Tuning Parameters in Fuzzy Growing Hierarchical Self-Organizing Networks

Miguel Arturo Barreto-Sanz[1,2], Andrés Pérez-Uribe[2], Carlos-Andres Peña-Reyes[2], and Marco Tomassini[1]

Abstract. Hierarchical Self-Organizing Networks are used to reveal the topology and structure of datasets. These methodologies create crisp partitions of the dataset producing tree structures composed of prototype vectors, permitting the extraction of a simple and compact representation of a dataset. However, in many cases observations could be represented by several prototypes with certain degree of membership. Nevertheless, crisp partitions are forced to classify observations in just one group, losing information about the real dataset structure. To deal with this challenge we propose Fuzzy Growing Hierarchical Self-Organizing Networks (FGHSON). FGHSON are adaptive networks which are able to reflect the underlying structure of the dataset in a hierarchical fuzzy way. These networks grow by using three parameters which govern the membership degree of data observations to the prototype vectors and the quality of the hierarchical representation. However, different combinations of values of these parameters can generate diverse networks. This chapter explores how these combinations affect the topology of the

Miguel Arturo Barreto-Sanz
Université de Lausanne, Hautes Etudes Commerciales (HEC), Institut des Systèmes d'Information (ISI)
e-mail: Miguel-Arturo.Barreto-Sanz@heig-vd.ch

Andrés Pérez-Uribe
University of Applied Sciences of Western Switzerland (HEIG-VD)(REDS)
e-mail: andres.perez-uribe@heig-vd.ch

Carlos-Andres Peña-Reyes
University of Applied Sciences of Western Switzerland (HEIG-VD)(REDS)
e-mail: Carlos.Pena@heig-vd.ch

Marco Tomassini
Université de Lausanne, Hautes Etudes Commerciales (HEC), Institut des Systèmes d'Information (ISI)
e-mail: Marco.Tomassini@unil.ch

L. Franco et al. (Eds.): Constructive Neural Networks, SCI 258, pp. 261–279.
springerlink.com

network and the quality of the prototypes; in addition the motivation and the theoretical basis of the algorithm are presented.

1 Introduction

We live in a world full of data. Every day we are confronted with the handling of large amounts of information. This information is stored and represented as data, for further analysis and management. One of the essential means in dealing with data is to classify or group it into categories or clusters. In fact, as one of the most ancient activities of human beings [1], classification plays a very important role in the history of human development. In order to learn a new object or distinguish a new phenomenon, people always try to look for the features that can describe it and further compare it with other known objects or phenomena, based on the similarity or dissimilarity, generalized as proximity, according to some standards or rules.

In many cases classification must be done without a priori knowledge of the classes in which the dataset is divided (unlabeled pattern). This kind of classification is called clustering (unsupervised classification). On the contrary, discriminant analysis (supervised classification) is made by providing a collection of labeled patterns; so the problem is to label a newly encountered, unlabeled pattern. Typically, the given labeled patterns are used to learn descriptions of classes which in turn are used to label a new pattern. In the case of clustering, the problem is to group a given collection of unlabeled patterns into meaningful clusters. In a sense, labels are associated with clusters also, but these category labels are data driven; that is, they are obtained solely from the data [15, 23].

Even though the unsupervised classification presents many advantages over supervised classification[1], it is a subjective process in nature. As pointed out by Backer and Jain [2], "in cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i.e., chosen subjectively based on its ability to create "interesting" clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups". Clustering algorithms partition data into a certain number of clusters (groups, subsets, or categories). There is no universally agreed upon definition [8].

Thus, methodologies to evaluate clusters with different levels of abstraction in order to find "interesting" patterns are useful; these methodologies could help to improve the analysis of cluster structure creating representations, facilitating the selection of clusters of interest. Methods for tree structure

[1] For instance, no extensive prior knowledge of the dataset is required, and it can detect "natural" groupings in feature space.

representation and data abstraction have been used for this task, revealing the topology and organization of clusters.

On the one hand, hierarchical methods are used to help explain the inner organization of datasets, since the hierarchical structure imposed by the data produces a separation of clusters that is mapped onto different branches. Hierarchical clustering algorithms organize data into a hierarchical structure according to a proximity matrix. The results of Hierarchical clustering are usually depicted by a binary tree or dendrogram. The root node of the dendrogram represents the whole data set and each leaf node is regarded as a data object. The intermediate nodes describe to what extent the objects are proximal among them; and the height of the dendrogram usually expresses the distance between each pair of objects or clusters, or an object and a cluster. The ultimate clustering results can be obtained by cutting the dendrogram at different levels. This representation provides very informative descriptions and visualization for the potential data clustering structures, especially when real hierarchical relations exist in the data, like the data from evolutionary research on different species of organisms. Therefore, this hierarchical organization enables us to analyze complicated structures as well as the exploration of the dataset at multiple levels of detail [23].

On the other hand, data abstraction permits the extraction of a simple and compact representation of a data set. Here, simplicity is either from the perspective of automatic processing (so that a machine can perform further processing efficiently) or is human-oriented (so that the representation obtained is easy to comprehend and intuitively appealing). In the clustering context, a typical data abstraction is a compact description of each cluster, usually in terms of cluster prototypes or representative patterns such as the centroid of the cluster [7]. Soft competitive learning methods [11] are employed on data abstraction in a self-organizing way. These algorithms attempt to distribute a number of vectors (prototype vectors) in a potentially low-dimensional space. The distribution of these vectors should reflect (in one of several possible ways) the probability distribution of the input signals which in general is not given explicitly but through sample vectors. Two principal approaches have been used for this purpose. The first is based on a fixed network dimensionality (i.e. Kohonen maps [16]). In the second approach, non fixed dimensionality is imposed on the network; hence, this network can automatically find a suitable structure and size through a controlled growth process [19].

Different approaches have been introduced in order to combine the capabilities of tree structure of the hierarchical methods and the advantages of soft competitive learning methods used for data abstraction [20, 13, 6, 22, 12, 18], obtaining networks capable of representing the structure of clusters and their prototypes in a hierarchical self-organizing way. These networks are able to grow and adapt their structure in order to represent the characteristics of clusters in the most accurate manner. Although these hybrid models provide satisfactory results, they generate crisp partitions of the datasets. The crisp

segmentations tend to allocate elements of the dataset in just one branch of the tree in each level of the hierarchy and assign just one prototype to represent one cluster, so the membership to other branches or prototypes is zero. Nevertheless, in many applications crisp partitions in hierarchical structures are not the optimal representation of the clusters, since some elements of the dataset could belong to multiple clusters or branches with a certain degree of membership.

One example of this situation is presented in Geographic Information Systems (GIS) applications. One of the topics treated by GIS researchers refers to the classification of geographical zones with similar characteristics to climate, soil and terrain (conditions relevant to agricultural production) in order to create the so called agro-ecological zones (AEZ) [9]. AEZ provide the frame for various applications, such as quantification of land productivity, estimation of land's population supporting capacity, and optimization of land resource use and development. Many institutions, governments and enterprises need to know which AEZ a particular region belongs to (allocating the region to a certain AEZ cluster), in order to apply policies to invest, for instance in new cropping systems for economic viability, and sustainability. However, the geographical region of interest can vary in range of resolution depending on the application or context (i.e. countries, states, cities, parcels). In addition, the fuzzy and implicit nature of the geographic zones (in which geographical boundaries are not hard, but rather soft boundaries) transform the boundaries of the AEZ in zones of transition rather than sharp boundaries. Thus, the soft boundaries make it possible that regions in the middle of two AEZ have membership of both. The clustering method to deal with this situation has to provide views of AEZ at multiple levels, preferably in a hierarchical way. In addition, it should be capable of discovering fuzzy memberships of geographical regions to the AEZ.

For the purpose of representing degrees of membership, fuzzy logic is a feature that could be added to hierarchical self-organized hybrid models. We propose, thus Fuzzy Growing Hierarchical Self-Organizing Networks (FGHSON), with the intention of synergistically combining the advantages of Self-Organizing Networks, hierarchical structures, and fuzzy logic. FGHSON are designed to improve the analysis of datasets where it is desirable to obtain a fuzzy representation of a dataset in a hierarchical way, then discovering its structure and topology. This new model will be able to obtain a growing hierarchical structure of the dataset in a self-organizing fuzzy manner. This kind of network is based on the Fuzzy Kohonen Clustering Networks (FKCN) [4] and Hierarchical Self-Organizing Structures (HSS) [17, 21, 20, 22].

This book chapter is organized as follows: In the next section the Hierarchical Self-Organizing Structures and the Fuzzy Kohonen Clustering Networks will be explained then, our model will be described. Section 3 focuses on the application of the methodology using the Iris benchmark and an example dataset, a further example where model parameters are tuned is also

presented. Finally, in Section 4 conclusions are drawn and future extensions of the work described.

2 Methods

2.1 Hierarchical Self-Organizing Structures

The ability to obtain hierarchically structured knowledge from a dataset using autonomous learning has been widely used in many areas. This is due to the fact that hierarchical self-organizing structures permit unevenly distributed real-world data to be represented in a suitable network structure, during an unsupervised training process. These networks capture the unknown data topology in terms of hierarchical relationships and cluster structures.

Different methodologies have been presented in this area with various approaches. It is possible to classify hierarchical self-organizing structures in two classes taking into account the algorithm of self-organization used. The first family of models is based on Kohonen self-organizing maps (SOM), and the second on Growing Cell Structures (GCS) [10].

With respect to approaches based on GCS, Hierarchical Growing Cell Structures (HiGCS) [5], TreeGCS [13] and the Hierarchical topological clustering (TreeGNG) [6] have been proposed. The algorithms derived from GCS are based on periodic node deletion, node activity and the volume of the input space classified by the node. This approach tends to represent examples with high occurrence rates, and therefore takes low frequency examples as outliers or noise. As a result, examples with low presence rates are not represented in the model. Nevertheless, in many cases it is desirable to discover novelties in the dataset, so taking into account the observations with low occurrence rates could allow discovery of those exceptional behaviors.

For this reason, we focused our research on approaches based on SOM [17, 21, 20], particularly the Growing Hierarchical Self-Organizing Map (GHSOM)[22] due to its ability to take into account the observations with low presence rates as part of the model. This is possible since the hierarchical structure of the GHSOM is adapted according to the requirements of the input space. Therefore, areas in the input space that require more units for appropriate data representation create deeper branches than others. This process is done without eliminating nodes that represent examples with low occurrence rates.

2.2 Fuzzy Kohonen Clustering Networks

FKCN [4] integrate the idea of fuzzy membership from Fuzzy C-Means (FCM), with the updating rules of SOM. Thus, creating a self-organizing algorithm that automatically adjusts the size of the updated neighborhood

during a learning process, which usually terminates when the FCM objective function is minimized. The update rule for the FKCN algorithm can be given as:

$$W_{i,t} = W_{i,t-1} + \alpha_{ik,t}(Z_k - W_{i,t-1});\ for\ k = 1, 2, ..., n;\ for\ i = 1, 2, ..., c \quad (1)$$

where $W_{i,t}$ represents the centroid[2] of the i^{th} cluster at iteration t , Z_k is the k^{th} vector example from the dataset and α_{ik} is the only parameter of the algorithm and according to [14]:

$$\alpha_{ik,t} = (U_{ik,t})^{m(t)} \quad (2)$$

Where $m(t)$ is an exponent like the fuzzification index in FCM and $U_{ik,t}$ is the membership value of the compound Z_k to be part of cluster i. Both of these constants vary at each iteration t according to:

$$U_{ik} = \left(\sum_{j=1}^{c} \left(\frac{\|Z_k - W_i\|}{\|Z_k - W_j\|} \right)^{2/(m-1)} \right)^{-1} \quad ;\ 1 \le k \le n\ ;\ 1 \le i \le c \quad (3)$$

$$m(t) = m0 - m\Delta \cdot t \quad ; \quad m\Delta = (m0 - m_f)/iterate\ limit \quad (4)$$

Where $m0$ is a constant value greater than the final value (m_f) of the fuzzification parameter m. The final value m_f should not be less than 1.1, in order to avoid a divide by zero error in equation (3). The iterative process will stop if $\left\|W_{i,(t)} - W_{(i,t-1)}\right\|^2 < \epsilon$, where ϵ is a termination criterion or after a given number of iterations. At the end of the process, a matrix U is obtained, where U_{ik} is the degree of membership of the Z_k element of the dataset to the cluster i. In addition, the centroid of each cluster will form the matrix W where W_i is the centroid of the i^{th} cluster. The FKCN algorithm is given below:

1. Fix c, and $\epsilon > 0$ to some small positive constant.
2. Initialize $W_0 = (W_{1,0}, W_{2,0}, \cdots, W_{c,0}) \in \Re^c$.
   Choose $m_0 > 1$ and $t_{max} = max.\ number\ of\ iterations.$
3. For $t = 1, 2, \cdots, t_{max}$
   a. Compute all cn learning rates $\alpha_{ik,t}$ with equations (2) and (3).
   b. Update all c weight vectors $W_{i,t}$ with
   $W_{i,t} = W_{i,t-1} + [\sum_{k=1}^{n} \alpha_{ik,t}(Z_k - W_{i,t-1})] \,/\, \sum_{j=1}^{n} \alpha_{ij,t}$
   c. Compute $E_t = \left\|W_{i,(t)} - W_{(i,t-1)}\right\|^2 = \sum_{i=1}^{c} \left\|W_{i,(t)} - W_{(i,t-1)}\right\|^2$
   d. If $E_t < \epsilon$ stop.

[2] In the perspective of neural networks it represents a neuron or a prototype vector. So the number of neurons or prototype vectors will be equal to the number of clusters.

2.3 Fuzzy Growing Hierarchical Self-Organizing Networks

Fuzzy Growing Hierarchical Self-Organizing Networks (FGHSON) are based on a hierarchical fuzzy structure of multiple layers, where each layer consists of several independent growing FKCNs. This structure can grow by means of an unsupervised self-organizing process in two manners (inspired by [22]):

a. Individually, in order to find the more suitable number of prototypes (which compose a FKCN) that may represent in an accurate manner the input dataset.
b. On groups of FKCNs in a hierarchical mode, permitting the hierarchy to reveal a particular set of characteristics of data.

Both growing processes are modulated by three parameters that regulate the breadth (growth of the layers), depth (hierarchical growth) and membership degree of data to the prototype vectors.

The FGHSON works as follows:

1) Initial Setup and Global Network Control
The main motivation of the FGHSON algorithm is to properly represent a given dataset. The quality of this representation is measured in terms of the difference between a prototype vector and the example vectors represented by this. The *quantization error qe* is used for this purpose. The *qe* measures the dissimilarity of all input data mapped onto a particular prototype vector, hence it can be used to guide a growth process with the aim of achieving an accurate representation of the dataset reducing the *qe*. The *qe* of a prototype vector W_i is calculated according to (5) as the mean Euclidean distance between its prototype and the input vectors Z_c that are part of the set of vectors C_i mapped onto this prototype.

$$qe_i = \sum_{Z_c \in Ci} \|W_i - Z_c\| \,;\; C_i \neq \phi \tag{5}$$

The first step of the algorithm is focused on obtaining a global measure that allows us to know the nature of the whole dataset. For this purpose the training process begins with the computation of a global measure of error qe_0. qe_0 represents the *qe* of the single prototype vector W_0 that forms the layer 0, see figure 1(a), calculated as shown in (6). Where, Z_k represents the input vectors from the whole data set Z and W_0 is defined as a prototype vector $W_0 = [\mu_{0_1}, \mu_{0_2}, \ldots, \mu_{0_n}]$, where μ_{0_i} for $i = 1, 2, \ldots, n$; is computed as the average of μ_{0_i} in the complete input dataset. In other words W_0 is a vector that corresponds to the mean of the input variables.

$$qe_0 = \sum_{Z_k \in Z} \|W_0 - Z_k\| \tag{6}$$

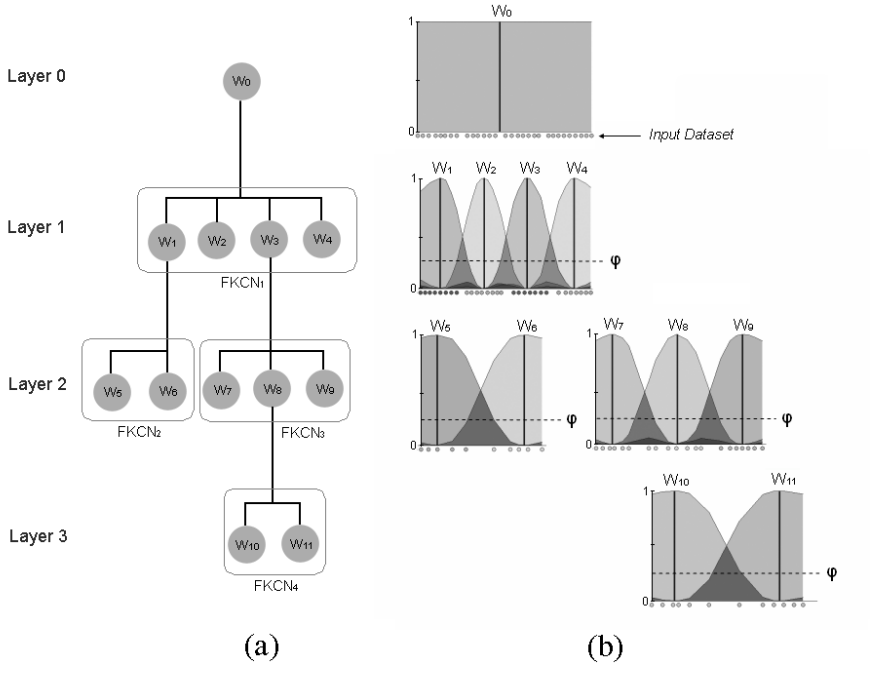

Fig. 1 (a) Hierarchical structure showing the prototype vectors and FKCNs created in each layer for a supposed case. (b) Membership degrees in each layer, corresponding to the network shown in the diagram. The parameter φ (the well known $\alpha - cut$) represents the minimal degree membership of an observation to be part of the dataset represented by a prototype vector, the group of data with a desired membership to a prototype will be used for the training of a new FCKN in the next layer (depth process). In this particular diagram the dataset is unidimensional (represented by the small circles below the membership plot) in order to simplify the example

The value of qe_0 will help to measure the minimum quality of data representation of the prototype vectors in the subsequent layers. Succeeding prototypes have the task of reducing the global representation error qe_0.

2) Breadth growth process

The construction of the first layer starts after the calculation of qe_0. This first layer consists of a FKCN ($FKCN_1$) with two initial prototype vectors. The growth process of the $FKCN_1$ begins by adding a new prototype vector and training it until a suitable representation of the dataset is achieved. Each of these prototype vectors is an n-dimensional vector W_i (with the same dimensionality as the input patterns), which is initialized with random values. The $FKCN_1$ is trained as shown in section 2.2, taking as input (in the exceptional case of the first layer) the whole dataset. More precisely, the $FKCN_1$ is allowed to grow until the qe of the prototype for its preceding

layer (qe_0 in the case of layer 1) is reduced to at least a fixed percentage τ_1. Continuing with the creation of the first layer, the number of prototypes in the $FKCN_1$ will be adapted. To achieve this, the *mean quantization error of the map* (MQE) is computed according to expression (7), where d refers to the number of prototype vectors contained in the FKCN, and qe_i represents the quantization error of the prototype W_i.

$$MQE_m = \frac{1}{d} \cdot \sum_i qe_i \tag{7}$$

The MQE is evaluated using (8) to measure the quality of data representation, and is used also as stopping criterion for the growing process of the FKCN. In (8) qe_u represents the qe of the corresponding prototype u in the upper layer. In the specific case of the first-layer, the stopping criterion is shown in (9).

$$MQE < \tau_1 \cdot qe_u \tag{8}$$

$$MQE_{layer1} < \tau_1 \cdot qe_0 \tag{9}$$

If the stopping criterion (8) is not fulfilled, it is necessary to aggregate more prototypes for a more accurate representation. For this aim, the prototype with the highest qe is selected and is denoted as the error prototype e. A new prototype is inserted in the place where e was computed. After the insertion, all the FKCN parameters are reset to the initial values (except for the values of the prototype vectors) and the training begins according to the standard training process of FKCN. Note that the same value of the parameter τ_1 is used in each layer of the FGHSON. Thus, at the end of the process, a layer 1 is obtained with a $FKCN_1$ formed by a set of prototype vectors W, see figure 1(a). In addition, a membership matrix U is obtained. This matrix contains the membership degree of the dataset elements to the prototype vectors, as explained in section 2.2.

3) Depth growth process

As soon as the breadth process of the first layer is finished, its prototypes are examined for further growth (depth growth or hierarchical growth). In particular, those prototypes with a large quantization error will indicate which clusters need a better representation by means of new FKCNs. The new FKCNs form a second layer, for instance W_1 and W_3 in figure 1(a). The selection of these prototypes is regulated by qe_0 (calculated previously in step 1) and a parameter τ_2 which is used to describe the desired level of granularity in the data representation. More precisely, each prototype W_i in the first layer that does not fulfill the criterion given in expression (10) will be subject to hierarchical expansion.

$$qe_i < \tau_2 \cdot qe_0 \tag{10}$$

After the expansion process and creation of the new FKCNs, the breadth process described in stage 2 begins with the newly established FKCNs, for instance, $FKCN_2$ and $FKCN_3$ in figure 1(a). The methodology for adding new prototypes, as well as the termination criterion of the breadth process, is essentially the same as used in the first layer. The difference between the training processes of the FKCNs in the first layer and all subsequent layers, is that only a fraction of the whole input data is selected for training. This portion of data will be selected according to a minimal membership degree (φ). This parameter φ (an $\alpha - cut$) represents the minimal degree of membership for an observation to be part of the dataset represented by a prototype vector. Hence, φ is used as a selection parameter, so all the observations represented by W_i have to fulfill expression (11), where U_{ik} is the degree of membership of the ${Z_k}^{th}$ element of the dataset to the cluster i. As an example, figure 1(b) shows the membership functions of the FKCNs in each layer, and how φ is used as a selection criteria to divide the dataset.

$$\varphi < U_{ik} \tag{11}$$

At the end of the creation of layer two, the same procedure described in step 2 is applied to build layer 3 and so forth.

The training process of the FGHSON is terminated when no prototypes require further expansion. Note that this training process does not necessarily lead to a balanced hierarchy, i.e., a hierarchy with equal depth in each branch. Rather, the specific distribution of the input data is modeled by a hierarchical structure, where some clusters require deeper branching than others.

3 Experimental Testing

3.1 Iris Data Set

In this experiment the Iris dataset[3] is used in order to show the adaptation of the FGHSON to those areas where an absolute membership to a single prototype is not obvious. Therefore, FGHSON must (in an unsupervised manner) look at the representation of the dataset on the areas where observations of the same category share similar zones. For instance in the middle of the data cloud formed by the Virginica and Versicolor observations (see figure 2(a)).

The parameters of the algorithm were set to $\tau_1 = 0.2$, $\tau_2 = 0.03$, and $\varphi = 0.2$. After training, a structure of four layers was obtained. The zero layer is used to measure the whole deviation of the dataset as was presented in section 2.3. The first layer consist of a FKCN with three prototype vectors as shown in figure 2(b), this distribution of prototypes aim to represent three Iris

[3] There are three categories in the data set : Iris Setosa, Iris Versicolor and Iris Virginical. Each having 50 observations with four features: sepal length (SL), sepal width (SW), petal length (PL), and petal width (PW).

categories. The second layer (figure 2(c)) reaches a more fine-grained description of the dataset, placing prototypes in almost all of the data distribution, adding prototypes in the zones where more representation was needed. Finally in figure 2(d), it is possible to observe an over population of prototypes in the middle of the cloud of Virginica and Versicolor observations. This occurs because this part of the dataset presents observations with ambiguous membership in the previous layer, then, several prototypes are placed in this new layer for proper representation. Hence, permitting those observations to obtain a higher membership of its new prototypes. The outcome of the process is a more accurate representation of this zone.

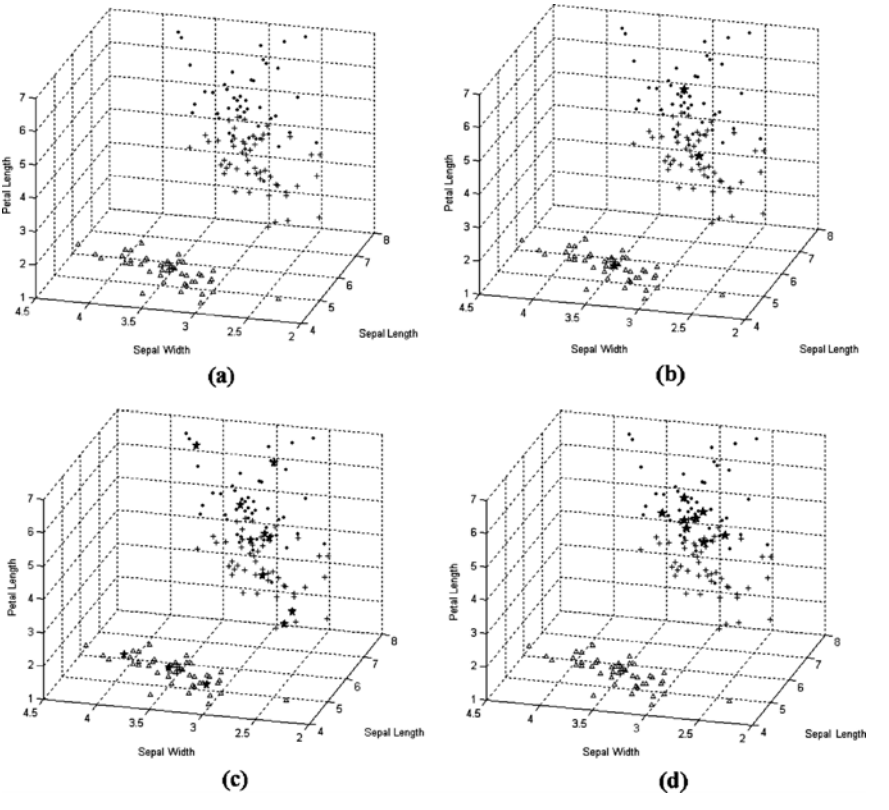

Fig. 2 Distribution of the prototype vectors, represented by stars, in each layer of the hierarchy. (a) Iris data set. There are three Iris categories: Setosa, Versicolor, and Virginica represented respectively by triangles, plus symbols, and dots. Each has 50 samples with 4 features. Here, only three features are used: PL, SW, and SL. (b) First layer (c) Second layer and (d) Third layer of the FGHSON, in this layer prototypes are presented only in the zone where observations of Virginica and Vesicolor share the same area, so the new prototypes represent each category in a more accurate manner.

3.2 Example Set

An exmaple set, as presented by Martinez et al [19] is used in order to show the capabilities of the FGHSON to represent a dataset that has multiple dimensionalities. In addition, it is possible to illustrate how the model stops the growing process in those parts where the desired representation is reached and keep growing where a low membership or poor representation is present. The parameters of the algorithm were set to $\tau_1 = 0.3$, $\tau_2 = 0.065$, and $\varphi = 0.2$. Four layers were created after training the network. In figure 3(a) the first layer is shown, in this case seven prototypes were necessary to represent the dataset at this level, one for the 1D oval, one for the 2D plane and five for the 3D parallelepiped (note that there are no prototypes clearly associated to the line).

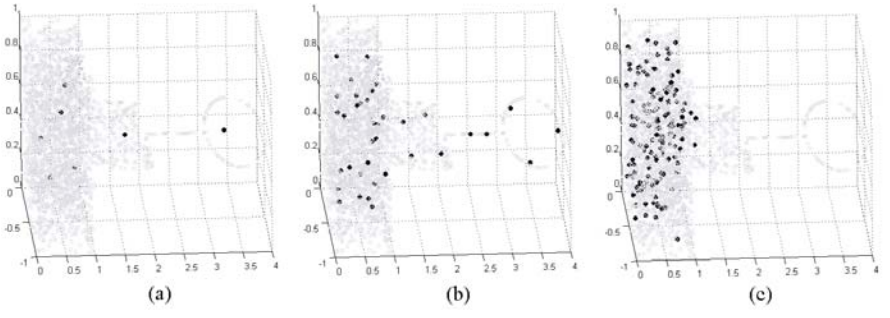

Fig. 3 Distribution of the prototype vectors (represented by black points) (a) First layer (b) Second layer (c) Third layer.

In the second layer shown in the figure 3, a more accurate distribution of prototypes is reached, so it is possible to observe prototypes adopting the form of the dataset. Additionally, in regions where the quantization error was large, the new prototypes allow a better representation (e.g., along the line). In layer three (see figure 3(c)), no more prototypes are needed to represent the circle, the line and the plane; but a third hierarchical expansion was necessary to represent the parallelepiped. In addition, due to the data density in the parallelepiped, many points are members of multiple prototypes, so several prototypes were created.

3.3 Tuning the Model Parameters

In order to explore the performance of the algorithm, different values for the parameters φ, τ_1 and τ_2 were tested using the Iris dataset. The tests were performed using ten different values of τ_1 (breadth parameter), ten of τ_2 (depth parameter) and eight of φ ($\alpha - cut$), forming 800 triplets. For each

triplet (φ, τ_1 and τ_2) a FGHSON was trained using the following fixed parameters: $t_{max} = 100$ (maximum number of iterations), $\epsilon = 0.0001$ (termination criterion) and $m_0 = 2$ (fuzzification parameter).

Several variables were obtained in order to measure the quality of the networks created for every FGHSON generated; for instance the number of hierarchical levels of the obtained network, the number of FKCNs created for each level, and finally the quantization error by prototype and level. The analysis of these values will allow discovery of the relationships between the parameters (φ, τ_1 and τ_2) and the topology of the networks (represented in this experiment by the levels reached for each network and the number of FKCN created). In addition, it will be possible to observe the relationship between the quantization errors of prototypes by level and the parameters of the algorithm. This activity makes it possible for us to find values of the parameters that allow us to build the most accurate structure, based on the number of prototypes, the quantization error and the number of levels present in the network.

Due the large amount of information involved, a graphical representation of the obtained data was used in order to facilitate visualization of the results. For this, 3D plots were used as follows: the parameter τ_1 (which regulates the breadth of the networks) and the parameter τ_2 (which regulates the depth of the hierarchical architecture) are shown on the x-axis and y-axis respectively. The z-axis shows the quantity of levels in the hierarchy (see figure 4 and figure 5). Each 3D plot corresponds to one fixed value of φ. Hence, eight 3D plots represent the eight different values evaluated for φ, then each 3D plot contains 100 possible combinations of the duple (τ_1,τ_2) for a specific φ. Therefore, analysis of τ_1, τ_2 and φ and the levels of the 800 networks were generated and plotted.

Furthermore, additional information was added to the 3D plots. The number of FKCNs created for level were represented by a symbol in the 3D plots (see figure 4 and figure 5 left side). The higher quantization error of the prototypes that were expanded is shown in a new group of 3D plots; in others words this prototype is the "father" of the prototypes in that level[4]. The rounded value of the quantization error is shown as a mark in the 3D plot for each triplet of values, in each level (see figure 4 and figure 5 right side).

Examining the obtained results, there are some interesting results related to the quantization error and the topology of the network. For instance, figure 4 and figure 5 show the different networks created. It can be seen that for values of τ_2 above 0.3 the model generates networks with just one level, so an interesting area to explore lies between the values of $\tau_2 = 0.1$, 0.2 and 0.3. With respect to the quantization error (figure 4 and figure 5 right side) for almost all values of φ, the lower quantization error with the lower number of

[4] For this reason, in level one all the values are 291 (see figure 4 and figure 5 right side) because the prototype "father" that is expanded has the same quantization error for all networks; in the case of the first level this error is called qe_0, as is described in section 2.3.

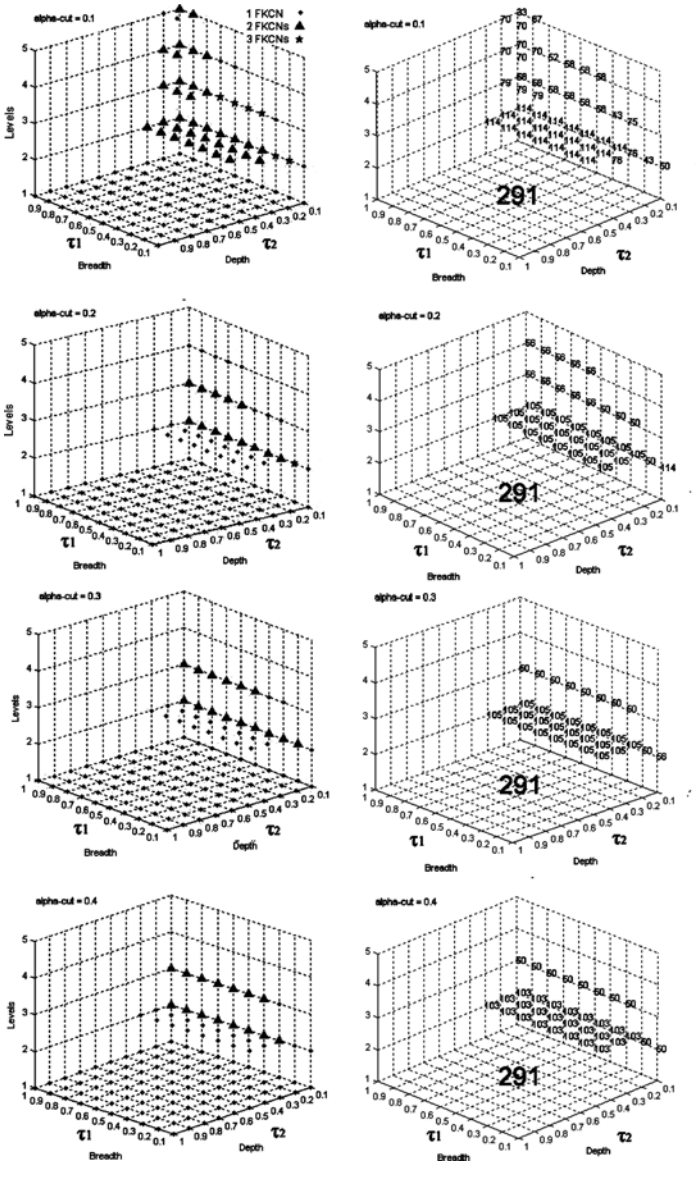

Fig. 4 The figure has 3D plots showing the results obtained using $\varphi = 0.1$, 0.2, 0,3 and 0.4. On the left side it is possible observe the levels obtained for each triplet (φ, τ_1, τ_2), in addition the number of FKCN created for each level are represented by a symbol. On the right side the higher quantization error of the prototypes that were expanded is shown; in others words this prototype is the "father" (with higher quantization error of the prototypes in that level). In the special case of the first level all the values are 291, because the prototype "father" that is expanded has the same quantization error for all the networks; in the case of the first level this error is called qe_0, as described in section 2.3.

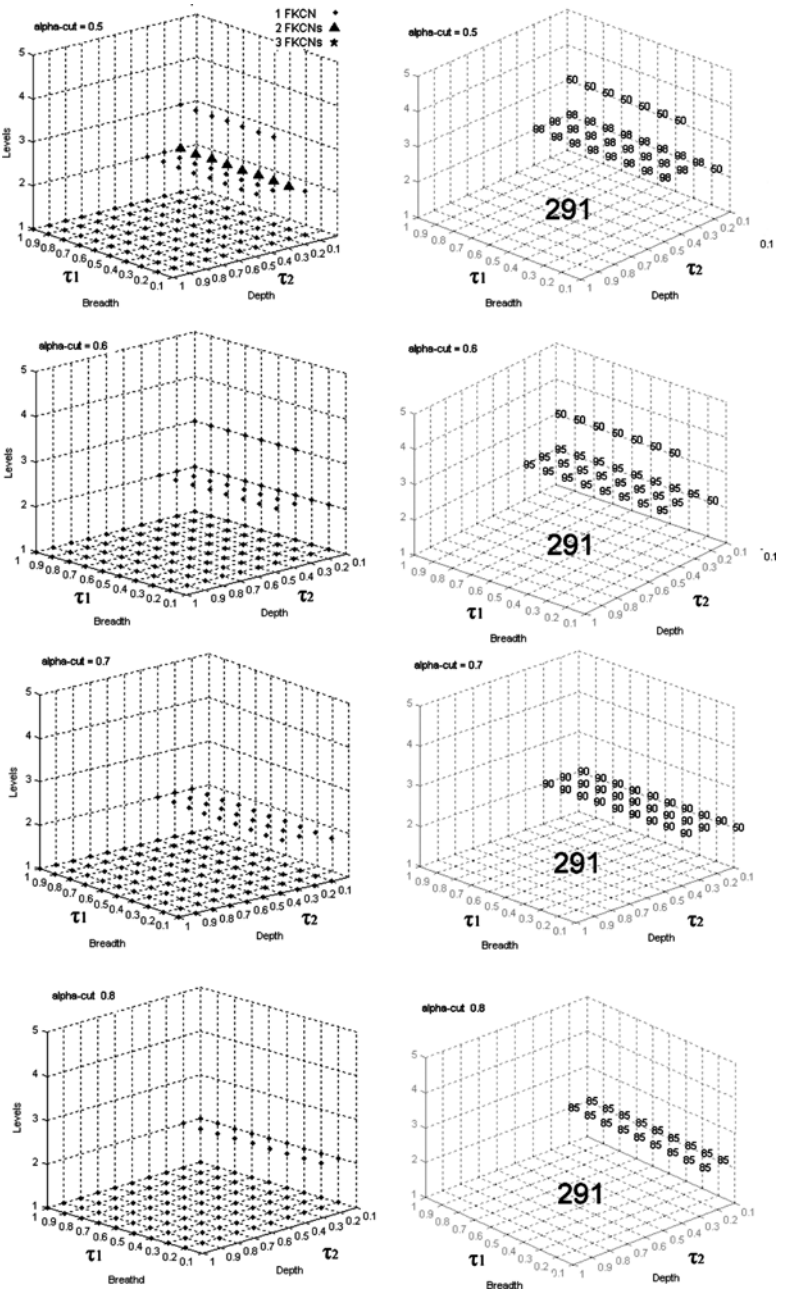

Fig. 5 The figure shows 3D plots of the results obtained using $\varphi = 0.5$, 0.6, 0.7 and 0.8, in addition the number of FKCN created for each level are represented by a symbol. On the right side the higher quantization error of the prototypes that were expanded is shown; in others words this prototype is the "father" (with higher quantization error) of the prototypes in that level.

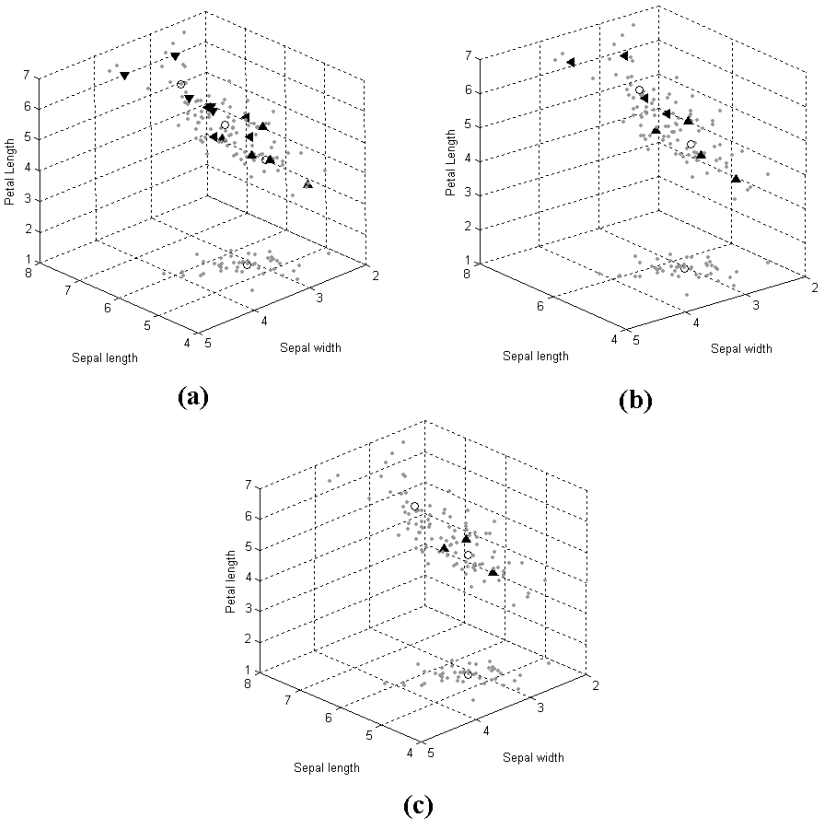

Fig. 6 Structures obtained tuning the model with the values (a) $\varphi = 0.1$, $\tau_1 = 0.2$, $\tau_2 = 0.1$ (b) $\varphi = 0.3$, $\tau_1 = 0.2$, $\tau_2 = 0.1$, and (c) $\varphi = 0.3$, $\tau_1 = 0.2$, $\tau_2 = 0.1$. In this figure it is possible to observe the distribution of the prototype vectors; the prototypes of the first level are represented by circles and the prototypes of the second level are represented by triangles.

levels was presented in the points ($\tau_1 = 0.1$, $\tau_2 = 0.2$) and(or) ($\tau_1 = 0.2$, $\tau_2 = 0.1$) (see figure 4 and figure 5).

In the next step of the analysis, the value of the parameters which generated the best networks so far were selected, based on the premise that the most accurate network has to present a lower number of levels, FKCNs, and a lower quantization error. For a selected group of three networks (see table 1), the distribution of the prototypes on the dataset were plotted in order to analyze how the prototypes of these selected networks had been adapted to the dataset (see figure 6).

Some remarks could be made about the plots obtained. In the first example ($\varphi = 0.1$, $\tau_1 = 0.2$, $\tau_2 = 0.1$) shown in figure 6(a), there are four prototypes in

Table 1 Parameters and results of the best networks selected

φ	τ_1	τ_2	Levels	qe[a]	FKCNs
0.1	0.2	0.1	2	43	3
0.3	0.2	0.1	2	50	2
0.6	0.2	0.1	2	50	1

[a] The higher quantization error of the "father" prototype of the prototypes in that level.

the first level of the hierarchy; these prototypes represent four classes[5]. Then, the prototypes of this layer represent the three classes of iris, in addition they also take the problematic region between Versicolor and Virginica as a fourth class. Furthermore, new prototypes are created in the second layer in order to obtain a more accurate representation of the dataset, creating a proliferation of prototypes. This phenomena is due to the low φ (0.1) being selected. This is because the quantity of elements represented for each prototype is large (due to low membership, a lot of data can be a member of one prototype) so, many prototypes are necessary to reach a low quantization error.

In the next example with $\varphi = 0.3$, $\tau_1 = 0.2$, $\tau_2 = 0.1$, it is possible observe (figure 6(b)) the three prototypes created in the first level. In this case the number of the prototypes matches the number of classes in the iris dataset. Nevertheless, there is (as in the previous example) an abundance of prototypes in the Virginica-Versicolor group. But in this case the number of prototypes is lower compared with the preceding example, showing how φ affects the quantity of prototypes created.

Finally, in the last example with $\varphi = 0.6$, $\tau_1 = 0.2$, $\tau_2 = 0.1$. Three prototypes are created in the first level of the network matching the classes of the Iris dataset (figure 6(c)); additionally, in the second layer one of the previous prototypes is expanded to three prototypes in order to represent the fuzzy areas of the data set. This last network presents the lower values of vector quantization, levels of hierarchy, and FKCNs; so it is possible to select this as the more accurate topology. Consider the previously defined premise which said that the most accurate network had to present lower number of levels, number of FKCNs, and the lower quantization error.

4 Conclusion

The Fuzzy Growing Hierarchical Self-organizing Networks are fully adaptive networks able to hierarchically represent complex datasets. Moreover, they allow a fuzzy clustering of the data, allocating more prototype vectors or

[5] It knows that there are three classes (iris Setosa, Virginica and Versicolor) but the fourth exists in an area where Versicolor and Virginica present similar characteristics.

branches to heterogeneous areas or to regions where the data have similar membership degree to several clusters. This property can help to better describe the structure of the dataset and the inner data relationships.

In this book chapter the effects of using different values for the parameters of the algorithm, have been presented using the Iris dataset as an example. It was shown how the different parameters affect the topology and quantization error of the networks created. In addition, some of the better networks created were examined in order to show how different representations of the same dataset can be obtained with similar accuracy.

Acknowledgements. This work is part of a cooperation project between BIOTEC, CIAT, CENICAÑA (Colombia) and HEIG-VD (Switzerland) named Precision Agriculture and the Construction of Field-crop Models for Tropical Fruits. The financial support is given by several institutions in Colombia (MADR, COLCIENCIAS, ACCI) and the State Secretariat for Education and Research (SER) in Switzerland.

References

1. Anderberg, M.: Cluster Analysis for Applications. Academic, New York (1973)
2. Backer, E., Jain, A.: A clustering performance measure based on fuzzy set decomposition. IEEE Trans. Pattern Anal. Mach. Intell. PAMI-3(1), 66–75 (1981)
3. Baraldi, A., Alpaydin, E.: Constructive feedforward ART clustering networks-Part I and II. IEEE Trans. Neural Netw. 13(3), 645–677 (2002)
4. Bezdek, J., Tsao, K., Pal, R.: Fuzzy Kohonen clustering networks. In: IEEE Int. Conf. on Fuzzy Systems, pp. 1035–1043 (1992)
5. Burzevski, V., Mohan, C.: Hierarchical Growing Cell Structures. Tech Report: Syracuse University (1996)
6. Doherty, J., Adams, G., Davey, N.: TreeGNG - Hierarchical topological clustering. In: Proc. Euro. Symp. Artificial Neural Networks, pp. 19–24 (2005)
7. Diday, E., Simon, J.C.: Clustering analysis. In: Digital Pattern Recognition, pp. 47–94. Springer, Heidelberg (1976)
8. Everitt, B., Landau, S., Leese, M.: Cluster Analysis. Arnold, London (2001)
9. Fischer, G., van Velthuizen, H.T., Nachtergaele, F.O.: Global agro-ecological zones assessment: methodology and results. Interim Report IR-00-064. IIASA, Laxenburg, Austria and FAO, Rome (2000)
10. Fritzke, B.: Growing cell structures: a self-organizing network for unsupervised and supervised learning. Neural Networks 7(9), 1441–1460 (1994)
11. Fritzke, B.: Some competitive learning methods, Draft Doc. (1998)
12. Taniichi, H., Kamiura, N., Isokawa, T., Matsui, N.: On hierarchical self-organizing networks visualizing data classification processes. In: Annual Conference, SICE 2007, pp. 1958–196 (2007)
13. Hodge, V., Austin, J.: Hierarchical growing cell structures: TreeGCS. IEEE Transactions on Knowledge and Data Engineering 13(2), 207–218 (2001)
14. Huntsberger, T., Ajjimarangsee, P.: Parallel Self-organizing Feature Maps for Unsupervised Pattern Recognition. Int. Jo. General Sys. 16, 357–372 (1989)

15. Jain, A., Murty, M., Flynn, P.: Data clustering: a review. ACM Comput. Surv. 31(3), 264–323 (1999)
16. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics 43(1), 59–69 (1982)
17. Lampinen, J., Oja, E.: Clustering properties of hierarchical self-organizing maps. J. Math. Imag. Vis. 2(2–3), 261–272 (1992)
18. Luttrell, S.: Hierarchical self-organizing networks. In: Proceedings of the 1st IEE Conference on Artificial Neural Networks, London, UK, pp. 2–6. British Neural Network Society (1989)
19. Martinez, T., Schulten, J.: Topology representing networks. Neural Networks 7(3), 507–522 (1994)
20. Merkl, D., He, H., Dittenbach, M., Rauber, A.: Adaptive hierarchical incremental grid growing: An architecture for high-dimensional data visualization. In: Proc. Workshop on SOM, Advances in SOM, pp. 293–298 (2003)
21. Miikkulainen, R.: Script recognition with hierarchical feature maps. Connection Science 2, 83–101 (1990)
22. Rauber, A., Merkl, D., Dittenbach, M.: The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. IEEE Transactions on Neural Networks 13(6), 1331–1341 (2002)
23. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on Neural Networks 16(3), 645–678 (2005)