DOI: https://doi.org/10.1016/j.ins.2018.12.007

Spark-GHSOM: Growing Hierarchical Self-Organizing Map for Large Scale Mixed Attribute Datasets

Ameya Malondkar[a], Roberto Corizzo[b,c], Iluju Kiringa[a], Michelangelo Ceci[b,c], Nathalie Japkowicz[d]

[a] *School of Electrical Engineering and Computer Science (EECS) University of Ottawa - Ottawa, Canada.*
[b] *Department of Computer Science, University of Bari Aldo Moro - Bari, Italy*
[c] *National Interuniversity Consortium for Informatics (CINI) - Rome, Italy*
[d] *Department of Computer Science, American University - Washington D.C*

Abstract

The Growing Hierarchical Self-Organizing Map (GHSOM) algorithm has shown its potential for performing several tasks such as exploratory analysis, anomaly detection and forecasting on a variety of domains including the financial and cyber-security domains. GHSOM is a dynamic variant of the SOM algorithm which generates a multi-level hierarchy of SOM maps based solely on input data. However, in order to generate this multi-level structure, GHSOM requires multiple iterations over the input dataset, thus making it intractable on large datasets. Moreover, the conventional GHSOM algorithm is designed to handle datasets with numeric attributes only. This represents an important limitation as most modern real-world datasets are characterized by mixed attributes - numerical and categorical. In this work, we propose an extension of the conventional GHSOM algorithm called Spark-GHSOM, which exploits the Spark platform to process massive datasets in a distributed manner. Moreover, we leverage a method known as the *distance hierarchy* approach to modify the optimization function of GHSOM so that it can (also) coherently handle mixed-attribute datasets. We test our new method with respect to accuracy, scalability and descriptive power. The results obtained using different datasets demonstrate the superior predictive

Email addresses: ameya.malondkar@gmail.com (Ameya Malondkar), roberto.corizzo@uniba.it (Roberto Corizzo), iluju.Kiringa@uottawa.ca (Iluju Kiringa), michelangelo.ceci@uniba.it (Michelangelo Ceci), japkowic@american.edu (Nathalie Japkowicz)

and descriptive capabilities of Spark-GHSOM, as well as its applicability to large-scale datasets which could not be analyzed before.

Keywords: GHSOM, Self-Organizing Map, Clustering, Distributed, Big Data

---

1. Introduction

The Self-Organizing Map (SOM) [27] is a neural-network-based clustering and data analysis algorithm that operates by mapping high-dimensional input data onto a 2-dimensional space implemented by a grid of neurons called a feature map. The shape and number of neurons in the feature map need to be established prior to training the SOM. This is difficult when nothing is known about the underlying dataset. Furthermore, hierarchical relations within the dataset cannot be identified when using conventional SOM, as they appear in the same feature space.

The Growing Hierarchical Self-Organizing Map (GHSOM) was proposed in [11] to overcome these limitations. The GHSOM corresponds to a multilevel tree-like architecture that consists of individual SOMs. The size of each SOM layer adapts to the data incident on it. This way, the hierarchical structure of the GHSOM provides a zooming capability: starting from the first map which represents the complete data, one can zoom into the next hierarchical layer to see the finer details of the dataset. Thus, the GHSOM has the capability to grow horizontally to adapt to the underlying dataset as well as to expand vertically to provide details at finer levels of granularity. The GHSOM has been successfully used in many applications such as exploratory analysis [39], knowledge representation in the legal field [42] [45], financial diagnosis [43], financial fraud detection [51, 20], network anomaly detection [10] and product demand forecasting [31].

One shortcoming of the GHSOM is the fact that it can only process numerical attributes. Yet, many real world datasets contain a mixture of numerical and categorical attributes. This is not a trivial problem since handling categorical attributes along with numerical ones would require new optimization functions that coherently consider any type of attributes so that no bias related to the attribute type is introduced in the learning phase. A second, and probably even more limiting, problem is that GHSOM involves multiple passes over the input dataset. This makes it intractable on large datasets if applied sequentially on a single machine.

We overcome these limitations by introducing a new optimization function based on the *distance hierarchy*, originally presented in [19]. The approach provides a uniform method for handling both the numeric and categorical components of a mixed attribute dataset. Our second contribution is a new and distributed version of the GHSOM algorithm, called Spark-GHSOM, which is capable of analyzing massive datasets representing real-world use cases. This was achieved by formulating the two dimensional and hierarchical growth processes in terms of Spark's *map* and *reduceByKey* functions. Combining our two contributions, we propose a new hierarchical clustering algorithm able to process large data sets composed of both numeric and categorical attributes accurately and efficiently.

The paper is structured as follows: in Section 2 we briefly review existing work related to this paper. Section 3 introduces background notions necessary for the description of our work along with the original GHSOM algorithm. In Section 4, we propose and describe our distributed Spark-GHSOM algorithm. In Section 5,we describe our experimental evaluation method and present the results obtained in these experiments. Specifically, the results show that our method is able to make accurate predictions and is efficient when dealing with massive data sets. Section 6 concludes the paper and suggests new directions for future work.

2. Related work and contribution

In the literature, there is always significant interest in new algorithms for learning SOMs, mainly due to their versatility in different tasks and fields. For example, the authors in [16] exploit the characteristic of SOMs to preserve the topological properties of the input data, to generate a set of uniformly distributed weight vectors, which can be integrated with decomposition-based algorithms tailored for solving many-objective optimization problems. SOMs are also powerful in data collection and exploration tasks. [12] exploits SOMs as an autonomous data collection tool from predeployed sensors, applying it to the traveling salesman problem. [40] implements SOMs in a time series analysis framework, which supports the analyst with data exploration, cluster visualization, and pattern identification capabilities. [1] exploits SOMs in combination with crisp sets as an efficient information retrieval system. [7] addresses the problem of identifying complex spatio-temporal patterns in groundwater data with SOMs. The proposed soft-computing methodology is capable to display complex high-dimensional data sets into visible topo-

logical maps, which accurately represent the current status of groundwater resources, providing a way for their future management. [36] adopts SOMs to seek for non-linear and complex relationships in ecological data.

Lately, SOMs have also been exploited to support supervised learning tasks. [50] accomplishes near-optimal classification of industrial polymer species employing SOMs to separate polymers of different types, and then using K-Means to separate polycarbonate and polystyrene. [4] proposes coalitions and complex network topologies for connecting the neurons within a SOMs, which are then used to perform time series prediction tasks. [34] focuses on rock penetrability classification exploiting Artificial Bee Colony (ABC) and SOM algorithms. The study demonstrates that SOM can be of valuable help also in classification tasks. [55] exploits SOMs for modeling the operating conditions of an industrial process reflected by available auxiliary signals. The Best Matching Unit is subsequently fed in a neuro fuzzy inference system tailored for forecasting tasks.

All these recent works, are directly related to our paper for multiple reasons: the analysis of spatio-temporal and sensor data, the adaptation to the analysis of time series and the usage of SOMs for supervised learning tasks. These aspects, have been considered in our paper because of the different tasks presented in Section 5.

In the remaining part of this section we focus on works that are similar to ours from a *methodological* viewpoint. Specifically, we first discuss some works which extend GHSOM in the direction of managing mixed types of attributes and then mention some works that provide distributed implementations of SOM (there is no work, to the best of our knowledge, which proposes a distributed implementation of GHSOM).

2.1. Extending GHSOM for Mixed Attributes

The original GHSOM algorithm is applicable to datasets with numeric attributes only. One of the most popular and naïve approaches for extending an algorithm to mixed attribute domains is *One Hot Encoding*. In this approach, categorical attributes are transformed into a set of binary attributes, where the number of binary attributes created is equal to the number of discrete values in the domain of the categorical attribute. After the transformation, all binary attributes are treated as numerical attributes and the dataset can be processed normally. However, the resulting dataset has an increased dimensionality which results in high computational complexity of the learning phase. Moreover, the transformed dataset is very sparse and

an addition of a new value to the domain of categorical attributes needs a change in schema of the dataset.

Another popular approach is *Simple Matching* in which two categorical values have a distance of 0 if they are the same, 1 otherwise. This approach, has been originally used in k-modes [22] and k-prototypes [21] algorithms, which are variants of the k-means clustering algorithm for handling categorical and mixed-attribute datasets, respectively. For updating the cluster centres in the variants of k-means algorithm, a frequency-based approach has been adopted. The value of the j^{th} categorical attribute of a cluster centre is set to the most frequent value of the j^{th} attribute for the instances in that cluster. Following this initial research, a similar simple matching approach for dealing with categorical attributes in the SOM training has been proposed in NCSOM [8]. In this work, however, the computed distances are not fractional (i.e. they are either 0 or 1, with no intermediate values). This inevitably leads to a loss in precision.

Tai et al. [49] address the problem of clustering mixed attributes datasets by devising a distance measure which considers information embedded in concept hierarchies, to properly find similarities between the data instances and neurons. Alternatively, SCM models [13] have been proposed to address symbol strings clustering by extracting a lattice of nodes on a 2D map. Each node of the SCM is associated with a symbol string and a weight vector. TCSOMs [18] have been proposed for clustering transaction data with categorical attributes. Differently from SOMs, the update phase involves only the winner neuron. Similarly, [29] proposes a distance measure which takes information embedded in concept hierarchies into consideration for clustering transactional data. The major drawbacks of these methods, however, is that they typically focus on a specific data type, and none of them addresses the large-scale data clustering issue, thus resulting ineffective with real world massive datasets.

Hsu [19] proposed a distance hierarchy method for handling categorical attributes in serial SOM. This variant of SOM was called GSOM (Generalized Self Organizing Map). The approach extends the concept hierarchy technique [17] by giving weights to the links. It provides a method for calculating the distance between mixed, numeric and/or categorical data in a uniform manner. This is accomplished by mapping the values to the distance hierarchy of attributes and calculating their distance in the hierarchies. The distance hierarchy approach provides a better representation of the similarities or dissimilarities between the categorical values. For example, if we

consider a categorical attribute of Drink with values Mocha, Expresso, Pepsi, Coke, Mocha and Expresso (types of coffee) are more similar to each other than Mocha when compared with Pepsi (carbonated drink). Obviously, to compute the distance between such categorical values, a distance hierarchy for each categorical attribute must be provided in advance. Similar values according to the concept hierarchy are placed under a common parent (Coffee or Carbonated Drink) which represents an abstract concept. This work, although relevant for the study reported in our paper because we use the same distance hierarchy, does not propose to learn Hierarchical Self-Organizing Maps and learning is not distributed.

2.2. Distributed Implementations of SOM

There are different recent works involving SOM and GHSOM. Some approaches tackle the issue of treating categorical data, while others propose distributed implementations.

The work in [54] presents a MapReduce variant of SOM implemented on the Hadoop framework. The number of key-value pairs shuffled from the map to the reduce task is equal to $n \cdot k$, where n is the total number of input instances and k is the number of neurons in the layer. This algorithm does not use the concept of *combiner*, which would greatly reduce the number of key-value pairs transmitted over the network.

The authors in [48] overcome the limits of the serial SOM algorithm exploiting an MPI implementation (Message-Passing Interface) called MapReduce-MPI to distribute the SOM training process in a cluster environment. However, these approaches are implemented in MapReduce, which presents the limitation of iterative disk spill of data partitions, which reduce the efficiency of the computation [44].

The study in [41] is the first attempt to implement the SOM algorithm in Apache Spark. In the paper, the authors propose two implementations: the first one is similar to the one mentioned in [54] where the number of key-value pairs generated is $n \cdot k$, where n is the total number of input instances and k is the number of neurons in the SOM layer. In the second algorithm, the output of the mapper task is a matrix constituted by rows of input vectors multiplied by the neighborhood factor and a neighborhood vector consisting of the neighborhood factors. The size of the matrix is $k \cdot n$ where k is the number of neurons and n is the number of input vectors, while the size of neighborhood vector is equal to the number of neurons.

All the papers mentioned do not tackle the problem of treating categorical data. Moreover, they can only learn SOM models and not a hierarchy of models as in the GHSOM algorithm. In fact, to the best of our knowledge, a distributed implementation of GHSOM has never been proposed in the literature.

Other variants of SOM and GHSOM tackle the problem of mixed attributes, but they are capable to work only on a single machine (see, for instance [19] [8]), therefore, they are limited in terms of amount of data that can be processed.

This paper makes two contributions to the existing literature: 1) it proposes a distributed implementation of hierarchical SOM; and 2) it proposes an implementation for handling mixed attributes in the hierarchical context.

The former, performs all training steps exploiting *map* and *reduce* transformation functions in Apache Spark. Particular effort has been put to exploit data locality where possible, thus performing local aggregations on worker nodes, minimizing network communication costs and computational load on the driver node.

The latter, leverages the distance hierarchy approach to modify the optimization function of GHSOM in order to coherently handle mixed-attribute datasets during the step of SOM adaptation. Moreover, the distance hierarchy approach has been exploited as a distance measure between an instance and a neuron in a SOM, required for the identification of the winner neuron during the training process.

The same measure has been also profitably exploited for predictive purposes as discussed in Section 5, where we evaluate the accuracy of our algorithm in regression and forecasting tasks, also focusing on industrial problems of practical importance, such as energy forecasting.

Moreover, a qualitative evaluation has been carried out, which demonstrates the usefulness of the proposed algorithm for data exploration tasks. Finally, the efficiency of the algorithm has been assessed with a scalability evaluation.

3. Background: Growing Hierarchical Self-Organizing Map (GHSOM)

In this section, we provide a brief theoretical background of the SOM and the GHSOM algorithms. A SOM layer consists of a grid of neurons

where each neuron is associated with a weight vector. The training of the SOM involves two basic steps. In the first step, an instance is provided to the SOM and the neuron with the shortest distance from the input instance is selected as the *winner* neuron. In the next step, the *winner* neuron and its surrounding neighbor neurons are adapted towards the input instance. This training process requires a defined number of iterations over the input dataset called *epochs*.

The GHSOM extensively uses a metric called the Mean Quantization Error (MQE) [39] [32]. The MQE of a neuron is the total deviation of the neuron from its mapped input instances, whereas the MQE for a SOM layer is the average MQE of all the neurons representing instances. For the GHSOM training, first, the MQE of the level-0 neuron, mqe_0 is calculated with respect to all the input instances. The first neuron map is created at level-1 consisting of 2×2 neurons. This level-1 map is trained using the conventional SOM training process. After the training is complete, the map is analyzed and the MQE for the map MQE_m is computed. A higher value of the MQE_m signifies that the map m does not represent the input data well and requires more neurons to better represent the input domain. Formally, this is governed by Equation 1 (τ_1 criterion):

$$MQE_m < \tau_1 \cdot mqe_p, \text{ with } 0 \leq \tau_1 \leq 1 \tag{1}$$

where, mqe_p is the MQE of the parent neuron from which this map m is expanded. The map grows until the condition in Equation 1 is satisfied. To grow the map, the neuron with the highest MQE is identified as the *error* neuron e. Then, its most dissimilar direct neighboring neuron d is selected and a new row or column of neurons is inserted into the grid between e and d. The vectors of these new neurons are initialized as the average of the weight vectors of their corresponding adjacent neighbors. This process produces an updated (grown) layer, which is then trained and analyzed again. These iterations of growth and training continue until the τ_1 criterion is satisfied.

Once the τ_1 criterion is satisfied, each neuron in the map is analyzed according to the criterion in Equation 2 (τ_2 criterion):

$$mqe_k < \tau_2 \cdot mqe_0, \text{ with } 0 \leq \tau_2 \leq 1. \tag{2}$$

The neurons which do not satisfy the τ_2 criterion are expanded into new maps at the next level of hierarchy. These new maps undergo the same

process of training, growth and hierarchical expansion as the level-1 map. The training of the GHSOM stops when all the neurons in the lowest level maps satisfy Equation 2. The resulting GHSOM structure thus contains multiple SOM layers arranged in a hierarchy with each SOM representing the data at a finer granularity than its parent layer.

4. Method

In this section we discuss how we redesigned the GHSOM algorithm in order to process mixed attributes datasets and to distribute the computation. For the first aspect, the main difficulty comes from the computation of a hierarchical distance function which can be used to modify the optimization function of GHSOM. For the second aspect, the main difficulty comes from the implementation, using the Spark framework, of the adaptation of the SOM, where several partial updates have to be efficiently combined to reach a complete adaptation of the SOM from epoch to epoch.

4.1. Extending the GHSOM algorithm for mixed attributes

The first step in the GHSOM algorithm is to compute the inherent dissimilarity in the input data set indicated by mqe_0 (mean quantization error of the level 0 neuron). In fact, in the original GHSOM, to compute mqe_0 for the level-0 neuron, we need to compute the mean of all input vectors m_0 in the dataset. We then compute the mean distance of the input vectors from the mean vector to obtain the value of mqe_0, which is defined as follows:

$$mqe_0 = \frac{1}{n} \sum_{x_{(\cdot,i)} \in C_n} \left|\left| m_0 - x_{(\cdot,i)} \right|\right|, \tag{3}$$

where C_n is the set of all n input instances, $x_{(\cdot,i)}$ is the vector which represents the i-th instance and mqe_0 represents the overall dissimilarity in the input dataset.

Extending this definition in order to deal with mixed attributes is not straightforward because we have to replace the computation of the mqe_0 with a measure that is valid for both numeric and categorical attributes. The main problem is that there is no standard definition of *mean* for categorical attributes. For this reason, we replaced the mean quantization error by variance and used it as a measure for assessing the quality of map and neurons.

Variance, while being a valid way to compute the deviation in the input (and more robust to outliers than mean quantization error), has also a counterpart for categorical attributes. In fact, [24] states that, for categorical attributes, "unalikeability" is a good measure to estimate the "variation about mean" or, more precisely, how often the categorical values differ from one another.

The *coefficient of unalikeability* for a categorical attribute l is defined as:

$$u_2(l) = \sum_{i \in Domain(l)} p_i(1 - p_i) \tag{4}$$

where $p_i = \dfrac{frequency(l_i, C_n)}{n}$, l_i is the i-th value of the attribute l in its domain and $frequency(l_i, C_n)$ is the absolute frequency of l_i for the attribute l in C_n. This equation is similar (and inspired by) the "within data" variance for continuous attributes defined as follows:

$$W(l) = \frac{\sum_{i=1}^{n} \sum_{k=1}^{n} (x_{(l,i)} - x_{(l,k)})^2}{(n^2 - n)}. \tag{5}$$

where $x_{(l,i)}$ represents the value of attribute l for the i-th instance. It can be proved (see [15]) that with a simple shift of values of the attribute l such that the average $m_{0l} = \frac{1}{n} \sum_{i=1}^{n} x_{(l,i)} = 0$, we have that $W(l) = 2 \cdot Var(l)$, where $Var(l)$ is the classical variance.

Formulae (4) and (5) represent deviation of input from the mean or a reference, they are defined for categorical and continuous attributes (resp.), they have an intuitive interpretation and, more importantly, they are comparable. Hence, we use *variance* as a measure of dissimilarity in place of the *mean quantization error* in the GHSOM training. This means that the total variance in the input data var_0 can be computed as:

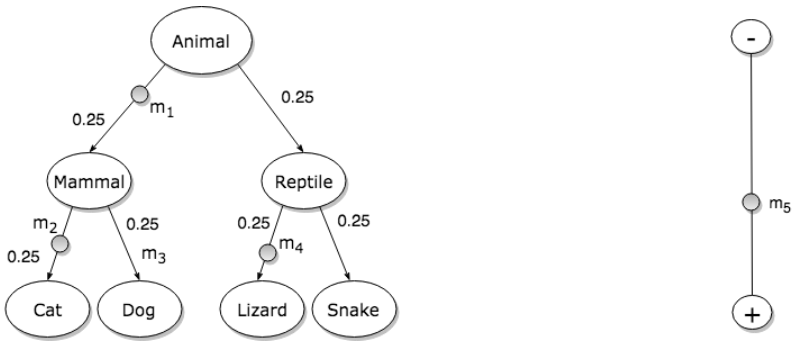
$$var_0 = \sum_{l=1}^{L} \left(\mathbb{1}^{numerical(l)} \cdot Var(l) + \mathbb{1}^{categorical(l)} \cdot \frac{u_2(l)}{2} \right) \tag{6}$$

where, L is the number of input attributes, $\mathbb{1}^{numerical(l)}$ (resp. $\mathbb{1}^{categorical(l)}$) is 1 if l is numerical (resp. categorical), 0 otherwise.

4.1.1. Using distance hierarchy *in GHSOM for mixed attributes*

The following step in the GHSOM training process consists, according to the competitive learning setting, in finding the winner neuron for each input instance, i.e., the least distant neuron from the input instance. Both *simple matching* and *distance hierarchy* techniques support this operation for mixed attributes. In this work we adopt the *distance hierarchy* approach, since it provides a uniform mechanism for handling numeric and categorical attributes.

A distance hierarchy is a tree-like structure with a root and several leaves as shown in Fig. 2. In the distance hierarchy, the value of an instance attribute is represented by a point in the tree. A point X is represented by a pair of (N_X, d_X) where N_X is the anchor or the leaf symbol and d_X is the offset of the point from the root. Typically, the non-leaf points in the hierarchy represent neurons' attributes while the leaf points represent the instance attributes. During the SOM training, when we present an instance to the neuron map, the neuron's weight vector is adapted by a certain amount to match the instance vector. In terms of distance hierarchy, when an instance is presented, it pulls the neuron point towards its leaf.



(a) Distance hierarchy for categorical attribute

(b) Distance hierarchy for numeric attribute

Figure 1: Distance hierarchy for the mixed attribute GHSOM

As stated earlier, training an individual SOM layer is a two step process. In the first step we identify the *winner* neuron $w(x_{(\cdot,i)})$ for the instance $x_{(\cdot,i)}$ at time t.

$$w(x_{(\cdot,i)}) = arg\min_{k} dist(x_{(\cdot,i)}, m_{(\cdot,k)}(t)) \tag{7}$$

where $dist(x_{(\cdot,i)}, m_{(\cdot,k)}(t))$ is the distance between $x_{(\cdot,i)}$ and neuron $m_{(\cdot,k)}$ at time t using the *distance hierarchy*, which can be computed as:

$$dist(x_{(\cdot,i)}, m_{(\cdot,k)}(t)) = \left(\sum_{l=1}^{L} |dh(l, x_{(l,i)}) - dh(l, m_{(l,k)}(t))|^2 \right)^{1/2} \quad (8)$$

where, $dh(l, x_{(l,i)})$ and $dh(l, m_k(t))$ represent the distance hierarchy mappings (for the hierarchy of the l-th attribute) of $x_{(\cdot,i)}$ and m_k at time t and L is the number of attributes. That is, $dh(l, x_{(l,i)}) = (N_{x_{(l,i)}}, d_{x_{(l,i)}})$ and $dh(l, m_k(t)) = (N_{m_{(l,k)}(t)}, d_{m_{(l,k)}(t)})$.

The difference, used in formula (8), between any two points X $(= dh(l, x_{(l,i)}))$ and Y $(= dh(l, m_{(l,k)}(t)))$ in a distance hierarchy is computed using Equation 9.

$$|X - Y| = d_X + d_Y - 2 \cdot d_{LCP(X,Y)} \quad (9)$$

where, X and Y represent distance hierarchy points, d_X and d_Y represent the offset of the points in the distance hierarchy and $d_{LCP(X,Y)}$ is the offset of the *least common point* of X and Y. $LCP(X, Y)$ is defined as one of the following:

1. either X or Y, if X and Y refer to the same point, or
2. X if X is an ancestor of Y, i.e. X lies on the path from the root to Y, or
3. the least common ancestor of X and Y.

For example, with reference to Figure 1a, the distance for the attribute Animal between $m_2(t)$ and an instance for which the value of the attribute "Animal" is "Dog", is 0.375 + 0.5 - 2 * 0.25 = 0.375. Here we assume that the distance between every node and its parent node is 0.25 and that $d_{m_2(t)} = 0.375$ or, in other terms, $m_2(t)$ is in the middle between "Mammal" and "Cat".

Once the winner $w(x_{(\cdot,i)})$ for each instance $x_{(\cdot,i)}$ has been identified, it is used to adapt the whole SOM at epoch t, where the SOM S can be represented by a matrix of neurons of size L (i.e., $S(t) = \{m_{(\cdot,k,k')}(t)\}_{(k,k')}$. If we represent as $m_{(\cdot,k,k')}(t)$ the neuron of size L in position k and k' of S, we have that:

$$m_{(\cdot,k,k')}(t+1) = \left[\frac{\bigoplus_{i=1}^{n} \Big(h(w(x_{(\cdot,i)}), m_{(\cdot,k,k')}(t)) \cdot dh(l, x_{(l,i)}) \Big)}{\sum_{i=1}^{n} \Big(h(w(x_{(\cdot,i)}), m_{(\cdot,k,k')}(t)) \Big)} \right]_{l=1,\ldots,L}, \tag{10}$$

where $h(\cdot,\cdot)$ is defined as:

$$h(m^{(1)}_{(\cdot,k_1,k'_1)}(t), m^{(2)}_{(\cdot,k_2,k'_2)}(t)) = exp\left(-\frac{(|k_1 - k'_1| + |k_2 - k'_2|)^2}{2\sigma(t)^2} \right), \tag{11}$$

and $\sigma(t)$ corresponds to the width of the neighborhood function:

$$\sigma(t) = \left(\frac{\sqrt{R^2 + C^2}}{2} \right) \cdot exp\left(-\frac{t}{numEpochs} \cdot log\left(\frac{\sqrt{R^2 + C^2}}{2} \right) \right), \tag{12}$$

In formula (10) we have at the numerator the multiplication operation between a scalar and a distance hierarchy point. This multiplication returns a new distance hierarchy point where the node remains the same and the offset is the multiplication between the offset value of the distance hierarchy point by the neighborhood factor.

To define the addition operation of two points in the distance hierarchy (used in the summation of the numerator of (10)), we introduce the concept of a *pliable point*. A *pliable point* P is a point in the distance hierarchy that moves along the paths in the hierarchy as other points are applied to it. Initially, the location of P is the root of the tree. Hence, the initial value of P is (N_P, d_P) where the value of $N_P = Any$ and the value of $d_P = 0$. The addition operation of two or more distance hierarchy points can be considered as applying each point one after the other to P and pulling P towards the anchor of the applied point. Formally, this can be written as:

$$(N_A, d_A) \oplus (N_B, d_B) = \begin{cases} (N_A, d_A + d_B) & \text{if } N_A = N_B \\ (N_A, d_A - d_B) & \text{if } N_A \neq N_B \text{ and } d_A \geq d_B \\ (N_B, d_B - d_A) & \text{if } N_A \neq N_B \text{ and } d_A < d_B \end{cases} \tag{13}$$

Similarly, for the division operation (used in (10)), we can divide the offset value by the dividend value.

Intuitively, during the adaption process, the distance hierarchy points for the neurons move towards the respective leaf nodes of the input instance. Let (N_M, d_M) represent the mapping for the neuron M, (N_X, d_X) be the mapping for the instance X, P be the conceptual parent of X at an offset d_P from the root and δ be the adaption amount of M towards X (formally, $\delta = |m_{(l,k,k')}(t+1) - m_{(l,k,k')}(t)|$ due to a single instance). Then, the following cases arise during adaption:

- *Case* 1: When M is in the path between the root and X and P. If, after adjustment $d_M + \delta$, M does not cross over P, then the new value M' is $(N_M, d_M + \delta)$. This is illustrated in Fig. 2a. The anchor of M does not change.

- *Case* 2: When M is in the path between the root and X and P. If, after adjustment $d_M + \delta$, M crosses over P, then the new value M' is $(N_X, d_M + \delta)$. This is illustrated in Fig. 2b. The anchor of M becomes equal to the anchor of X, $N_M = N_X$.

- *Case* 3: When P is in the path between the root and M, M is in the path between P and X (with $(N_M = N_X)$), then the new value M' is $(N_X, d_M + \delta)$. This is illustrated in Fig. 2c. The anchor of M does not change (it is already same as the anchor of X).

- *Case* 4: When P is in the path between the root and M, M is not in the path between P and X $(N_M \neq N_X)$ and M does not cross over P, then the new value M' is $(N_M, d_M - \delta)$. This is illustrated in Fig. 2d. The anchor of M does not change $(N_M \neq N_X)$.

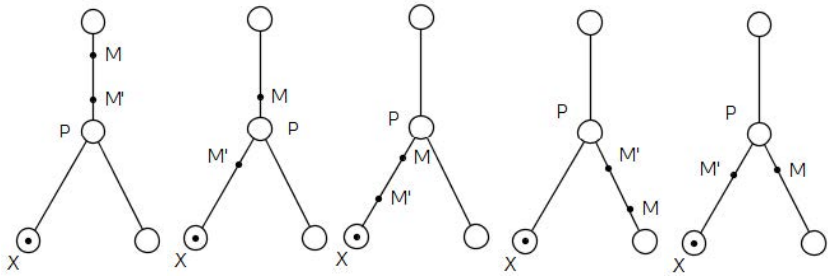
- *Case* 5: When P is in the path between the root and M, M is not in the path between P and X $(N_M \neq N_X)$ and M crosses over P, then the new value M' is $(N_X, 2d_P - d_M + \delta)$. This is illustrated in Fig. 2e. The anchor of M changes to N_X.

Once the SOM has been trained, the growing process, which consists in the addition of new rows and columns in the SOM, and the hierarchical growth, which consists in the creation of new SOMs at a finer level of granularity, follow the classical process of GHSOM described in Section 3.

In the next section, we formally present the distributed algorithms for Spark-GHSOM.

4.2. Distributed Algorithms

In this section, we describe the algorithms for training Spark-GHSOM models, adopting the Apache Spark distributed computing engine.



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4 (e) Case 5

Figure 2: Distance hierarchy for the mixed attribute GHSOM

The *map* function is executed in parallel on the mapper nodes and emits an output or a set of outputs for each input record. The *reduceByKey* function is similar to the *reduce()* function in the conventional Map-Reduce. It works in a distributed manner on reducer nodes and each reducer node is responsible for an exclusive subset of keys. The *reduce()* function in Spark is an aggregation function which combines all the input values and returns a single output, while the *reduceByKey* function returns a single output per input key. The $emit()$ function used in the algorithms, indicate the output produced by the $map()$ and $reduceByKey()$ functions on a per input record or per key basis, respectively.

4.2.1. Complete distributed algorithm

Before the training takes place, the dataset is transformed as follows,

$$dataset(instance) \rightarrow \\ dataset(parent\_layer, parent\_neuron, instance) \tag{14}$$

This is important to maintain mapping information for the hierarchical growth of Spark-GHSOM, since each spawned-off new layer is trained only on the subsets of instances represented by the parent neuron.

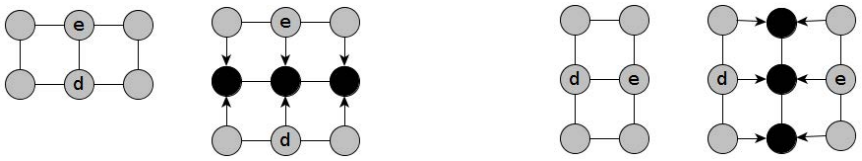
Algorithm 1 outlines the Spark-GHSOM algorithm. The first step computes the overall variance var_0 of the dataset. This can be done using a simple iteration exploiting the *map* and *reduce* functions. Since we train the SOM layers in a breadth-first manner, we maintain a queue to manage these layers. Next, we transform the dataset as per Equation 14. Then we start processing the SOM layers. We first filter out the instances for the current layer. Then we train the individual SOM layer. We describe this SOM training process in the next subsection. Once the SOM layer is trained, it is evaluated for the τ_1 criterion for two-dimensional growth (see Fig. 3). Once

that criterion is satisfied, we evaluate each neuron for the hierarchical growth using the τ_2 criterion. Both growth processes can be achieved adopting the *map* and *reduce* functions. Once we identified the neurons for hierarchical expansion (see Fig. 4), we filter and map the associated instances to the neurons (line 37).

4.2.2. Training of an individual SOM

A high-level algorithm for training an individual SOM is shown in Algorithm 2. First, a 2×2 SOM layer is created. If this is the layer at level 1, then the neuron weight vectors are initialized randomly. For any subsequent levels, the layers can be initialized as a function of their parent neuron and their neighbors. For this we used the approach mentioned in [6]. The training of the SOM involves a defined number of epochs over the dataset. Hence before we start with the training, we cache the dataset in memory (Algorithm 2, line 2). During each epoch, the neuron layer is broadcast to the mapper nodes. The new weight vectors for the neurons (*neuron_updates*) are computed in a *map* and *reduce* iteration. At the end of the epoch, the neuron updates are collected at the driver. The driver updates the neuron layer with the new weight vectors (see Formula (10)) and broadcasts it to the mapper nodes for the next epoch.

Algorithm 3 shows a SOM training iteration using *map* and *reduce* functions. The *map* function computes the numerator and the denominator of (10). It emits a *key-value* pair, where key is the neuron identifier and value is the computed numerator and denominator part. The $reduceByKey$ function computes the updates by collating all the numerator and denominator parts for each neuron efficiently, exploiting the data locality of the records with the same key which are processed on the same node.



(a) Insertion of a new row in a SOM model

(b) Insertion of a new column in a SOM model

Figure 3: Adding a new row or column between the error neuron and the most dissimilar neighbor neuron

Algorithm 1 Spark-GHSOM

function SPARK-GHSOM($dataset, \tau_1, \tau_2, epochs$)
  compute var_0 using *map* and *reduce* functions for Formula 6
  $parent\_queue \leftarrow$ create queue to track parent layer and
    neuron of current layer
  $parent\_layer \leftarrow 0$ ▷ For level-0 layer
  $parent\_neuron \leftarrow 0$ ▷ Only neuron in level-0 layer
  $parent\_queue.enqueue(parent\_layer, parent\_neuron)$
    ▷ for the first level layer
  $mapped\_dataset \leftarrow$ transform dataset as in Equation 14
  $curr\_layer\_id \leftarrow 0$
  while $parent\_queue$ is not empty **do**
    $(curr\_parent\_layer, curr\_parent\_neuron) \leftarrow$
     $parent\_queue.dequeue()$
    $curr\_dataset \leftarrow$ filter instances from $mapped\_dataset$ for
      current layer
    $curr\_dataset.cache()$
    $curr\_layer \leftarrow$ create a new layer of 2×2 neurons
    $curr\_layer\_id \leftarrow curr\_layer\_id + 1$
    $is\_2d\_growth \leftarrow false$
    repeat
      train $curr\_layer$ SOM as in Algorithm 2
      evaluate quality (variance) of $curr\_layer$
      if $curr\_layer$ does not satisfy 2-D growth τ_1 crit. **then**
        grow the layer
        $is\_2d\_growth \leftarrow true$
      else
        $is\_2d\_growth \leftarrow false$
      end if
    until is_2d_growth is $true$
    $expand\_neuron\_set \leftarrow$ create empty set
    for all $neuron \in curr\_layer$ **do**
      if $neuron$ does not satisfy hier. growth τ_2 crit. **then**
        add $neuron$ to $expand\_neuron\_set$
        $parent\_queue.enqueue(curr\_layer\_id, neuron.id)$
      end if
    end for
    $mapped\_dataset \leftarrow$ associate instances to winner neurons
    save the current SOM
    $curr\_dataset.uncache()$
  end while
end function

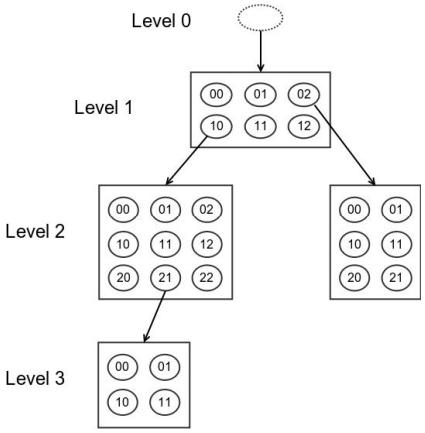


Figure 4: Hierarchical structure of a trained GHSOM model

Algorithm 2 Training of SOM

1: **function** SOMTRAIN($datasetRDD, neuron\_layer, numEpochs$)
2: datasetRDD.cache()
3: initialize $neuron\_layer$
4: **for** $current\_epoch \leftarrow 0, \ldots, numEpochs$ **do**
5: $neuron_updatesRDD \leftarrow$Adapt SOM from $datasetRDD$, $neuron\_layer$ /*(see Algorithm 3)*/
6: apply the updates $neuron\_updatesRDD$ to $neuron\_layer$
7: $current\_epoch \leftarrow current\_epoch + 1$
8: **end for**
9: **end function**

Algorithm 3 SOM Adaptation

function MAP(instance x, $neuron\_layer$)
    find the winning neuron w in the SOM layer for instance x
    for all $m \in neuron\_layer$ **do**
        for all $l \in 1, \ldots, L$ **do**
            $num\_part[l] \leftarrow h(w, m) \cdot dh(l, x)$
        end for
        $den\_part \leftarrow h(w, m)$
        $emit(neuron.id, (num\_part, den\_part))$
    end for
end function

function REDUCEBYKEY($neuron\_id, val\_list =$
        $[(num\_part, den\_part), \ldots])$
    $numerator \leftarrow 0$
    $denominator \leftarrow 0$
    for all $partial\_update \in val\_list$ **do**
        for all $l \in 1, \ldots, L$ **do**
            $numerator[l] \leftarrow numerator[l] \oplus partial\_update.num\_part[l]$
        end for
        $denominator \leftarrow denominator + partial\_update.den\_part$
    end for
    $updated\_weight\_vector \leftarrow numerator \cdot 1/denominator$
    $emit(neuron\_id, updated\_weight\_vector)$
end function

This section outlined the crucial distributed algorithms involved in the training of the complete Spark-GHSOM. It also demonstrated how we leveraged the in-memory data processing capability provided by the Apache Spark computing engine. In the next section we focus on the experimental results obtained using Spark-GHSOM.

5. Empirical Evaluation

In this section, we describe our evaluation approach to assess Spark-GHSOM from different perspectives (accuracy, scalability and descriptive power of the models extracted) and we present the results obtained.

5.1. Accuracy evaluation

In order to assess quantitatively the accuracy of the models extracted by our algorithm, we designed a predictive module which, once the model is built using training data, is capable of assigning a prediction to unlabeled instances at testing time.

Let G be a Spark-GHSOM model built using a training set $T_R = \{x_{(\cdot,i)}\}_{(i)}$: G consists of a set of SOM models ($G = \{S(t)\}_{1 \leq t \leq numEpochs}$) each of which defined as $S(t) = \{m_{(\cdot,k,k')}(t)\}_{(k,k')}$.

We represent winner neurons, namely, neurons which are chosen at least once as best matching units during the training process, as a subset W of all neurons in S. Formally, W is defined as $W = \{w|\ \exists m \in S\ s.t.\ m = w(x_{(\cdot,i)}) \wedge x_{(\cdot,i)} \in T_R\}$ ($w(\cdot)$ is the function which returns the winner neuron).

During the training phase, when a training instance $x_{(\cdot,i)}$ is processed, a winner neuron $m = w(x_{(\cdot,i)})$ is found in the current SOM $S(t)$ considered: it is then possible to assign a target attribute value $x_{(y,i)}$ (y is the index of the target attribute) to m and keep track of any subsequent target attribute assignments received by m using a vector $\mathbf{V_m}$.

At the end of the training process, each neuron m for the SOMs in G might have an empty vector $\mathbf{V_m}$ (if it has never been activated as winner for any training instance) or a non-empty vector ($|\mathbf{V_m}| >= 1$), if it has been activated at least once as winner. In the second case, we assign one definitive target attribute value m_y to m for each $m \in W$, calculated as the average of all assignments received during the training process:

$$m_y = \frac{1}{|\mathbf{V_m}|} \sum_{j=1}^{|\mathbf{V_m}|} \mathbf{V_m}[j] \tag{15}$$

In the prediction phase, let T_S be the set of instances $T_S = \{x_{(\cdot,i)}\}_{(i)}$, for which the value of the (target) attribute y is unknown. Then, the prediction for each instance in T_S can be done in the following way: for each instance $x_{(\cdot,i)} \in T_S$ and for each SOM $S(t)$, we find the closest neuron $w(x_{(\cdot,i)})$ between $x_{(\cdot,i)}$ and all neurons $m \in S(t)$. If $m = w(x_{(\cdot,i)})$ has a target attribute value assigned (according to formula (15)), we assign that value as potential prediction to $x_{(\cdot,i)}$. Otherwise, we find the next closest neuron in the current SOM $S(t)$ according to their distance. This process leads to a candidate neurons set $C(x_{(\cdot,i)})$ consisting of one chosen candidate neuron $w(x_{(\cdot,i)})$ for each SOM $S(t)$. Subsequently, we calculate the distances between $x_{(\cdot,i)}$ and all candidate neurons in C once again and select the closest neuron $c \in C$. The target attribute value of c is then assigned to $x_{(y,i)}$. The algorithm is described in Algorithm 4.

It is noteworthy that all the distances both for training and prediction phases are computed according to formula (8) and, thus, 1) we consider the hierarchy and 2) we work with continuous and categorical attributes indifferently.

Algorithm 4 Prediction Module

function PREDICT($x_{(\cdot,i)}, G$)
  for all *SOM* $S(t) \in G$ **do**
    repeat
      $m = w(x_{(\cdot,i)}) \in S(t)$;
      if m_y is defined **then** add m to C;
      else remove m from neurons considered in $S(t)$;
      end if
    until m_y is defined;
  end for
  $c = \underset{c_{(\cdot,j)} \in C}{\operatorname{argmin}} \{dist(c_{(\cdot,j)}, x_{(\cdot,i)})\}$;
  $emit(c_y)$;
end function

To evaluate the predictive module introduced above, we have focused our attention on two different tasks: regression with mixed attributes datasets and sensor data forecasting, discussed in the following subsections.

For both tasks, the Spark-GHSOM algorithm has been compared with state of the art systems and, specifically, with SVR, linear regression and isotonic regression (distributed implementation available in Spark MLLib [33]). Since Linear regression requires to set the regularization parameter,

Dataset	Features	Numerical	Categorical	Targets
Energy Efficiency	8	6	2	2
Forest	12	10	2	1
Facebook Metrics	7	4	3	2
Automobile	25	15	10	1

Table 1: Brief description of datasets adopted for the regression with mixed attributes datasets task.

we performed a grid search to identify the best value in the following set of choices: $(0.15, 0.3, 0.45)$. Moreover, in order to allow the algorithms to handle the categorical attributes, we adopted the *One Hot Encoding* approach.

Only for sensor data forecasting, we have also used K-Means and ARIMA (which predicts time series) algorithms. In order to obtain predictions from K-Means, we use clusters. In particular, when the value of the target attribute of a new instance has to be predicted, we assign to it the value of the target variable of the closest centroid of the obtained clusters. Also in this case we allow each method to perform in its best conditions. For this purpose, we adopted the following approach:

- **ARIMA**: we select the best ARIMA model according to Akaike's Information Criterion (AIC), as in the AUTO-ARIMA algorithm [23];
- **K-Means**: we select the best value of the parameter k via grid search considering the following set of choices: ($sqrt(n)/8$, $sqrt(n)/4$, $sqrt(n)/2$, $sqrt(n)$, $sqrt(n) * 2$, $sqrt(n) * 4$, $sqrt(n) * 8$), where n is the number of data instances in the training window.

5.1.1. Regression with mixed attributes datasets

We evaluated the performance of Spark-GHSOM in the regression task, using datasets with mixed attributes obtained from the UCI repository, such as Automobile [26], Energy Efficiency [52], Forest [9] and Facebook Metrics [35]. The details can be found in Table 1. In particular, the Facebook Metrics and Energy Efficiency datasets propose more than one target variable which can be chosen for the regression task. Thus, we perform the regression task separately, as shown in Table 3.

Experiments are performed according to a 5 fold cross validation procedure. The final results obtained in terms of the average Root Mean Square

Error (RMSE) and average Mean Average Error (MAE) are presented in Table 2 and 3.

Form the results reported in Table 2, we can see that, globally, the best configuration is $\tau_1 = 0.3$ and $\tau_2 = 0.7$, that is, models with fine granularity and small hierarchies. With these values, in fact, we have the best accuracy for most of the datases.

In Table 3 we report the results of the comparison with other approaches. For a fair evaluation, for all the systems, only the best value obtained varying parameters' configurations is compared. The results show that Spark-GHSOM is capable to achieve a noteworthy improvement margin with respect to other approaches for the considered regression tasks. We observe that only for the Forest dataset, the linear regression method obtains the best performance, although the difference with Spark-GHSOM is only 0.16% in terms of improvement with respect to Polynomial SVR. This is probably due to the intrinsic difficulty of the task, motivated by the fact that none of the methods is able to outperform the others by a significant margin.

5.1.2. Sensor data forecasting

In this section we propose the application of Spark-GHSOM for sensor data forecasting. In details, we focus on the one-day-ahead renewable energy forecasting task. The datasets considered consists of a set of weather variables (such as temperature, humidity, etc.) monitored at hourly granularity (time series) by sensors placed on photovoltaic plants, located in different geographical areas. The task consists in forecasting the production of each photovoltaic plant for the next 24 hours.

We considered three datasets in all. Their description is reported in Table 4. Additional details about the preliminary data preprocessing steps performed for the datasets can be found in [5].

For the evaluation, a random sampling of 10% of the days has been performed for each dataset, considered as testing days. For each testing day, a time based sliding window model [14] has been adopted to train the model considering only the previous 60 days of historical data collected by the sensors. The trained model has been employed to predict the following 24 hours of the target variable (*power*) considering the geographical coordinates (latitude and longitude) of the plants and the weather conditions of the next day as independent variables.

In Table 5 we report the results which allow us to investigate the sensitivity of the system to the values of τ_1 and τ_2. They show that, also for this

Dataset: Energy Efficiency				**Target:** heating load					
	τ_2=0.7			τ_2=0.5			τ_2=0.3		
	RMSE	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
τ_1=0.7	0.1252	0.0950	75.80	0.1252	0.0950	75.80	0.1252	0.0950	75.80
τ_1=0.5	0.1202	0.0896	76.77	0.1190	0.0896	77.00	0.1252	0.0950	75.80
τ_1=0.3	**0.1044**	**0.0664**	**79.82**	0.1066	0.0668	79.40	**0.1044**	**0.0664**	**79.82**
Dataset: Energy Efficiency				**Target:** cooling load					
	τ_2=0.7			τ_2=0.5			τ_2=0.3		
	RMSE	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
τ_1=0.7	0.1146	0.0856	74.50	0.1146	0.0856	74.50	0.1146	0.0856	74.50
τ_1=0.5	0.1050	0.0790	76.63	0.1146	0.0856	74.50	0.1146	0.0852	74.50
τ_1=0.3	0.1038	0.0696	76.90	**0.1024**	**0.0694**	**77.21**	0.1028	0.0688	77.12
Dataset: Forest				**Target:** area					
	τ_2=0.7			τ_2=0.5			τ_2=0.3		
	RMSE	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
τ_1=0.7	0.0516	0.0178	78.65	**0.0514**	**0.0174**	**78.73**	**0.0514**	**0.0174**	**78.73**
τ_1=0.5	0.0524	0.0182	78.32	0.0526	0.0182	78.23	0.0524	0.0180	78.32
τ_1=0.3	0.0566	0.0180	76.58	0.0572	0.0188	76.33	0.0558	0.0180	76.91
Dataset: Facebook Metrics				**Target:** lifetime_post_consumers					
	τ_2=0.7			τ_2=0.5			τ_2=0.3		
	RMSE	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
τ_1=0.7	0.0692	0.0406	32.59	0.0738	0.0452	28.11	0.0768	0.0478	25.19
τ_1=0.5	0.0702	0.0406	31.62	0.0676	0.0394	34.15	0.0684	0.0400	33.37
τ_1=0.3	**0.0676**	**0.0380**	**34.15**	0.0694	0.0398	32.39	0.0684	0.0396	33.37
Dataset: Facebook Metrics				**Target:** total interactions					
	τ_2=0.7			τ_2=0.5			τ_2=0.3		
	RMSE	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
τ_1=0.7	**0.0526**	**0.0262**	**15.93**	0.0534	0.0262	14.65	0.0532	0.0258	14.97
τ_1=0.5	0.0558	0.0268	10.82	0.0542	0.0264	13.37	0.0544	0.0266	13.05
τ_1=0.3	0.0574	0.0274	8.26	0.0556	0.0270	11.14	0.0564	0.0276	9.86
Dataset: Automobile				**Target:** price					
	τ_2=0.7			τ_2=0.5			τ_2=0.3		
	RMSE	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
τ_1=0.7	0.1394	0.0896	58.48	0.135	0.0832	59.79	0.1378	0.0838	58.96
τ_1=0.5	0.1272	0.082	62.11	0.1272	0.082	62.11	0.1272	0.0820	62.11
τ_1=0.3	**0.1250**	**0.0692**	**62.77**	**0.1250**	**0.0692**	**62.77**	**0.1250**	**0.0692**	**62.77**

Table 2: Sensitivity analysis of Spark-GHSOM for the regression task with mixed attributes datasets, considering different values of τ_1 and τ_2. The improvement (%) is calculated with respect to the polynomial SVR baseline method. Best results for each dataset are highlighted in bold.

Dataset:	Energy Efficiency			Energy Efficiency			Forest		
Target:	heating load			cooling load			area		
Method	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
SVR (Linear)	>1	>1		>1	>1		>1	> 1	
SVR (Polyn.)	0.5176	0.4411		0.4495	0.3702		0.2417	0.1945	
SVR (Sigm.)	0.2734	0.2475	47.17	0.2568	0.2316	42.85	0.0511	0.0125	78.84
Isotonic Reg.	0.1472	0.1084	71.55	0.1224	0.0922	72.76	0.0536	0.0200	77.82
Linear Reg.	0.1808	0.1496	65.06	0.1736	0.1442	61.37	**0.0510**	**0.0174**	**78.89**
Spark-GHSOM	**0.1044**	**0.0664**	**79.82**	**0.1024**	**0.0694**	**77.21**	0.0514	0.0174	78.73
Dataset:	Facebook Metrics			Facebook Metrics			Automobile		
Target:	lifetime_post_consum.			total interactions			price		
Method	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**	**RMSE**	**MAE**	**Imp.%**
SVR (Linear)	>1	>1		>1	>1		>1	> 1	
SVR (Polyn.)	0.1027	0.0699		0.0626	0.0332		0.3357	0.2859	
SVR (Sigm.)	0.0758	0.0428	26.17	0.0531	0.0229	15.19	0.1788	0.1275	46.74
Isotonic Reg.	0.0754	0.0470	26.55	0.0528	0.0258	15.61	0.1774	0.1288	47.15
Linear Reg.	0.0748	0.0468	27.13	0.0526	0.0260	15.93	0.1566	0.1158	53.35
Spark-GHSOM	**0.0676**	**0.0380**	**34.15**	**0.0526**	**0.0262**	**15.93**	**0.1250**	**0.0692**	**62.77**

Table 3: Experimental results for the regression with mixed attributes datasets task, and improvement (%) with respect to the polynomial SVR baseline method. Each method is executed in its best configuration. The best results for each dataset are highlighted in bold.

Dataset	**Plants**	**Days**	**Hours**	**Instances**
PV Italy	17	856	19	276488
PV NREL	48	365	19	331968
Burlington	1	366	24	16464

Table 4: Brief description of datasets adopted for the sensor data forecasting task.

task, the best configuration is $\tau_1 = 0.3$ and $\tau_2 = 0.7$.

In Table 6 we compare the average RMSE obtained by different systems. As for the regression experiment, we have only considered the best values obtained by each method by varying parameters' configuration. The results clearly show that Spark-GHSOM globally outperforms the other algorithms. The only case in which Spark-GHSOM does not outperform the others is PV-NREL, where SVR Poly performs very well. Here the motivation can be found in the fact that PV NREL, differently from other datasets, is simulated: probably SVR Poly is able to catch the underlying data distribution according to which data have been generated.

To confirm the superiority of Spark-GHSOM, in Table 7 we report the results of the signed Wilcoxon rank test. As we can see, Spark-GHSOM significantly outperforms other methods (with $\alpha = 0.05$), even if it is not designed for regression (it is designed for clustering). This confirms that the extracted clusters well describe the data distribution, without unnecessary overfitting.

5.2. Scalability evaluation

In this section we introduce scalability results performed using the KDD CUP 1999 network dataset. This dataset has been chosen because it represents a good example of a big dataset (it contains 4.2 millions of instances) with mixed attributes (it contains 41 features, 7 of which categorical). Additionally, we performed a stress test considering augmented versions of the dataset, which allow to test the performances of our method with up to 100 millions of instances.

The experiments aim to demonstrate that Spark-GHSOM is capable of processing a large mixed attribute dataset in a distributed environment.

All the experiments have been conducted on a Spark cluster consisting of one driver node (6 cores, 32GB RAM) and 4 worker nodes (24 cores, 128GB RAM in total), each of them equipped with an Intel Core i7 Processor and 512GB SSD hard drives. In particular, to assess the behavior of the algorithm with different data sizes, we performed the experiments with three different samples (100%, 1000% and 2000%) of the total number of instances.

The running times, with the different samples, are shown in Fig. 5. The results highlight the linear complexity of the algorithm in an enough complicate situation ($\tau_1 = 0.5$, $\tau_2 = 0.5$). The results also show that the algorithm

Dataset: PV NREL			**Target:** power			
	τ_2=0.7		τ_2=0.5		τ_2=0.3	
	RMSE	**Impr.%**	**RMSE**	**Impr.%**	**RMSE**	**Impr.%**
τ_1=0.7	0.2397	15.85	0.2406	15.52	0.2410	15.38
τ_1=0.5	0.2353	17.40	0.2331	18.15	0.2351	17.48
τ_1=0.3	0.2331	18.17	**0.2309**	**18.92**	0.2328	18.25

Dataset: PV Italy			**Target:** power			
	τ_2=0.7		τ_2=0.5		τ_2=0.3	
	RMSE	**Impr.%**	**RMSE**	**Impr.%**	**RMSE**	**Impr.%**
τ_1=0.7	0.1543	7.87	0.1543	7.85	0.1521	9.15
τ_1=0.5	0.1436	14.22	0.1427	14.80	0.1443	13.82
τ_1=0.3	**0.1340**	**19.94**	0.1341	19.89	0.1352	19.26

Dataset: Burlington			**Target:** power			
	τ_2=0.7		τ_2=0.5		τ_2=0.3	
	RMSE	**Impr.%**	**RMSE**	**Impr.%**	**RMSE**	**Impr.%**
τ_1=0.7	0.1518	32.91	0.1587	29.87	0.1549	31.57
τ_1=0.5	**0.1370**	**39.45**	0.1402	38.05	0.1411	37.63
τ_1=0.3	0.1440	36.36	0.1467	35.18	0.1451	35.87

Table 5: Sensitivity analysis of Spark-GHSOM for the sensor data forecasting task with different datasets, considering different values of τ_1 and τ_2, and improvement (%) with respect to the ARIMA baseline method. Best results for each dataset are highlighted in bold.

Dataset:	PV NREL		PV Italy		Burlington	
Target:	power		power		power	
Method	**RMSE**	**Impr.%**	**RMSE**	**Impr.%**	**RMSE**	**Impr.%**
ARIMA	0.2849		0.1675		0.2263	
K-Means	0.2336	17.99	0.1507	10.00	0.1397	38.26
SVR (Linear)	0.1781	37.49	0.1967	-17.43	0.1953	13.70
SVR (Poly)	**0.1491**	47.67	0.1758	-4.96	0.1623	28.28
SVR (Sigmoid)	>1		0.1966	-17.37	0.1952	13.74
Isotonic Reg.	0.2621	8.00	0.2000	-19.40	0.4388	-93.90
Linear Reg.	0.2307	19.02	0.1503	10.27	0.1451	35.88
Spark-GHSOM	0.2309	18.92	**0.1340**	**19.94**	**0.1370**	**39.45**

Table 6: Experimental results for the sensor data forecasting task, and improvement (%) with respect to the ARIMA baseline method. Each method is executed in its best configuration. The best results for each dataset are highlighted in bold.

Table 7: p-values of the signed Wilcoxon rank tests for all pairwise combinations of the methods in common for the two predictive tasks considered (regression with mixed attributes and sensor data forecasting). In bold statistically significant values (confidence=0.05, unless specified otherwise).

Pairwise comparison	p-value	winner
RMSE criterion		
Spark-GHSOM VS		
SVR (Linear)	**0.011**	**Spark-GHSOM**
SVR (Poly)	**0.038**	**Spark-GHSOM**
SVR (Sigmoid)	**0.011**	**Spark-GHSOM**
Linear Reg.	**0.036**	**Spark-GHSOM**
Isotonic Reg.	**0.008**	**Spark-GHSOM**

is not affected by computational bottlenecks such as complex operations performed on the driver node.

Another important perspective of the results is provided by Figure 6 (left), where we report the speedup factor obtained when processing 4M instances (the whole KDD CUP 1999 dataset) with an increasing number of cores. Moreover, Figure 6 (right) shows the scaleup performances obtained with an increasing number of data instances and cores (4M - 8 cores, 8M - 16 cores, 12M - 24 cores). Both, speedup and scaleup curves are quite close to the ideal curves (linear and constant curves, respectively). This confirms that Spark-GHSOM can be profitably used in a cluster enviroment with large datasets.

Finally, in Figure 7 we report the performances of Spark-GHSOM in terms of execution time with an increasing number of features, considering a sample of the KDD CUP 1999 dataset, where features are replicated by a factor of 2x, 4x, 8x and 16x. Although Spark-GHSOM does not scale linearly in the number of features because the distribution of the workload is carried out in terms of horizontal partitioning, the results show that Spark-GHSOM is still capable to handle datasets characterized by a large number of features since its time complexity is still polynomial (in the number of features).

5.3. Hierarchical Growth Evaluation

In this evaluation, we analyze the hierarchical behavior of Spark-GHSOM. Since there does not exist any variant of the GHSOM capable of handling

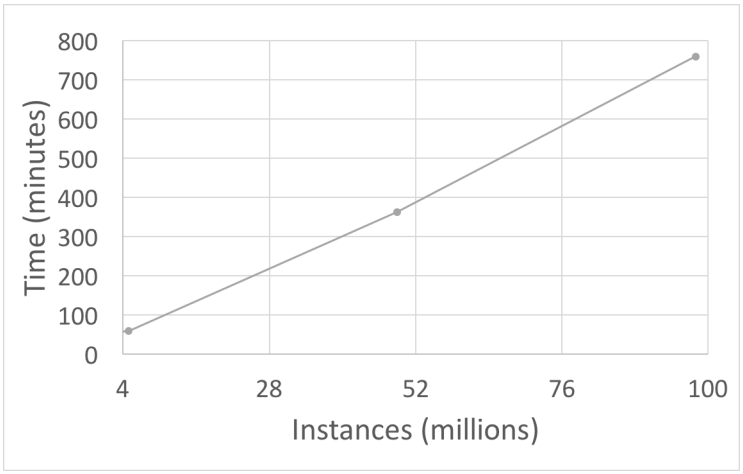


Figure 5: Stress test results (KDD CUP 1999 dataset) with τ_1=0.5, τ_2=0.5 and $epochs = 15$.

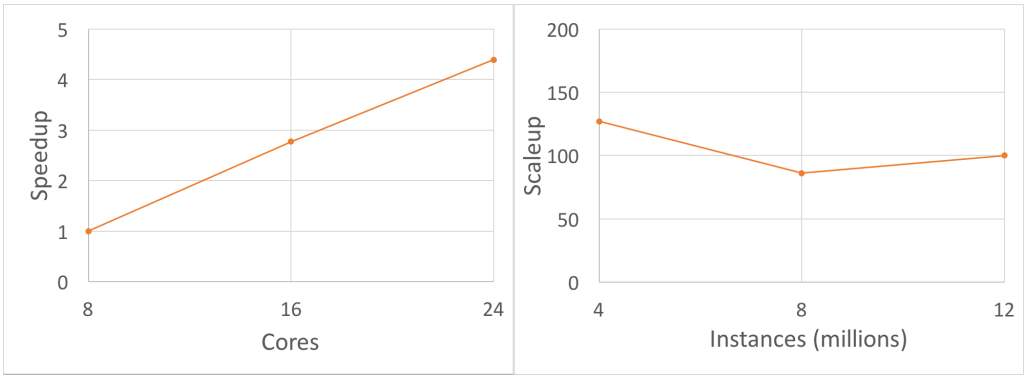


Figure 6: Speedup and scaleup results (KDD CUP 1999 dataset) with $\tau_1 = 0.5$, $\tau_2 = 0.5$ and $epochs = 15$.

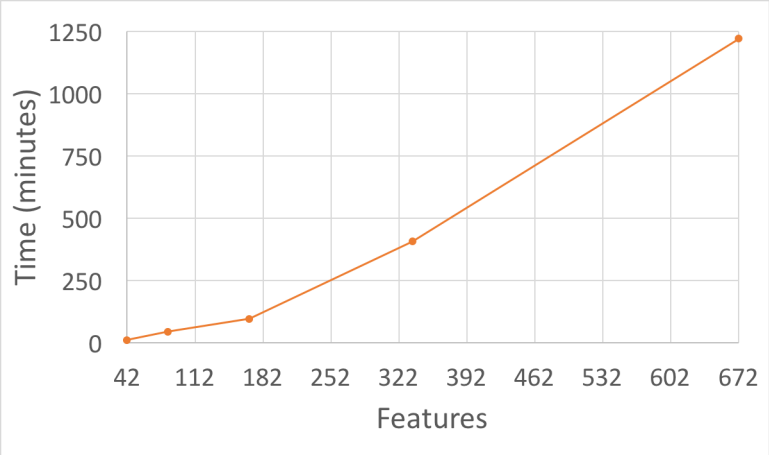


Figure 7: Stress test results with a sample of the KDD CUP 1999 dataset (1 million instances) with τ_1=0.8, τ_2=0.8 and *epochs* = 15 and increasing number of features.

mixed attributes, we analyze the results in this section empirically. Here, we try to show that Spark-GHSOM produces layers arranged in a hierarchy such that the layers lower in the hierarchy represent the data at a finer granularity than the data represented by the parent neuron in the parent layer.

We used the Zoo[30] dataset consisting of 101 instances of animals. Each instance has 16 attributes (excluding the name of animal and the type attribute for the class). We treated all the boolean attributes in the dataset as categorical attributes. Thus, the dataset had 15 categorical attributes and 1 numeric attribute. To understand the hierarchical structure produced by Spark-GHSOM, we present the label distribution (animal names) of the SOM layers.

Figure 8 shows the resulting GHSOM structure. The first layer of the SOM is shown by the blue coloured grid. The lower level layers are projected into the first layer (shown by small red coloured grids). The first layer of the generated SOM had dimensions of 5×2. It splits the dataset into two major groups - mammals (cells $[0, 0]$, $[0, 1]$ and $[1, 0]$) and non-mammals (rows 2 to 4). The last row represents the bird family. Further, we can see that some of the cells ($[0, 1]$, $[2, 0]$, etc.) spawned into new layers and represent the data into finer groups. The cell $[0, 1]$, representing land animals, is further expanded into a new layer of in which columns 0 and 1 represent predators while the remaining are non-predators. In this sub-layer, the orientation of

the predator related cells is towards the cell $[0, 0]$ containing aquatic predators - mink, seal and sealion. This ascertains that Spark-GHSOM preserves the orientation of the sublayers with respect to the parent neuron and its neighbors in the parent layer. From the generated map, we can conclude that Spark-GHSOM reflects the hierarchical relations in the data.

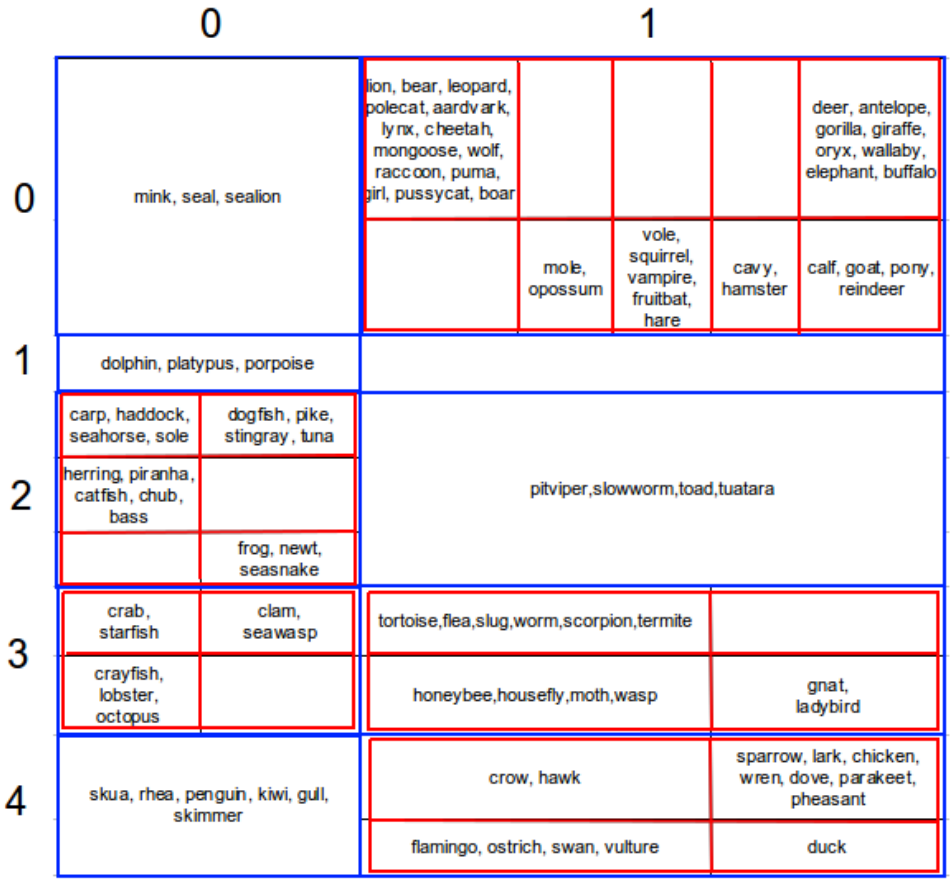


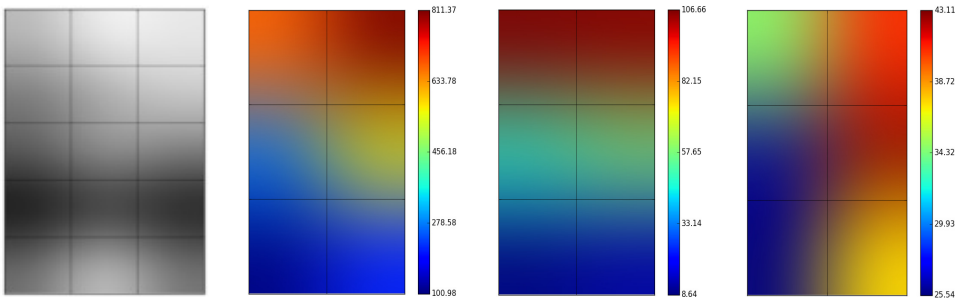
Figure 8: Hierarchical SOMs for Zoo dataset

5.4. Qualitative evaluation

We now proceed to our analysis of a large census dataset, to show a possible use of Spark-GHSOM. The analysis of a Census dataset can have applications in election campaign steering or market segmentation, just to

mention few applications. We used the *Census-Income (KDD) Dataset* containing $\sim 300K$ records with mixed-attributes from UCI [30]. Owing to the size and the presence of mixed-attributes, it would not be feasible to analyze and visualize this dataset using the conventional GHSOM. The classification attribute for this dataset (*income class*) is derived from the attribute of *total person income* in the original survey. As we wanted to model this dataset as a typical census microdata of a population, we treated the prediction class attribute as one of the analysis attributes. We combined the train and test datasets to form our dataset of $\sim 300K$ instances. The original dataset contains 42 attributes which are of mixed type - numeric and categorical. We excluded the attributes that do not give information about the instance such as year (the year of census survey) and instance weight (attribute related to stratified sampling used for creating this dataset). Further, we ignored the missing values in the dataset. For presentation purposes, we generated layers of size that would be legible for the paper. The generated GHSOM structure contained two levels. The values of τ_1 and τ_2 were set to 0.7 and 0.6 respectively.

The first level map is a small map of 3×2 neurons. It represents the dataset at a very coarse level. The U-Matrix [53] and relevant component planes for this map are shown in Fig. 9. The U-Matrix is a popular SOM visualization technique which marks the clusters in the data on a gray scale (boundaries are represented by dark colours).



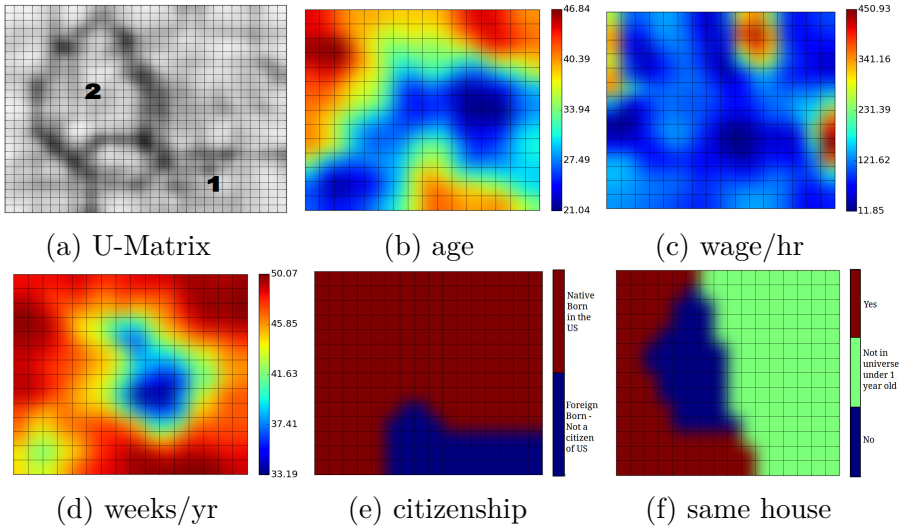
(a) U-Matrix (b) cap.gains (c) wage/hr (d) age

Figure 9: U-Matrix and Component Planes for Level 1 SOM of the Census Dataset

From the U-Matrix(Fig. 9a, we can see two major clusters. The cluster in the lower part of the map corresponds to the population of individuals with low capital gains. From these map visualizations, we can also see that

the people in the mid-age of around 40 (orange shade in Fig. 9d) have higher capital gains and more wage per hour (red shade in Fig. 9b) and 9c). Three neurons from the level-1 map expanded into new layers. We discuss the layer spawned from the cell $[0, 0]$.

From the neuron $[0, 0]$ in the level 1 map, a SOM layer of size 14×16 neurons is created. Fig. 10 shows the U-Matrix and the component planes for this layer. The U-Matrix (Figure 10a) shows several clusters in this



(a) U-Matrix (b) age (c) wage/hr

(d) weeks/yr (e) citizenship (f) same house

Figure 10: U-Matrix and Component Planes for Level 2 SOM of the Census Dataset

layer. It indicates that the data represented by the parent neuron was indeed diversified and needed to be refined further. We can see some major clusters (comprising of the several minor ones) marked by fairly darker boundaries. The cluster labelled as 1 is a cluster denoting immigrants (those not born in USA) as asserted by the component plane of the attribute *citizenship* (blue region). Further, the majority of this population is in the age range of 30-40 (light green, yellow and light orange region near the bottom edge of Figure 10b). Further, this population works for almost 50 hours in a year (red color in Figure 10d). The cluster labelled as 1 represents the population who didn't stay in their current house for more than one year (Figure 10f). Moreover, this cluster contains the age group of 25 to 35 on average. Also, this population has an average wage per hour in the range of 110 to 150 (Figure 10c).

Dataset	Instances	Features	Classes	SOM Layers	Time (mins)
Breast Cancer [47]	85	456	5	12	138
Colon Cancer [2]	62	2000	2	5	1
CNS Tumor [38]	60	7128	2	7	19.7
Huntington's Disease [3]	31	22283	2	6	37.3
Lymphoma [46]	58	6817	2	10	176
SRBCT [25]	63	2308	4	3	0.5

Table 8: Experimental results with high-dimensional micro-array datasets. Results are obtained with a configuration of τ_1=0.8 and τ_2=0.8 and $epochs = 15$.

We observed that Spark-GHSOM generated a GHSOM structure in which the first layer depicted the data at a very coarse level to be refined at lower levels. Using the example of the Census dataset above, we also demonstrated the use of the U-Matrix and the component plane visualizations to analyze the results.

In order to assess the behavior of the algorithm with high-dimensional data, we have conducted additional experiments with micro-array data, also because hierarchical clustering on gene-expression data is highly required in bioinformatics [37]. The datasets used for this evaluation are described in Table 8, where we also report the number of layers extracted and running times. Experiments have been performed using a fixed configuration of τ_1=0.8 and τ_2=0.8 and $epochs = 15$.

In the following we illustrate the results obtained on the Lymphoma dataset, in terms of U-matrix representations at different SOM levels. Application of our approach on the other micro-array datasets are similar. The Lymphoma dataset [46] contains data about B-cell lymphoid malignancies in adults. It consists of 6817 features and two classes: 58 examples are of class DLBCL (Diffuse Large B-Cell Lymphom) and 19 examples of class FL (Follicular Lymphoma). However, it is known in the literature that FLs frequently evolve over time and acquire the morphologic and clinical features of DLBCLs, and some subsets of DLBCLs have chromosomal translocations characteristic of FLs [28]. This aspect, in combination with the large number of features, makes data exploration and classification challenging with this data.

Our algorithm extracts a hierarchy of 10 SOM levels (see Fig. 11 for a partial view of the hierarchy). At level 1 (map1), the model describes data

at a coarse level. In fact, there are cases in which examples of different classes are mapped to the same neurons. Successive levels offer a finer level of detail in the representation. For example, map2 (at level 2) specializes the neuron at position $[0, 0]$ in map1: data instances associated to this level are exclusively of class DLBCL. The same applies for map4 (specializing the neuron at position $[0, 0]$ in map2), map5 (which specializes the neuron at position $[0, 1]$ in the map2), map7 (which specializes the neuron at position $[1, 1]$ in map5) and map9 (which specializes the neuron at position $[1, 1]$ in map6). The remaining maps represent instances coming from different classes mapped on the same SOM model: this phenomenon happens on a single neuron in map3 (which specializes the neuron at position $[2, 0]$ in map1), map6 (which specializes the neuron at position $[1, 0]$ in map3) and map8 (which specializes the neuron at position $[1, 0]$ in map6). These levels would likely be expanded further with lower values of τ_1 and τ_2. A different case is represented by map10, in which the originating SOM model maps instances belonging to different classes on different rows of neurons.

Overall, these representations can, in principle, offer a wide and concise perspective of the data for a domain expert, who can easily gain insights thanks to the hierarchy of two-dimensional grids.

Additional results in terms of component planes and U-Matrix representations for all datasets are available online on the supplemental material website.

5.5. Availability

The system and the datasets are available at the following hyperlink for replication purposes: `http://www.di.uniba.it/~ceci/ghsom/`.

6. Conclusions

In this work, we proposed a variant of the GHSOM algorithm called Spark-GHSOM, which scales GHSOM to large real-world datasets on a distributed cluster. Moreover, we also proposed a new *distance hierarchy* approach to extend the GHSOM for mixed attribute datasets. We also defined the arithmetic operations for the distance hierarchy points, required for the training process in the GHSOM algorithm.

From a quantitative viewpoint, we evaluated the scalability and the accuracy of the models extracted with the algorithm, also presenting two real applications, namely, regression with mixed attributes datasets and sensor data

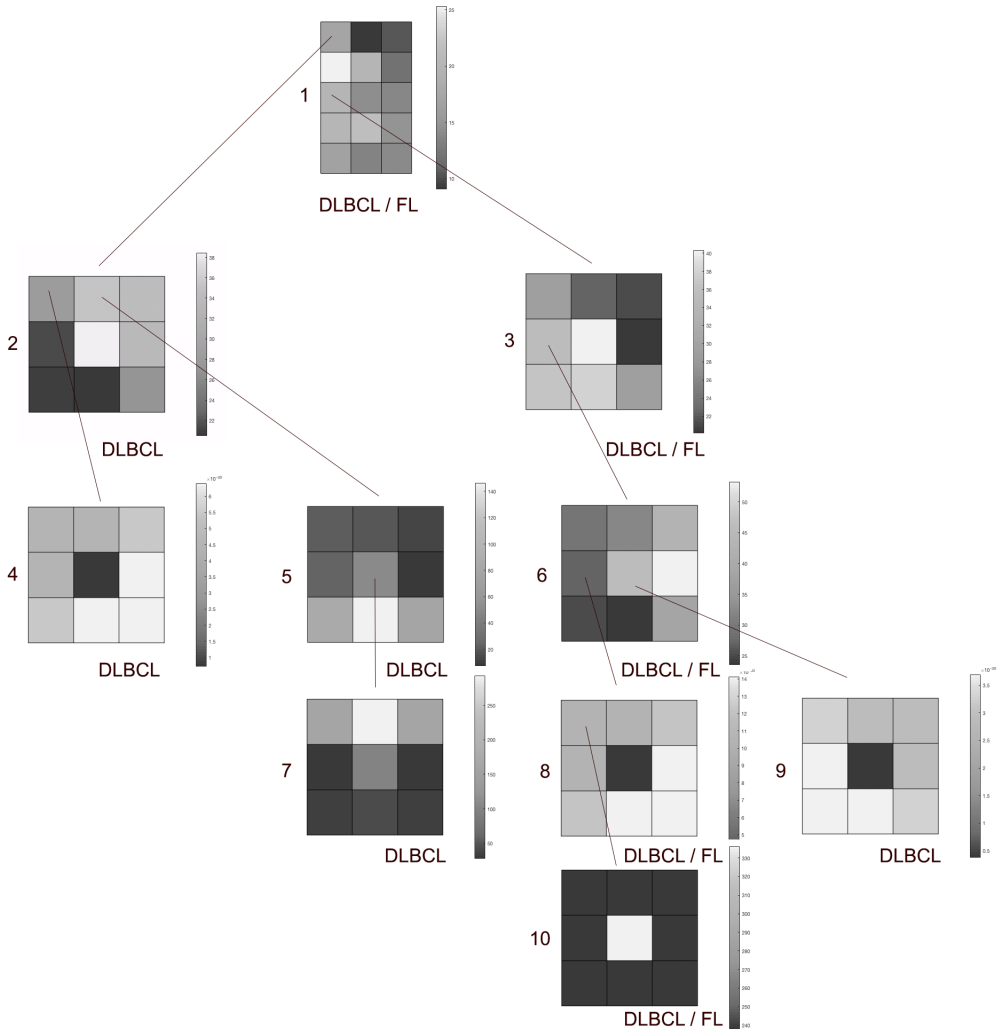


Figure 11: Hierarchy of U-Matrix representations extracted with the Lymphoma Dataset. The results are obtained with τ_1=0.8, τ_2=0.8 and *epochs* = 15. Maps are numbered from map1 to map10. "DLBCL/FL" indicates that there are cases in which examples of different classes are mapped to the same neurons. "DLBCL"("FL") indicates that there is no case in which examples of different classes are mapped to the same neurons.

forecasting, comparing the results obtained with state of the art algorithms, specifically designed for regression. Lastly, we analyzed our Spark-GHSOM using some popular datasets from UCI [30]. The results ascertained that Spark-GHSOM produced satisfactory results, both in terms of modelling an individual SOM as well as depicting the hierarchical relations in the dataset. We generated a multi-level hierarchy of SOM layers and discussed the results using the U-Matrix and the component plane visualizations.

Overall, we can conclude that the Spark-GHSOM is an useful extension of the GHSOM algorithm, which not only scales GHSOM to large datasets but also extends it to process high-dimensional mixed attribute datasets. The models extracted can be profitably exploited in a variety of contexts, from data visualization to predictive modeling.

Directions for future work include efficient treatment of high-dimensional data. For example, it would be possible to include a dimensionality reduction or feature extraction step to be performed before the modeling step, in order to address issues such as feature collinearity and computational bottlenecks deriving from high-dimensional data. Alternatively, it could be possible to formulate a distributed variant of the algorithm which would perform vertical data partitioning instead of horizontal partitioning. One other aspect worth investigating is the extension or the integration of GHSOM models with supervised techniques to perform tasks of recent interest, such as multi-label classification and multi-target regression.

7. Acknowledgments

We acknowledge the financial support of the European Commission, via the grants ICT-2013-612944 MAESTRA and H2020-ICT-688797 TOREADOR, and of the Italian Ministry of Education, University and Research (MIUR) via the PON project ComESto - Community Energy Storage: Aggregate Management of Energy Storage Systems in Power Cloud (Grant N. ARS01_01259). We also acknowledge the support of Rotary Foundation via a district grant (District 2120 - Molfetta, Italy).

References

[1] M. Aggarwal, A. K. Tiwari, Approach for Information Retrieval by Using Self-Organizing Map and Crisp Set, in: Speech and Language Processing for Human-Machine Communications, Springer, 51–56, 2018.

[2] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, A. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, Proceedings of the National Academy of Sciences 96 (12) (1999) 6745–6750.

[3] F. Borovecki, L. Lovrecic, J. Zhou, H. Jeong, F. Then, H. D. Rosas, S. M. Hersch, P. Hogarth, B. Bouzou, R. V. Jensen, D. Krainc, Genome-wide expression profiling of human blood reveals biomarkers for Huntington's disease, Proceedings of the National Academy of Sciences of the United States of America 102 (31) (2005) 11023–11028.

[4] J. C. Burguillo, J. García-Rois, Time Series Prediction Using Coalitions and Self-organizing Maps, in: Self-organizing Coalitions for Managing Complexity, Springer, 171–205, 2018.

[5] M. Ceci, R. Corizzo, F. Fumarola, D. Malerba, A. Rashkovska, Predictive Modeling of PV Energy Production: How to Set Up the Learning Task for a Better Prediction?, IEEE Transactions on Industrial Informatics 13 (3) (2017) 956–966, ISSN 1551-3203.

[6] A. Chan, E. Pampalk, Growing hierarchical self organising map (ghsom) toolbox: visualisations and enhancements, in: Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on, vol. 5, IEEE, 2537–2541, 2002.

[7] I.-T. Chen, L.-C. Chang, F.-J. Chang, Exploring the spatio-temporal interrelation between groundwater and surface water by using the self-organizing maps, Journal of Hydrology 556 (2018) 131–142.

[8] N. Chen, N. C. Marques, An extension of self-organizing maps to categorical data, in: Progress in Artificial Intelligence, Springer, 304–313, 2005.

[9] P. Cortez, A. d. J. R. Morais, A data mining approach to predict forest fires using meteorological data, in: Proc. of the Conference of the Portuguese Association for Artificial Intelligence (APPIA), 2007.

[10] E. De la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, A. Martínez-Álvarez, Feature selection by multi-objective optimisation: Applica-

tion to network anomaly detection by hierarchical self-organising maps, Knowledge-Based Systems 71 (2014) 322–338.

[11] M. Dittenbach, D. Merkl, A. Rauber, The growing hierarchical self-organizing map, in: ijcnn, IEEE, 6015, 2000.

[12] J. Faigl, G. A. Hollinger, Autonomous data collection using a self-organizing map, IEEE transactions on neural networks and learning systems 29 (5) (2018) 1703–1715.

[13] J. A. Flanagan, Unsupervised clustering of symbol strings, in: Neural Networks, 2003. Proceedings of the International Joint Conference on, vol. 4, IEEE, 3250–3255, 2003.

[14] J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall / CRC Data Mining and Knowledge Discovery Series, CRC Press, ISBN 978-1-4398-2611-9, 2010.

[15] T. Gordon, Is the standard deviation tied to the mean?, Teaching Statistics 8 (2) (1986) 40–42.

[16] F. Gu, Y.-M. Cheung, Self-organizing map-based weight design for decomposition-based many-objective evolutionary algorithm, IEEE Transactions on Evolutionary Computation 22 (2) (2018) 211–225.

[17] J. Han, M. Kamber, J. Pei, Data mining: concepts and techniques: concepts and techniques, Elsevier, 2011.

[18] Z. He, X. Xu, S. Deng, Tcsom: Clustering transactions using self-organizing map, Neural Processing Letters 22 (3) (2005) 249–262.

[19] C.-C. Hsu, Generalizing self-organizing map for categorical data, Neural Networks, IEEE Transactions on 17 (2) (2006) 294–304.

[20] S.-Y. Huang, R.-H. Tsaih, The prediction approach with Growing Hierarchical Self-Organizing Map, The 2012 International Joint Conference on Neural Networks (IJCNN) (2012) 1–7.

[21] Z. Huang, Clustering large data sets with mixed numeric and categorical values, in: Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining,(PAKDD), Singapore, 21–34, 1997.

[22] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data mining and knowledge discovery 2 (3) (1998) 283–304.

[23] R. J. Hyndman, Y. Khandakar, et al., Automatic time series for forecasting: the forecast package for R, Tech. Rep., 2007.

[24] G. D. Kader, M. Perry, Variability for categorical variables, Journal of Statistics Education 15 (2) (2007) 1–17.

[25] J. Khan, J. S. Wei, M. Ringnér, L. H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. R. Antonescu, C. Peterson, P. S. Meltzer, Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks., Nature Medicine 7 (6) (2001) 673–679.

[26] D. Kibler, D. W. Aha, M. K. Albert, Instance-based Prediction of Real-valued Attributes, Comput. Intell. 5 (2) (1989) 51–57.

[27] T. Kohonen, The self-organizing map, Proceedings of the IEEE 78 (9) (1990) 1464–1480.

[28] B. Li, B.-B. Tian, J. Liu, Tumor gene expressive data classification based on locally linear representation fisher criterion, in: International Conference on Intelligent Computing, Springer, 443–449, 2013.

[29] W.-C. Liao, C.-C. Hsu, A self-organizing map for transactional data and the related categorical domain, Applied Soft Computing 12 (10) (2012) 3141–3157.

[30] M. Lichman, UCI Machine Learning Repository, 2013.

[31] C.-J. Lu, Y.-W. Wang, Combining independent component analysis and growing hierarchical self-organizing maps with support vector regression in product demand forecasting, International Journal of Production Economics 128 (2) (2010) 603–613.

[32] A. M. Malondkar, Extending the Growing Hierarchical Self Organizing Maps for a Large Mixed-Attribute Dataset Using Spark MapReduce, Ph.D. thesis, Université d'Ottawa/University of Ottawa, 2015.

[33] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, A. Talwalkar, MLlib: Machine Learning in Apache Spark, Journal of Machine Learning Research 17 (34) (2016) 1–7, URL http://jmlr.org/papers/v17/15-237.html.

[34] R. Mikaeil, S. S. Haghshenas, S. H. Hoseinie, Rock penetrability classification using artificial bee colony (ABC) algorithm and self-organizing map, Geotechnical and Geological Engineering 36 (2) (2018) 1309–1318.

[35] S. Moro, P. Rita, B. Vala, Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach, Journal of Business Research 69 (9) (2016) 3341–3351.

[36] Y.-S. Park, T.-S. Chon, M.-J. Bae, D.-H. Kim, S. Lek, Multivariate Data Analysis by Means of Self-Organizing Maps, in: Ecological Informatics, Springer, 251–272, 2018.

[37] G. Pio, M. Ceci, D. Malerba, D. D'Elia, ComiRNet: a web-based system for the analysis of miRNA-gene regulatory networks, BMC Bioinformatics 16 (S-9) (2015) S7, URL https://doi.org/10.1186/1471-2105-16-S9-S7.

[38] S. L. Pomeroy, P. Tamayo, M. Gaasenbeek, L. M. Sturla, M. Angelo, M. E. McLaughlin, J. Y. H. Kim, L. C. Goumnerova, P. M. Black, C. Lau, J. C. Allen, D. Zagzag, J. M. Olson, T. Curran, C. Wetmore, J. A. Biegel, T. Poggio, S. Mukherjee, R. Rifkin, A. Califano, G. Stolovitzky, D. N. Louis, J. P. Mesirov, E. S. Lander, T. R. Golub, Prediction of central nervous system embryonal tumour outcome based on gene expression, Nature 415 (6870) (2002) 436–442.

[39] A. Rauber, D. Merkl, M. Dittenbach, The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data, Neural Networks, IEEE Transactions on 13 (6) (2002) 1331–1341.

[40] D. Sacha, M. Kraus, J. Bernard, M. Behrisch, T. Schreck, Y. Asano, D. A. Keim, SOMFlow: Guided Exploratory Cluster Analysis with Self-Organizing Maps and Analytic Provenance, IEEE transactions on visualization and computer graphics 24 (1) (2018) 120–130.

[41] T. Sarazin, H. Azzag, M. Lebbah, SOM Clustering Using Spark-MapReduce, in: Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, IEEE, 1727–1734, 2014.

[42] E. Schweighofer, A. Rauber, M. Dittenbach, Automatic text representation, classification and labeling in European law, in: ICAIL, 2001.

[43] C. Serrano-Cinca, Self organizing neural networks for financial diagnosis, Decision Support Systems 17 (3) (1996) 227–238.

[44] J. Shi, Y. Qiu, U. F. Minhas, L. Jiao, C. Wang, B. Reinwald, F. Özcan, Clash of the Titans: MapReduce vs. Spark for Large Scale Data Analytics, PVLDB 8 (13) (2015) 2110–2121, URL `http://www.vldb.org/pvldb/vol8/p2110-shi.pdf`.

[45] J. Shih, Y. Chang, W. Chen, Using GHSOM to construct legal maps for Taiwan's securities and futures markets, Expert Syst. Appl. 34 (2) (2008) 850–858.

[46] M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, et al., Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning, Nature medicine 8 (1) (2002) 68.

[47] T. Sørlie, C. M. Perou, R. Tibshirani, T. Aas, S. Geisler, H. Johnsen, T. Hastie, M. B. Eisen, M. van de Rijn, S. S. Jeffrey, T. Thorsen, H. Quist, J. C. Matese, P. O. Brown, D. Botstein, P. E. Lønning, A.-L. Børresen-Dale, Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications, Proceedings of the National Academy of Sciences 98 (2001) 10869–10874.

[48] S.-J. Sul, A. Tovchigrechko, Parallelizing BLAST and SOM algorithms with MapReduce-MPI library, in: Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, IEEE, 481–489, 2011.

[49] W.-S. Tai, C.-C. Hsu, Growing Self-Organizing Map with cross insert for mixed-type data clustering, Applied Soft Computing 12 (9) (2012) 2856–2866.

[50] Y. Tang, Y. Guo, Q. Sun, S. Tang, J. Li, L. Guo, J. Duan, Industrial polymers classification using laser-induced breakdown spectroscopy combined with self-organizing maps and K-means algorithm, Optik 165 (2018) 179–185.

[51] R.-H. Tsaih, W.-Y. Lin, S.-Y. Huang, Exploring fraudulent financial reporting with GHSOM, in: Pacific-Asia Workshop on Intelligence and Security Informatics, Springer Berlin Heidelberg, 31–41, 2009.

[52] A. Tsanas, A. Xifara, Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools, Energy and Buildings 49 (2012) 560–567.

[53] A. Ultsch, Self-organizing neural networks for visualisation and classification, Springer, 1993.

[54] C. Weichel, Adapting Self-Organizing Maps to the MapReduce Programming Paradigm, in: STeP, 119–131, 2010.

[55] D. Zurita, M. Delgado, J. A. Carino, J. A. Ortega, Multimodal Forecasting Methodology Applied to Industrial Process Monitoring, IEEE Transactions on Industrial Informatics 14 (2) (2018) 494–503.