Neural Computing and Applications (2023) 35:17579–17597
https://doi.org/10.1007/s00521-023-08630-w

ORIGINAL ARTICLE

Prototype generation method using a growing self-organizing map applied to the banking sector

Sara Ruiz-Moreno[1] · Amparo Núñez-Reyes[1] · Adrián García-Cantalapiedra[2] · Fernando Pavón[2]

Received: 4 April 2022 / Accepted: 24 April 2023 / Published online: 12 May 2023

Abstract
In fields like security risk analysis, Fast Moving Consumer Goods, Internet of Things, or the banking sector, it is necessary to deal with large datasets containing a great list of variables. In these situations, the analysis becomes intricate and computationally expensive, so data reduction techniques play an important role. Prototype generation methods provide a reduced dataset with the same properties as the original. GSOMs (growing self-organizing maps) reduce the data size without the need for prefixing the number of neurons needed to represent the input space. To the best of the authors' knowledge, this is the first time that the GSOM is applied for reduction and generation of prototypes, posing an advantage over their predecessors, the SOMs (self-organizing maps), which do not have the automatic growth feature. This work addresses the use of a GSOM to reduce the number of prototypes to use in a 1-NN (1 nearest neighbor) classifier. The proposed methodology is applied to an income dataset for testing and a large bank dataset that contain classifications into two different groups. The 1-NN classifier is used to obtain predictions using the nodes of the GSOM as prototypes. This article demonstrates that GSOMs save a significant amount of time in obtaining nearly the same validation results as SOMs by comparing the classifications obtained in the bank dataset. The results show data reductions of more than 99%, and accuracies greater than 80% for the income dataset and 74% for the bank dataset.

Keywords Growing self-organizing map · Data reduction techniques · Prototype generation · k-NN · Banking

1 Introduction

Nowadays, the development of new technologies and computing systems gives humans the ability to collect large amounts of data. The more data we have on a system, the better representation we obtain of it, and this will give us greater capacity to extract information and describe it [1]. The problem arises when, in real-world applications, the amount of data needed is too large. For instance, in banking applications, where there are millions of clients, with many different behaviors and operations, the computational resources needed are unaffordable [2]. In this paper, the results for a bank with more than 2 million customers will be presented.

One of the most thrilling problems in the banking sector is the classification of customers to detect unusual behaviors. The k-nearest neighbors (k-NN) technique is widely used for recognition tasks. It consists of obtaining a representative dataset formed by prototypes for which the classes are known. When new data are fed to the algorithm, the class corresponding to its k nearest neighbors is assigned within the prototype space. It is one of the most useful algorithms in data mining, but it has the drawbacks of high storage requirements, low efficiency, and low noise tolerance, especially in the case of 1-NN, since it gives importance to all data [3]. One way to improve the

✉ Sara Ruiz-Moreno
srmoreno@us.es

Amparo Núñez-Reyes
anreyes@us.es

Adrián García-Cantalapiedra
adrian.garcia@gamco.es

Fernando Pavón
fernando.pavon@gamco.es

[1] Department of Systems and Automatic Control Engineering, Escuela Técnica Superior de Ingeniería, University of Seville, Camino de los Descubrimientos, 41092 Seville, Spain

[2] GAMCO S.L., Alcalá 20, 28014 Madrid, Spain

performance of the k-NN classifier is to use data reduction techniques, more specifically, prototype reduction, which can be achieved by either prototype selection or generation.

Prototype selection consists of choosing some representative instances from the dataset to use them as prototypes. The objective is to obtain a training set of smaller size than the original one, but with the same characteristics, achieving similar or higher classification accuracy for new data [3]. On the other hand, prototype generation consists of creating new training data that represent the previous one with the same objective as prototype selection.

The self-organizing map, also called Kohonen map after the first person who described it as a neural network, is an unsupervised method that organizes data and also provides a low-dimensional representation employing competitive learning [4–7]. This algorithm consists of a non-linear, ordered, and smooth assignation of high-dimensional input data to a low-dimensional matrix–typically 2D—by assigning weights to each node of its topology. Since a SOM organizes the data in groups, it allows for a better understanding of the input space and is very useful in applications like data mining or pattern recognition. A SOM can be understood in two different ways: considering the output as the low-dimensional map gives a dimensional reduction and can be used as a clustering technique; considering the output as the weights gives a data reduction in which each weight conforms a new prototype. A drawback of the SOMs is that the number of neurons must be pre-specified before the training phase. A low number of nodes leads to general information and a relatively straightforward explanation of the data, whereas more nodes allow for more detailed information. The growing self-organizing map (GSOM) addresses this issue by creating new neurons during the training process so that it can adapt its structure [8]. This, along with a high reduction in training times [9], is a great advantage over the SOM. This work addresses the problem of data reduction for k-NN using a GSOM.

The main contribution of this work is the implementation of a GSOM for prototype generation in large datasets. To the best of the authors' knowledge, this is a novel approach for prototype generation that has not been previously published. The main objective of the work is to reduce the number of instances in large datasets without compromising the accuracy. Moreover, a real dataset with 350000 clients was used. The GSOM is applied over the Census Income dataset and a bank dataset to obtain a representative space of the data used afterward in a 1-NN classification problem. This way, it is easier to distinguish patterns of behavior in clients and make predictions. The advantages of this method are the decrease in time needed to apply the 1-NN algorithm and the use of a machine learning technique that allows a visual understanding of the problem. To enhance the adaptation of the network, we have added the use of several iterations of the algorithm in each of its phases. This aspect is pre-selected.

This document organizes as follows. Section two provides a literature review and discusses some related work. In section three, the proposed methodology is defined and the techniques are described, as well as the evaluation and visualization metrics. The experimental setup is outlined in section four, showing the results obtained for two different datasets. Next, a discussion is provided in section five and, finally, section six extracts some conclusions from the work.

2 Literature overview

2.1 k-nearest neighbors

The k-NN classifier is an algorithm that assigns a class to an input vector based on the classes of the k nearest prototypes in the input space. It is a well-known technique used in many different applications. For example, Konieczny and Stojek [10] use a k-NN classifier to classify the wear condition of a pump, Santos Ruiz et al. [11] apply the k-NN algorithm to locate leaks in water distribution networks and Tharwat et al. [12] classify human activities with information from smart devices.

Despite their wide use and ease of application, k-NN classifiers have low efficiency, low noise tolerance, and high data storage requirements. To overcome these problems, there are different approaches [13]. Some of them are based on modifying the parameters and equations of the k-NN, for example, by selecting the optimal value of k [14] or doing it adaptively [15], combining with genetic algorithms in the selection of neighbors [16], designing distance functions [17] or using frameworks prepared for big data applications [18]. Other techniques include reducing the number of prototypes, either by prototype selection or by prototype generation.

2.2 Prototype reduction

There are many approaches to prototype selection. Rosero-Montalvo et al. [19] present an analysis of neighborhood criterion using the condensed neighbor algorithm to eliminate redundant data. The work by Suyal and Singh [20] approaches the problem of prototype selection by using multi-label k-NN. Gurumoorthy et al. [21] propose a framework for prototype selection based on optimal transport and compare it with other methods evaluating it with a 1-NN classifier on several datasets. The study by Kasemtaweechok and Suwannik [22] proposes a technique based on geometric median and compares it with different methods that show high accuracy with low times.

Prototype generation, on the other hand, also has various applications in literature. Triguero et al. [23] classify and compare different types of prototype generation algorithms applied to k-NN classifiers. Ougiaroglou et al. [24] use an algorithm based on reduction through homogeneous clustering for multilabel k-NN classification. Elkano et al. [25] propose the use of a new distributed MapReduce method called CHI-PG for big data classification problems. Few works in the literature address the problem of prototype reduction by using a SOM. For instance, Lechevallier and Ciampi [26] integrate a SOM with other clustering methods and apply it to nutritional data and, with regard to financial applications, Sarlin and Peltonen [27] use a SOM for data and dimensionality reduction to monitor vulnerabilities and map the state of financial stability.

2.3 Self-organizing maps

The SOM and all its variations have great potential in applications where it is necessary to analyze great amounts of data, and it becomes extremely expensive computationally. For that reason, it can be used in many preprocessing tasks in areas such as data mining: data and dimensionality reduction, master and multiple curves approximation, clustering, and classification. Self-organizing maps have also been widely used in cybersecurity since ten years after the appearance of the algorithm [28]. Ichimura et al. [29] use a SOM and automatically defined groups to obtain the distribution of spam email and classification that would serve to adjust email filtering. Sarkar et al. [30] use a k-means algorithm combined with SOM to extract patterns in accidents at work. Christyawan et al. [31] propose the use of a type of GSOM with a clustering reference vector for an intrusion detection system. Since the 1990 s, SOMs have been used in financial applications [32]. Regarding performance analysis, Shanmuganathan [33] states that the SOM is a useful tool to examine the returns and measures implemented in financial sectors. Concerning financial crisis monitoring, López Iturriaga and Pastor Sanz [34] use neural networks based on SOMs to compare macroeconomic imbalances in European countries. In these fields, SOMs allow a visual representation of the data and their relationship, which leads to a better understanding of its triggering factors. Barman and Chowdhury [35] use a SOM and a minimum spanning tree for customer segmentation. In the field of fraud detection, Quah and Sriganesh [36] analyze the behavior of credit card customers in real time to find hidden patterns without the need for previous information, and Balasupramanian et al. [37] propose a fraud detection and prevention method that uses a SOM to detect patterns. Ganegedara and Alahakoon [38] address the use of parallel GSOM and propose a method to reduce redundant neurons. Studies like the work by Kuo et al. [39] prove that GSOMs perform computationally better than SOMs, in this case, combined with bee colony optimization.

3 Techniques applied

The outline of this work is presented in figure 1. The first step is to train the neural network with a three-phase process [9] over a training set of data. After that, different error measures are calculated, as well as the U-matrix, which allows a visual explanation of the relationships between neurons. Finally, we use the GSOM as prototypes for a one-neighbor k-NN in order to classify the clients and assess the neural network's capacity to represent the input space.

3.1 GSOM

This subsection aims to describe the GSOM and the metrics used in this paper to evaluate it. The general idea of a self-organizing map consists of a network with a given topology that adapts the weights of its nodes as it receives input data without losing topological properties. A neighborhood accomplishes the adaptation of the weights to the net so that the arrival of new data during the training process will lead to a major adaptation in those neurons whose weights are closer to these data. A two-classes

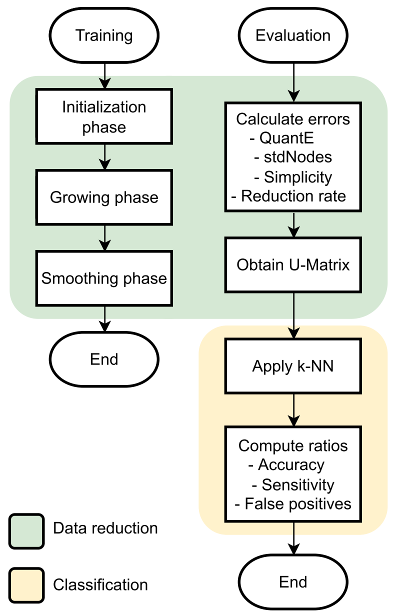


Fig. 1 General schema of the work, with the data reduction stage marked in green and the classification stage marked in yellow

dataset with 700 vectors is used for exemplification purposes, obtaining a GSOM with 88 nodes.

The growing self-organizing map includes the possibility of automatically modifying the size of the network, which allows the use of the number of neurons necessary for each application without the need for it to be fixed beforehand.

The GSOM training process is as follows [9]. Initially, there is a small net, generally composed of four nodes or neurons, each with a weight vector of the same size as the input data that will fit in the input space. In SOMs and GSOMs, as new data arrive, the neuron with the greatest activation–this is generally calculated as the lowest Euclidean distance between the weights and the input data–will be elected as the winner (best matching unit or BMU), as shown in Fig. 2. With a decreasing learning rate, this weight vector will be modified to resemble the input data. This learning rate can be linear, potential, or inverse time. Similarly, the nearest nodes to the BMU weights will adapt too by a space-time neighborhood function–the most common is the Gaussian function. The adaptation is usually expressed as:

$$w_j(k+1) = w_j(k) + LR(k)\epsilon(k)\big(v(k) - w_j(k)\big) \tag{1}$$

where k denotes the current learning epoch, j is the node, $LR(k)$ is the learning rate, $\epsilon(k)$ is the neighborhood, and $v(k)$ represents each element of the input data. This learning procedure leads to an ordered topological mapping of the represented input data.

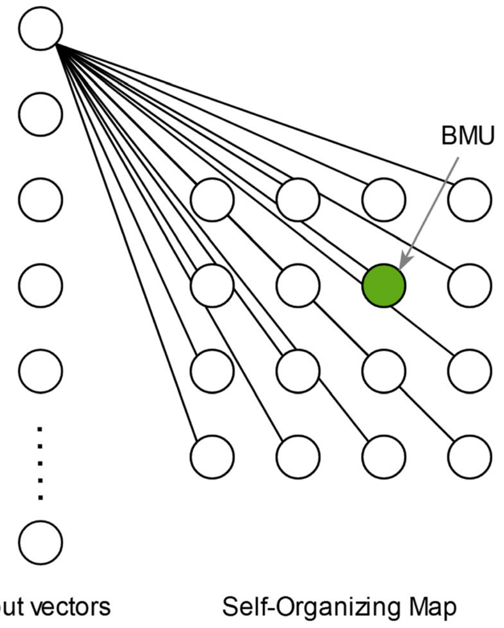


Fig. 2 Scheme of the training process of a SOM. For each input vector, the BMU is selected and the SOM weights are adapted

The learning rate LR is calculated from the following equation, where α and R are parameters that must be preselected and $n(k)$ is the number of nodes.

$$LR(k) = LR(k-1)\cdot\alpha\cdot\left(1 - \frac{R}{n(k)}\right) \tag{2}$$

In a GSOM [40], it is required to keep track of the accumulated error so that when a given threshold is reached, it will be necessary to expand the number of nodes. The growth of the map GT is controlled by the spread factor SF as given by the following equation, where D is the dimension of the data.

$$GT = -D\cdot ln(SF) \tag{3}$$

As the network receives new data, the maximum error of the nodes is calculated. If it reaches GT, new nodes will be added around the winner. When a node not belonging to the frontier is selected for growing, the error of its neighbors increases by the distribution factor FD, as in Eq. 5, with the computation of the BMU given by Eq. 4.

$$BMU(k) = \arg\min_j ||v(k) - w_j(k)||, \quad j = 1, \ldots, n(k) \tag{4}$$

$$E = \begin{cases} GT/2 & \text{if} \quad \text{node is BMU} \\ E(1+FD) & \text{if} \quad \text{node is neighbor of BMU} \end{cases} \tag{5}$$

The neighborhood function is given by Eq. 6, where d_t is the topological distance between each neuron j and the winner BMU, and σ represents the selection of neighbors and decreases each epoch a given value $\Delta\sigma$.

$$\epsilon = e(k)^{-\frac{d_t(j,BMU(k))}{2\sigma^2}} \tag{6}$$

For a major control over the neighborhood, we added the possibility of selecting the value of σ with a certain number of iterations. The decrease $\Delta\sigma$ is generally selected directly, but this is not as intuitive when it comes to parameter tuning. This work will fix it through the number of iterations in each phase of the algorithm so that the increment will be equal to an initial value of σ minus a minimum value divided by the number of total iterations N.

$$\Delta\sigma = \frac{\sigma_{ini} - \sigma_{min}}{N} \tag{7}$$

This process is accomplished in three phases: initialization, growing and smooth. The last one consists of refining the weights of the nodes once the net has reached its definite size. This is detailed in algorithm 1 [9].

Several tests with different network topologies (squared and hexagonal) have been conducted. In squared networks, a neuron can have up to four neighbors, while the number of neighbors may reach six in the hexagonal topologies. Thus, when new nodes are added to the net, the growth will be more intense in the case of hexagonal topologies.

Algorithm 1 Training process of the GSOM [9].

```
Initialization phase
1: Randomly initialize weight vectors
2: Calculate $GT$
Growing phase
1: for each input vector in the training set do
2:     Read a new input vector
3:     Initialize $LR$, $\sigma$ and total error of the nodes
4:     while $\sigma \neq \sigma_{min}$ do
5:         Find the BMU
6:         Update weights of the BMU and its neighbors
7:         Increase the error of the BMU
8:         if total error of node $i > GT$ then
9:             Adapt error of the winner and its neighbors
10:            Actualize $LR$ and $\sigma$fa
11:        end if
12:    end while
13: end for
Smoothing phase
1: for each input vector in training set do
2:     Read a new input vector
3:     Reinitialize $LR$ and $\sigma$
4:     while $\sigma \neq \sigma_{min}$ do
5:         Find the BMU
6:         Update weights of the BMU and its neighbors
7:         Actualize $LR$ and $\sigma$
8:     end while
9: end for
```

3.1.1 U-matrix

The U-matrix (Unified Distance Matrix) is a bidimensional representation of the neurons in the input space to visualize the distances between them. The mean distance between the weight vector and its neighbors is calculated for each neuron, and a color is associated. The nodes whose colors correspond to lower values will have more similar weights [41].

The self-organizing maps allow a bidimensional representation of the data space that will be able to be visualized through the U-matrix. This work employs a GSOM representation with a U-matrix intending to facilitate its comprehension and favor the following analysis process.

3.1.2 Evaluation

Three tools have been employed to compare the performance of the various networks obtained: standard deviation, quantification error, and simplicity. The first one (stdNodes, in Eq. 8) collects the mean standard deviation of the distances between the input data and the associated BMUs. The quantification error or heterogeneity (quantE, in Eq. 9) provides a measure of the adjustment of neurons to training vectors using the mean of the distances to the corresponding winning node [42]. Simplicity [43], in Eq. 10, is the accumulated neighborhood distance. It measures the network expansion and collects the sum of the distance between the BMUs and their neighbors. Also, the percentage of prototype reduction will be computed.

$$stdNodes = \frac{1}{n(N)} \sum_{j=1}^{n(N)} s_{X(j)}(||w_j(N) - v(N)||) \tag{8}$$

with s the standard deviation and $X(j) = \{x : j = BMU(x)\}$

$$quantE = \frac{1}{N_p} \sum_{i=1}^{N_p} ||v(i) - w_{BMU(v(i))}|| \tag{9}$$

$$simplicity = \sum_{i:X(i)\neq\emptyset} \sum_{j:j\ neighbor\ of\ i} ||w_i - w_j|| \tag{10}$$

3.1.3 Illustrative examples

GSOM alone can be used for classification and generation of new neurons presenting the initial topology. To validate the proposed methodology and visually check the adaptation of the net, two preliminary experiments are presented. The GSOM features shown in the examples (ring and letter G) will be exploited as a prototype generation method for a k-NN classifier in the results section.

The use of the U-matrix for classification is exemplified in Figs. 3 and 4. They show, respectively, the weight vectors and U-matrix of a GSOM applied to a 700 vectors dataset with two features (x,y) in which two classes (z-axis) are distinguished. Neurons with blue tones are associated with minor distances to neighboring nodes, and reddish tones associate with major distances. The U-matrix allows one to visually distinguish both classes, since orange neurons constitute a frontier between them.

The second example was conducted on a set of tridimensional data distributed in the space and forming a figure. The dataset is conformed by points randomly scattered building a letter G. It has been divided into three subsets (s1, s2 and s3) that contain 1000, 10000 and 20000 data, respectively. The variables are x, y and z. This example shows the advantages of GSOMs since they can generate the number of nodes necessary to adapt to datasets of different sizes without the need to know it beforehand. Not necessarily a set with more data must be represented with a larger map.

Table 1 shows the parameters used for training the different GSOMs. *Id* is the GSOM identifier (which starts with the subset identifier x, meaning 1, 2 or 3), type is the shape of the net (GRID means squared network and HEX means hexagonal network), LR^{g}_{ini} is the initial learning rate in the growth phase and LR^{s}_{ini} is the initial learning rate in the smooth phase. In all cases, it has been used an initial network of four nodes, a spread factor $SF = 0.5$, a distribution factor $FD = 0.1$, a value of $R = 3.8$, [9], and initial and final σ values of 1 y 0.1, respectively. Some of the GSOMs have been trained setting $\Delta\sigma$ and others fixing the number of iterations in each phase.

The final numbers of neurons and error measures are collected in Table 2, where Neurons indicate the final size of the GSOM (which is the result of the data reduction) and Reduction is the reduction rate = (1 - number of samples in the reduced set /number of samples in the original set)$\cdot$ 100. Figure 5 shows the weight vectors of the GSOMs $s1\_2$, $s2\_2$ and $s3\_2$ in the input space. The results show similar adaptations of the GSOMs independently of the number of inputs, and the reduction increases with the size of the input dataset. The values of quantE, stdNodes and simplicity differ, as they depend on the amount of input data, but the variation between subsets is low. More detailed visualizations of the adaptability of growing self-organizing networks are made by [44].

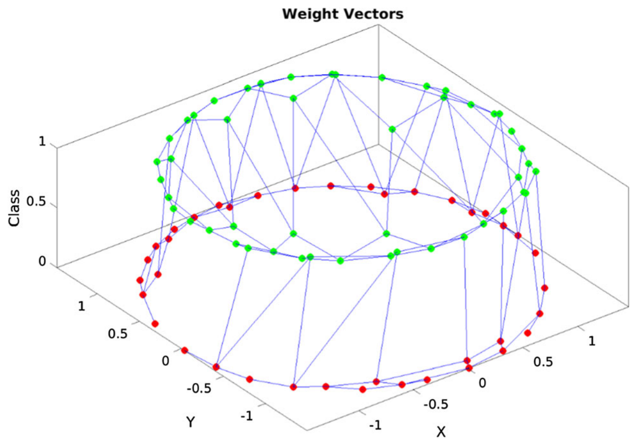


Fig. 3 Ring with two classes, where the red and green dots differentiate both of them

3.2 k-NN

Finally, the k-nearest neighbor algorithm for classification [45] is used, which assigns a class for each input vector based on the class of the k closest prototypes in the space. Initially, a set of labeled prototypes is required. These are vectors that represent the data the best as possible to obtain good classifications. A set of prototypes could be a real dataset, but this will carry a great computational cost. It is essential to use a helpful dataset, given that poorly balanced sets will falsify predictions. If the real data and the prototypes contain a very different proportion of positives, the assignments will not be correct. The weight vectors of the GSOM will be used as prototypes for the classifier.

3.2.1 Evaluation

To measure the performance of the binary classifier [45], three different ratios that take into account the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are used:

- Average accuracy (ACC): the hit rate.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

- Sensitivity (SEN) or recall: the rate of properly detected defaults.

$$SEN = \frac{TP}{TP + FN} \tag{12}$$

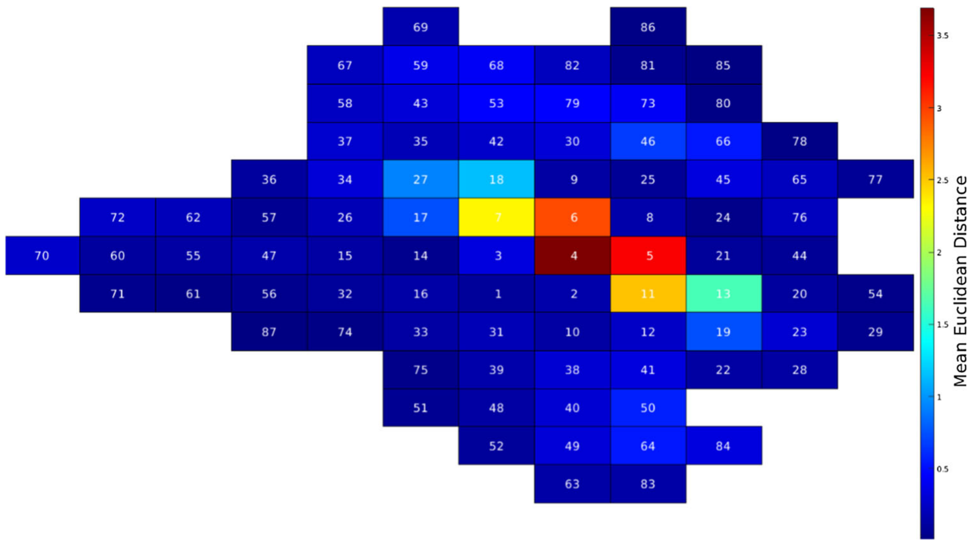


Fig. 4 U-matrix of a GSOM applied to the dataset with ring shape

Table 1 Training parameters for each letter *G* GSOM, where *sx* indicates subsets 1, 2 or 3

Id	Type	LR^g_{ini}	LR^s_{ini}	α	$\Delta\sigma^g$	$\Delta\sigma^s$
sx_1	GRID	0.3	0.5	1	0.1	0.05
sx_2	GRID	0.5	0.3	1	0.1	0.05
sx_3	GRID	0.5	0.5	0.7	0.1	0.05
sx_4	HEX	0.5	0.5	1	0.1	0.05

Id	Type	LR^s_{ini}	LR^s_{ini}	α	Growth it	Smooth it
sx_5	GRID	0.5	0.5	1	50	100
sx_6	GRID	0.5	0.5	1	10	100
sx_7	GRID	0.5	0.5	1	50	50
sx_8	GRID	0.5	0.5	1	100	100

Table 2 Results obtained for each GSOM of the letter G

Id	quantE	stdNodes	simplicity	Neurons	Reduction (%)
s1_1	0.226	0.083	291.085	128	87.200
s1_2	0.353	0.118	131.097	46	95.400
s1_3	0.274	0.091	207.833	123	87.700
s1_4	0.329	0.113	271.289	57	94.300
s1_5	0.420	0.158	105.066	42	95.800
s1_6	0.376	0.128	117.729	49	95.100
s1_7	0.342	0.113	156.099	49	95.100
s1_8	0.397	0.137	114.764	44	95.600
s2_1	0.229	0.078	790.437	207	97.930
s2_2	0.332	0.108	188.857	63	99.370
s2_3	0.229	0.072	303.373	193	98.070
s2_4	0.332	0.110	519.052	77	99.230
s2_5	0.343	0.105	163.300	59	99.410
s2_6	0.341	0.108	154.000	57	99.430
s2_7	0.333	0.105	157.527	61	99.390
s2_8	0.336	0.104	170.616	60	99.400
s3_1	0.217	0.072	1134.208	251	98.745
s3_2	0.332	0.107	215.894	64	99.680
s3_3	0.213	0.067	334.224	243	98.785
s3_4	0.327	0.109	588.972	80	99.600
s3_5	0.323	0.100	168.357	66	99.670
s3_6	0.314	0.097	188.892	71	99.645
s3_7	0.314	0.097	190.444	70	99.650
s3_8	0.320	0.099	175.056	80	99.600

- False Positive rate (FPR): the rate or improperly detected defaults.

$$FPR = \frac{FP}{FP + TN} \tag{13}$$

- Balanced accuracy (BA): the mean between the sensitivity and the true negative rate, which is 1 - FPR.

$$BA = 1 - \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{14}$$

- Precision (PRE): the rate of true positives over all predicted as positive classes.

$$PRE = \frac{TP}{TP + FP} \tag{15}$$

- F1-score (F1): the harmonic mean of precision and sensitivity.

$$F1 = 2\frac{PRE \cdot SEN}{PRE + SEN} \tag{16}$$

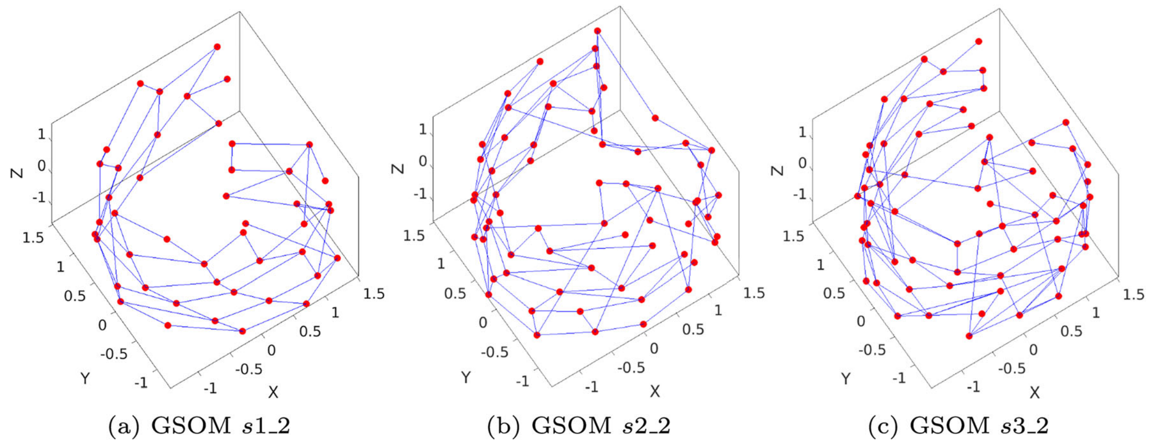


Fig. 5 Weight vectors of the GSOMs applied to the three different subsets of the letter G

- Cohen's Kappa: a measure of the agreement between the actual and predicted classes that compensates for the effect of randomness.

$$\kappa = \frac{P_o - P_e}{1 - P_e}, where \quad \begin{cases} P_o = \dfrac{TP + TN}{TP + TN + FP + FN} \\ P_e = \dfrac{(TP + FP)\cdot(TP + FN)}{(TP + TN + FP + FN)^2} + \dfrac{(TN + FN)\cdot(TN + FP)}{(TP + TN + FP + FN)^2} \end{cases} \tag{17}$$

4 Application results

This section shows the different tests carried out in two datasets to observe the adaptation of the GSOMs to the input space. These datasets correspond to an annual income register and bank records. A 1-NN classifier has been applied to obtain predictions using the GSOMs as prototypes. All neural networks have been trained using Intel® Xeon® CPU E3-1220 v6 at 3.00GHz and 48 GB DDR3.

4.1 Census income results

In this work, different experiments were carried out using the Census Income dataset, also known as the Adult dataset, from the UCI Machine Learning Repository [46]. Its use is intended for the classification between two types of incomes to predict whether a given person makes more than $50000 a year based on attributes such as age, sex, education, marital status, etc. It contains 48842 instances of multivariable data with 14 attributes with categorical and integer variables. The last variable corresponds to the output of the dataset, which represents the classification, meaning 1 a positive value and 0 otherwise. These data are divided into two sets. After a preprocessing step by joining the data and eliminating incomplete instances, a dataset of 45222 instances is obtained. As the original data are unbalanced, a balanced subset of 22416 instances with 50% of each class is created. Both were divided into training sets of 80% (36177 instances for the unbalanced data and 17933 for the balanced data) and a test set of 20% (9045 unbalanced instances and 4483 balanced). As a baseline, the case without data reduction in the unbalanced dataset is used. The results on the test set are as follows: accuracy of 79.67%, sensitivity of 58.95%, false positive ratio of 13.56%, F1-score of 58.83% and Cohen's Kappa of 45.33%.

4.1.1 GSOM and k-NN performance

Many experiments have been conducted on this dataset modifying the training parameters and the network topology, defined by the size–It is automatically obtained from the GSOM. Table 3 contains the parameters of the different

Table 3 Training parameters for each GSOM applied to Census Income dataset

Id	Type	LR^g_{ini}	LR^s_{ini}	α	Growth it.	Smooth it
1,11	GRID	0.5	0.1	1	10	20
2,12	GRID	0.5	0.5	1	50	100
3,13	GRID	0.5	0.5	1	10	100
4,14	GRID	0.5	0.5	1	10	50
5,15	GRID	0.3	0.1	1	10	20
6,16	GRID	0.5	0.5	0.9	10	20
7,17	HEX	0.5	0.1	1	10	20
8,18	HEX	0.5	0.5	1	50	100
9,19	HEX	0.5	0.5	1	10	100
10,20	HEX	0.5	0.5	0.9	10	20

GSOMs applied to the Census Income dataset, either the unbalanced data (GSOMs 1 to 10) and the balanced data (GSOMs 11 to 20). The same parameters have been used for each type of dataset to compare the results. *Id* is the GSOM identifier and LR^{g}_{ini} and LR^{s}_{ini} are the initial learning rates in growth and smooth phases. In every GSOM, it has been employed a spread factor $SF = 0.5$, a factor of distribution $FD = 0.1$, a value of $R = 3.8$, [9], and initial and final σ values of 1 and 0.1, respectively.

Before training the GSOMs, several SOMs have been trained to compare them with the GSOMs and validate the method. Each SOM was trained with 75 iteration steps, and a hexagonal topology. Four grid sizes were tested: 10 nodes $\times$ 10 nodes, 15 nodes $\times$ 15 nodes, 20 nodes $\times$ 20 nodes and 25 nodes $\times$ 25 nodes. The error measures are gathered in Tables 4 and 5. The time to train each model was recorded in the "Tr. time" column. As the SOM maps the space obtaining non-binary variables, the output was binarized, assigning to each neuron the mode output of the training vectors for which it is the BMU. Both datasets produce similar measures, although the training times are significantly lower using the balanced dataset.

The error measures and training times for each GSOM are listed in Table 6 for the unbalanced data and 7 for the balanced data. As well as with the SOMs, the training times are lower with the balanced dataset, and the rest of the measures are of the same order. However, the values of simplicity vary significantly from one experiment to other. In general, for models of the same size, GSOMs require much less training time than SOMs.

The accuracy, sensitivity, and false positives rates calculated after applying the 1-NN classifier are listed in Tables 8 and 9 for the SOMs, and 10 and 11 for the GSOMs. The models with the unbalanced dataset have slightly higher accuracy, but lower sensitivity compared to the ones with the balanced dataset. Moreover, the increase in sensitivity is more significant with GSOMs.

The results from both the unbalanced and balanced data have values of quantE, stdNodes, and reduction that are similar, indicating that the networks are equally adapted to the input data and near the BMUs. Nevertheless, the similarity values are lower for the balanced data, indicating that the models are less spread out. Although the reduction is similar in most cases, as shown in Fig. 6, the balanced dataset was smaller, so it provides GSOMs with fewer nodes, obtaining better training times. With unbalanced data, the accuracy is higher than with the balanced data, but the false positive rate is higher too, and the sensitivity is lower. Compared to the baseline accuracies, better results are obtained: with accuracies in the test subset over 80% for GSOMs 6, 7, 10 and 14. Figure 7 gathers the mean reductions, accuracies, balanced accuracies, F1-scores, Cohen's Kappa values and inverse of training times for the five best experiments in the test set. This graph allows associating the best models with larger areas covered by their corresponding polygon, which highlights the superior performance of the GSOMs on the balanced dataset compared to the other models.

To obtain a better understanding of the results, the U-matrix and an output map of GSOM 16 have been plotted in Fig. 8. The U-matrix helps to distinguish the zones where the neurons are more similar, displaying clustering capabilities, while the output map prints a color associated to the classification value (or output) of each neuron. Given that the GSOM is adapted, its values need to be binarized, considering that values closer to 1 correspond to positive classifications. Bluer neurons in the U-matrix relate to homogeneous zones in the output map and red colors in the U-matrix correspond to zones with more color variety in the output map. With GSOM 16, a reasonably expanded network is obtained, and it is possible to distinguish clearly two great areas in the U-matrix, implying that there are two types of people in the dataset. By further analysis, even three or four groups could be identified.

4.2 Bank data results

Different GSOMs have been implemented in GAMCO's ARM (Advanced Risk Management) solutions using bank clients datasets. Starting from an initial network of four nodes, the GSOMs have been trained to sample the data. The aim is to obtain a small set of prototypes for a 1-NN classifier. It also provides a low-dimensional representation of them that allows faster analysis and easier comprehension at a human level. The weight vectors of the GSOM are clusters that constitute a new dataset themselves. They can be subsequently combined with further analysis and classification algorithms such as k-NN. The dataset used is too large, so it is not feasible to apply a k-NN classifier without

Table 4 Training results for each SOM with the unbalanced CI dataset

Id	QuantE	stdNodes	Simplicity	Neurons	Reduction (%)	Tr. time (h)
1	2.224	0.483	801	100	99.723	65.559
2	1.971	0.353	1800	225	99.378	126.575
3	1.790	0.326	3452	400	98.943	225.649
4	1.657	0.303	4842	625	98.272	408.930

Table 5 Training results for each SOM with the balanced CI dataset

Id	QuantE	stdNodes	Simplicity	Neurons	Reduction (%)	Tr. time (h)
1	2.195	0.429	854	100	99.442	33.709
2	1.943	0.359	4658	225	99.745	72.040
3	1.777	0.327	3396	400	97.769	106.432
4	1.650	0.278	5924	625	96.515	179.303

Table 6 Training results for each GSOM applied to the Census Income Dataset with unbalanced training set

Id	quantE	stdNodes	Simplicity	Neurons	Reduction (%)	Tr. time (s)
1	1.618	0.581	2068	208	99.425	30.223
2	1.573	0.561	3644	279	99.229	180.601
3	1.628	0.589	2169	197	99.455	110.301
4	1.732	0.599	3382	200	99.447	59.810
5	1.281	0.463	6373	751	97.924	88.449
6	1.530	0.556	2546	271	99.250	37.257
7	1.605	0.582	3596	229	99.367	32.423
8	1.556	0.541	8635	368	98.983	212.803
9	1.611	0.576	5268	262	99.276	136.627
10	1.530	0.558	4128	292	99.123	40.465

Table 7 Training results for each GSOM applied to the Census Income dataset with balanced straining set

Id	quantE	stdNodes	Simplicity	Neurons	Reduction (%)	Tr. time (s)
11	2.027	0.630	1669	91	99.493	9.545
12	1.712	0.584	882	94	99.476	46.171
13	1.716	0.591	841	88	99.509	33.607
14	1.713	0.592	1008	96	99.465	19.587
15	1.314	0.460	3506	421	97.652	25.229
16	1.590	0.558	1216	142	99.208	11.766
17	1.720	0.588	1617	103	99.426	10.166
18	1.671	0.570	1830	121	99.325	52.758
19	1.684	0.586	1700	111	99.381	38.806
20	1.537	0.540	2764	195	98.913	14.318

Table 8 Results obtained for each SOM+1-NN applied to Census Income dataset with unbalanced data, expressed in %

	Train					Test				
Id	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ
1	81.1	46.7	7.6	54.9	43.5	80.4	46.5	7.97	54.7	42.7
2	80.6	47.1	8.3	54.7	42.8	80.9	47.3	8.38	54.5	42.7
3	81.2	52.7	9.45	58.1	46.2	80.9	50.6	9.13	56.7	44.7
4	81.5	53.5	9.33	58.8	47.1	81	53.4	9.72	58.6	46.4

Table 9 Results obtained for each SOM+1-NN applied to Census Income dataset with balanced data, expressed in %

	Train					Test				
Id	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ
1	75.1	76.9	26.6	75.6	50.3	74.8	76.5	27	75.2	49.6
2	76.8	77.2	23.6	76.9	53.7	76.7	77.1	23.7	77	53.5
3	76.6	77	23.8	76.7	53.2	76.6	76.9	23.7	76.5	53.2
4	78.5	79.9	22.9	78.8	56.9	78.4	78.8	21.9	78.5	56.9

Table 10 Results obtained for each GSOM+1-NN applied to Census Income dataset with unbalanced data, expressed in %

	Train					Test				
Id	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ
1	80.9	57.8	11.5	60.1	47.5	79.6	56.1	12.8	57.4	44.0
2	80.4	62.6	13.7	61.3	48.2	79.4	60.7	14.5	59.1	45.3
3	80.4	60.7	13.1	60.6	47.5	78.9	58.0	14.3	57.4	43.4
4	78.9	64.3	16.3	60.3	46.0	78.2	62.2	16.7	58.3	43.6
5	82.6	61.8	10.5	63.8	52.4	79.7	56.1	12.6	57.5	44.2
6	81.4	62.2	12.2	62.5	50.2	80.6	61.3	13.1	60.8	47.9
7	81.5	60.4	11.6	61.8	49.6	80.4	59.2	12.7	59.7	46.7
8	78.8	60.9	15.2	58.8	44.6	77.7	58.0	16.0	56.0	41.0
9	80.3	62.0	13.7	61.0	47.9	79.7	60.9	14.2	59.5	46.0
10	81.5	55.4	9.8	59.9	48.0	80.1	51.3	10.6	55.8	43.1

Table 11 Results obtained for each GSOM+1-NN applied to the Census Income dataset with balanced data, expressed in %

	Train					Test				
Id	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ
11	77.7	86.8	31.4	79.5	55.4	78.6	87.6	30.7	78.3	52.7
12	78.2	82.6	26.2	79.1	56.4	79.2	82.7	24.5	80.1	58.3
13	78.6	83.9	26.7	79.7	57.2	79.0	83.1	25.1	80.1	58.0
14	78.4	82.1	25.3	79.2	56.8	80.4	83.7	23.0	81.2	60.7
15	80.1	81.2	21.1	80.3	60.1	79.3	79.5	21.0	79.5	58.5
16	79.2	82.3	23.9	79.8	58.4	79.5	81.4	22.5	80.1	59.0
17	77.9	82.6	26.9	78.9	55.7	79.6	83.8	24.8	80.6	59.1
18	77.8	80.7	25.1	78.4	55.6	78.3	80.0	23.4	78.9	56.6
19	78.0	82.1	26.0	78.9	56.1	77.8	81.0	25.5	78.7	55.5
20	79.3	82.4	23.9	79.9	58.6	79.4	82.2	23.4	80.2	58.9

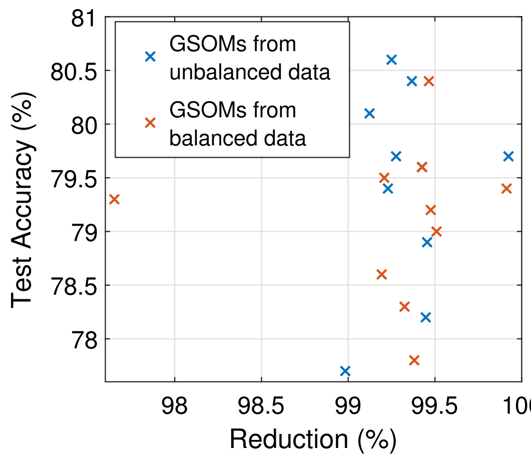


Fig. 6 Accuracy against data reduction percentage for the GSOMs of the Census Income dataset

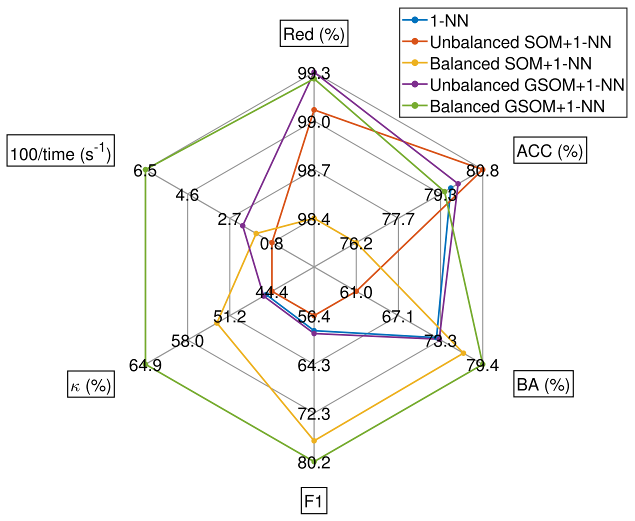


Fig. 7 Mean reductions, accuracies, balanced accuracies, F1-scores, Cohen's Kappa values and inverse of training times for the Census Income dataset

first applying a prototype reduction method. Therefore, a comparison between SOM+k-NN and GSOM+k-NN has been performed.

The dataset corresponds to different anonymous clients of a bank, and the objective is to determine well in advance when a customer is going to default. The total client portfolio is made up of around 2 million clients, and their data is obtained from four different sources: customers,

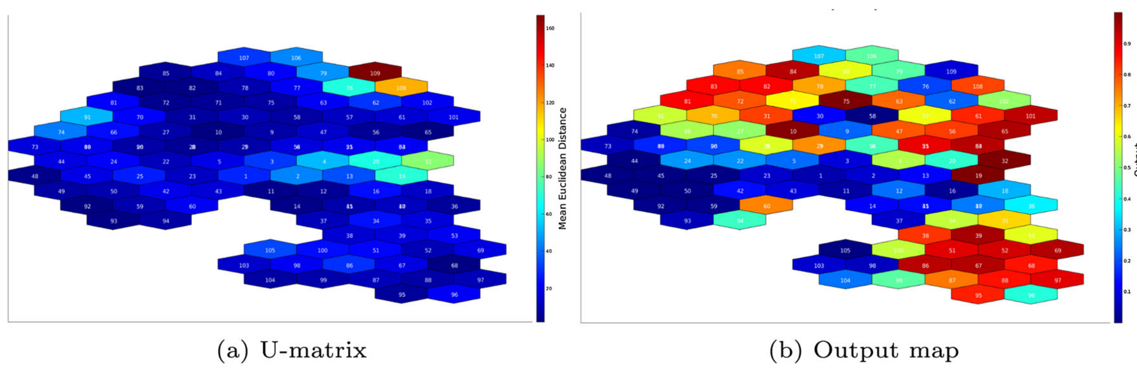


Fig. 8 U-matrix and Output visualizations of the GSOM_CI 16

bank transactions, contracts, and defaults. There are approximately 2.5 billion transactions and 500 million hires. Three-year historical data were used to build the training data, looking for examples of customer defaults and payment cases with different previous histories. Table 12 shows a summary of the type of data used.

To obtain the dataset, every month, more than 7 million bank customers were processed and more than 1.6 million customers were evaluated by the models. The steps to verify to obtain the customers to be evaluated by the model are as follows:

1. Customers associated with any loan contract (asset).
2. The customer must be the holder of at least one contract of both assets and deposits.
3. Any loan contract was still in vigor.
4. With a contract that did not have a grace period on the date of evaluation of the client's risk by the model.
5. With no incidences in the credit payments (they are healthy clients).
6. They have some movement in their bank accounts.

On the other hand, for the training of the models, sets of input/output vectors had to be created, each vector corresponding to the features of a customer. These feature vectors are divided into three sets of input/output vectors: training, validation, and test. For this purpose, we worked primarily with the following information sources or datasets:

- Customers: 7 million records.
- Transactions: more than 2.9 billion transactions.
- Contracts: more than 6.5 million loan contracts and 494 million contracts for other products.
- Incidents: more than 14.3 million defaults.

A very important and costly step is the labeling of the vectors corresponding to a client and a date, that is to say, to mark if that client will default or not at a given time from a date. For example, customers on day d will be labeled as unpaid, if on $d + 180$ they will have a default of more than 30 days on any of their receivables. These vectors will be used to create models that detect defaults of more than 30 days and 180 days before a given date. In the sets of input and output vectors, great care was taken to include customers with the following characteristics, thus achieving to have feature vectors of the different typologies of customers in terms of defaults on a specific historical date, in which the feature vector is labeled:

- Customers with no defaults
- Customers who have defaults prior to the date on which it is labeled, but no subsequent defaults.
- Customers who have defaults after the date on which it is labeled.

Table 12 Brief description of the data used in ARM

Source	Variables
Customers	Age, ZIP code, segmentations, civil status, gender, housing type, patrimonial value, economic sector, country, country of birth, level of studies, number of children
Transactions	Monthly averages, debits and credits, balances, product categories, number of transactions
Hiring	Quantity of products, types, products without ownership, customer seniority
Loans	Number and type of loans, amounts contracted in consumer loans and mortages, interests
Defaults	Time with default, last default, maximum default, defaults on consumer loans and mortages

- Customers who have defaults on the date being labeled (in the example above: 180 days after the date being labeled are at least 30 days late on any of their loans).
- Customers who have not defaulted, but who have had and will have defaults (beyond the 180 days that serves as an example).

Finally, a balanced dataset was built (same number of payments as of defaults) with more than 350,000 clients and is divided into a training set of 284259 clients (75%), a validation set of 56852 (15%) clients and a test set of 37900 clients (10%). The vectors contain 125 features of the individuals and their bank records. One of the variables corresponds to the output, which indicates if the client has committed default.

4.2.1 GSOM and k-NN performance

Table 13 contains the parameters used for training the different GSOMs implemented. The GSOM is expected to obtain similar results to those achieved with a SOM in a much shorter time. To verify it, several SOMs have been trained with different topologies. Tables 14 and 16 collect the results of best SOMs obtained for four different topologies (squared of 20 nodes $\times$ 20 nodes for SOM 1, hexagonal of 23 nodes $\times$ 23 nodes for SOM 2, hexagonal of 25 nodes $\times$ 25 nodes for SOM 3 and hexagonal of 27 nodes $\times$ 27 nodes for SOM 4). Each of the selected SOMs was trained using 75 iteration steps, which were necessary to obtain good results. The training times are around 3 and 4 h, and the accuracies in the test set are all over 70%. Only one of the false positive rates is lower than 20%.

The error measures described in Sect. 3 have been calculated, as well as the definite number of neurons and the training duration in hours for each one of the GSOMs implemented. The results are gathered in Table 15. The most significant reductions are in GSOMs 1, 3 and 4, which

Table 13 Training parameters for each GSOM with the bank dataset

Id	Type	LR^g_{ini}	LR^s_{ini}	α	Growth it.	Smooth it
1	GRID	0.5	0.1	1	10	20
2	GRID	0.5	0.5	1	50	100
3	GRID	0.5	0.5	1	10	100
4	GRID	0.5	0.5	1	10	50
5	GRID	0.3	0.1	1	10	20
6	GRID	0.5	0.5	0.9	10	20
7	HEX	0.5	0.1	1	10	20
8	HEX	0.5	0.5	1	50	100
9	HEX	0.5	0.5	1	10	100
10	HEX	0.5	0.5	0.9	10	20

had only 10 growth iterations, a squared topology and an initial growth learning rate of 0.5. The GSOM with a lower initial growth learning rate obtains the lowest stdNodes and QuantE values, and also one of the lowest simplicities. It should be mentioned that GSOM 1 is comparable to SOM 1, GSOM 7 to SOM 2, SGOM 9 to SOM 3 and GSOM 10 to SOM 4, with higher simplicities and much shorter training times.

After applying 1-NN to the GSOMs to classify the clients in default (positive output) versus payment (negative output), the number of true and false positives and negatives have been obtained. Those numbers serve to obtain the previously mentioned performance ratios: Accuracy (hit rate), Sensitivity (correctly detected defaults) and False Positive rate (improperly detected defaults). These have been calculated and presented in Table 17. The accuracies in the test set are above 70% in most GSOMs, with 74.7% in the case of GSOM 5. The false positives only exceed 20% in GSOMs 1 and 6.

The obtained accuracies are represented over the reduction percentages of all GSOMs in Fig. 9. The highest accuracy is associated with a low reduction percentage, although the highest reduction also obtains a high accuracy in the test subset.

The average results of the best five experiments of each type are collected in Fig. 10, including mean reductions, accuracies, balanced accuracies, F1-scores, Cohen's Kappa values, and inverse of training times. The mean Kappa values, F1-scores, balanced accuracies, and reductions are similar between the SOMs and the GSOMs, however, the GSOMs tend to provide slightly higher accuracies and take much less training time.

Figure 11-left shows the adaptation of the GSOM 3 (with squared topology) to the input space in different features chosen randomly–feature 4 is related to the customer client, feature 8 refers to monthly charges, feature 21 relates to payments and feature 30 takes account of daily balances. The data output is the class, which indicates if the client has committed payment (0) or default (1). After a discretization process, the arrangement of the GSOM is shown in Fig. 12-left. For a better visualization of the performance using a GSOM of different topology, the weight vectors of the GSOM 9 and their adaptation to the input space are represented in Figs. 11-right and 12-right. This map has a hexagonal topology and has been trained using the same parameters as for the GSOM 3. As previously mentioned, hexagonal topologies have faster growth. The expansion of the GSOM can be perceived, as well as its adaptation to the training data, providing more prototypes in the zones with higher data density. At first sight, there is not a huge difference in the adaptability of the two topologies used.

Table 14 Training results for each SOM with the bank dataset

Id	QuantE	stdNodes	Simplicity	Neurons	Reduction (%)	Tr. time (h)
1	6.075	4.912	26952	400	99.859	3.254
2	5.979	3.690	35410	529	99.814	3.780
3	7.862	3.535	2543	625	99.780	4.334
4	5.885	3.015	35310	729	99.744	3.408

Table 15 Training results for each GSOM with the bank dataset

Id	QuantE	stdNodes	Simplicity	Neurons	Reduction (%)	Tr. time (h)
1	6.168	5.200	36947	439	99.846	0.751
2	6.520	3.446	100082	507	99.822	4.144
3	6.535	3.637	90178	439	99.846	2.931
4	6.936	3.714	107209	439	99.846	1.568
5	5.883	2.780	48758	998	99.649	1.917
6	6.287	4.249	60855	511	99.820	0.840
7	6.128	4.812	53224	503	99.823	0.869
8	6.572	3.142	179350	936	99.671	5.118
9	6.560	3.458	160374	564	99.802	3.741
10	5.949	3.659	60376	766	99.731	1.372

Table 16 Performance of each SOM+1-NN, expressed in % with the bank dataset

	Train					Validation					Test				
Id	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ
1	74.5	67.2	19.3	70.8	48.2	74.2	67.1	19.7	70.5	47.8	74.0	66.9	20.0	70.3	47.2
2	75.0	69.3	20.1	71.8	49.4	74.3	68.7	21.0	71.1	48.0	74.3	68.8	21.0	71.1	48.0
3	71.7	59.8	18.2	66.1	42.2	71.7	59.6	18.1	66.0	42.2	71.2	59.6	18.9	65.5	41.2
4	75.7	69.3	18.9	72.4	50.7	74.7	68.2	19.6	71.3	48.8	74.6	68.5	20.1	71.3	48.7

Table 17 Performance of each GSOM+1-NN, expressed in % with the bank dataset

	Train					Validation					Test				
Id	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ	ACC	SEN	FPR	F1	κ
1	73.8	66.6	20.1	70.0	46.8	73.6	66.1	20.0	69.7	46.4	73.7	66.3	20.1	69.6	46.3
2	72.3	61.1	18.2	67.0	43.5	72.2	60.9	18.2	71.3	43.3	72.2	61.0	18.2	66.8	43.3
3	71.5	58.3	17.3	65.3	41.7	71.6	58.4	17.1	65.4	41.9	71.5	58.4	17.2	65.1	41.5
4	69.9	56.4	18.6	63.3	38.5	68.9	56.3	18.6	63.2	38.3	69.9	56.4	18.6	63.0	38.1
5	74.9	67.3	18.7	71.1	49.0	74.5	67.0	19.0	70.7	48.3	74.7	67.1	18.9	70.7	48.4
6	72.7	64.1	20.0	68.3	44.5	72.7	64.1	19.9	68.4	44.6	72.7	64.1	20.0	68.0	44.2
7	73.6	63.8	18.0	69.0	46.3	73.3	63.3	18.2	68.6	45.7	73.5	63.6	18.1	68.6	45.8
8	71.4	60.7	19.4	66.1	41.8	71.3	60.4	19.4	65.9	41.5	71.4	60.5	19.4	66.0	41.6
9	71.3	55.8	15.5	64.2	41.1	71.4	55.7	15.2	64.2	41.3	71.4	55.8	15.4	63.9	40.9
10	74.5	67.4	19.4	70.9	48.3	74.0	66.7	19.7	70.3	47.4	74.3	67.0	19.6	70.3	47.5

As well as for the Census Income case, the U-matrices and output maps have been represented in Figs. 13 and 14 to obtain a visual explanation of the mapping made by the GSOMs and the subsequent classifications carried out by

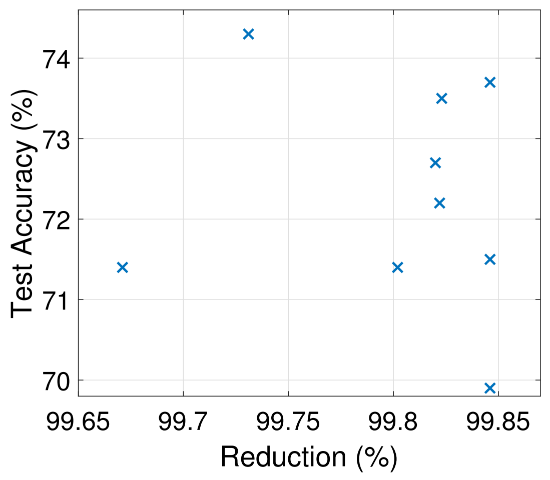


Fig. 9 Accuracy against data reduction percentage for the GSOMs of the bank dataset

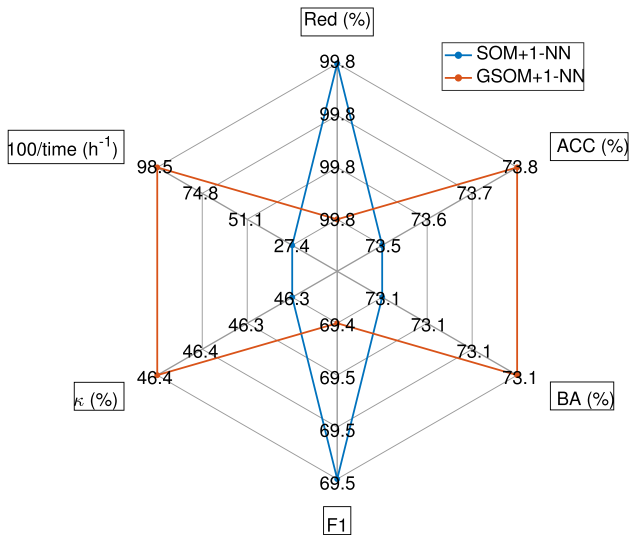


Fig. 10 Mean reductions, accuracies, balanced accuracies, F1-scores, Cohen's Kappa values and inverse of training times of each experiment for the bank dataset

the 1-NN classifiers. Higher values in the U-matrix relate to more variability in the output map. Two main groups can be distinguished for GSOM 3, whereas in GSOM 9 the visual explanation is not straightforward.

The GSOMs have lower training times than the SOMs and, although the accuracies are lower in most cases, the false positive rates are lower too, giving similar balanced accuracies. Better evaluations are related to a combination of high sensitivity and low false positive rate, instead of only taking into account the accuracy. For comparing better the results, the sensitivity (SEN) and false positive rate (FPR) of the GSOMs and SOMs in the test subset are represented in Fig. 15. The points with the highest SEN are given by SOMs, while the points with the lowest FPR are given by GSOMs. However, in the central area of the graph, there are points with high SEN and FPR below 20% that correspond to GSOMs.

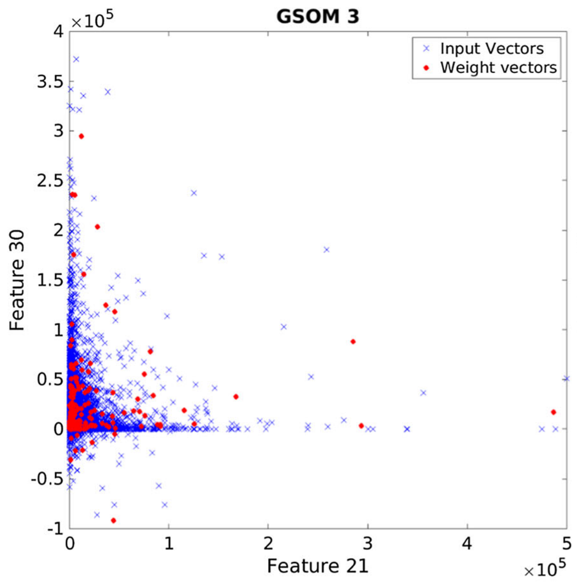
5 Discussion

This section aims to provide a brief discussion of the results obtained. Some observations are extracted:

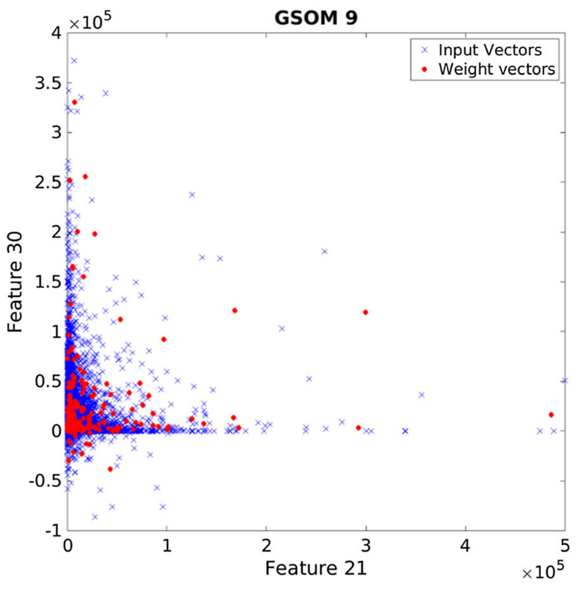
- The Census Income dataset was used to create balanced and unbalanced sets. The unbalanced set showed better accuracy and false positive rate, but lower sensitivity. Balancing the data prevented a significant decrease in detecting true positives, improving training times, Kappa values, F1-scores, and balanced accuracies.
- The results reveal that the GSOMs reduce the training data size in more than a 99% with a low loss of information by mapping the input space.
- A spatial adaptation of the data is observable. In the illustrative example, it is possible to recognize perfectly a 3-dimensional letter G, and in the bank dataset, the weight vectors are capable of adapting to the input space with more neurons in the zones with a major density.
- The use GSOMs allows the creation of a group of prototypes that can be used with algorithms like 1-NN, obtaining accuracies of more than 70% with false positives lower than 20% in a banking application and saving a great computational time.

The resulting number of nodes in each one of the GSOMs indicates that it is possible to represent the different datasets with no need for the complete growth of the maps. With more neurons, it is expected that the standard deviation will be lower, as there will be more neurons, but this will not necessarily lead to much better results. For example, GSOM 8 of the bank dataset contains 936 neurons, almost twice GSOM 7, but the quantification error remains stable and training time is quintupled. The results also highlight the need for a suitable parametrization, as too low initial learning rates in the growing phase lead to an overgrow of the map, a great difference between growing and smoothing iterations reduce the sensitivity and a lower α increases the false positive rate.

The U-matrix representation offers the possibility of separating clients into different groups, which serves as a preliminary step to customer classification and analysis, and its potential is strengthened when combined with tools as the output map. This feature is noticeable in the Census Income dataset. However, due to the great number of features in the bank dataset and the unknown correlation between them, the clusters obtained with the U-matrix are not easy to extrapolate, which highlights the difficulty of

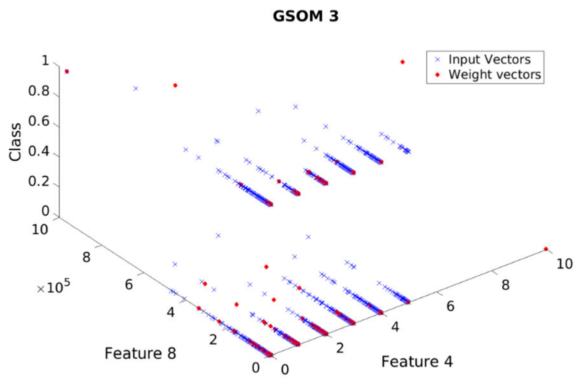


(a) Weight and input vectors of GSOM 3 with the bank dataset

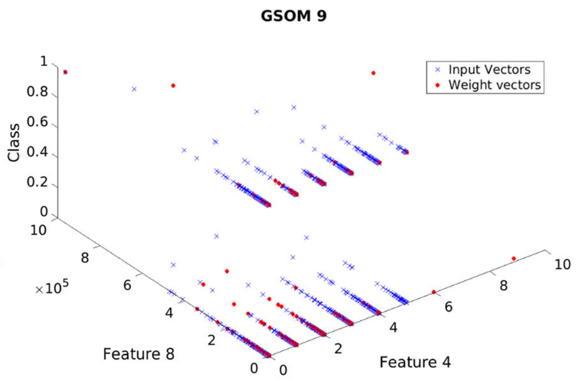


(b) Weight and input vectors of GSOM 9

Fig. 11 Representation of features 21 and 30 with the bank dataset

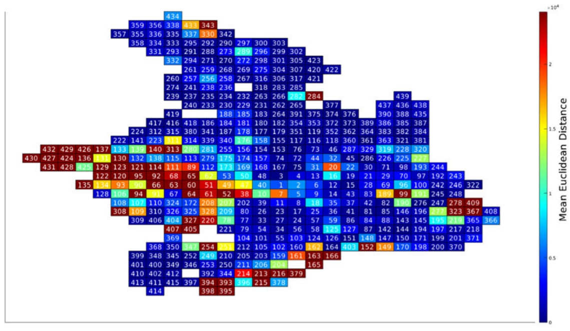


(a) Weight and input vectors of GSOM 3

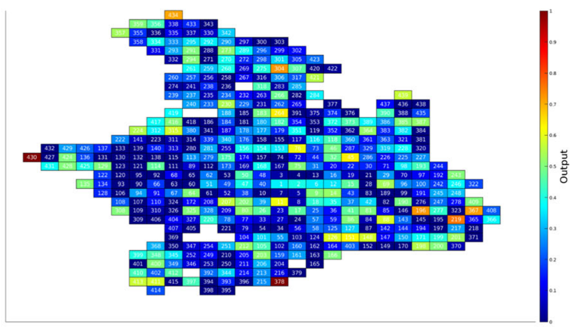


(b) Weight and input vectors of GSOM 9

Fig. 12 Representation of features 4, 8 and the class with the bank dataset



(a) U-matrix



(b) Output map

Fig. 13 U-matrix and Output visualizations of the GSOM 3 with the bank dataset

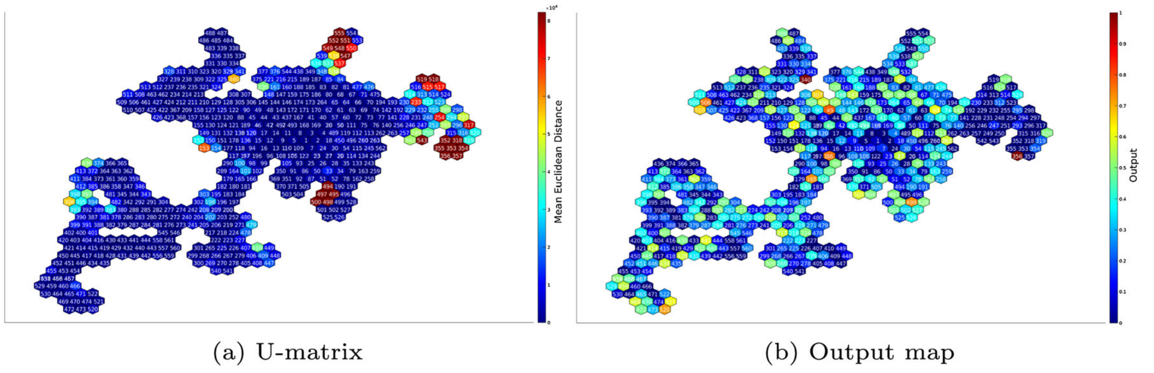


Fig. 14 U-matrix and Output visualizations of the GSOM 9 with the bank dataset

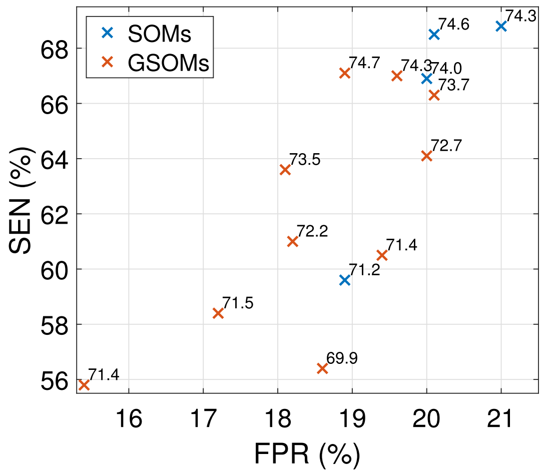


Fig. 15 Sensitivity versus False positive rate for all SOMs and GSOMs with the training subset of the bank dataset. The numbers in the graphic represent the accuracy

finding patterns between data when it diverges from a theoretical distribution.

6 Conclusions

This work addresses the application of a GSOM with 1-NN to different datasets to address the problem of prototype generation for a binary classifier. First, the Census Income dataset is used for an academic application of the method and then, a real application is carried out on an actual dataset from a bank. Specifically, in the case of banking data, the number of clients and the challenge of selecting a low number of features for tasks as classification lead to the need for reducing its size. This work addresses the problem of classifying the clients into two categories using a 1-NN classifier and, for training the classifiers, the use of prototypes obtained from a GSOM is proposed.

The proposed methodology divides the data into training, validation and test subsets and trains different GSOMs to represent the training set. Then, the weight vectors of the GSOMs are used as prototypes for a 1-NN classifier that makes predictions on new data. In the case of the Census Income dataset, two types of experiments were conducted, using either balanced and unbalanced data. Different SOMs and GSOMs were used for comparison purposes for the bank dataset.

The application to Census Income (a smaller dataset where it is possible to apply the 1-NN classifier without any data reduction method) has been carried out to demonstrate that the proposed method is effective since the classification performances are maintained or even improved from 1-NN to GSOM+1-NN, with very high reduction rates. This application gives us guarantees of the good performance of the proposed method. In the application to the banking sector, where very large datasets are used, it is not possible to apply k-NN without data reduction, so a comparison is made between SOM+k-NN and GSOM+k-NN.

The results obtained demonstrate that the use of a GSOM in data with a great number of vectors and variables achieves a small size representation that facilitates further classification tasks, with reductions of more than 99% in the training data. The GSOMs obtain similar results to the SOMs with even higher accuracy in some experiments (accuracies over 70% and false positive rates under 20%) and much lower training times (from more than 3 or 4 h to even less than 1 h in some cases).

When dealing with large datasets, not only the reduction of the size is of great interest. A dimensionality reduction with correct feature selection can also be crucial in the performance and cost of the classification process. A complete improvement in computational cost will come from a combination of data and dimensionality reduction techniques. This issue is part of future lines of development. Another aspect to analyze as future work is the normality of the data and its deviations, e.g., the work by Mantalos et al. [47].

Acknowledgements This work has been made possible by the projects SEKAS. Self-growing learning Knowledge Architecture for Security, supported by FEDER / Ministry of Science, Innovation and Universities, State Research Agency/Project [RTC-2017-6178-8]; and Model Predictive Control based on Machine Learning in Big Data Applications [PI-1914/24/2019 and PI-2030/24/2020].

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and investigation were performed by R-M and G-C. Supervision was performed by N-R and resources were provided by Pavón. The first draft of the manuscript was written by R-M and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Funding for open access publishing: Universidad de Sevilla/CBUA.

Availability of data and materials The Census Income dataset analyzed during the current study is available in [46]. The rest of the data that support the findings of this study are property of GAMCO, SL. and restrictions apply to the availability of these data, which were used under licence for the current study, and so are not publicly available.

Code availability Restrictions apply to the availability of the codes.

Declarations

Conflict of interest The authors declare that there are not conflicts of interest.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication The authors consent to the submission of this article.

References

1. Zinner S, Ivanenko V, Tynchenko V, Volegzhanin P, Stashkevich A (2021). Using machine learning methods in problems with large amounts of data. https://doi.org/10.47813/dnit-mip3/2021-2899-181-187
2. Mohammad SM (2019) Cloud computing in it and how it's going to help United States specifically. Int J Comput Trends Technol (IJCTT) https://doi.org/10.14445/22312803/IJCTT-V67I10P118
3. Garcia S, Derrac J, Cano J, Herrera F (2012) Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans Pattern Anal Mach Intell 34(3):417–435. https://doi.org/10.1109/TPAMI.2011.142
4. Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69. https://doi.org/10.1007/BF00337288
5. Kohonen T (1990) The self-organizing map. Proc IEEE 78:1464–1480. https://doi.org/10.1109/5.58325
6. Kohonen T (1995) The self-organizing maps. Springer, Berlin/Heidelberg, Germany
7. Vesanto J (1999) SOM-based data visualization methods. Intell Data Anal 3:111–126. https://doi.org/10.3233/IDA-1999-3203
8. Alahakoon D, Halgamuge SK, Srinivasan B (1998) A self-growing cluster development approach to data mining. In: SMC'98 Conference Proceedings 1998 IEEE international conference on systematics man, and cybernatics (Cat. No.98CH36218), vol. 3, pp. 2901–29063. https://doi.org/10.1109/ICSMC.1998.725103
9. Alahakoon D, Halgamuge SK, Srinivasan B (2000) Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Trans Neural Netw 11(3):601–614. https://doi.org/10.1109/72.846732
10. Konieczny J, Stojek J (2021) Use of the k-nearest neighbour classifier in wear condition classification of a positive displacement pump. Sensors. https://doi.org/10.3390/s21186247
11. Santos Ruiz Idl, López Estrada FR, Puig Cayuela V, Blesa Izquierdo J, Javadiha M (2019) Localización de fugas en redes de distribución de agua mediante k-nn con distancia cosenoidal. In: CNCA-Congreso Nacional de Control Automático, pp. 370–375
12. Tharwat A, Mahdi H, Elhoseny M, Hassanien AE (2018) Recognizing human activity in mobile crowdsensing environment using optimized K-NN algorithm. Expert Syst Appl 107:32–44. https://doi.org/10.1016/j.eswa.2018.04.017
13. Triguero I, García-Gil D, Maillo J, Luengo J, García S, Herrera F (2019) Transforming big data into smart data: an insight on the use of the k-nearest neighbors algorithm to obtain quality data. Wiley Interdiscip Rev Data Min Knowl Discov 9(2):1289
14. Ghosh AK (2006) On optimum choice of k in nearest neighbor classification. Comput Stat Data Anal 50(11):3113–3123. https://doi.org/10.1016/j.csda.2005.06.007
15. Ghosh AK (2007) On nearest neighbor classification using adaptive choice of k. J Comput Graph Stat 16(2):482–502. https://doi.org/10.1198/106186007X208380
16. Suguna N, Thanushkodi K (2010) An improved k-nearest neighbor classification using genetic algorithm. Int J Comput Sci 7(2):18–21
17. Hsu C-M, Chen M-S (2008) On the design and applicability of distance functions in high-dimensional data space. IEEE Trans Knowl Data Eng 21(4):523–536. https://doi.org/10.1109/TKDE.2008.178
18. Maillo J, Ramírez S, Triguero I, Herrera F (2017) KNN-is: an iterative spark-based design of the k-nearest neighbors classifier for big data. Knowl Based Syst 117:3–15. https://doi.org/10.1016/j.knosys.2016.06.012
19. Rosero-Montalvo PD, Umaquinga-Criollo AC, Flores S, Suarez L, Pijal J, Ponce-Guevara KL, Nejer D, Guzman A, Lugo D, Moncayo K (2017) Neighborhood criterion analysis for prototype selection applied in wsn data. In: 2017 international conference on information systems and computer science (INCISCOS), pp. 128–132. https://doi.org/10.1109/INCISCOS.2017.47. IEEE
20. Suyal H, Singh A (2021) Improving multi-label classification in prototype selection scenario. Comput Intell Healthc Inform. https://doi.org/10.1002/9781119818717.ch6
21. Gurumoorthy KS, Jawanpuria P, Mishra B (2021) Spot: a framework for selection of prototypes using optimal transport.

arXiv preprint arXiv:2103.10159. https://doi.org/10.1007/978-3-030-86514-6_33

22. Kasemtaweechok C, Suwannik W (2019) Adaptive geometric median prototype selection method for k-nearest neighbors classification. Intell Data Anal 23(4):855–876. https://doi.org/10.3233/IDA-184190
23. Triguero I, Derrac J, Garcia S, Herrera F (2011) A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans Syst Man Cybern Part C (Appl Rev) 42(1):86–100. https://doi.org/10.1109/TSMCC.2010.2103939
24. Ougiaroglou S, Filippakis P, Evangelidis G (2021) Prototype generation for multi-label nearest neighbours classification. In: international conference on hybrid artificial intelligence systems, pp. 172–183. https://doi.org/10.1007/978-3-030-86271-8_15
25. Elkano M, Galar M, Sanz J, Bustince H (2018) Chi-pg: a fast prototype generation algorithm for big data classification problems. Neurocomputing 287:22–33. https://doi.org/10.1016/j.neucom.2018.01.056
26. Lechevallier Y, Ciampi A (2007) Multilevel clustering for large databases, pp. 263–274. https://doi.org/10.1007/978-0-8176-4542-7_17
27. Sarlin P, Peltonen TA (2011) Mapping the state of financial stability. BOFIT discussion papers https://doi.org/10.1016/j.intfin.2013.05.002
28. Fox KL, Henning RR, Reed JH, Simonian R (1990) A neural network approach towards intrusion detection. In: Proceedings of the 13th national computer security conference, pp. 125–134. https://www.bibsonomy.org/bibtex/20f1a2a115ba200e7cdbe77cc0c8b80ad/schaul
29. Ichimura T, Hara A, Kurosawa Y (2007) A classification method for spam e-mail by self-organizing map and automatically defined groups. In: 2007 IEEE international conference on systems, man and cybernatics, pp. 2044–2049. https://doi.org/10.1109/ICSMC.2007.4413626
30. Sarkar S, Ejaz N, Maiti J (2018) Application of hybrid clustering technique for pattern extraction of accident at work: a case study of a steel industry. In: 2018 4th international conference on recent advance in information technology (RAIT), pp. 1–6. https://doi.org/10.1109/RAIT.2018.8389052. IEEE
31. Christyawan TY, Supianto AA, Mahmudy WF (2019) Anomaly-based intrusion detector system using restricted growing self organizing map. Indones J Electr Eng Comput Sci 13(3):919–926
32. Deboeck G, Kohonen T (2013) Visual explorations in finance: with self-organizing maps. Springer, London. https://doi.org/10.1007/978-1-4471-3913-3
33. Shanmuganathan M (2018) Visualized financial performance analysis: self-organizing maps (MS)
34. López Iturriaga FJ, Pastor Sanz I (2013) Self-organizing maps as a tool to compare financial macroeconomic imbalances: the European, Spanish and German case. Span Rev Financ Econ 11(2):69–84. https://doi.org/10.1016/j.srfe.2013.07.001
35. Barman D, Chowdhury N (2019) A novel approach for the customer segmentation using clustering through self-organizing map. Int J Bus Anal (IJBAN) 6(2):23–45
36. Quah JTS, Sriganesh M (2008) Real-time credit card fraud detection using computational intelligence. Expert Syst Appl 35(4):1721–1732. https://doi.org/10.1016/j.eswa.2007.08.093
37. Balasupramanian N, Ephrem BG, Al-Barwani IS (2017) User pattern based online fraud detection and prevention using big data analytics and self organizing maps. In: 2017 international conference on intelligent computing, instrumentation and control technologies (ICICICT), pp. 691–694. https://doi.org/10.1109/ICICICT1.2017.8342647. IEEE
38. Ganegedara H, Alahakoon D (2012) Redundancy reduction in self-organising map merging for scalable data clustering. In: The 2012 international joint conference on neural networks (IJCNN), pp. 1–8. https://doi.org/10.1109/IJCNN.2012.6252722
39. Kuo R-J, Rizki M, Zulvia FE, Khasanah A (2018) Integration of growing self-organizing map and bee colony optimization algorithm for part clustering. Comput Ind Eng 120:251–265. https://doi.org/10.1016/j.cie.2018.04.044
40. Ahmad N, Alahakoon D, Chau R (2010) Cluster identification and separation in the growing self-organizing map: application in protein sequence classification. Neural Comput Appl 19:531–542. https://doi.org/10.1007/s00521-009-0300-0
41. Ultsch A (2003) U*matrix: a tool to visualize clusters in high dimensional data
42. Uriarte EA, Martín FD (2005) Topology preservation in SOM. Int J Appl Math Comput Sci 1(1):19–22
43. Decker R, Monien K (2003) Market basket analysis with neural gas networks and self-organising maps. J Target Meas Anal Mark 11(4):373–386. https://doi.org/10.1057/palgrave.jt.5740092
44. Piastra M (2009) A growing self-organizing network for reconstructing curves and surfaces. In: 2009 international joint conference on neural networks, pp. 2533–2540. https://doi.org/10.1109/IJCNN.2009.5178709. IEEE
45. Zhang Z (2016) Introduction to machine learning: k-nearest neighbors. Ann Transl Med https://doi.org/10.21037/atm.2016.03.37
46. Dua D, Graff C (2017) UCI machine learning repository. http://archive.ics.uci.edu/ml
47. Mantalos P, Karagrigoriou A, Střelec L, Jordanova P, Hermann P, Kiselák J, Hudák J, Stehlík M (2020) On improved volatility modelling by fitting skewness in arch models. J Appl Stat 47(6):1031–1063. https://doi.org/10.1080/02664763.2019.1671323