The GH-EXIN neural network for hierarchical clustering

The GH-EXIN Neural Network for Hierarchical Clustering

Giansalvo Cirrincione[a,b], Gabriele Ciravegna[c], Pietro Barbiero[d], Vincenzo Randazzo[e], Eros Pasero[e]

[a] *University of South Pacific, Suva, Fiji*
[b] *University of Picardie Jules Verne, Amiens, France*
[c] *Università degli Studi di Siena, DIISM, Siena, Italy*
[d] *Politecnico di Torino, DISMA, Turin, Italy*
[e] *Politecnico di Torino, DET, Turin, Italy*

Hierarchical clustering is an important tool for extracting information from data in a multi-resolution way. It is more meaningful if driven by data, as in the case of divisive algorithms, which split data until no more division is allowed. However, they have the drawback of the splitting threshold setting. The neural networks can address this problem, because they basically depend on data. The growing hierarchical GH-EXIN neural network builds a hierarchical tree in an incremental (data-driven architecture) and self-organized way. It is a top-down technique which defines the horizontal growth by means of an anisotropic region of influence, based on the novel idea of neighborhood convex hull. It also reallocates data and detects outliers by using a novel approach on all the leaves, simultaneously. Its complexity is estimated and an analysis of its user-dependent parameters is given. The advantages of the proposed approach, with regard to the best existing networks, are shown and analyzed, qualitatively and quantitatively, both in benchmark synthetic problems and in a real application (image recognition from video), in order to test the performance in building hierarchical trees. Furthermore, an important and very promising application of GH-EXIN in two-way hierarchical clustering, for the analysis of gene expression data in the study of the colorectal cancer is described.

Keywords: Convex Hull, DGSOT, Dynamic Tree, GHNG, Hierarchical Divisive Clustering, Neural Network, Self-Organization, Two-way Clustering

---

1. Introduction

The hierarchical cluster analysis (HCA, [1]) is a multi-resolution clustering technique. It builds a tree of clusters with different levels of data interpretation. In data mining, for instance, HCA can yield a richer information than plain clustering. It is generally performed in two ways:

- Hierarchical Agglomerative Clustering (HAC), where each data in the beginning corresponds to a singleton cluster, and then pairs of clusters are merged until only one cluster containing all data is reached (bottom-up approach).

- Hierarchical Divisive Clustering (HDC), in which all data start in one cluster and splits are performed recursively until all clusters are singletons (top-down approach).

The top-down approach of HDC represents better the dataset because it starts taking into account all data. Instead, the bottom-up approach of HAC is, in this sense, more arbitrary in the initial steps, thus influencing the quality of the resulting tree. Also, HAC is intractable in case of large datasets. However, the way HDC splits the clusters is still an open problem. A promising approach is represented by the use of neural networks for clustering.

The neural algorithms for HDC can be classified according to both the way the neural tree is trained and the kind of basic neural network used for each node (basic neural unit). As a first taxonomy, there are two approaches: synchronous training (ST), where the training is performed on the whole tree, and asynchronous training (AT), where the training is performed node by node. The Dynamic Neural Tree Network (DNTN, [2]) adapts a dynamic hierarchy to data, as an output layer fed by the input: all growing nodes are fed by the same input and are trained simultaneously (ST). It requires a tolerance for determining the new neurons and a threshold for the child growth. It is not able to represent correctly the outliers. A variant of DNTN, the Competitive Evolutionary Neural Tree (CENT, [3]) claims it does not require user dependent parameters. Indeed, they become internal parameters which are dependent on data, but are empirical and not justified. CENT is based on the neuron activity (which is decreased in time for avoiding the poor initialization) and addresses the DNTN problem of outlier detection. Another ST approach uses the Self-Organizing Map (SOM), considered as a tree (TreeSOM, [4]). It is based on the interpretation of SOM as a tree when a distance threshold is decreased in time [5]. TreeSOM addresses the problem of the sensitivity of the tree to the SOM topology and initialization by using the idea of consensus tree, which is a virtual tree averaging the trees resulting from different initial conditions. The best tree is the closest to the consensus tree.

Most neural approaches fall into the AT category. They can be classified according to the clustering technique used for each node. In [6], the k-means algorithm is exploited. It divides data in a fixed number of clusters (HDC), but uses an additional HAC for refinement. The Hierarchical Clustering Algorithm based on k-means with Constraints (HCAKC, [7]) uses an improved Silhouette for determining the k parameter. In [8] and [9], a preprocessing based on Principal Component Analysis and divide-and-conquer, respectively, is used for dealing with high-dimensional data. In [10], the Growing Hierarchical Tree SOM (GHTSOM) uses an elementary SOM given by three connected neurons as basic module (triangle). It requires two kinds of links: the train links for defining the neural triangles and the class links for clustering at each level of the tree. The use of triangles, which does not yield necessarily a Delaunay triangulation, limits heavily the performance of the network. If the basic neural unit is a Growing Cell Structure (GCS, [11]) the hierarchical version is called Hierarchical GCS (HiGCS, [12]). If, instead, the Growing Grid (GG, [13]), with decreasing learning rate and neighborhood range, is used, the algorithm is called Growing Hierarchical SOM (GH-SOM, [14]). The vertical and horizontal tree growths are controlled by using the mean quantization error, by means of two parameters whose setting is tricky, as discussed in [15].

In [15], a novel algorithm, called Growing Hierarchical Neural Gas (GHNG), is proposed, which exploits the Growing Neural Gas (GNG, [16]) as basic neural unit. In [15] it is proved it has a better performance than GHSOM. Another algorithm performing AT is the Dynamically Growing Self Organizing Tree (DGSOT, [17]) which is an enhanced dynamic version of the Self Organizing Tree Algorithm (SOTA, [18]), which builds a binary hierarchy.

In the following sections, the last two algorithms (GHNG and DGSOT) are discussed, along with the proposed AT approach (GH-EXIN), highlighting similarities, differences and novelties. More specifically, in Sec. 2 the GHNG and DGSOT algorithms are briefly summarized. In Sec. 3 the novel neural network is presented, together with considerations on the required user-dependent parameters and the analysis of the time complexity. Results and comparison of the three algorithms on the synthetic experiments are reported and discussed in Sec. 4. Real-world applications are presented in Secs. 5 and 6, in a typical problem in video recognition and in two-way clustering, respectively.

2. Benchmark Networks

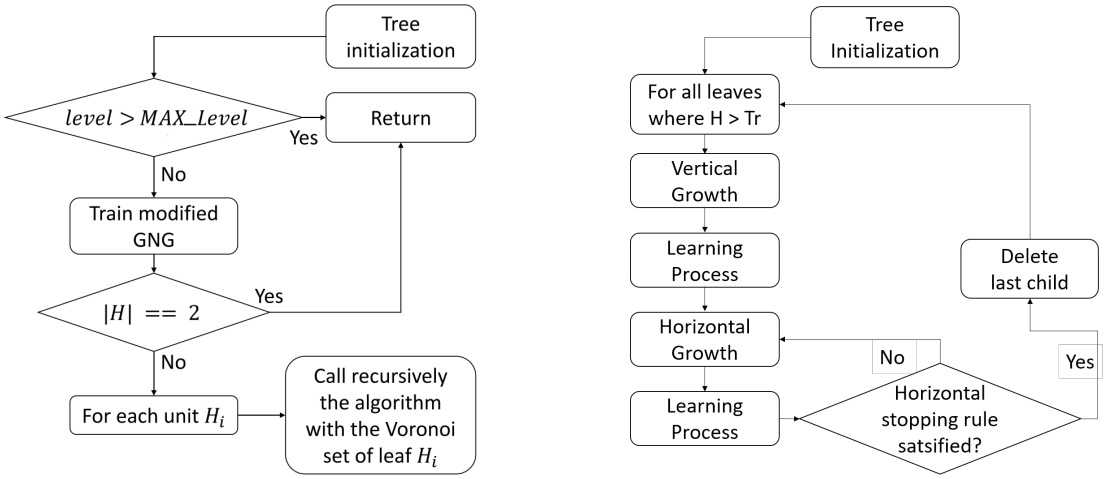
Considering that GHNG improves over GHSOM and is a hierarchical version of GNG, and DGSOT builds its hierarchy in a more general way than SOTA, and taking into account that both have the best results in applications and share some ideas of GH-EXIN, they have been chosen as benchmark neural networks to assess the performance of GH-EXIN.

2.1. GHNG

The Growing Hierarchical Neural Gas is an AT hierarchical self-organizing neural model, which learns a tree of Growing Neural Gas (GNG, [16]) networks where each subgraph (i.e. each GNG) is the child of a processing unit (a.k.a. neuron) of the upper level. At each hierarchical level a GNG network is created by using the Voronoi set of the father neuron (see Fig. 1(a)). The vertical growth in a branch of the hierarchy stops when the deepest GNG enters the convergence phase having only two neurons ($|H| == 2$) or when the maximum depth is reached ($level > MAX\_Level$, where MAX_Level is a user-dependent parameter). In the first case, this neural unit is pruned because is considered too small to represent any relevant distribution.

2.2. DGSOT

The Dynamically Growing Self-Organizing Tree (DGSOT), see Fig. 1(b), is an enhanced version of the Self Organizing Tree Algorithm (SOTA) [18]. DGSOT adds a horizontal growth to each vertical growth performed by SOTA. This kind of growth consists in the addition of a node to the current group of child nodes. This process allows to better determine the cluster partitioning at each level. It is always followed by a further learning process in which the correct number of neurons is automatically determined for the required quantization. As in SOTA, the learning process continues until the relative heterogeneity of all child nodes compared to the previous epoch is less than a user-dependent parameter T_R. These two steps are repeated until the Cluster Separation index falls below another user-dependent parameter CS_{min} (horizontal stopping rule, a.k.a. T_E).



(a) GHNG flowchart (b) DGSOT flowchart

Figure 1: Benchmark network flowcharts

3. The GH-EXIN Hierarchical Tree

The proposed approach builds a divisive hierarchical tree in an incremental and self-organized way. It is data driven (self-organization), in the sense that the final tree is automatically estimated. Also, it does not require a predefined number of units and levels (incremental with pruning phase). The resulting tree is neither binary nor balanced, because of its dependence on data. Both the GH-EXIN (Figs. 6 and 7), GHNG (Fig. 1(a)) and DGSOT (Fig. 1(b)) algorithms follow these criteria.

3.1. Tree Building

The hierarchical divisive clustering algorithms build a tree starting from a root node. By means of vertical and horizontal growths, successive splits are determined. These splits correspond to the transformation of the corresponding leaf into a node, called father neuron, whose sons are its associated leaves. For each father neuron a neural network (i.e. a basic neural unit) is trained on its corresponding Voronoi set, i.e. the set of data represented by the father neuron. The sons are then the neurons of the associated basic neural unit and determine a subdivision of the father Voronoi set. For each leaf the procedure is repeated.

3.1.1. Root Node

Both DGSOT and GH-EXIN associate the whole data set to a fictitious neuron (a.k.a. root node). The first basic neural unit is then trained on the Voronoi set of this fictitious unit. Conversely, GHNG does not have any fictitious father, all nodes created at the first layer are orphans (they do not originate from a father neuron). It could be argued that GHNG builds a forest other than a single tree.

3.1.2. Vertical Growth

Vertical growth is the process in which a leaf becomes a node and a deeper layer is added. In these algorithms, it is always required the creation of a *seed*, i.e. a pair of neurons, which represents the starting structure of a new basic neural unit. This kind of growth is exploited as long as a higher resolution is needed. In order to evaluate whether a vertical growth is necessary, all algorithms check if the quantization error of the basic neural unit is below a user-defined threshold. DGSOT checks data heterogeneity, defined as the average distance of the data to the neuron reference vector. GHNG, instead, checks whether the number of levels in the hierarchy exceeds a user-dependent parameter MAX_Level. GH-EXIN simultaneously checks both data heterogeneity using a task-dependent index, say H_{max}, and the data cardinality (min_{card}), i.e. the size of the leaf Voronoi set. The H index is based on the quantization error and determines the quality of the clustering. Several choices are possible, in general depending on the application. In this paper the H_{cc} index [19] has been chosen. It is designed for biclustering problems, but, here, it is extended to two-way clustering. Its description is given in Sec. 6. Considering the characteristics of the biclusters that can be found, it can also be used for normal clustering, as seen in Sec. 4.

3.1.3. Horizontal Growth

Horizontal growth refers to the addition of further neurons to the initial seed, i.e. the creation of siblings. This method allows to expand a layer of the tree and build more complex hierarchical structures other than binary. This process is performed by all algorithms through the respective neuron creation mechanism later described (see Sec. 3.2.1).

3.2. sG-EXIN

The basic neural unit is intended as the neural network chosen for the clustering of the input data. All these methods use a basic neural network for the processing of each leaf. GH-EXIN is based on the stationary variant of G-EXIN [20], say sG-EXIN. GHNG uses a variant of GNG, while DGSOT, deriving from SOTA, exploits as basic neural network an enhanced version of SOM. All these neural units do not employ any fixed topology (the induced topology is generated in the linking phase). Neural networks are composed of units called neurons, which are represented by weight (a.k.a. reference) vectors. As an abuse of language, the terms neuron and weight vector are used with the same meaning. With regard to training, both sG-EXIN and DGSOT are based on the idea of *epoch*, which is the presentation, in a random order, of the complete training set. After each epoch controls are made for the horizontal growth by means of user-dependent parameters: H_{perc} (GH-EXIN), T_E (DGSOT). However, this is not the classical batch learning, which requires the weights to be updated after the presentation of the whole batch. For these two neural networks, instead, weights are updated or created at each iteration (data presentation). Also, GHNG learns at each iteration, but it is not clear if it requires epochs. Indeed, its algorithm does not use epochs; however, in the examples in [15], epochs are mentioned. There are some pros and cons with the choice of *epoch-learning*. Despite the fact that it lowers the training cost (GHNG is the fastest one), the use of epochs before controls implies the exploitation of the complete batch, which means all information is used for building a tree. It must be

highlighted that the onset of the hierarchical tree is fundamentally a static problem, thus requiring the processing of the whole database for an accurate tree.

3.2.1. Neuron Creation

The basic neural units use incremental neural networks, i.e. they have a variable number of neurons (driven by data), achieved by the mechanism of neuron creation and pruning. Both DGSOT and GHNG decide in advance when to create a new neuron: DGSOT by considering the batches of the whole set of samples, at the end of an epoch, GHNG every λ (user-dependent parameter) iterations. On the contrary, for GH-EXIN, the creation of a neuron, is related to a novelty test: if the existing neurons are not able to describe the new data, say x_i, a new neuron is created. The test approach is less rigid because it is only driven by data. As discussed in the following, the choice of the test is not trivial.

Semi-Isotropic Region of Influence. The novelty test requires, in general, a model of the region of influence of a neuron in the input space. All existing algorithms determine, in a way or another, a threshold value representing the radius of an hypersphere which models this region. An exhaustive description can be found in [21]. This model is isotropic, in the sense that it does not take into account the orientation of the vector connecting the new data to the winner, but only its norm. As a drawback it does not consider the topology of the data manifold of the winner Voronoi set. The GH-EXIN proposed approach considers, at the same time, the shape (anisotropic criterion) and the extent (isotropic criterion) of the neuron neighborhood (determined by the linking phase, see Sec. 3.2.3). The anisotropic criterion consists of the analysis of the convex hull built on the neighbor neurons of the winner as explained in Fig. 2 and in more detail in [20]. In case x_i lies inside the convex hull, it is assigned to the winning neuron (w_γ), as shown in Fig. 3, and the weight adaptation is performed (see Sec. 3.2.2). Otherwise, the isotropic criterion is applied to check if the data is really novel w.r.t the existing units. The neuron isotropic threshold, say T_γ, is computed as the average of the Euclidean distances between the winning neuron (w_γ) and its N neighbors (w_i):

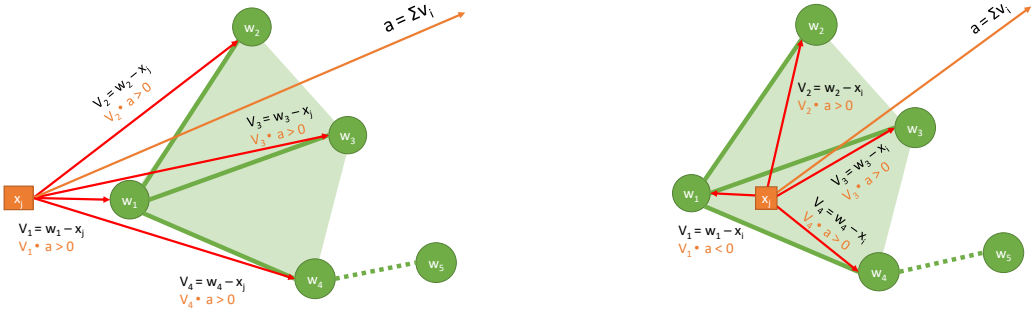
$$T_\gamma = \frac{1}{N} \sum_{i=1}^{N} ||w_\gamma - w_i||^2 \tag{1}$$

In case x_i is farther from the winner than T_γ, a new neuron is created on the data. Otherwise, it is assigned to the winner, and the weights are updated according to the mechanism described later (see Sec. 3.2.2).

The use of an anisotropic criterion is justified by the need of representing better the data manifold, which is not guaranteed by using only the isotropic threshold. Fig. 3 shows one of these cases. It is proved in [20] that this approach better models the border of the data manifold.

On the contrary, GHNG and DGSOT do not require a region of influence, but keep adding neurons as far as the quantization error (a.k.a. *cluster separation* in [17]) does not fall below a user-dependent threshold. GHNG computes the quantization error from data, by considering the average distance between the reference vectors and their Voronoi sets, while

DGSOT from network topology, by considering the rate between the minimum and the maximum distance among neurons. As a consequence, GHNG and DGSOT are prone to the choice of a predetermined parameter, instead of exploiting the neighborhood topology, as in the case of GH-EXIN.



(a) If all the dot products between the vectors v_i from the new data x_j and the neuron w_i, and the vector a, sum of the v_i's, have equal sign, then the new data is outside the convex hull of the winner (w_1)

(b) If all the dot products, between the vectors v_i from the new data x_j and the neuron w_i, and the vector a, sum of the v_i's, have not the same sign, then the new data is inside the convex hull of the winner (w_1)

Figure 2: A new data x_j is presented to the sG-EXIN neural network composed of four connected neurons

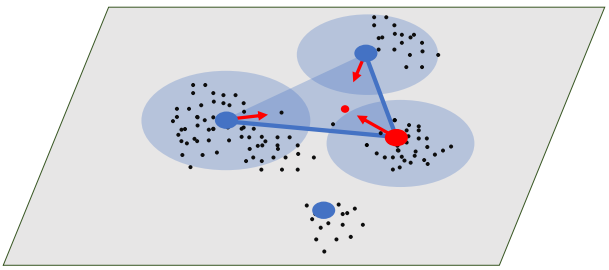


Figure 3: The anisotropic criterion: if a new data (small red dot) is presented and lies outside the hypershphere centered on the winner (big red dot), but within the convex hull created on its neighbours (blue connected nodes), it is also assigned to the winner. Furthermore, according to SCL both the winner and its neighbours move towards it.

Lonely Neuron. A neuron with no edges is named *lonely neuron*: in Fig. 3 the neuron at the bottom with no edges is a lonely neuron. Since DGSOT does not make use of edges, all its neurons are lonely. On the contrary, both GHNG and GH-EXIN exploit the concept of

lonely neurons to determine leaf neuron pruning. In both algorithms, a neuron may become lonely in case all its edges are pruned. However, in GHNG neurons already have connections when created. In GH-EXIN, instead, new neurons are created lonely. Connections may be generated only in the next iterations.

3.2.2. Soft-Competitive Learning

The weight computation (training) is based on the Soft Competitive Learning (SCL) paradigm [22], which requires a winner-take-most strategy and the network topology, whose setting is explained in Sec. 3.2.3. The closest neuron to the new data is named (first) winner. The set of potential winners differs according to the clustering algorithm. In both GHNG and GH-EXIN, only neurons belonging to the same basic neural unit are competitive in the learning phase. In DGSOT, instead, all leaf neurons belonging to the sub-tree below the K-ancestor of the current node are competitive.
At each iteration, both the winner and its neighbors change their weights but in different ways as shown in Fig. 3. The winner w_γ and its direct topological neighbors w_i are moved towards x_j by fractions $\alpha_\gamma(t)$ and $\alpha_i(t)$ (learning rates), respectively, of the vector connecting the weight vectors to the data. The update is given by:

$$\Delta w = \alpha(t) \cdot (w - x_j) \tag{2}$$

where $\alpha(t) = \alpha_0/t$, and α_0 is a user dependent parameter, higher for the winner and smaller for the neighbours, and t is the number of times a neuron wins (conscience). Regarding the weight adaptation, the difference among the three algorithms consists on the neighborhood determination. DGSOT considers as neighbors all neurons belonging to the same basic neural unit, plus the father neuron. On the contrary, the neighborhood of both GHNG and GH-EXIN is composed of only those neurons connected to the winner through an edge. Hence, GHNG and GH-EXIN exploit a more local information.

3.2.3. Edge Creation and Network Topology

Edges are exploited in order the determine the topology (neighborhood) of a network. Both GH-EXIN and GHNG use the Competitive Hebbian Learning (CHL) rule [22] for creating the neuron connections: each time a neuron wins (first winner), an edge is created, linking it to the second nearest neuron (second winner), if it does not exist yet. If there is already an edge, its age is set to zero. Furthermore, in both GHNG and GH-EXIN the same aging procedure is applied: the age of all links emanating from the winner is incremented by one. In case a link age is greater than the age_{max} scalar parameter, it is eliminated (pruned).

3.3. Neuron Pruning

Neuron pruning is the process through which neurons can be removed if redundant. DGSOT does not exploit any pruning technique, in the sense that the redundancy is only checked each time a neuron is added, but old neurons cannot be removed. Vice versa, GHNG and GH-EXIN remove all lonely neurons. GH-EXIN checks for lonely neurons at the end of each epoch (see Fig. 7), while GHNG checks at each iteration. Besides, GHNG may prune an entire set of neurons if its modified GNG enters the convergence phase. However, the onset of this phase is empirically determined.

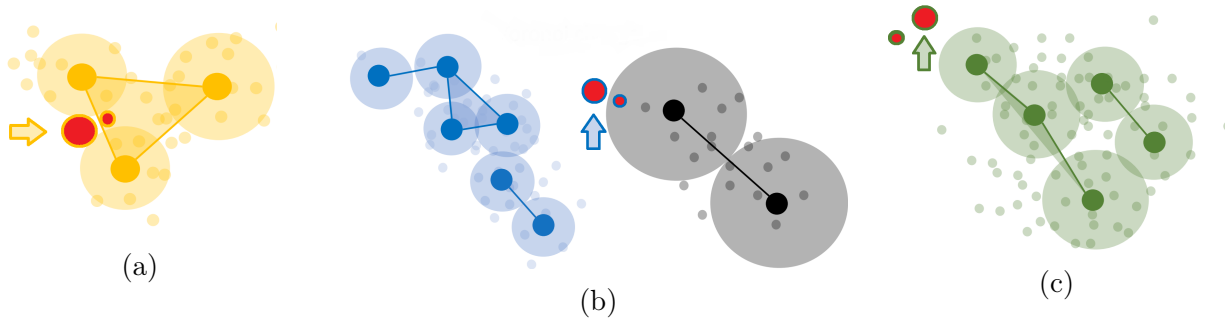


Figure 4: GH-EXIN data reallocation: small dots represent data while big dots represent neurons. Big red dots, indicated by arrows (colored as the neurons of the same sG-EXIN), mark pruned neurons, while small red dots represent data belonging to them. Different colors represent different network units

Data Reallocation. Data reallocation is the mechanism by which orphan data, i.e. associated to pruned neurons (their Voronoi sets) or lonely neurons (data coincident with the neuron) are possibly reassigned to other neurons. This mechanism is a novelty introduced in GH-EXIN (see Fig. 7), and is an improvement of the KLD method used in DGSOT. In fact, in GHNG, when neurons with no edges are removed, the associated data are not reallocated. In GH-EXIN, instead, all orphan data are labelled as potential outliers at the end of each epoch. For each potential outlier, GH-EXIN seeks a new winner among all leaf neurons. The data is reassigned in case it is inside the hypersphere (or the convex-hull) of another neuron within the same neural unit (Fig. 4(a)) or in case the nearest neuron belongs to another neural unit (Fig. 4(b)). If, instead, the winner belongs to the same neural unit of the pruned neuron, but the data is outside its hypersphere, the data is definitely marked as outlier and is not reassigned (Fig. 4(c)). This outlier identification can be useful in a lot of applications.
Resuming, only GH-EXIN and DGSOT may correct possible cluster errors made in the previous layers. Instead, this is not possible for GHNG: the advantage of the speed of its algorithm is counterbalanced by the fact that the errors always affect the final tree. As said before, the building of a tree is basically a static process. So, GHNG is less suited to hierarchical clustering than the other two techniques.

3.4. Connected Graph Test

Another remarkable novelty introduced by GH-EXIN consists of a possible double vertical growth. As shown in Fig. 6, at the end of the sG-EXIN training process, the resulting graph of the basic neural unit is analyzed by searching for connected components. If more than one connected component is detected, the algorithm tries to extract an abstract representation of data. Hence, each connected component, representing a cluster of data, is associated with a novel abstract neuron. The reference vectors of abstract neurons are placed in the centroids of the respective clusters. The tree structure is therefore modified by inserting a new layer between the leaf nodes and the father node, resulting in a double simultaneous vertical growth, as shown in Fig 5.

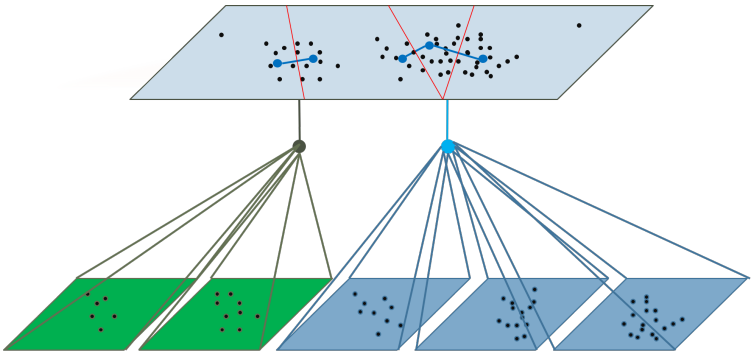


Figure 5: Connected Graph Test: the Voronoi regions of each neuron are represented with solid red lines; chains in the father sG-EXIN become sons

The proposed test is justified by the exploitation of the topology graph built by GH-EXIN. The estimated connected components are directly translated into the hierarchical tree through this additional vertical growth. In this sense, GH-EXIN does not only partition the data into nested Voronoi sets, but exploits its induced Delaunay triangulation, created by the CHL rule.

3.5. The GH-EXIN Algorithm

Resuming, for each node, an sG-EXIN neural network is trained on its corresponding Voronoi set (set of data represented by the father neuron). For each leaf, the vertical growth is performed until either the H index of the leaf has fallen below H_{max} or the cardinality of the leaf is less the min_{card} (both are user-dependent parameters). For each epoch, the basic iteration starts at the presentation of a new data, say x_i. All neurons are ranked according to the Euclidean distances between x_i and their weight vectors. In case the data is considered new, i.e. it is both outside the convex polytope and the hypersphere of radius T_γ of the winner w_1 (novelty test), a new neuron x_{new} is created (left branch of Fig. 7). The initial weight vectors and neuron thresholds T_γ are given by heuristics: the novel neuron has its weight x_{new} equal to x_i, and its threshold is set equal to the w_1 threshold. No edge is created at this time: x_{new} is labelled as *lonely neuron*.

Otherwise, in case the data is not new (right branch of Fig. 7), the first winner w_1 and the second winner w_2 are linked by an edge (CHL), if it does not exist yet. If there is already an edge, its age is set to zero. Also, the age of all other links emanating from the winner is incremented by one; if a link age is greater than the age_{max} scalar parameter, it is eliminated (edge pruning). Reference vectors of w_1 and its direct neighbors are updated according to Eq. 2. Thresholds of the winner and of its neighbors are recomputed, as their position has been modified. This process is repeated for all data of the father Voronoi set. At the end of each epoch, if a neuron remains unconnected (no neighbors) or is still lonely, it is pruned, but the associated data are analyzed and possibly reassigned.

The training epochs, i.e. the horizontal growth, are stopped when the estimated H average value falls below a percentage (H_{perc}) of the H value of the father neuron.

This technique builds a vertical growth of the tree. The horizontal growth is generated

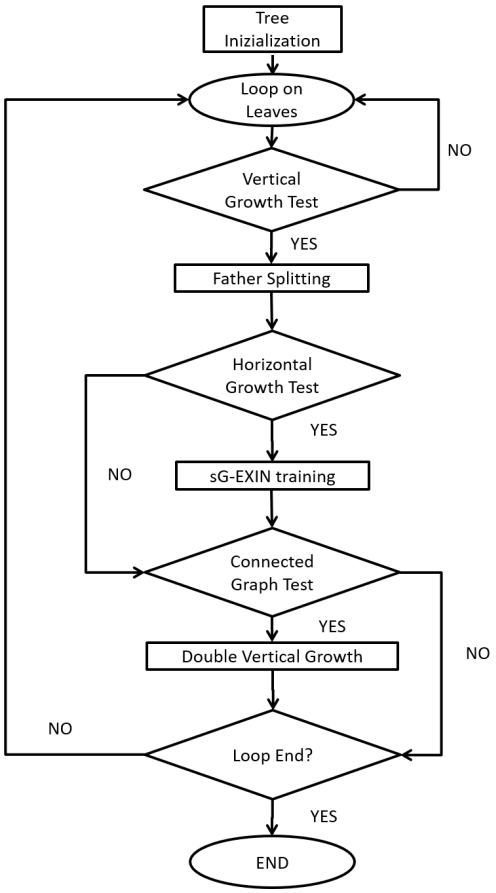


Figure 6: GH-EXIN flowchart

by the neurons of each network. However, a simultaneous double vertical growth is possible, as specified in Sec. 3.4.
GH-EXIN has been developed in MATLAB. The code is freely available at [23].

3.6. Analysis of the User-Dependent Parameters

The user dependent parameters can be grouped according to their function in three classes: learning, hierarchy and design variables. The learning parameters handle the training of the basic neural unit. GH-EXIN uses sG-EXIN, which requires CHL and SCL. They are performed by using:

- the two learning rate constants, $\alpha_{\gamma 0}$ and α_{i0}, used to update reference vectors of the winner and its neighbours, respectively;
- the scalar age_{max} used for edge pruning: it has to be lowered if more edges (and neurons) have to be pruned, indirectly controlling the leaf cardinality.

Instead, GHNG requires three more parameters, λ, α, and D, which are related to the creation of a new neuron. In particular, deciding in advance when to insert it (it depends

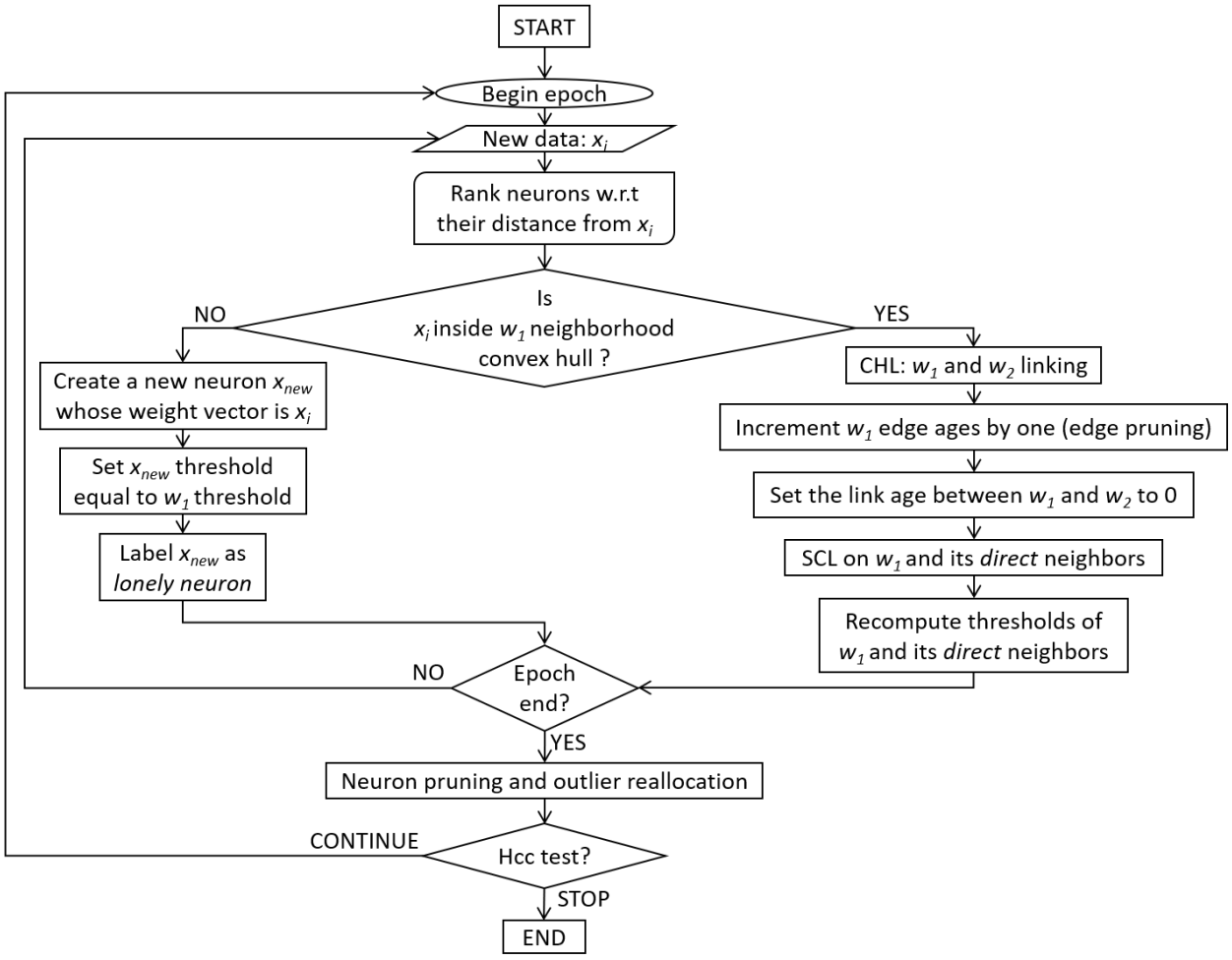


Figure 7: GH-EXIN: horizontal growth flowchart

on λ) is a serious drawback. In the case of DGSOT, there are three learning rates, for the winner, the parent and the siblings, respectively. Its horizontal growth can be compared to a neuron creation and is controlled by a threshold, T_E. The stop criterion for GH-EXIN depends on H_{perc} (other criteria can be used according to the application). GHNG controls the growth process by the parameter τ. It is then followed by a refinement step, terminated by a maximum number of epochs, decided in advance. DGSOT stops learning when the relative error of the entire three is less than the error threshold T_E.

The hierarchy parameters control the growth of the tree. GH-EXIN uses the H_{max}, which depends on the task. Instead, more rigidly, GHNG uses a MAX_Level threshold, which is not guided by the application at end, and can also be considered as a design variable. DGSOT controls the vertical growing using the threshold T_R for controlling the heterogeneity of any leaf.

The design parameters are not important for the hierarchical clustering but help in deciding in advance a preferred tree depth. In this sense, they cannot be considered as

relevant for the user dependent setting. GH-EXIN uses min_{card}, the minimum cardinality of leaves, used to avoid small clusters. The same holds true for GHNG, which seeks only for clusters with a cardinality more than three.

Both GH-EXIN and DGSOT have the possibility to reallocate data. However, this process is completely automatic in GH-EXIN, while DGSOT requires a parameter K.

Resuming the meaningful parameters to be set are 5 for GH-EXIN, 8 for GHNG and 7 for DGSOT. As a consequence, GH-EXIN is easier to be calibrated.

3.7. Analysis of Complexity

Define N as the number of data in the whole training set, d as the dimensionality of the input, J as the average number of epochs for the basic neural unit training, and k as the average number of neighbors for each neuron. Let be b as the average branching factor of the tree. Then, the height of the tree is $h = log_b M$, where M is the number of leaves in the hierarchy. For a full tree (each leaf node associated with one data), M is $O(N)$ where N is the number of data.

3.7.1. GH-EXIN Complexity

The computational cost of GH-EXIN can be estimated by considering the algorithm step-by-step (see Figs. 6 and 7).

sG-EXIN Iteration. At each iteration in an epoch, see Fig. 7, an input x_i is fed to the network and the two closest neurons (weights), w_1 and w_2, are found according to their Euclidean distances from the data. Let m_i the number of neurons of sG-EXIN at the i-th iteration. Then, the distance estimation is $O(m_i d)$. If a simple min algorithm is employed, the second step (i.e. first and second minimum search) is performed in $O(m_i)$. In order to take into account an average of the cardinality of all the Voronoi sets in an horizontal growth, m_i can be safely replaced by the average branching factor b. Resuming, both steps have a complexity of $O(bd) + O(b) = O(b)$, because d is constant w.r.t. the evaluation of the algorithm complexity. Once the winner is determined, the neighborhood convex hull test is performed. It requires the identification of the convex hull of the winner, i.e. its neighbors. For considering, in a global way, the vicinity of all neurons in the network, the average of the neighborhood cardinality, say k, is considered here. As explained in Fig. 2, to check if the input belongs to the winner convex hull, the vector a, sum of the difference vectors, v_i, between the winner neighborhood and the input x_i, needs to be computed; this requires $O(kd)$ operations. Then, the inner products between a and all the v_i's cost $O(kd)$ in the worst case (all comparison are needed). Resuming, the test costs $O(kd) = O(k)$, according to the previous considerations. After the novelty check, two scenarios can occur: either a new neuron is created or w_1 and its neighbors adapt their weights according to the SCL. It is easy to prove that the former costs $O(1)$ because it is just a sequence of atomic operations whose cost is, by definition, $O(1)$. The latter case, i.e. the SCL weight adaptation, is slightly more complex: the CHL linking is $O(1)$; the aging and pruning phase exactly needs $O(k)$ operations; the SCL adaptation requires $O(kd)$ because it performs a vector adjustment, k times (i.e. the size of the neighborhood), by means of adding a scaled difference vector to the

weight, whose computational cost is, of course, $O(d)$; the threshold re-estimation implies to evaluate, for each of the k neurons moved by the SCL, the distances from its neighbors (i.e. $O(k^2 d)$), assuming as negligible the search for the maximum neighbor distance. Resuming, it can cost either $O(1)$ in case of neuron creation or $O(k^2)$ in case of SCL weight adaptation. In conclusion, a single sG-EXIN training iteration employs $O(b) + O(k^2)$ operations. However, it must be taken into account that the branching factor represents the number of neurons of a neural unit, and that GH-EXIN builds, by construction, a tree and not a fully connected graph. Hence, $b >> k$ and the overall cost becomes $O(b)$.

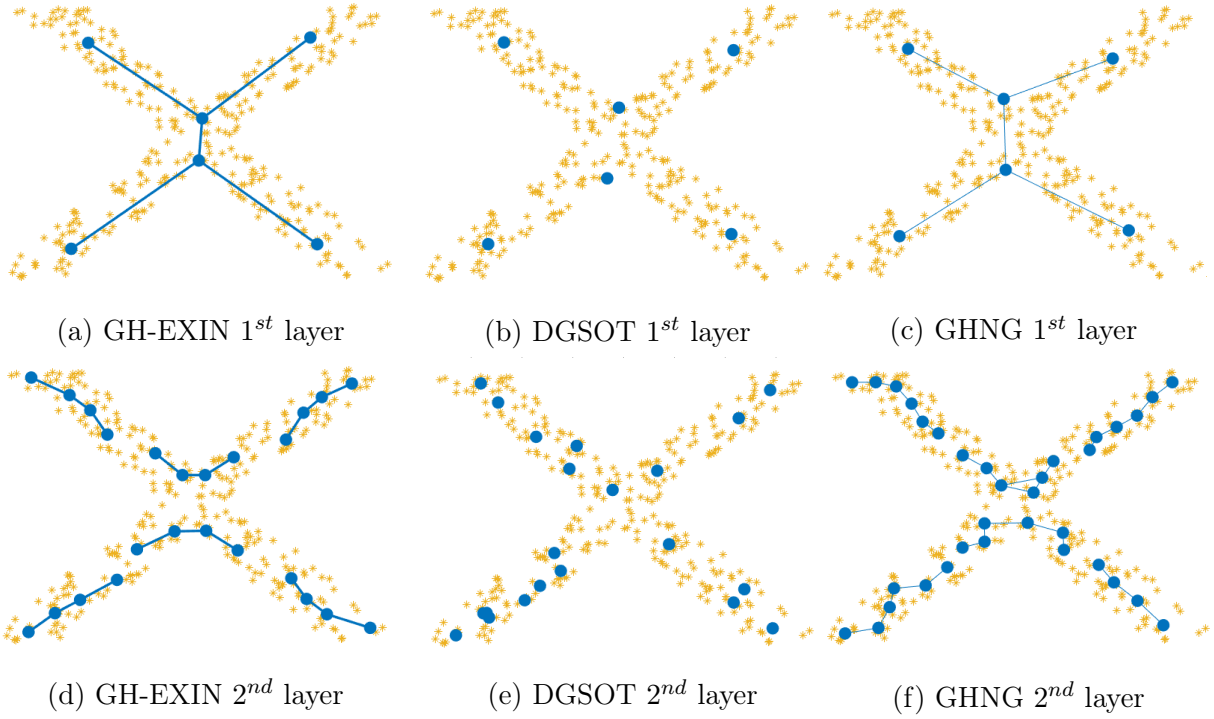
GH-EXIN Horizontal Growth. The horizontal growth of GH-EXIN corresponds to the training of an sG-EXIN neural network (see Fig. 7), which means presenting all the father node Voronoi set to the sG-EXIN neural unit for several epochs. In other words, it implies to repeat the sG-EXIN iteration for an number of times equal to one epoch (i.e. the cardinality of the Voronoi set) and then, repeat this procedure for the necessary number of epochs. For considering, in a global way, all the horizontal growths (i.e. sG-EXIN training) of a single level of the hierarchy, the average number of epochs, J, is considered here. Furthermore, in the worst case all leaves become father nodes; then, the sG-EXIN networks are trained on input sets whose cardinality sums exactly to N. As a consequence, a GH-EXIN horizontal growth costs $O(JNb)$ (neuron pruning and outlier reallocation have a negligible cost).

GH-EXIN Cost. The overall complexity of GH-EXIN can be estimated by considering the cost of repeating a full horizontal growth (i.e. expansion of all the leaves) for all the levels of the hierarchy, that is, the height h of the tree. According to the previous considerations, $h = log_b M$, and $M = O(N)$ then the overall training is $O(b * J * N * log_b N)$. Note that both J and b are usually smaller compared to N; it implies that J and b can be considered as constants. The overall GH-EXIN complexity is then $O(N * log_b N)$.

3.7.2. Complexity Comparison

As pointed out in [24], DGSOT has the same cost as GH-EXIN. However, the DGSOT analysis of the cost in [24] does not take into account the complexity of the horizontal growth. Indeed, both the approach in [24] based on the Minimum Spanning Tree and the approach in [17] based on the CS index, are very time consuming. Hence, the DGSOT complexity is probably underestimated.
On the contrary, GHNG is cheaper. Indeed, its cost is $O(N)$, according to our personal analysis, because there is no complexity estimation of this algorithm in the literature. However, a so simple technique prevents from building a really adaptive tree: the neuron creation does not depend on the data at hand, the tree height is predetermined (the number of levels is a hyperparameter) and the horizontal growth is only controlled by the leaf minimum cardinality. The latter is probably the worst problem, because it tends to flatten the hierarchical tree, in the sense that all the detected subgroups in a cluster are represented in the same level, even if they still contain nested levels.

4. Synthetic Experiments



(a) GH-EXIN 1^{st} layer (b) DGSOT 1^{st} layer (c) GHNG 1^{st} layer

(d) GH-EXIN 2^{nd} layer (e) DGSOT 2^{nd} layer (f) GHNG 2^{nd} layer

Figure 8: First **(up)** and second **(down)** layers of GH-EXIN, GHNG and DGSOT on the X-shape distribution

GH-EXIN, GHNG and DGSOT are here tested on artificial datasets, for comparing their performances. At first, the same two planar datasets used in [15] have been chosen both for the analysis of their partitioning properties and for direct visual inspection. With regard to their ability in building a hierarchical tree, a third database has been expressly created.

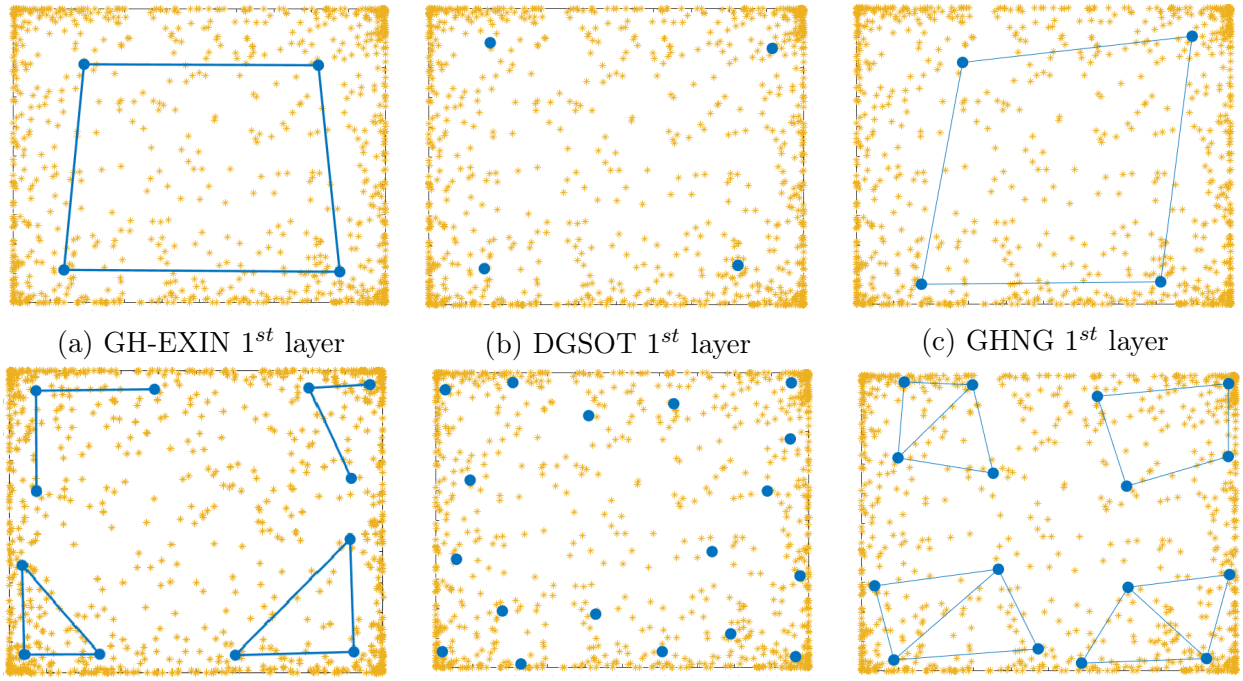
The first two databases are randomly drawn from i) a planar uniform X-shape manifold and ii) a planar square-shaped manifold having a Beta distribution (higher density in the borders). The parameters of each network, used in these experiments and in the next ones, are reported in Tabs. 1, 2 and 3.

For evaluating the quality of clustering, some internal indexes are used: the peak-signal to noise ratio ($PSNR$) index [15], the Davies–Bouldin index (DB) [25] and the global Silhouette value (S) [26].

The $PSNR$ index is defined as follows (in decibel, higher is better):

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \tag{3}$$

where MAX_I^2 is the squared Euclidean norm of the vector which joins the two most distant points in the input distribution support and MSE is the Mean Squared Error computed as the sum of the Euclidean distances between the reference vector of each leaf neuron and its associated data. $PSNR$ takes into consideration only the intra-cluster compactness,



(a) GH-EXIN 1^{st} layer (b) DGSOT 1^{st} layer (c) GHNG 1^{st} layer

(d) GH-EXIN 2^{nd} layer (e) DGSOT 2^{nd} layer (f) GHNG 2^{nd} layer

Figure 9: First **(up)** and second **(down)** layers of GH-EXIN, GHNG and DGSOT on the square distribution

while ignoring the inter-cluster separation. For this reason, it is not a very accurate index of the clustering quality. However, it is here introduced because of its use in [15].

The DB index, instead, takes into consideration both aspects, and is defined as follows:

$$DB = \frac{1}{N} \sum_{i=0}^{N} \max_{j \neq i} \frac{RMSE_i + RMSE_j}{D_{i,j}} \tag{4}$$

where $RMSE_i$ is the Root Mean Squared Error for the ith cluster, $D_{i,j}$ is the Euclidean distance between the centroids of the ith and jth clusters and N is the number of clusters. The lower the value, the better is DB.

Also the S index takes into account both the intercluster and the intracluster distances and is computed as follows:

$$S = \frac{1}{C} \sum_{i=1}^{C} \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{5}$$

where $a(i)$ is the average distance of the ith point from the points in the same cluster, while $b(i)$ is the minimum among the average distances of the ith point from the points in the other clusters and C is the cardinality of the current dataset. The Silhouette index, in general, is defined for each point in the dataset. Hence, the average value is considered. While DB aims at identifying sets of clusters that are compact and well separated, the S

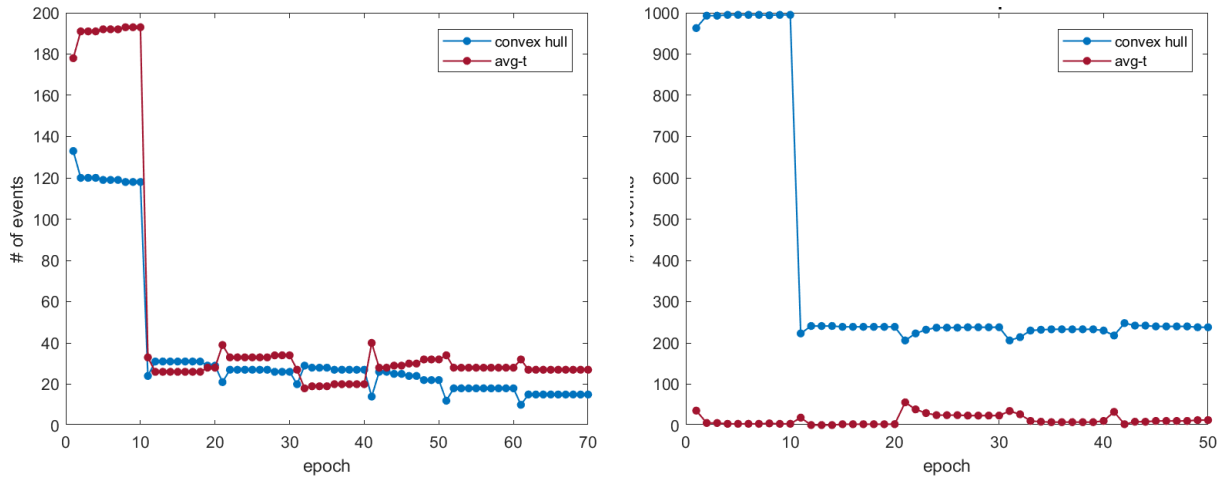
index is more suitable for estimating if, on average, samples are correctly assigned to the nearest neighbouring cluster.

On the X-shape distribution (Fig. 8), at the first layer, the three algorithms have learnt approximately well the manifold. With regard to the second level of the hierarchy, GH-EXIN uses less neurons than GHNG for covering the manifold. Furthermore, the proposed approach represents the symmetry of the manifold using symmetric branches composed of the same number of neurons. On the contrary, GHNG neurons and edges do not respect the symmetry (branches have not the same number of neurons). Also there are some higher neuron densities in branches. More specifically, DGSOT yields the worst results in terms of symmetry (e.g. the overlap of three neurons in the SW branch of the manifold), because of the absence of connections. For instance, the SW branch and the NE branch have eight and three neurons, respectively. Bar plots in Fig.16 report some statistics about the algorithms and the quality of the clustering. Results have been validated by running all the algorithms 10 times for each data distribution and the bar plots report the mean value and the standard error mean. With regard to the X-letter experiment, GH-EXIN uses less neurons on average and takes a few more seconds to end. In this experiment, the quality of GH-EXIN and GHNG clusterings is similar for all indexes, except for DB, which is slightly worse for GH-EXIN. They both overcome DGSOT results (S and DB). Nonetheless, GHNG and DGSOT are less stable than GH-EXIN in terms of number of neurons created. While this does not seem to affect GHNG performance on average, DGSOT DB index, instead, changes too much at each run.

On the square dataset (Fig. 9), with regard to the first layer, GH-EXIN and DGSOT distribute the four neurons in a more symmetrical way with respect to GHNG. However, the GH-EXIN distribution represents better the rectangle manifold (isosceles trapezoid). Concerning the second layer, GH-EXIN uses less neurons than GHNG and DGSOT for covering the manifold. Further, the proposed approach represents the symmetry of the manifold, correctly placing nodes along the sides of the squares, according to the point distribution, while mostly ignoring the emptier central part. On the contrary, GHNG and above all DGSOT place many neurons also in the central part and do not respect the symmetry of the dataset. From a quantitative point of view, GH-EXIN is the best algorithm for all indexes but $PSNR$, while using far fewer neurons. Nevertheless, also in this case it takes longer to terminate than DGSOT and, above all, GHNG. On this dataset, algorithms seems to be quite stable on average.

With regard to the GH-EXIN novelty test, Figs. 10(a) and 10(b) show the importance of the convex-hull mechanisms with regard to the isotropic threshold for explaining the X-shape and the square distribution, respectively. The convex-hull criterion is extensively exploited for the square distribution because of the importance of the border.

Resuming, in both experiments GH-EXIN yields the best clustering, as confirmed by the visualization, and in terms of the S and DB indexes. It requires fewer neurons, but is more time consuming.



(a) X-shape distribution (b) Square distribution

Figure 10: Number of times the two novelty test approaches are called during the training phase (see Paragraph 3.2.1 and Fig. 3). The blue line represents the number of calls to the convex-hull technique (see Fig. 2), while the red line refers to the isotropic threshold criterion (see Equation 2). Each node training lasts ten epochs. The first ten epochs refer to the root node, while the following ones to the second layer nodes

4.1. Hierarchical Synthetic Experiment

The previous experiments highlight the performance in quantization of the three algorithms for each level. However, these techniques have been conceived not for partitional, but for hierarchical clustering. The proposed experiment checks for the quality of the estimated tree, by using, as a ground truth, a predefined hierarchical clustering. At this aim, a dataset composed of two Gaussian mixture models has been devised: the first model is made of three Gaussians, the second one of four Gaussians, as shown in Fig. 11.

The results, visualized in Fig.12 whose trees are shown in Fig. 13, clearly show that only GH-EXIN and DGSOT build the correct hierarchy: two nodes in the first layer (level), which represent the two clusters, and as many leaves as Gaussians in the second layer, which represent the mixtures. Neurons are also positioned correctly w.r.t. the centers of the Gaussians. On the contrary, GHNG spreads all the information in the first level. In this sense, it only partitions, but does not reveal the hierarchy. The quality indexes, as illustrated in Fig. 16, only refer to the final partition. They show a slightly better PSNR and S for GH-EXIN, but a similar DB. This assessment is supported also by the analysis of the Silhouette plots reported on Fig. 14, which shows the S values for each neuron: the S values for GH-EXIN are mostly positive and the other values are only slightly negative, unlike the other two methods. All the algorithms require a similar number of neurons, but GHNG is much faster.

Resuming, GHNG is a faster algorithm, but is better for partitional clustering, even if it has been conceived for finding hierarchies in the data. It opens the question if its performance simply results from its modified GNG module.

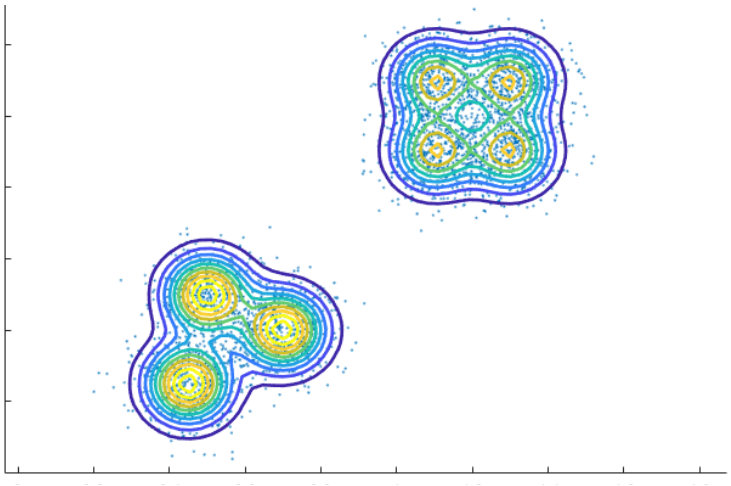
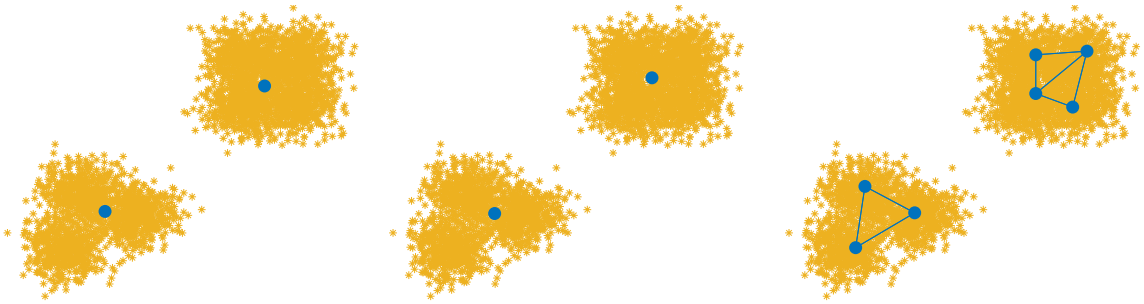
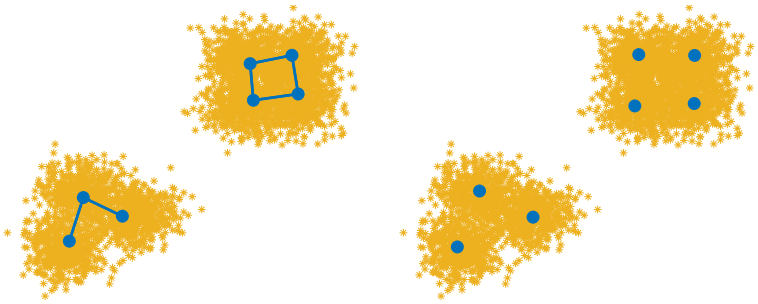


Figure 11: Two mixtures of Gaussians: data and contours



(a) GH-EXIN 1^{st} layer (b) DGSOT 1^{st} layer (c) GHNG 1^{st} layer

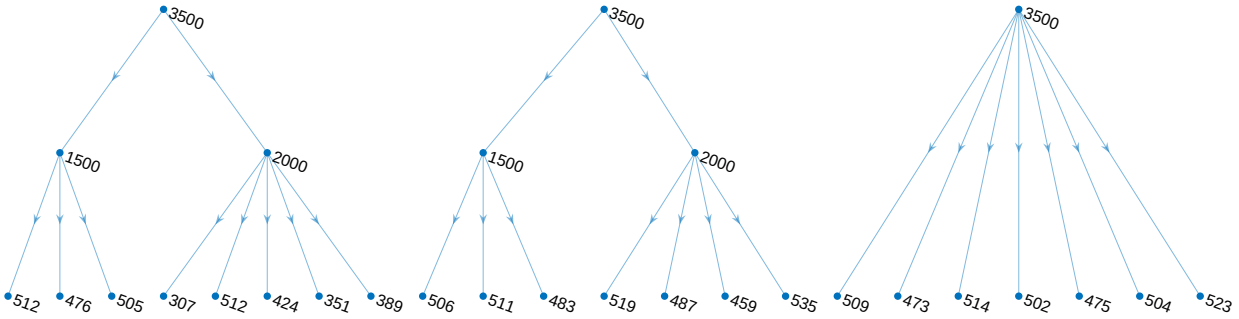


(d) GH-EXIN 2^{nd} layer (e) DGSOT 2^{nd} layer

Figure 12: First **(up)** layers of GH-EXIN, DGSOT and GHNG and second **(down)** layers of GH-EXIN and DGSOT on the Gaussian distribution; GHNG second layer is not reported as it already covers all Gaussian distribution at the first level

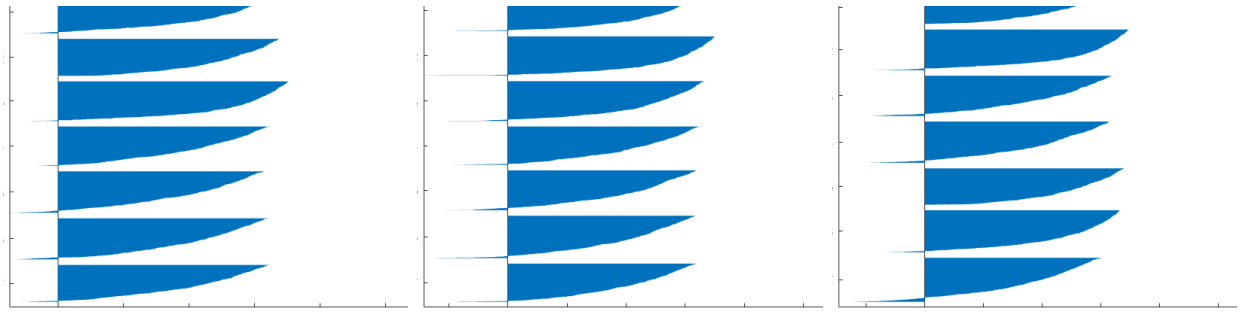
5. Hierarchical Clustering for Video Sequences

The first real experiment is the same proposed in [15], where GHNG is compared to GHSOM, GNG and SOM (GHNG is comparable to GNG and better than GHSOM and SOM). It has been devised for checking the quality of the hierarchical clustering. A database



(a) GH-EXIN tree (b) DGSOT tree (c) GHNG tree

Figure 13: Tree structures (labelled by the cluster cardinality) of the three algorithms for the Mixture of Gaussians dataset

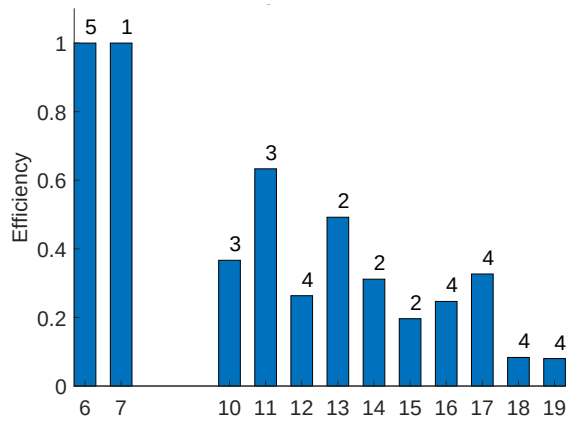


(a) GH-EXIN Silhouette (b) DGSOT Silhouette (c) GHNG Silhouette

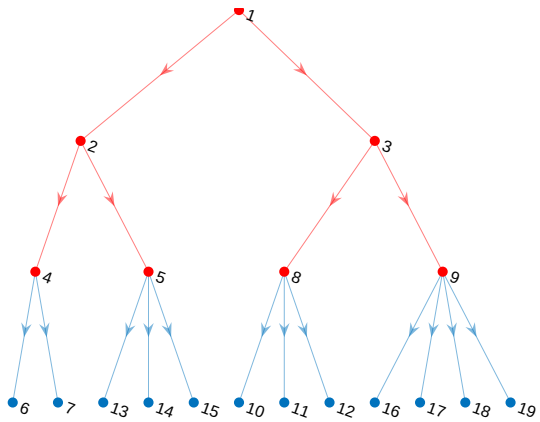
Figure 14: Silhouette scores for the three algorithms on the Mixture of Gaussian Dataset

[27] composed of five video sequences, each representing exclusively either one of four people (two men, classes 2 and 4, and two women, classes 1 and 5) or one container (class 3), is used, with the goal of grouping frames of the same class. There are 1432 input frames of dimensionality 25344 (176×144 pixels), each with 3 channels (RGB). At first the color images are converted to grayscale images by eliminating the hue and saturation information while retaining the luminance. Because of the high dimensionality of data, the inputs are linearly projected to dimension 8 by using the Principal Component Analysis (PCA), performed by the eigenface method [28]. The projection retains 83% of the original data variance.

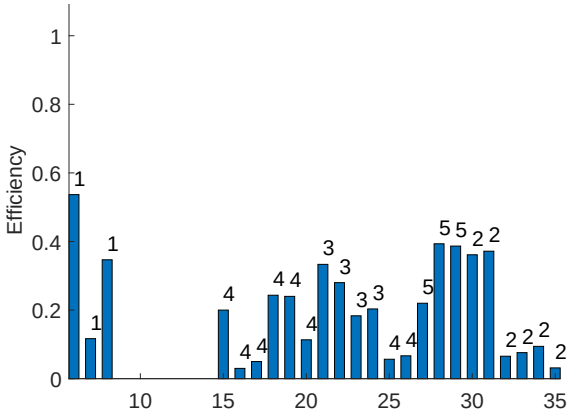
Fig.15 shows the hierarchical trees (labeled by the numbers of the nodes and leaves) and the associated best leaf efficiencies. The efficiency of a class in a cluster is defined as the percentage (w.r.t. the whole database) of elements of the class in the cluster. The best efficiency reported for each leaf in Fig.15 is the maximum of these values (the number on the top of the bar corresponds to the class) and represents an external qualitative index of the clustering. It has been observed that all leaves of the GH-EXIN and DGSOT trees have a 100% purity, while a few GHNG leaves do not share this property, where the purity is defined as the percentage of elements in a cluster belonging to the most common class. The following conclusions about the experiment can be drawn.



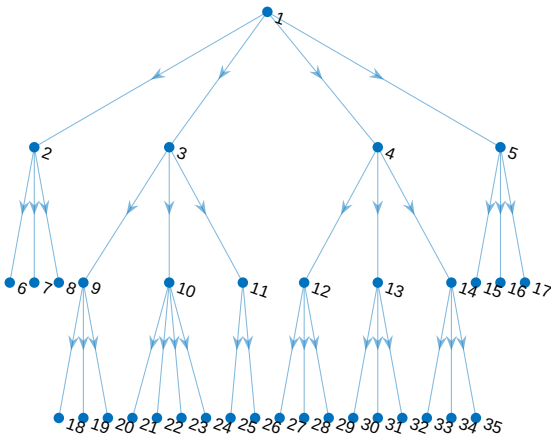
(a) GH-EXIN leaves efficiency



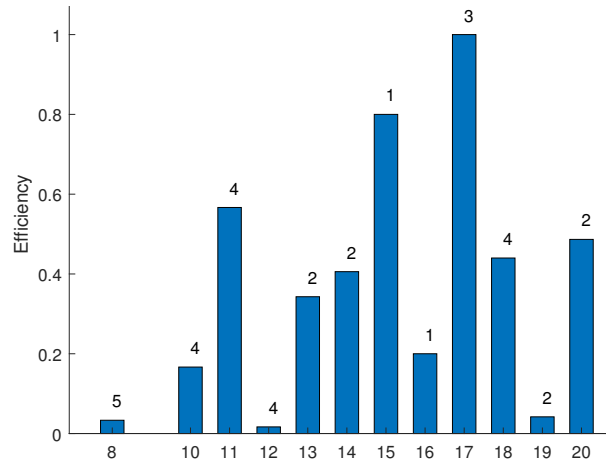
(b) GH-EXIN tree



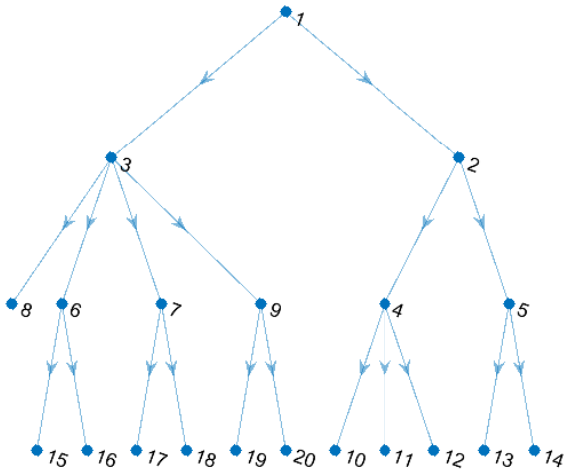
(c) DGSOT leaves efficiency



(d) DGSOT tree



(e) GHNG leaves efficiency



(f) GHNG tree

Figure 15: Leaf efficiency (left) and tree structure (right) of the three algorithms for the videos dataset. Regarding the efficiency bar plot, the row represents the number of the corresponding leaf on which the efficiency has been computed. Each bar is labelled on top by the class of the group of data with the highest efficiency within the leaf.

- GH-EXIN nodes 2 and 3 have been built by the double simultaneous vertical growth (red edges in Fig.15(b)) and reflect the fact two chains have been found in the global dataset. The node 2 Voronoi set only contains data from classes 1, 2 and 5. The node 3 Voronoi set is only composed of data from classes 3 and 4. Hence, the first level of the GH EXIN tree perfectly divides in two clusters.

- DGSOT, which has not this double vertical mechanism, has a first level which is comparable with the second level of GH-EXIN. DGSOT finds exactly all data of class 1 in node 2, all data of class 3 and 65% data of class 4 in node 3, all data of class 2 and 5 in node 4 and 35% data of class 4 in node 5. Resuming, node 2 is 100% pure and efficient, but node 5 is 100% pure and only 35% efficient.

- The GH-EXIN second level finds exactly all data of class 2 in node 5, all data of class 3 and 25% data of class 4 in node 8, all data of class 1 and 5 in node 4 and 75% data of class 4 in node 9. W.r.t. DGSOT, node 9 is 100% pure but 75% efficient for the same class of DGSOT node 5. Also, node 4 perfectly identifies the two women classes.

- The GH-EXIN third level and the DGSOT second level neurons have Voronoi sets composed of a unique class. However, 65% class 4 data for DGSOT and 25% class 4 data for GH-EXIN are nested in another cluster (together with class 3).

- The GHNG tree first level does not compute a correct clustering. Instead, node 2 retains a portion of class 2 and 4 data, with the consequence that these classes will be grouped in the next levels by clusters in different branches. The absence of a reallocation tool in the algorithm prevents from correcting this problem.

- The GHNG tree second level has more nodes than the same level for the other two algorithms. Class 2 is shared, nearly fifty-fifty, by node 9 and node 5, belonging to different branches. The same can be repeated for node 4 and 7 w.r.t. class 4. Class 1 is perfectly retrieved in node 6, while class 3 is only retrieved at the third level. The worst result is yielded in the third level by node 8, which only retains less than 0.1% class 5 data, thus preventing a correct clustering of class 5 in the tree. Indeed, the node 8 Voronoi set is empty (this is allowed by the GNG algorithm[1]). However, this value of efficiency derives from the recall phase, in which it is possible that an *empty* neuron wins because it has moved in the Voronoi set of another neuron. The same considerations can be repeated for the empty neuron 12. On the contrary, this problem is avoided in GH-EXIN because of the reallocation technique.

With regard to the results in Fig.16, GH-EXIN requires fewer neurons, before being automatically stopped. GHNG, as usual, is by far the fastest. The quality indexes, which

[1]GNG creates a neuron in the middle between the father and the mother neurons. Its position does not depend on the presence of a data. It is linked to its parent neurons. If it never wins, but one of its two neighbors wins, it is possible that it changes position (SCL) and approaches data of another cluster, which will not be presented anymore to the network. Hence, it remains empty, but may win in the recall phase.

do not take into account the hierarchy, but only the quality of the final partitioning, show comparable PSNR and a far better S index for GH-EXIN, despite the better DB for DGSOT. Notice the very high value of DB for GHNG. It has been found experimentally that it is related to the presence of empty neurons.

Summing up the previous observations, GH-EXIN and DGSOT find an optimal hierarchical clustering. GH-EXIN is also able to detect the difference between male and female faces. On the contrary, GHNG yields a very poor hierarchy. Probably, this explains why, in [15], the authors of GHNG do not show the entire tree, but only the results of some nodes, with the associated leaves.

Table 1: GH-EXIN hyperparameters

	H_{max}	H_{perc}	$\alpha_{\gamma 0}$	α_{i0}	age_{max}	min_{card}
X-shape	0.00002	0.9	0.1	0.01	5	10
Square	0.00002	0.9	0.35	0.001	10	30
Gaussians	0.001	0.9	0.5	0.05	5	300
Videos	0.8	0.9	0.8	0.1	20	10

Table 2: DGSOT hyperparameters

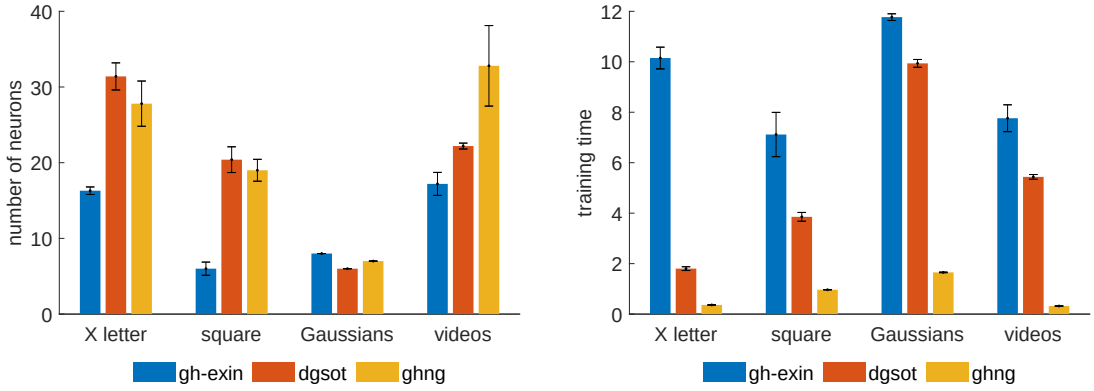
	α	σ_0	T_R	T_E	ϵ_{AD}	ϵ_{ET}	K
X-shape	0.2	1	0.3	10	0.046	0.03	1
Square	0.1	1	0.001	10	0.09	0.03	0
Gaussians	0.2	1	200	2	0.2	0.05	1
Videos	0.2	1	250	2	0.2	0.05	1

Table 3: GHNG hyperparameters

	MAX_{LEVEL}	τ	λ	ϵ_B	ϵ_N	α	A_{max}	D
X-shape	2	0.25	100	0.1	0.01	0.5	50	0.995
Square	2	0.3	100	0.35	0.01	0.5	50	0.995
Gaussians	2	0.1	100	0.4	0.01	0.5	14	0.995
Videos	3	0.2	100	0.001	0.001	0.5	50	0.995

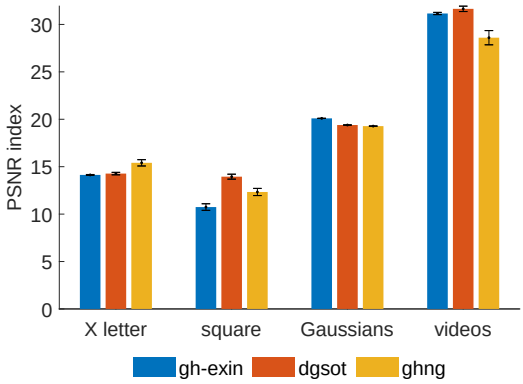
6. Application to Two-Way Clustering

The second real experiment deals with an application in a high-dimensional space, where data manifolds are embedded in subspaces. This is a more challenging problem with regard to normal clustering, because also the more meaningful features have to be identified. While this approach is called subspace clustering, there is no universal consensus about the definition of the corresponding techniques. In a way, it depends on how it is implemented: two-way clustering, if a clustering technique is alternatively applied both in the row and in

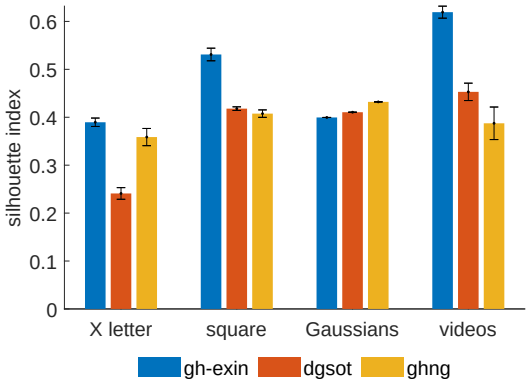


(a) Number of neurons

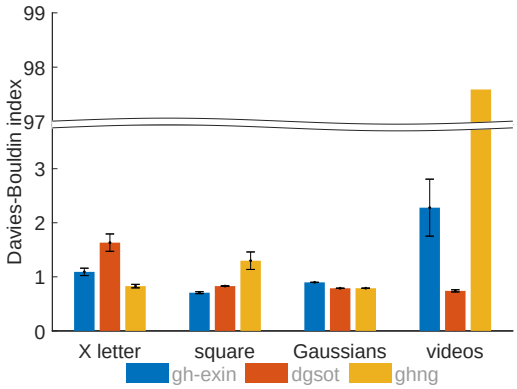
(b) Training time



(c) PSNR index



(d) Silhouette index



(e) Davies-Bouldin index. Notice that the s.e.m. error bar of GHNG on the videos dataset is not reported as it would have impaired the visualization of the other bars

Figure 16: Bar plots with the standard error of the mean (s.e.m.) bars showing the mean of some relevant statistics for each dataset and for each neural network

the column space of the matrix representing the dataset [29]; biclustering, if these operations are carried out at the same time [19].

Two-way clustering searches for biclusters with constant values, with constant values on rows or columns and with coherent values, respectively. It can be proved that the rank of the corresponding submatrices is less than or equal to three in the noiseless case. Hence, the numerical rank can be used as a figure of merit of the quality of the bicluster. The H_{cc} index has been chosen for controlling the quality of the bicluster as it also takes into account the noise in data. It is expressed as:

$$H_{cc} = \frac{\sum_i^{N_r} \sum_j^{N_c} r_{ij}^2}{N_r N_c} \tag{6}$$

where N_c represents the total number of columns of the matrix, N_r represents the total number of rows and $r_{i,j}$ is the residue, which is calculated as:

$$r_{ij} = a_{ij} - \frac{\sum_k^C a_{ik}}{C} - \frac{\sum_h^R a_{hj}}{R} + \frac{\sum_i^R \frac{\sum_j^C a_{ik}}{C}}{R} \tag{7}$$

The components a_{ij} are the elements of the matrix representing the dataset. C and R are the number of columns and of rows of the bicluster at hand, respectively. The second term is the average value of the ith row, the third term is the average value of the jth column, while the last one is the average value of the whole bicluster. This index decreases as the values in the bicluster tend to be constant, differing for a constant on the rows or a constant on the columns. It goes to zero for the trivial *1*x*1* bicluster. This fact implies an additional control on the cardinality of the biclusters in order to avoid this drawback.

GH-EXIN, driven by H_{cc}, is applied alternatively to the rows and columns, as shown in Algorithm 1. It is not necessary to require the same minimum cardinality for both rows and columns. The column clustering can be considered as a feature selection step, which corresponds to an orthogonal projection in the column space. In this sense, this approach can be also named as *projected clustering*, because it does not allow overlapped biclusters.

6.1. Application to Gene Expression Analysis

The cancer phenomenon seems to be the result of a different sequence of genetic alterations. In this difficult setting, clinical treatments add an external complexity to the tumor behavior. In recent years, Patient-Derived Xenografts (PDXs) have emerged as powerful tools for biomarker discovery and drug development in oncology [30][31][32]. The PDX technology has been leveraged to conduct large-scale preclinical analyses to identify reliable correlations between genetic or functional traits and sensitivity to anti-cancer drugs. In this context, during the last decade, the Cancer Institute of Candiolo (IRCC, Italy) has assembled the largest collection of PDXs from metastatic colorectal cancer (mCRC) available worldwide in an academic environment. Such resource has been widely characterized at the molecular level through the Illumina bead array technology [33] and has been annotated for response to therapies, including cetuximab, an anti-EGFR antibody approved for clinical use [34], [35], [36]. The data consists of a DNA microarray, with the expression of 20.023 genes

Algorithm 1 two-way (projected) clustering pseudo-code

```
1: two-way clustering:
2: clustering on rows
3: for all leaves do
4:     if leaf.cardinality ≤ min_card1 then
5:         skip leaf
6:     else
7:         clustering on the columns of the leaf (projection)
8:         for all leaves do
9:             if projectedLeaf.cardinality ≤ min_card2 then
10:                skip leaf
11:            else
12:                save projected leaf
13:                goto two-way clustering
14:            end if
15:        end for
16:    end if
17: end for
18: return
```

in 403 CRC murine tissues. Each cancerous tissue is associated with a Boolean variable describing the tumor response to cetuximab (responsive or not responsive to treatments), as described in previous works [37], [38], [39].

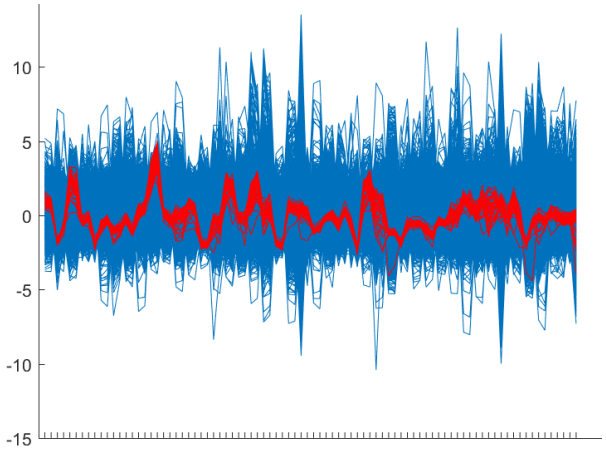
6.1.1. Neural Framework

The two-way clustering approach works on a matrix whose rows are given by the genes and the columns by the murine tissues, while the entries are the gene expressions. Considering the very large number of genes, the first hierarchical clustering is in the row space. In order to better analyze genetic expressions common for different patients, the dataset has been divided into three parts (classes). This division follows the murine tissues response to anti-cancer drugs. At the end, three datasets have been derived, one for the mice which started recovering after three weeks of treatments, a second one for the mice which had a stable situation and at last one also for the case in which drugs had no effect and the cancer kept growing. After a second hierarchical clustering on the column space, the resulting biclusters are analyzed.

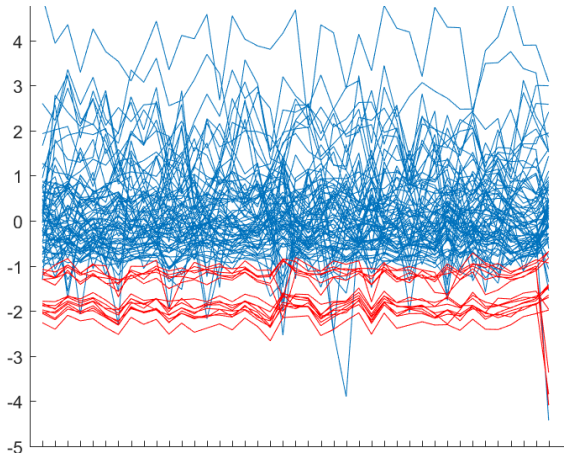
6.1.2. Analysis of the Results

In order to validate the two-way clustering of GH-EXIN (first clustering: $H_{max} = 0.1$, $H_{perc} = 0.9$, $\alpha_{\gamma 0} = 0.8$, $\alpha_{i0} = 0.08$, $age_{max} = 20$, $min_{card} = 20$; second clustering: $H_{max} = 0.001$, $H_{perc} = 0.5$, $\alpha_{\gamma 0} = 0.5$, $\alpha_{i0} = 0.05$, $age_{max} = 3$, $min_{card} = 20$), the parallel coordinate technique [40] is used. Fig. 17(a) shows this kind of plot by visualizing genes as samples (colored polylines) and murine tissues as features (parallel vertical axes) on a leaf of GH-EXIN in the gene space, whose characteristics are shown in the top line of the figure. Blue polylines represent all genes available in the dataset, while red polylines stand for

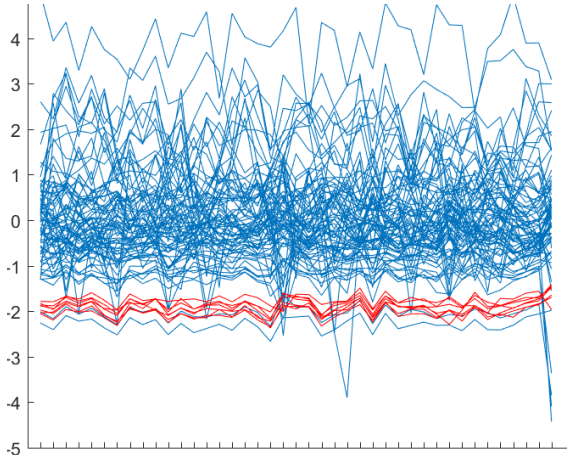
genes collected in the gene cluster. The red grouping of polylines shows coherency, thus confirming the quality of gene clustering. A similar validation analysis is used after the GH-EXIN clustering in the tissue space which is run after projecting the Voronoi set of the 19th gene leaf (cluster), as shown in Figs. 17(b) and 17(c).



(a) Parallel coordinates of a cluster of genes



(b) Parallel coordinates of a bicluster



(c) Parallel coordinates of a smaller bicluster

Figure 17: Parallel coordinates

6.1.3. Biological Analysis

As a biological feedback, the scientific relevance of the selected genes has been taken into account. Among all the biclusters found, the one that grouped the most interesting genes in the cancer field has also the lowest H_{cc} index value. Indeed, the 7 genes found in the bicluster are the following:

- "CSAG1", "CSAG3", "CSAG3A", which belong to the same CSAG family. These genes are well known in medical literature as associated with chondrosarcomas, but they are also found in normal tissues. Furthermore, CSAG3 and CSAG3A are genes

coding the "Chondrosarcoma-associated gene 2/3 protein", which is a "drug-resistance related protein, its expression is associated with the chemotherapy resistant and neoplastic phenotype. May also be linked to the malignant phenotype" [41].

- "MAGEA2", "MAGEA3", "MAGEA12", "MAGEA6", which belong to the same MAGEA family. These genes are melanoma antigens which "Reduce p53/TP53 transactivation function" and also "Repress p73/TP73 activity" [42]. Both p53 and p73 are tumor suppressor proteins which regulate cell cycle and induce apoptosis.

This analysis suggests that, at least in the observed conditions, these gene families are not only important by themselves, but may also co-regulate each other. It is also important to notice that this bicluster phenomenon has been observed within the tissues belonging to the third class, to which tissues unresponsive to drugs belong.

7. Conclusion

Complex data, especially in high dimensional spaces, are better analyzed by the hierarchical (multiresolution) clustering. Divisive algorithms are better suited for this task, but have drawbacks that can be overcome by unsupervised neural networks. The GH-EXIN neural network is a novel algorithm for building a hierarchical tree. It is based on a growing self-organizing neural unit (s-GEXIN) and is based on new ideas:

- semi-isotropic novelty detection, by employing the neighborhood convex hull, for a better representation of the data manifold;
- original data reallocation, for self-correcting tree building and for outlier detection;
- double vertical growth, for exploiting the neuron topology.

This network requires only a few user-dependent parameters and so it is easy to be calibrated. Unlike most algorithms of the same kind, a complete analysis of its complexity is here reported.

For assessing the validity of GH-EXIN, two important hierarchical clustering algorithms have been chosen for comparison. The first, GHNG, is based on an algorithm which has some points in common with GH-EXIN (e.g., SCL, HCL), but, above all, is better than other neural networks for clustering, like SOM, GH-SOM and GNG. The second, DGSOT, is well suited for applications in medicine, and is an improvement of the important algorithm SOTA. These three algorithms have been tested on more and more difficult synthetic and real problems, in order to check their hierarchical properties. GH-EXIN has shown to be better than DGSOT and far better than GHNG. In the end, GH-EXIN has also been used for an important application of two-way clustering, and the achieved results are very promising.

Future work will deal with the analysis of nonstationary data and with the "simultaneous" biclustering by using a variable distance metric.

Bibliography

[1] B. Everitt, S. Landau, M. Leese, D. Stahl, Cluster Analysis, Wiley Series in Probability and Statistics, Wiley, 2011.
URL https://books.google.it/books?id=w3bE1kqd-48C

[2] T. Li, Y. Tang, S. Suen, L. Fang, A. Jennings, A Structurally Adaptive Neural Tree for Recognition of Large Character Set, Proceedings of the 11th IAPR International Joint Conference on Pattern Recognition 2 (1992) 187 – 190. doi:10.1109/ICPR.1992.201751.

[3] R. Adams, K. Butchart, N. Davey, Hierarchical Classification with a Competitive Evolutionary Neural Tree, Neural Networks 12 (3) (1999) 541 – 551. doi:https://doi.org/10.1016/S0893-6080(99)00010-6.
URL http://www.sciencedirect.com/science/article/pii/S0893608099000106

[4] E. Samsonova, J. Kok, A. Ijzerman, TreeSOM: Cluster Analysis in the Self-Organizing Map, Neural networks : the official journal of the International Neural Network Society 19 (2006) 935–49. doi:10.1016/j.neunet.2006.05.003.

[5] J. Himberg, A SOM Based Cluster Visualization and its Application for False Coloring, IEEE Int. Joint Conf. on Neural Networks 3 (2000) 587 – 592 vol.3. doi:10.1109/IJCNN.2000.861379.

[6] M. Venkat Reddy, M. Vivekananda, R. U. V. N. Satish, Divisive Hierarchical Clustering with K-means and Agglomerative Hierarchical Clustering, International Journal of Computer Science Trends and Technology (IJCST) 5.

[7] G. Hang, D. Zhang, J. Ren, C. Hu, A Hierarchical Clustering Algorithm Based on K-Means with Constraints, Innovative Computing ,Information and Control, International Conference on 0 (2009) 1479–1482. doi:10.1109/ICICIC.2009.18.

[8] G. Aloysius, Efficient High Dimension Data Clustering using Constraint-Partitioning K-Means Algorithm, International Arab Journal of Information Technology 10.

[9] M. Khalilian, N. Mustapha, N. Suliman, A. Mamat, A Novel K-Means Based Clustering Algorithm for High Dimensional Data Sets, in: International MultiConference of Engineers and Computer Scientists, 2010, pp. 17–19.

[10] A. Forti, G. L. Foresti, Growing Hierarchical Tree SOM: An Unsupervised Neural Network with Dynamic Topology, Neural networks 19 (10) (2006) 1568–1580.

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

[12] V. Burzevski, C. K. Mohan, Hierarchical Growing Cell Structures, in: IEEE int. conference on neural networks, 1996, pp. 207–218.

[13] B. Fritzke, Growing Grid - A Self-Organizing Network with Constant Neighborhood Range and Adaptation Strength, Neural Processing Letters 2 (5) (1995) 9–13. doi:10.1007/BF02332159.
URL https://doi.org/10.1007/BF02332159

[14] A. Rauber, D. Merkl, M. Dittenbach, The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data, IEEE Transactions on Neural Networks 13 (6) (2002) 1331–1341. doi:10.1109/TNN.2002.804221.

[15] E. J. Palomo, E. López-rubio, The Growing Hierarchical Neural Gas Self-Organizing Neural Network, IEEE Transactions on Neural Networks and Learning Systems (2016) 1–10.

[16] B. Fritzke, A Growing Neural Gas Network Learns Topologies, in: Advances in neural information processing systems, 1995, pp. 625–632. arXiv:arXiv:1011.1669v3, doi:doi=10.1.1.31.4273.

[17] L. Khan, F. Luo, Hierarchical Clustering for Complex Data, International Journal on Artificial Intelligence Tools 14 (2005) 791–810.

[18] J. Dopazo, J. M. Carazo, Phylogenetic Reconstruction Using an Unsupervised Growing Neural Network that Adopts the Topology of a Phylogenetic Tree, Journal of Molecular Evolution 44 (2) (1997) 226–233.

[19] Y. Cheng, G. M. Church, Biclustering of Expression Data, Proceedings. International Conference on Intelligent Systems for Molecular Biology 8 (2000) 93–103.

[20] V. Randazzo, G. Cirrincione, G. Ciravegna, E. Pasero, Nonstationary Topological Learning with Bridges and Convex Polytopes: the G-EXIN Neural Network, in: 2018 International Joint Conference on Neural

Networks (IJCNN), IEEE, 2018, pp. 1–6. doi:10.1109/IJCNN.2018.8489186.
URL https://ieeexplore.ieee.org/document/8489186/

[21] M.-R. Bouguelia, Y. Belaid, A. Belaid, Online Unsupervised Neural-Gas Learning Method for Infinite Data Streams, in: Pattern Recognition Applications and Methods, Springer, 2015, pp. 57–70.

[22] G. Cirrincione, V. Randazzo, E. Pasero, The Growing Curvilinear Component Analysis (GCCA) neural network, Neural Networks 103 (2018) 108–117.

[23] G. Ciravegna, P. Barbiero, Gh-exin (version 1.0.1)., https://bitbucket.org/machine_learning_research/ghexin/src/master/ (2018).

[24] F. B. I.-L. Y. Feng Luo, Latifur Khan, J. Zhou, A dynamically growing self-organizing tree (DGSOT) for hierarchical clustering gene expression profiles, Bioinformatics 20 (2004) 2605–2617.

[25] D. L. Davies, D. W. Bouldin, A Cluster Separation Measure, IEEE Transactions on Pattern Analysis and Machine Intelligencedoi:10.1109/TPAMI.1979.4766909.

[26] P. J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematicsdoi:10.1016/0377-0427(87)90125-7.

[27] M. Reisslein, L. J. Karam, P. Seeling, F. H. Fitzek, and T. K. Madsen , YUV Video Sequences , http://trace.eas.asu.edu/yuv/index.html , Accessed on 2019-06-07 (December 2010).

[28] M. A. Turk, A. P. Pentland, Face recognition using eigenfaces, in: Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 1991, pp. 586–591.

[29] G. Getz, E. Levine, E. Domany, Coupled Two-Way Clustering Analysis of Gene Microarray Data, Proceedings of the National Academy of Sciences 97 (22) (2000) 12079–12084. arXiv:https://www.pnas.org/content/97/22/12079.full.pdf, doi:10.1073/pnas.210134797.
URL https://www.pnas.org/content/97/22/12079

[30] M. Hidalgo, F. Amant, A. Biankin, E. Budinská, A. Byrne Phd, C. Caldas, R. Clarke, S. Jong, J. Jonkers, G. Mælandsmo, S. Roman-Roman, J. Seoane, L. Trusolino, A. Villanueva, Patient-Derived Xenograft Models: An Emerging Platform for Translational Cancer Research, Cancer discovery 4 (2014) 998–1013. doi:10.1158/2159-8290.CD-14-0001.

[31] J. Tentler, A. C. Tan, C. D Weekes, A. Jimeno, S. Leong, T. Pitts, J. J Arcaroli, W. A Messersmith, S. Eckhardt, Patient-Derived Tumour Xenografts as Models for Oncology Drug Development, Nature reviews. Clinical oncology 9 (2012) 338–50. doi:10.1038/nrclinonc.2012.61.

[32] A. Byrne Phd, D. Alferez, F. Amant, D. Annibali, J. Arribas, A. Biankin, A. Bruna, E. Budinská, C. Caldas, D. K Chang, R. Clarke, H. Clevers, G. Coukos, V. Dangles-Marie, S. Eckhardt, E. Gonzalez-Suarez, E. Hermans, M. Hidalgo, M. Jarzabek, L. Trusolino, Interrogating Open Issues in Cancer Medicine with Patient-Derived Xenografts, Nature reviews. Cancer 17. doi:10.1038/nrc.2017.85.

[33] Illumina. Array-based Gene Expression Analysis. Data Sheet Gene Expr. (2011).
URL http://res.illumina.com/documents/products/datasheets/datasheet_gene_exp_analysis.pdf

[34] C. Boccaccio, P. Luraghi, V. Bigatto, E. Cipriano, G. Reato, F. Orzan, F. Sassi, F. Bacco, C. Isella, S. Erika Bellomo, E. Medico, P. Comoglio, A. Bertotti, L. Trusolino, A Molecularly Annotated Model of Patient-Derived Colon Cancer Stem-Like Cells to Assess Genetic and Nongenetic Mechanisms of Resistance to Anti-EGFR Therapy, Clinical Cancer Research 24 (2017) clincanres.2151.2017. doi:10.1158/1078-0432.CCR-17-2151.

[35] E. R Zanella, F. Galimi, F. Sassi, G. Migliardi, F. Cottino, S. Leto, B. Lupo, J. Erriquez, C. Isella, P. Comoglio, E. Medico, S. Tejpar, E. BudinskĂˇ, L. Trusolino, A. Bertotti, IGF2 Is an Actionable Target that Identifies a Distinct Subpopulation of Colorectal Cancer Patients with Marginal Response to Anti-EGFR Therapies, Science translational medicine 7 (2015) 272ra12. doi:10.1126/scitranslmed.3010445.

[36] A. Bertotti, E. Papp, S. Jones, V. Adleff, V. Anagnostou, B. Lupo, M. Sausen, J. Phallen, C. A Hruban, C. Tokheim, N. Niknafs, M. Nesselbush, K. Lytle, F. Sassi, F. Cottino, G. Migliardi, E. R Zanella, D. Ribero, N. Russolillo, V. Velculescu, The Genomic Landscape of Response to EGFR Blockade in Colorectal Cancer, Nature 526 (2015) 263–267. doi:10.1038/nature14969.

[37] C. Isella, F. Brundu, S. E. Bellomo, F. Galimi, E. Zanella, R. Porporato, C. Petti, A. Fiori, F. Orzan, R. Senetta, C. Boccaccio, E. Ficarra, L. Marchionni, L. Trusolino, E. Medico, A. Bertotti, Selective Analysis of Cancer-Cell Intrinsic Transcriptional Traits Defines Novel Clinically Relevant Subtypes of

Colorectal Cancer, Nature Communications 8. doi:10.1038/ncomms15107.
[38] P. Barbiero, A. Bertotti, G. Ciravegna, G. Cirrincione, E. Pasero, E. Piccolo, Unsupervised Gene Identification in Colorectal Cancer, in: Quantifying and Processing Biomedical and Behavioral Signals, Springer International Publishing, 2018, pp. 219–227.
[39] B. Pietro, C. Gabriele, E. Piccolo, C. Giansalvo, C. Maurizio, B. Andrea, Neural biclustering in gene expression analysis, in: 2017 International Conference on Computational Science and Computational Intelligence (CSCI), 2017, pp. 1238–1243. doi:10.1109/CSCI.2017.361.
[40] E. J. Wegman, Hyperdimensional data analysis using parallel coordinates, Journal of the American Statistical Associationdoi:10.1080/01621459.1990.10474926.
[41] http://www.genecards.org/cgi-bin/carddisp.pl?gene=CSAG3, accessed: 2017-11-20.
[42] http://www.genecards.org/cgi-bin/carddisp.pl?gene=MAGEA2, accessed: 2017-11-20.