INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Scatter-partitioning RBF network for function regression and image segmentation: Preliminary results

Andrea Baraldi[*]

TR-98-017 [†]

June 1998

Abstract. Scatter-partitioning Radial Basis Function (RBF) networks increase their number of degrees of freedom with the complexity of an input-output mapping to be estimated on the basis of a supervised training data set. Due to its superior expressive power a scatter-partitioning Gaussian RBF (GRBF) model, termed Supervised Growing Neural Gas (SGNG), is selected from the literature. SGNG employs a one-stage error-driven learning strategy and is capable of generating and removing both hidden units and synaptic connections. A slightly modified SGNG version is tested as a function estimator when the training surface to be fitted is an image, i.e., a 2-D signal whose size is finite. The relationship between the generation, by the learning system, of disjointed maps of hidden units and the presence, in the image, of pictorially homogeneous subsets (segments) is investigated. Unfortunately, the examined SGNG version performs poorly both as function estimator and image segmenter. This may be due to an intrinsic inadequacy of the one-stage error-driven learning strategy to adjust structural parameters and output weights simultaneously but consistently. In the framework of RBF networks, further studies should investigate the combination of two-stage error-driven learning strategies with synapse generation and removal criteria.

Keywords: RBF networks, supervised and unsupervised learning from data, prototype vectors, synaptic links, Gestaltist theory, image segmentation, low-level vision.

[*]ICSI, 1947 Center Street, Suite 600, Berkeley, CA 94704-1198, Email: baraldi@icsi.berkeley.edu

[†]Internal report of the paper entitled "Image segmentation with scatter-partitioning RBF networks: A feasibility study," to be presented at the conference *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation*, part of SPIE's International Symposium on Optical Science, Engineering and Instrumentation, 19-24 July 1998, San Diego, CA.

1 Introduction

A Radial Basis Function (RBF) network consists of a hidden layer of neural units plus an output layer where linear combinations of activations provided by the hidden units form the network outputs. Among possible RBFs, the Gaussian function, $g(\cdot)$, is preferred due to the fact that it is localized and factorizable [1]. When the Gaussian Radial Basis Function (GRBF) is adopted, the approximating function provided by the GRBF network is

$$\hat{\mathbf{y}}(\mathbf{x}) = \sum_{j=1}^{c} \mathbf{w_j} g(\mathbf{x}; \mu_\mathbf{j}, \sigma_\mathbf{j}). \tag{1}$$

In Eq. (1), the following adjustable parameters, related to the hidden layer, are termed structural parameters: the number c of hidden units (also termed mixture components or basis functions), positions (templates or prototype vectors) $\mu_\mathbf{j}$ and covariances $\sigma_\mathbf{j}$, $j = 1, ..., c$, of the GRBFs. Related to the output layer, parameters $\mathbf{w_j}$, $j = 1, ..., c$, are termed output weights.

In both classification and function regression tasks, the role of RBF networks is to obtain good predictions for unobserved data by optimizing the values of the network adjustable parameters on the basis of a data set of observed input-output examples. This parameter adaptation process is called learning from data. For the class of RBF networks the learning task is typically implemented as a two-stage procedure, termed hybrid learning [2]. A hybrid learning algorithm combines unsupervised with supervised learning as described below. In the first stage, an unsupervised learning method adjusts the structural parameters of the RBF network on the basis of the density of the input vectors exclusively, i.e., without employing the input-output examples of the supervised training set. Unsupervised adaptation of the network parameters is also termed (input) data-driven learning. Once that structural parameters are set, the second stage of the hybrid learning scheme employs a supervised learning method to adjust the weights of the output layer on the basis of the supervised training set. Supervised adaptation of the network parameters is also termed error-driven learning.

RBF networks employing a hybrid learning procedure feature the following interesting properties: i) they employ simple architectures to perform complex mappings; ii) they train faster than multi layer perceptrons; iii) they may be particularly attractive for applications where input patterns are readily available but input-output sample pairs are difficult to be gathered; iv) they are easily interpretable when they employ RBFs that are well localized [1]; and v) the use of unsupervised learning methods can be quite successful in practice when the input distribution is highly nonuniform [3].

Disadvantages include: i) many unsupervised techniques require some parameters of the hidden layer be fixed in advance, typically the number of hidden units; and ii) the distribution of RBFs in the input space as it is computed by the unsupervised technique may be poor for the classification or regression problem at hand, e.g., unsupervised methods may form clusters of input vectors that are closely spaced in the input space but belong to different classes [1], [4]-[6]. This is tantamount to saying that in practical applications where RBF networks based on hybrid learning techniques are involved, if the number of hidden units (i.e., the number of adjustable parameters) is increased by the user, there is no guarantee of improving the system's performance on a test set, consisting of unobserved examples, because the unsupervised algorithm may locate basis functions in regions of the input space where they are either useless or harmful for implementing the desired input-output mapping [6].

To overcome this limitation, growing RBF networks based on error-driven learning techniques have been recently proposed in the literature [6]-[9]. In these systems localized basis functions are selectively positioned at those locations of the input space where it is difficult to approximate the target input-output mapping at the desired level of precision.

Growing RBF networks employ either scatter- [6]-[8], or grid-partitioning mechanisms [9], such that the input space is covered with localized basis functions, also termed small patches [4], or data windows [10]. Unlike grid-partitioning systems, scatter-partitioning systems position small patches at locations which are not known *a priori*. Among scatter-partitioning RBF networks, those proposed by Karayiannis and Fritzke employ learning strategies whose properties are complementary: Karayiannis' learning is two-stage, error-driven, and batch, while Fritzke's method, termed Supervised Growing Neural Gas (SGNG), adopts a one-stage, error-driven, on-line learning mechanism. One-stage learning referred to RBF networks means that structural parameters and output weights are trained simultaneously (in parallel).

The different attributes of the learning strategies adopted by the Karayiannis and SGNG algorithms reflect a difference in properties of their supervised training sets. In batch learning parameters are estimated once for every processing epoch, i.e., on the basis of the full training data set. This implies that batch learning can be employed only when the data set of observed examples is finite and small [11]. On the contrary, in on-line learning a steady (infinite) stream of data is assumed to be presented to the system; in this case, it may be that the input-output distribution is non-stationary, i.e., it may change with time, so that specific rules for relocating basis functions within the input space are enforced [12].

Additional major differences between SGNG and the Karayannis growing system are that: i) while Karayannis' system is incremental, i.e., the number of hidden units increases monotonically, SGNG may generate as well as delete hidden units dynamically; ii) by adopting the Competitive Hebbian Rule (CHR) [13], SGNG is capable of generating synaptic links between pairs of hidden units (intra-layer connections); and iii) according to a heuristic, SGNG may also remove synaptic links. This means that, from a theoretical point of view, the expressive power of the SGNG model is superior to that of the Karayannis growing system.

To the best of our knowledge, performances of these two systems reported in the literature regard classification tasks exclusively, i.e., no application to function regression problems has been described.

In this paper we discuss the application of a slightly modified version of SGNG to approximate images, which are 2-D signals. In the mammalian low-level visual system an image is partitioned into segments that are perceived as pictorially uniform [14]. The goal of this paper is to assess the ability of SGNG to approximate images while generating maps of hidden units somehow correlated to image segments considered uniform by a human photointerpreter. If this correlation exists, then the image segmentation problem would be addressed in a way that is new and interesting within the framework of complex system development.

The paper is organized as follows: first, the Karayannis and SGNG learning methods are summarized and the CHR criterion is presented. Next, an adaptation of CHR is proposed. Finally, experimental results obtained by applying SGNG to a test set of images are presented and discussed.

2 Scatter-partitioning RBF networks

In this section brief descriptions of the Karayiannis growing RBF network and SGNG algorithm are provided.

2.1 Karayiannis' learning scheme

For classification tasks, Karayiannis employs a GRBF network and a two-stage learning scheme where both learning stages, applied to the hidden and output layer respectively, are batch and error-driven. The Karayiannis learning scheme is presented below [6]:

1. Set $c = 2$. Initialize prototypes μ_{j}, $j = 1, ..., c$.
2. First learning stage.
   - (a) For all the training data, update prototypes μ_{j}, $j = 1, ..., c$, by means of a batch error-driven adaptation strategy (e.g., gradient descent of a cost function computed as the sum of class-conditional variances over all output classes and all the hidden units).
   - (b) For fixed prototypes μ_{j}, $j = 1, ..., c$, and for all the training data, update the spread parameters σ_{j}, $j = 1, ..., c$, by means of a batch error-driven adaptation strategy (e.g., gradient descent of a cost function computed as the sum of class-conditional variances over all output classes and all the hidden units).
3. Second learning stage.
   - (a) Initialize the weights of the output layer.
   - (b) For fixed prototypes and spread parameters, $(\mu_{\mathrm{j}}, \sigma_{\mathrm{j}})$, $j = 1, ..., c$, and for all the training data, update the weights of the output layer by means of a batch error-driven technique (e.g., gradient descent of the output sum-of-squares error). Note that Steps 2(a) to 3(b) form a so-called growing cycle.
   - (c) If a stopping criterion is satisfied, then stop. This stopping criterion detects overfitting as follows: after each growing cycle, a training and a testing error are computed and recorded. If the testing error does not decrease after a sufficient number of growing cycles, then the training is terminated while the network that resulted in the smallest error on the testing set is considered as the final product of the learning process. This is tantamount to saying that the proposed GRBF learning algorithm employs a back-tracking strategy to avoid overfitting. If the stopping criterion is not satisfied, then:

(d) Split one of the existing prototypes according to a supervised splitting criterion. For example, after presentation of the whole input data set (i.e., after one processing epoch), split the unit responsible of the highest number of misclassified patterns (note that before this step both structural parameters and output weights of the network are set).

(e) Set $c = c + 1$ and go to Step 2(a).

For each unit insertion the growing cycle, described by Steps 2(a) to 3(b), is repeated until convergence is reached. This procedure is computationally intensive, its computation time increasing as $\mathcal{O}(c \cdot n)$, where c is the size of the hidden layer and n is the size of the training data set.

2.2 SGNG learning scheme

For both classification and function regression tasks, SGNG employs a one-stage learning scheme where parameters in both the hidden and the output layer are trained simultaneously according to an on-line and error-driven (supervised) adaptation strategy. SGNG employs CHR to generate intra-layer connections. It has been proved that CHR guarantees topological preserving mapping [13]. The SGNG algorithm is presented below:

1. Set $c = 2$. Initialize prototypes $\mu_{\mathbf{j}}$ and output weights $\mathbf{w_j}$, $j = 1, ..., c$.

2. Select at random an input-output vector pair from the supervised training set.

3. Determine the winner, $w_1 \in \{1, c\}$, and the second-nearest unit, $w_2 \in \{1, c\}$, as the pair of hidden units featuring, respectively, the nearest and the second-nearest prototype to the input vector. If this inter-pattern distance is measured as the Euclidean distance, then receptive fields centered on prototype vectors are equivalent to Voronoi polyhedra [4], [13], [15].

4. According to CHR, if a synaptic link between units w_1 and w_2, identified as s_{w_1,w_2}, does not exist already, create it. Set (or reset) to zero the synaptic link-based local counter $age_{s_{w_1,w_2}}$, which is the age of synaptic link s_{w_1,w_2}. It is important to stress that synaptic links generated by CHR belong to an output graph or network of neural units; both synapses and neural units can be projected back onto the input space as a set of template vectors and inter-template connections [13].

5. Move prototype $\mu_{\mathbf{w_1}}$ toward the input pattern by a quantity equal to learning rate ϵ_a times the Euclidean distance between template $\mu_{\mathbf{w_1}}$ and the input vector.

6. The prototype of every hidden unit connected to the winner is moved toward the input pattern by a quantity equal to learning rate ϵ_b times the Euclidean distance between that prototype vector and the input vector (where $\epsilon_b << \epsilon_a$).

7. Increase by one the age of all synapses in the network of hidden units.

8. Remove all synapses whose age is above threshold $age_{s,max}$. If this results in units having no more emanating connection, then remove these hidden units as well.

9. If the number of input presentations is a multiple of parameter λ (equivalent to a number of processing steps), then split the hidden unit selected as the one featuring the highest value of its neuron-based (local) error counter (for more details about this splitting criterion, refer to the literature) [7], [8]. The output weights of the generated unit are initialized. Note that this neuron insertion policy allows the function estimator to average over the noise on the data (i.e., one noisy pattern, e.g., an outlier, which is poorly mapped does not cause a unit insertion), despite the fact that the learning strategy is on-line (i.e., although the network immediately reacts to an input presentation through parameter adaptation).

10. Decrease by a constant fraction, β (e.g., $\beta = 0.02$), the local error counter of all neurons, so that errors related to more recent signals are weighted stronger than previous ones.

11. In the input space, compute the spread parameters σ_{j}, $j = 1, ..., c$, as the mean euclidean length of all connections emanating from each prototype.

12. Compute the actual output of the GRBF network according to Eq. (1).

13. Compute the squared error between the actual output of the GRBF network and the desired (supervised) output.

14. Update the output weights of the GRBF network according to the delta rule, i.e., according to stochastic (on-line) gradient descent of the output squared error.

15. In function regression, add the computed output squared error to the local error counter of winner unit w_1, as each input vector is represented by its winner unit which provides the prototype nearest to the input pattern. In a classification task, the local error counter of the winner unit is increased by one if the input pattern is misclassified.

16. If a stopping criterion is met (e.g., network size is equal to user-defined parameter c_{max}) then stop; otherwise go to Step 2.

Note that SGNG requires six user-defined parameters to run: ϵ_a, ϵ_b, λ, β, c_{max}, $age_{s,max}$. The computational complexity of the algorithm increases as $\mathcal{O}(c_{max} \cdot \lambda)$, i.e., SGNG increases its computation time with the size of the network rather than with the size of the training set.

3 Constrained CHR based on perceptual grouping

The Gestaltist theory has long ago revealed the existence of innate perceptual mechanisms capable of organizing visual stimuli (observations) into perceived objects which are separated from their background [16]. For our purpose the problem of perceptual grouping can be described by considering the set of points shown in Fig. 1, hereafter referred to as Simpson's data set, consisting of 24 vectors [17]. Typically, different human observers would provide different partitions of the Simpson data set. This means that perceptual grouping is an ill-posed problem which allows different (subjective) solutions depending on the state of (subjective) prior world knowledge. This view is consistent with a Bayesian interpretation of the grouping percept. The difficulty, of course, is in specifying the prior world knowledge; some of it relates to low level visual processing based on obervations about intensity, color and texture features, but some of it also relates to higher processing levels involving symmetries of objects or object models [18].

Let us consider the GNG clustering algorithm, which is the unsupervised version of SGNG. GNG differs from the SGNG procedure described in Section 2.2 as follows: Steps 11 to 14 must be removed, while in Step 15 the computation of the supervised output squared error must be substituted with the computation of the input requantization error. The requantization error can be computed as the squared Euclidean distance between the current input pattern and the prototype vector of the winner unit (for more details, refer to the literature) [7], [8]. If parameter c_{max} is set to 24 (equal to the number of input patterns) and GNG is input with the data set shown in Fig. 1, then Fig. 2 is generated when convergence is reached. In Fig. 2, lines represent projections back onto the 2-D input space of connections linking neural units belonging to the output network, while squares depict the prototype vectors, equivalent to receptive field centers. Note that, in line with theoretical expectations, each prototype coincides with one separate input pattern. Fig. 2 clearly shows that GNG provided with CHR generates disjointed maps of prototypes that are not consistent with the results of perceptual grouping by a human observer. This is particularly evident for the point approximately located at the center of Fig. 1: this point is perceived as a cluster on its own by a human observer, while its matching prototype is linked to another group of prototype vectors in the clustering result depicted in Fig. 2.

To increase its consistency with perceptual grouping mechanisms, CHR is modified as suggested below. For a given input pattern, determine the winner unit, $w_1 \in \{1, c\}$, and the second-nearest unit, $w_2 \in \{1, c\}$, as those units featuring, respectively, the nearest and the second-nearest prototype to the input vector. If a synaptic link between units w_1 and w_2, s_{w_1,w_2}, whose length in the input space is identified as ls_{w_1,w_2}, does not exist already, then consider $ls_{w_1,min}$ as the length of the shortest connection currently emanating from prototype μ_{w_1} in the input space, and $ls_{w_2,min}$ as the corresponding length value related to prototype μ_{w_2}. If neural unit w_1 is currently isolated (i.e., there is no synaptic link emanating from it), then set $ls_{w_1,min} \to \infty$. Analogously, if neural unit w_2 is isolated, set $ls_{w_2,min} \to \infty$. If $ls_{w_1,w_2} \leq k \cdot \min\{ls_{w_1,min}, ls_{w_2,min}\}$, where $k \geq 1$ is a user-defined parameter, then: i) generate s_{w_1,w_2}; ii) if $ls_{w_1,w_2} < ls_{w_1,min}$, remove all connections from unit w_1 whose Euclidean length in the input space is bigger than k times ls_{w_1,w_2}; and iii) if $ls_{w_1,w_2} < ls_{w_2,min}$, remove all connections from unit w_2 whose Euclidean length in the input space is bigger than k times ls_{w_1,w_2}. This is tantamount to saying that the ratio between the length of the longest and shortest connection emanating from any template must always be $\leq k$. This connection generation policy is termed Constrained CHR (CCHR) [19]. When CCHR is adopted, slightly modified versions of GNG and SGNG must be developed, such that a new criterion for removing processing units replaces that described at Step 8 in Section 2.2. Our choice is to employ the following heuristic. At any processing step, a neuron-based age attribute, age_j, $j = 1, ..., c$, is set to zero for the winner, otherwise it is increased by one. Next, all neural units whose age is above threshold $age_{n,max}$ are removed.

A GNG version that employs CCHR in place of CHR is applied to the Simpson data set shown in Fig. 1. Input parameters are: $\lambda = 24$ (i.e., equal to the number of separate input patterns), $age_{s,max} = age_{n,max} = c_{max} = \lambda$, while parameter k is fixed equal to 1.2 and 1.6 respectively. In these two cases, after a few processing epochs, GNG generates Figs. 3 and 4 respectively. Although no single solution exists for the perceptual grouping problem, Figs. 3 and 4 are consistent with human visual percepts.

4 SGNG as an image segmentation algorithm

A digitized image, which is to say a discrete 2-D signal, is a finite data set equivalent to a surface defined on a 2-D input space. This means that exploitation of batch rather than on-line learning algorithms should be recommended for finding an approximating surface that provides the best fit to the image data points. On the other hand, CHR and CCHR may be applied only to on-line learning frameworks. Since SGNG should be capable of adapting the spread parameter of its hidden units on the basis of the localized spatial frequency content of the image (i.e., more GRBFs are expected to be positioned in the input space where the local spatial frequency content of the image increases), we expect that projections onto the input space (i.e., between prototypes) of the connections generated between hidden unit pairs may provide useful information for image segmentation, i.e., about the location of image areas perceived as pictorially uniform by a human observer. To verify how well SGNG performs as image segmenter we consider some synthetic images for testing.

It is known that contour pixels belong to: *edges* (step or ramp edges), *ridges* (e.g., a line represents a narrow ridge), *roofs*, or to a combination of these structures [20]. These image features are shown in Figs. 5 to 8, where images consist of 50×50 pixels. According to the psychophysical phenomenon of the Mach bands, which is one of the best known brightness illusions, when a luminance (radiance, intensity) ramp meets a plateau there is a spike of brightness (i.e., perceived luminance), although there is continuity in the luminance profile [20]. This phenomenon should be consistent with an ideal behavior of SGNG provided with CCHR in approximating the surfaces depicted in Figs. 5 to 8. In this case, projections onto the input space of maps of processing units generated by SGNG should look like those depicted in Figs. 9 to 12.

5 Experimental results

SGNG provided with CCHR is run over images shown in Figs. 5 to 8. Input parameters are: $\epsilon_a = 0.05$, $\epsilon_b = 0.0005$, $\lambda = 30$ (i.e., during each processing epoch, consisting of $50 \times 50 = 2500$ patterns, up to 83 hidden units can be generated), $age_{n,max} = age_{s,max} = 2500$, $c_{max} = 200$, $k = 1.6$, processing epochs $= 10$. Approximated surfaces corresponding to Figs. 5 to 8 are shown in Figs. 13 to 16 respectively. These results are largely unsatisfactory. Corresponding SGNG-generated actual maps are shown in Figs. 17 to 20. These maps are very different from the ideal ones depicted in Figs. 9 to 12.

6 Discussion and conclusions

Implementation errors may be one possible cause of the very poor behavior shown by SGNG as both function approximator and image segmenter. The fact that GNG, which is the core of SGNG, has been widely tested before completing the SGNG architecture actually reduces the chance that dramatic implementation errors have affected SGNG outcomes. Unfortunately, we are not aware of analogous function approximation results that have been presented in the literature to refer to for comparison.

Aside from implementation errors, SGNG unsatisfactory performance as both function approximator and image segmenter may be due to an excessive degree of dynamicity of the learning algorithm, i.e., parallel updating of structural parameters and output weights seems to lead to inconsistent learning behaviors, at least when SGNG is involved with function approximation tasks. One solution may be found by adopting two-stage, batch, error-driven learning strategies as those implemented in the Karayiannis learning algorithm proposed in Section 2.1. How to insert in such a scheme the on-line CHR or CCHR policy capable of augmenting the expressive power of the model by introducing a competitive mechanism among synaptic links may be the subject of further research. Additional interest may be focused on hierarchical grid-partitioning RBF networks [9], that should be related to wavelets and filter banks theory [21].

References

1. C. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford (UK), 1995.
2. J. Moody, and C. Darken, “Fast learning in networks of locally-tuned processing units,” *Neural Computation*, **1**, pp. 281-294, 1989.
3. V. Cherkassky, and F. Mulier, *Learning from data*, Wiley, New York, 1998.
4. B. Fritzke, “Incremental neuro-fuzzy systems,” *Proc. SPIE's Optical Science, Engineering and Instrumentation '97: Applications of Fuzzy Logic Technology IV*, San Diego, CA, July 1997.
5. E. Alpaydın, Soft vector quantization and the EM algorithm. *Neural Networks*, in press, 1998.
6. N. Karayiannis, “Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques,” *IEEE Trans. on Neural Neworks*, in press, 1998.
7. B. Fritzke, “Growing cell structures - A self-organizing network for unsupervised and supervised learning,” *Neural Networks*, **7**(9), pp. 1441-1460, 1994.
8. B. Fritzke, “Some competitive learning methods,” draft document, http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG, 1998.
9. N. A. Borghese, and S. Ferrari, “Hierarchical RBF networks and local parameter estimate,” *Neurocomputing*, in press, 1998.
10. T. Masters, *Signal and image processing with neural networks - A C++ sourcebook*, Wiley, New York, 1994.
11. J. Buhmann, “Learning and data clustering,” in *Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed., Bradford Books / MIT Press, 1995.

12. B. Fritzke, "A self-organizing network that can follow non-stationary distributions," *Proc. of the International Conference on Artificial Neural Networks '97*, Springer, 1997, pp. 613-618.

13. T. Martinetz, G. Berkovich, and K. Schulten, "Topology representing networks," *Neural Networks*, **7**(3), pp. 507-522, 1994.

14. D. Marr, *Vision*, Freeman, New York, 1982.

15. T. Mitchell, *Machine learning*, McGraw-Hill, New York, 1997.

16. M. Wertheimer, "Laws of organization in perceptual forms" (partial translation), in *A Source-book of Gestalt Psychology*, pp. 71-88, Harcourt, Brace and Company, 1938.

17. P. Simpson, "Fuzzy min-max neural networks - Part 2: clustering," *IEEE Trans. Fuzzy Systems*, **1**(1), pp. 32-45, 1993.

18. J. Shi, and J. Malik, "Normalized cuts and image segmentation," *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997.

19. A. Baraldi, and F. Parmiggiani, "Novel neural network model combining radial basis function, competitive Hebbian learning rule, and fuzzy simplified adaptive resonance theory," *Proc. SPIE's Optical Science, Engineering and Instrumentation '97: Applications of Fuzzy Logic Technology IV*, San Diego, CA, July 1997, vol. 3165, pp. 98-112.

20. D. Burr, and M. C. Morrone, "A nonlinear model of feature detection," in *Nonlinear Vision: Determination of Neural Receptive Fields, Functions, and Networks*, R. B. Pinter and N. Bahram, Eds., pp. 309-327, CRC Press, Boca Raton, 1992.

21. G. Strang, and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley (MA), 1997.

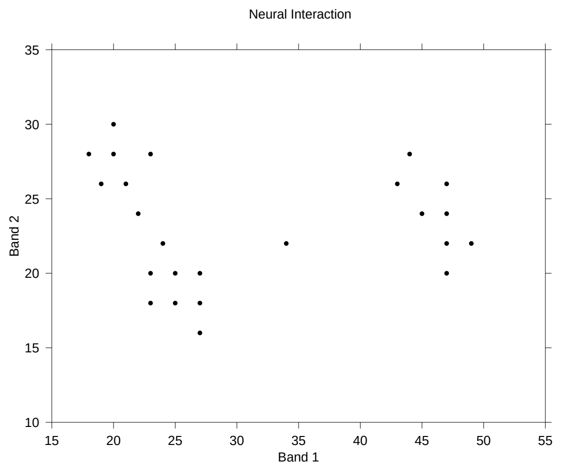

Figure 1: Simpson's data set consisting of 24 points.

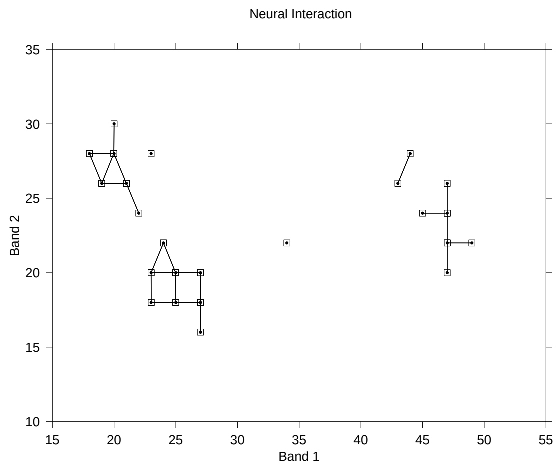

Figure 3: GNG processing of the Simpson data set when CCHR is employed: parameter $k = 1.2$.

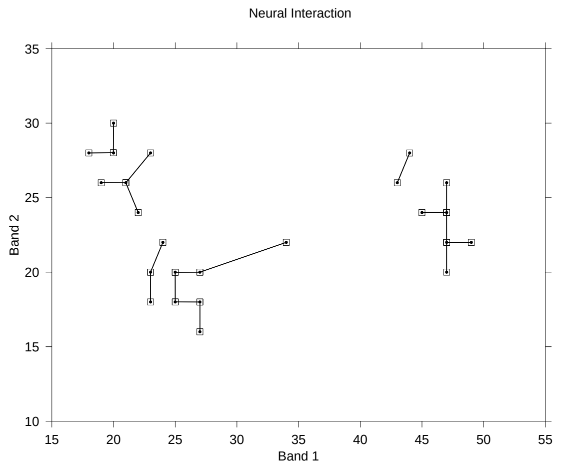

Figure 2: GNG processing of the Simpson data set when CHR is employed.

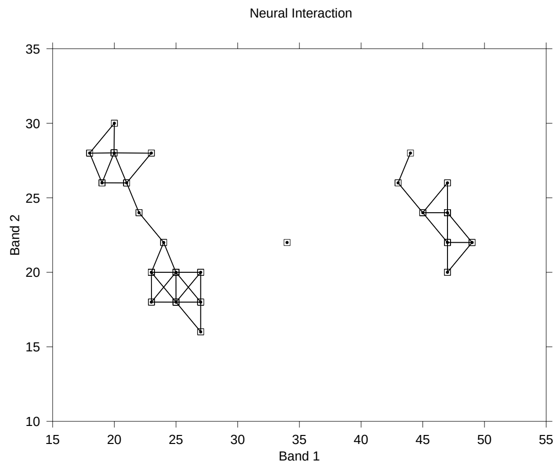

Figure 4: GNG processing of the Simpson data set when CCHR is employed: parameter $k = 1.6$.

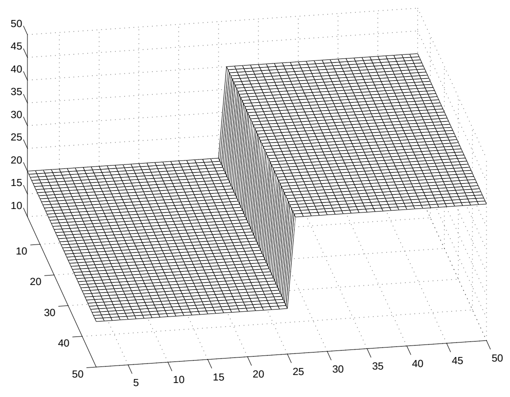

Figure 5: Step edge.

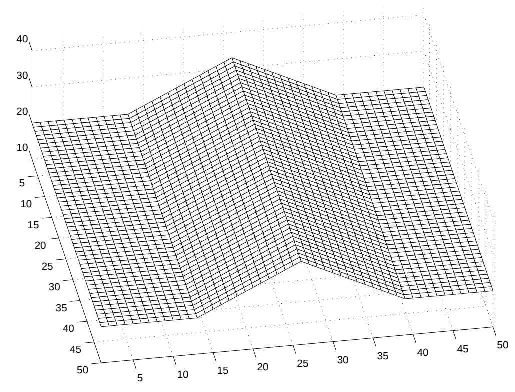

Figure 8: Roof.

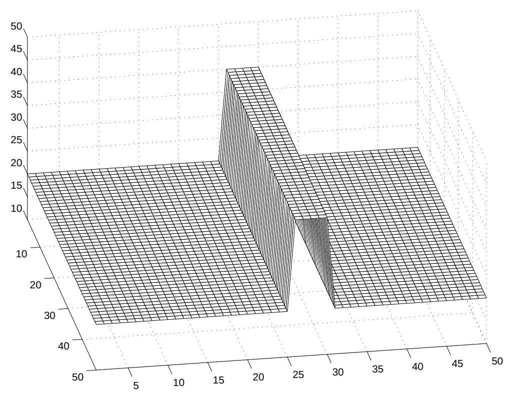

Figure 6: Ridge.

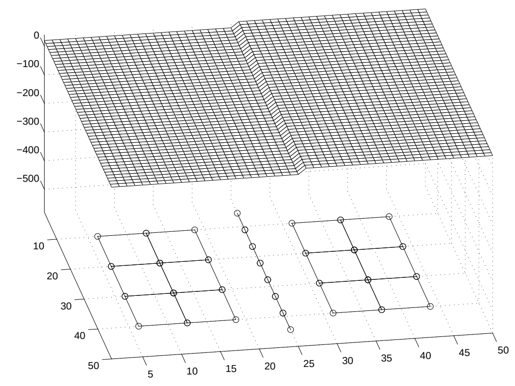

Figure 9: SGNG ideal mapping of the step edge.

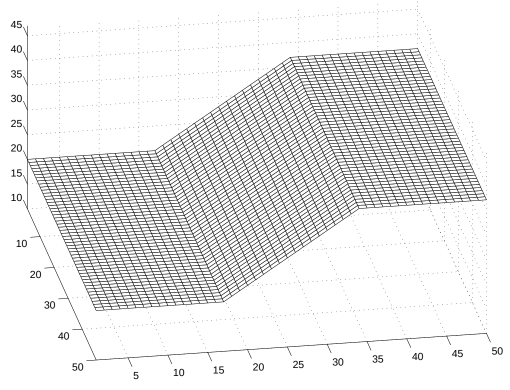

Figure 7: Ramp.

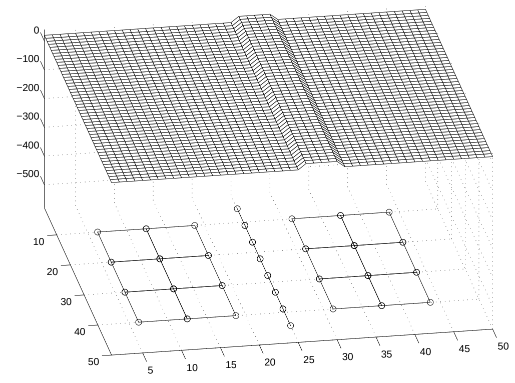

Figure 10: SGNG ideal mapping of the ridge.

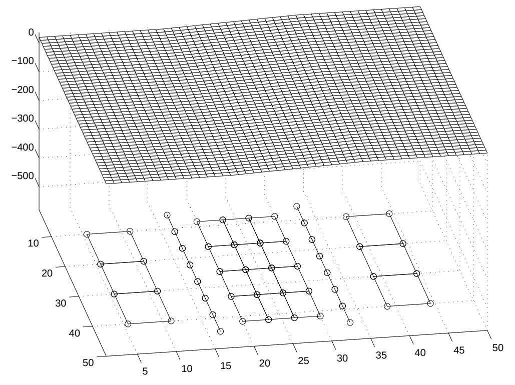

Figure 11: SGNG ideal mapping of the ramp.

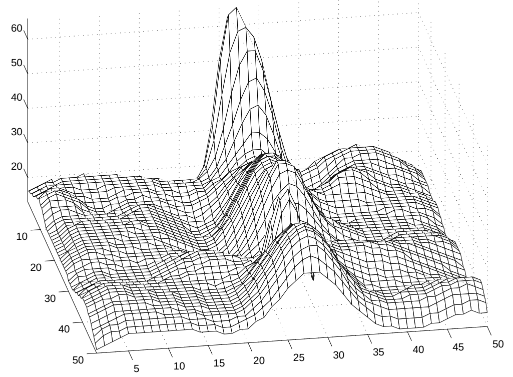

Figure 14: SGNG reconstruction of the ridge.

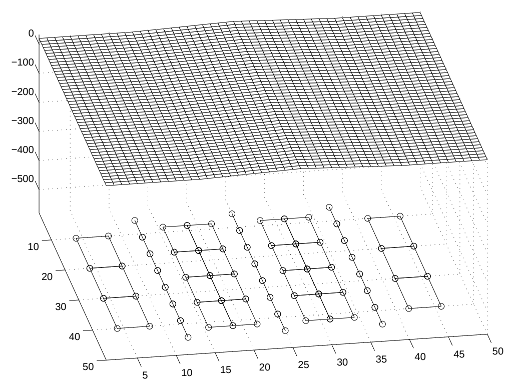

Figure 12: SGNG ideal mapping of the roof.

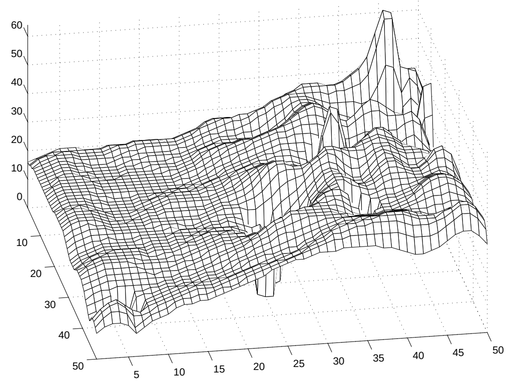

Figure 15: SGNG reconstruction of the ramp.

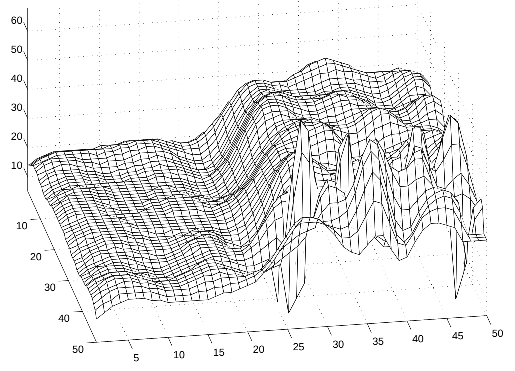

Figure 13: SGNG reconstruction of the step edge.

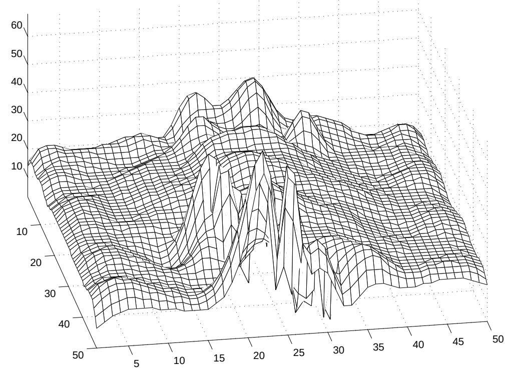

Figure 16: SGNG reconstruction of the roof.

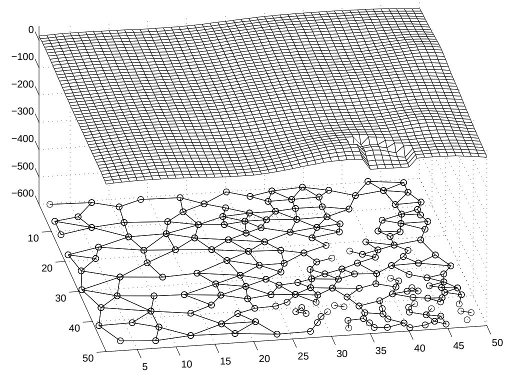

Figure 17: SGNG actual mapping of the step edge.

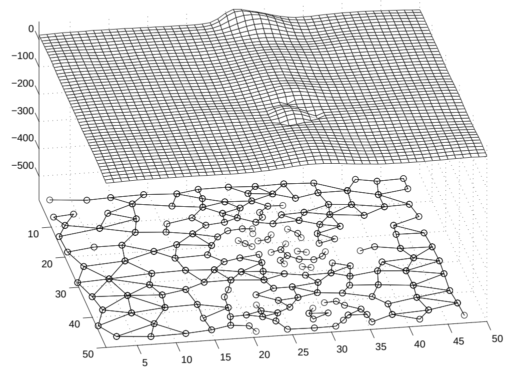

Figure 18: SGNG actual mapping of the ridge.

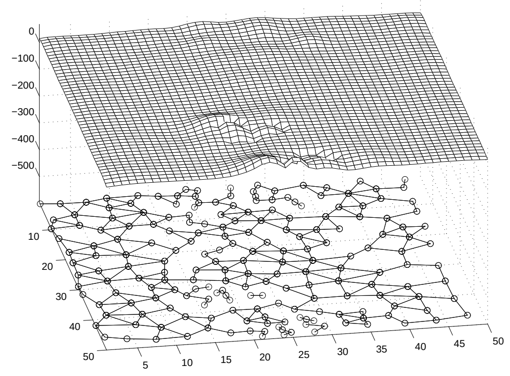

Figure 20: SGNG actual mapping of the roof.

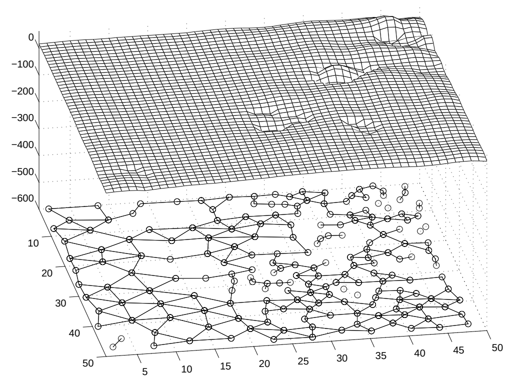

Figure 19: SGNG actual mapping of the ramp.