figure (4c), (d) its sample distribution, and (e) the approximation. Consider the representations at the boundaries in (e) compared to (4e) (the square at the bottom of (e) and that from figure (4e)), and the reduced amount of generated samples (figure (d)).

from locations which expose a lack of approximation accuracy due to a lower coherency of the function values.

To enable this feature we also modified the error resource term of the classical SGCS according to the L_n error norm.

The robustness and the efficiency of the algorithm have been shown by several examples, and the algorithm has successfully been applied in [1].

Subject of future work will be extensive testing of higher network dimensions and the adaption of the algorithm to recent approaches like [3].

References

[1] C.-A. Bohn. Efficiently representing the radiosity kernel through learning. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 123–132, Wien, Austria, 1996. Springer-Verlag Wien New York.

[2] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. Technical Report ICSI TR-93-026, International Computer Science Institute, Berkeley, CA, May 1993.

[3] B. Fritzke. Incremental learning of local linear mappings. In *Proceedings of the ICANN-95*, Paris, France, 1995.

[4] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, pages 59–99, 1982.

[5] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. Technical Report YALEU/DCS/RR-654, Dept. of Computer Science, Yale University, New Haven, CT, 1989.