A SEMI-SUPERVISED SELF ORGANIZING MAP FOR POST-PROCESSING THE SCENARIOS OF AN INTEGRATED DETERMINISTIC AND PROBABILISTIC SAFETY ANALYSIS

Francesco Di Maio[1], Roberta Rossett[1], Enrico Zio[1,2]

[1]Energy Department, Politecnico di Milano, Via La Masa 34, 20156 Milano, Italy
francesco.dimaio@polimi.it; enrico.zio@polimi.it
Chair System Science and the Energy Challenge, Fondation Electricité de France (EDF), CentraleSupélec, Université Paris Saclay, 92290 Paris, France
enrico.zio@supelec.fr; enrico.zio@edf.fr

In this paper, we propose the use of a Semi-Supervised Self Organizing Map (SSSOM) in a novel learning scheme based on the Manhattan distance, for post-processing the multi-valued dynamic scenarios that are collected during an Integrated Deterministic and Probabilistic Safety Analysis (IDPSA) of a dynamic system. The combinatorial explosion of the IDPSA scenarios makes the post-processing for safety information retrieval computationally very difficult. We, thus, propose SSSOM as a suitable approach for visualizing and classifying the IDPSA scenarios in safe, near misses, failed and prime implicants scenarios, whose consequences and probabilities of occurrence carry relevant information that are useful when assessing the risk of operation of the system. Results are shown with respect to a SIMULINK model that simulates the IDPSA scenarios of a U- Tube Steam Generator (UTSG) of a Nuclear Power Plant (NPP).

Keywords: Integrated Deterministic and Probabilistic Safety Analysis, scenarios post-processing, Semi-Supervised Self Organizing Map, Manhattan distance.

I. INTRODUCTION

Integrated Deterministic and Probabilistic Safety Analysis (IDPSA) can overcome the challenges of Deterministic Safety Analysis (DSA) and Probabilistic Safety Analysis (PSA) for complex dynamic systems, like nuclear and oil and gas plants[1], viz:

- Realistic uncertainty estimation;
- Realistic quantification of safety margins;
- Identification and characterization of undiscovered plant vulnerabilities;
- Increasing transparency and robustness of risk-informed decision making;
- Improvement of plant safety and operation.

Indeed, by means of a safety analysis one aims at estimating the unknown risk profile of a plant and at defining the safety margins for protecting it from known/unknown accidental events[2]. DSA attempts to do so by verifying the capability of the system (e.g. a Nuclear Power Plant (NPP)) to withstand to a limited set of postulated, conservative Design Basis Accidents (DBAs)[1,3]. PSA attempts to consider a wider set of accidental scenarios than DBAs for identifying the risk profile and quantifying the associated probabilities[2,4]. Both DSA and PSA are scenarios-based analyses with expert judgment scenario selection, which may lead to missing or underestimating potentially dangerous scenarios[5]. Indeed, it is recognized that DSA and PSA approaches provide information about what is known already to be an "issue", but they are not capable of revealing what is unknown, and to what extent[6].

IDPSA aims at improving the DSA and PSA analyses by coupling deterministic and probabilistic aspects of safety analyses, addressing both the stochastic nature of events (i.e., aleatory uncertainties), as well as the lack of knowledge about the processes relevant to the system (i.e., epistemic uncertainties), in a consistent manner[2,4]. In doing so, a key aspect of IDPSA is the attempt to reduce the reliance on expert judgment to avoid predetermining the analyzed scenarios in terms of time dependencies and sequences of events. In IDPSA, the scenarios emerge from the solution of the system stochastic dynamics: a comprehensive generation and analysis of the scenarios that may occur in the system due to the combination of components failures at all possible failures times is the main benefit of IDPSA. However, this also turns into being one of the major challenges of IDPSA, since the computational complexity of scenario generation and post-processing might become intractable

for very complex systems. Indeed, the number of dynamic scenarios to be taken into account increases with the number of failure events, rendering the computational cost enormous and the a posteriori information difficult to interpret[2,3,7].

As timing and sequencing of component failures determine the system final state (faulty or not), the complexity of the model increases from the binary variables (Boolean logic), where the modeling is limited only to the occurrence or not of certain events, to the Multi-Valued Logic (MVL) variables[2–4,8–10]. Then, also the post-processing (i.e., the classification of the generated dynamic scenarios in safe, failed, Near Misses (NM) and Prime Implicants (PIs) clusters) becomes challenging due to the large number of simulated scenarios. Safe scenarios are those that, even if including several components failures, keep the system working in safe conditions. Failed scenarios, instead, result from a combination of failure events that lead the system into a failed condition. Among failed scenarios, PIs are those scenarios containing events representing the minimal combinations of component failure necessary for system failure[11] (i.e., the equivalent of Minimal Cut Sets (MCSs) for dynamic systems). Among safe scenarios, NMs are dangerous sequences of events that lead the system to a quasi-faulty state.

Many methods have been proposed in literature for the classification task. A first screening would entail distinguishing failed scenarios from safe scenarios, for example by implementing a Fuzzy-c-Means (FCM) classifier[8], Mean-Shift Methodology (MSM)[12], or a decision tree[13]. Other methods have been proposed also for the identification of PIs and Near Misses, besides failed and safe scenarios. For example, the PIs identification among the whole set of scenarios can be performed with a differential evolution-based method[14] or a visual interactive method[5], where the number of components whose behavior is specified in the accident sequence, is selected as most important feature for the PIs identification: the accident sequences associated with the lowest literal cost are selected and stored as PIs, being, indeed, the most reduced sequences (i.e., with least number of events) that cannot be covered by any other implicant (i.e., PIs by definition). Regarding the identification of the Near Misses sequences an unsupervised clustering problem can be addressed and solved by an optimized wrapper algorithm around the K-means clustering algorithm[3,15].

With the abovementioned methods, it is possible to process the information and extract the characteristics of homogeneous groups (safe, failed, NMs and PIs), individually. However, there is no comprehensive method able to provide the analyst with a complete picture of the accidental scenarios characterization. The goal of this work is to propose such an IDPSA scenarios post-processing method that is: i) able to comprehensively deal with all the classes at the same time (safe, failed, NMs, PIs), ii) independent on the granularity of the MVL approximation iii) and whose interpretation of the provided classification results is simpler than in the other methods, thanks to an intuitive visual interface, which could be useful also for dynamic monitoring of the system. To this aim, we use Self-Organizing Maps (SOMs)[16] which have been widely used in various engineering and physical applications, including fault detection and diagnosis in complex systems[17,18]. SOMs capture non-linear relationships of high dimensional data and visualize them on a low dimensional interface, normally a 2-D structure of, so called, neurons. In this structure, data are assigned to the most similar neuron (usually by measuring the smallest Euclidean distance) so that the available data are divided into regions with common characteristics (i.e., data with high similarity are mapped close to each other). Three kinds of SOM exist: the Unsupervised SOM (USOM), the Semi-Supervised SOM (SSSOM) and the Supervised SOM (SSOM). We will show that SSSOM is the best for identifying safe, failed, NMs and PIs groups of scenarios. We tailor the SSSOM learning scheme to properly treat the MVL approximation as input data: more precisely, we resort to the Manhattan distance as similarity measure for assigning the discrete variables (i.e., the failure sequences) to the most similar neurons. The feasibility of using the Manhattan distance-based SSSOM method for post-processing IDPSA scenarios for their classification into safe, failed, NMs and PIs, is demonstrated with respect to a dynamic U-Tube Steam Generator (UTSG) of a NPP. For IDPSA scenarios generation, a dynamic simulation model has been implemented in SIMULINK and, as an approximation of the reality, a Multi-Valued Logic (MVL) scheme has been adopted for describing the different component operational states.

The paper is organized as follows. In Section 2, the UTSG and its SIMULINK model are presented. Section 3 characterizes the SOM both for unsupervised and semi-supervised algorithms. In Section 4, the results obtained on the UTSG scenarios post-processing are reported. In Section 5, some conclusions and final remarks are drawn.

II. CASE STUDY

II.A. The U-Tube Steam Generator (UTSG) model

We consider the UTSG represented in Fig. 1[19]. The reactor coolant enters the UTSG at the bottom, moves upward and then downward in the inverted U-tubes, transferring heat to the secondary fluid before exiting at the bottom. The secondary feedwater (Q_e) enters the UTSG at the top of the downcomer, through the space between the tube bundle wrapper and the SG shell.

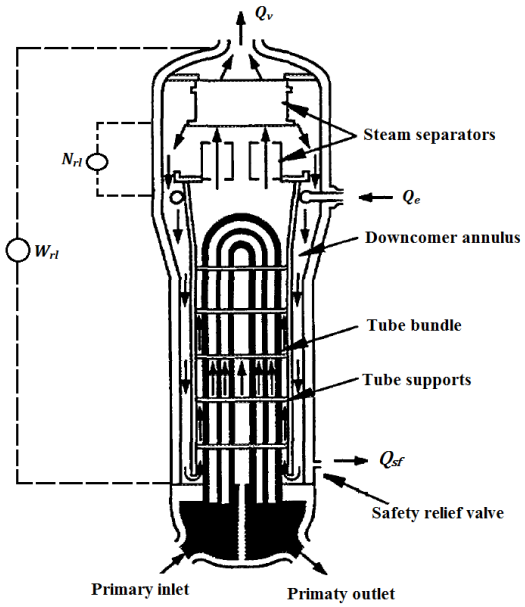


Fig. 1. Schematic of the UTSG[19].

The feed water Q_e heats up, reaches saturation, starts boiling, turns into a two-phase mixture and then the steam is separated from liquid water through the separator/riser section and the dryers, ensuring that the exiting steam (Q_v) is essentially dry. The separated water is re-circulated back to the downcomer. The balance between the exiting Q_v and the incoming Q_e governs the change in water levels: the narrow range level N_{rl}, due to the pressure difference between two points close to the water level and indicating the mixture level, and the wide range level W_{rl}, consisting in the difference between the two extremities of the SG and indicating the collapsed liquid level related with mass of water in the SG.

The goal of the control UTSG block (see Fig. 2) is to maintain the N_{rl} at a reference position (N_{ref}): if the N_{rl} goes above or below the thresholds N_{high} / N_{low}, a high/low level failure occurs.

Indeed, if $N_{rl} > N_{high}$ the steam separator and the dryer lose their functionally and excessive moisture is carried in Q_v, degrading the turbine blades profile and the turbine efficiency, whereas if $N_{ref} < N_{low}$ a sufficient cooling capability of the primary fluid cannot be guaranteed.

For IDPSA scenarios generation, a dynamic simulation model has been implemented in SIMULINK, to represent the UTSG behavior at different power level P_o [3]. A PID controller, that provides a flow rate Q_{pid} resulting from the residuals between N_{rl} and N_{ref} , and a feedforward controller, that is a safety relief valve to be opened if (and only if) $N_{rl} > N_{hl}$ (i.e., a pre-alarm level $N_{hl} < N_{high}$) to trigger a constant flow rate Q_{sf} removal from the UTSG, are used within the control block.

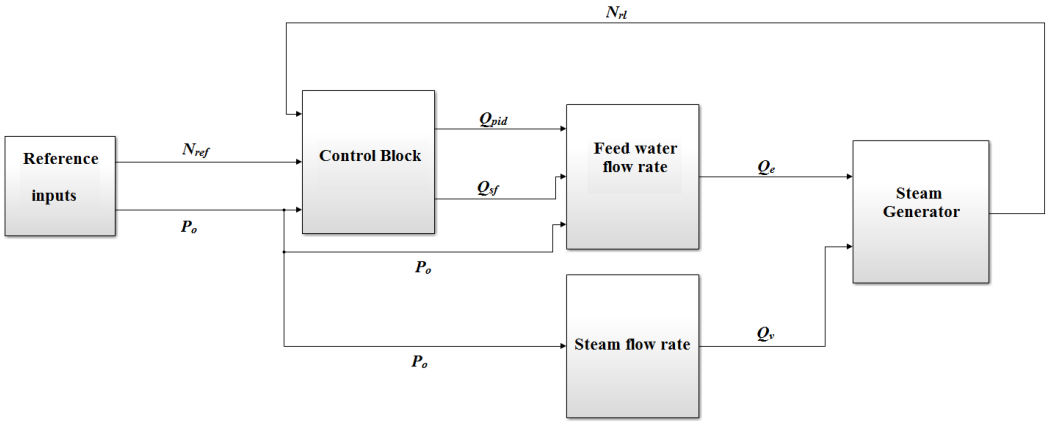


Fig. 2. SIMULINK model of the control block of the UTSG[3].

II.B. IDPSA scenarios generation

The set of multiple component failures that can occur in the UTSG are (Fig. 3): the steam valve failure, the safety relief valve failure, the interruption of the communication between the sensor that monitors N_{rl} and the PID controller, and the PID failure.

The failures assumptions have been arbitrarily made in order to i) favor multiple failures in the sequences, ii) capture the dynamic influence of all factors and iii) show a complex (but still manageable) real problem, whose IDPSA scenario generation requires post-processing the acquired information for guaranteeing a robust risk quantification of the UTSG operation. A mission time (T_{miss}) of 4000 (s) has been considered for allowing complete development also of slow dynamic accident scenarios occurring at early/medium time. The component failures are considered occurring at any continuous time instant, with any order in the sequence and magnitude[20].

For the tractability of the problem, we resort to a Multi Logic Value (MVL) computational framework in which the components can fail at discrete times and magnitudes[14]. The discretization consists in:

- Time: for each component the mission time (T_{miss}) is divided into four intervals, labelled $t = 1$ for failure in [0, 1000] (s), $t = 2$ in [1000, 2000] (s), $t = 3$ in [2000, 3000] and $t = 4$ in [3000,4000]. If $t = 0$ the component does not fail in T_{miss}.
- Component failure magnitudes:
  - the steam valve failure magnitude is indicated as 1, 2 or 3 for failure states corresponding to stuck at 0%, stuck at 50% and stuck at 150% of the Q_e value that should be provided at power level P_o, respectively; if the steam valve magnitude is indicated as 0, the component does not fail in T_{miss};
  - the safety relief valve failure magnitude is indicated as 1, 2, 3 and 4, if it is stuck between [0.5, 12.6] (kg/s), (12.6, 25.27] (kg/s), (25.27, 37.91] (kg/s) and (37.91, 50.5] (kg/s), respectively; if the safety relief valve magnitude is indicated as 0, the component does not fail in T_{miss};
  - the communication between the sensor measuring N_{rl} and the PID controller is labeled 0 if the communication works, 1 otherwise;
  - the PID controller failure magnitude is discretized into 8 equally spaced magnitude intervals, labeled from 1 to 8, representative of failure states corresponding to discrete intervals of output value belonging to [-18,18]% of the Q_e value that should be provided at P_o; if the PID controller magnitude is labeled as 0, the component does not fail in T_{miss}.
- the steam valve fails stuck at its maximum allowable value at a time in [1001, 2000] (s) and it is the first event occurring along the sequence;
- the safety relief valve fails third in [2001, 3000] (s), with a magnitude belonging to [0.5, 12.6] (kg/s);
- the communication between the sensor measuring N_{rl} and the PID controller is the second failure event in the sequence and occurs in [1001, 2000] (s);
- the PID controller fails stuck in [3001, 4000] (s), with a magnitude belonging to [6, 10] % of the Q_e value that should be provided at P_o.

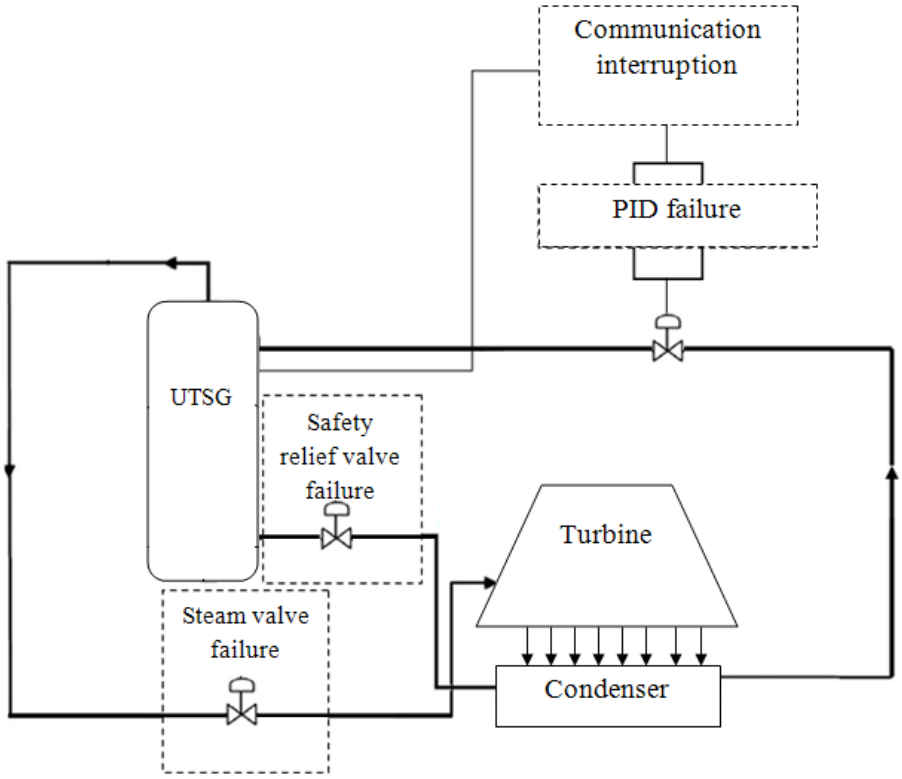


Fig. 3. Sketch of the failures that can be injected into the system.

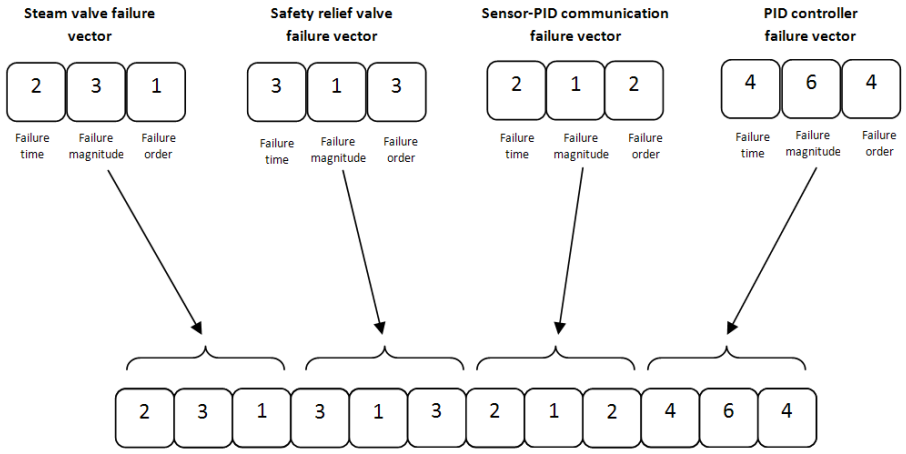


Fig. 4. Example of a sequence vector of an IDPSA scenario of the UTSG.

After scenarios generation, post-processing is needed to distinguish the safe, failed, NMs and PIs scenarios. For example, for Prime Implicants identification, a visual interactive method has been used[5], whereas for Near Misses identification a *k*-means clustering method has been adopted[3]. In the following, we propose the SOM as a comprehensive method able to provide the analyst with a complete picture of the accidental scenarios characterization, dealing with all the classes at the same time, independently on the granularity of the MVL approximation and allowing, for an intuitive interpretation of the results.

III. THE SELF-ORGANIZING MAP

III.A. Overview

Network architectures and signal processes that mimic the nervous system can be categorized in three groups: feedforward[21], feedback[22] and neighboring[16]. SOM belongs to this last category of neural networks where a map of neighboring cells compete in their activities with a mutual interaction in a competitive self-organizing manner[16]. The conceptualization of SOMs is inspired by the cerebral cortex, which is divided into different areas for processing different signals such as sight, hearing and tactile sensations. When signals are received, the cortex first classifies and, then, maps them into the corresponding cortex area, where neurons with similar functionality are closely related, allowing for a fast and accurate processing of the received signal[23]. This process is called topographic mapping and is one of the fundamental concepts underlying the SOM functioning[16]. Furthermore, the SOM is a powerful visualization tool for high-dimensional data, which are orderly mapped into a low-dimensional structure that usually consists of a 2-D regular grid of hexagonal nodes (neurons) that can vary from a few dozens up to several thousands[24]. The SOM concentrates all the information contained in a set of N input samples $\bar{X}$ belonging to a d-dimensional space, say $\bar{\bar{X}}=[\bar{X}_1, \bar{X}_2, \ldots \bar{X}_N]$, where the n-th sample is $\bar{X}_n = [x_1, x_2, \ldots, x_d]$, utilizing a set of M neurons, $C = [c_1, c_2, \ldots, c_M]$ (where $M < N$), each of which is associated to a weight vector $\bar{w}_m = [w_1, w_2, \ldots, w_d]$ (also called "prototypes" or "codebook" vectors)[25,26]. The weights of each neuron $\bar{w}_m$ are usually ramdonly initialized between 0 and 1; then, their values are adjusted during a training phase, so as to be able to optimally represent $\bar{\bar{X}}$ and its structure.

Essentially, three kind of SOM exist that differ in the manner they are trained (i.e., the way the topological structure is created): the unsupervised SOM (USOM), the semi supervised SOM (SSSOM), and the supervised SOM (SSOM).

III.B. The unsupervised SOM

The unsupervised SOM training mainly consists in three phases: competition, cooperation and adaptation[27], whose details are given in the following subsections. Briefly the training phase entails a stimulus (i.e., any of the input samples $\bar{X}_n$ from $\bar{\bar{X}}$) to be presented to the network and the neurons competition so as to indentify which is the Best Matching Unit (BMU), that is the most similar to $\bar{X}_n$ in terms of its weight values. Then, a subset of the neighborhood neurons to the BMU are modified by a neighborhood function. Figuratively, the region around the BMU is stretched towards the stimulus (shown in Fig. 5). As a result, the neurons on the grid become ordered: neighboring neurons tend to have similar weight vectors.

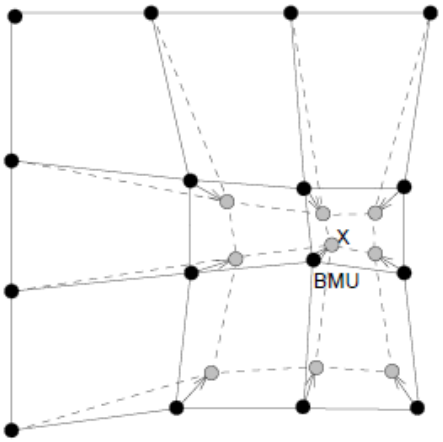


Fig. 5. Best matching unit (BMU) and its neighbors stretching towards the stimulus X. Solid and dashed lines correspond to the grid before and after the update, respectively.

III.B.1. Competitive process

The SOM is trained iteratively: for each training step t, one sample vector $\bar{X}_n$ is chosen randomly from the N available input data set $\bar{\bar{X}}$ and the distance between it and all the weight vectors of the SOM are calculated using some distance measure. The neuron c_i whose weight vector $\bar{w}_i$ is closest to the input vector $\bar{X}_n$ is called Best-Matching Unit (BMU):

$$c_i = arg\{min\{\|\bar{X}_n - \bar{w}_m\|\}\} \qquad with\ m = 1, \dots M \tag{1}$$

where c_i is the BMU and $\|\cdot\|$ is the distance measure (typically Euclidean, but also Binary[28] or, as original in this work, Manhattan).

III.B.2. Cooperation process

Once c_i is found, its weights vector $\bar{w}_i$ is updated proportionally to the difference between $\bar{w}_i$ and the values of $\bar{X}_n$, and accounting also for the characteristics of the neighborhing neurons of the BMU (that is, BMU and neighbors tightly cooperate to form a specific pattern on the lattice)[16,29].

III.B.3. Adaptation process

The tuning function that updates $\bar{w}_i$ is:

$$\bar{w}_{\,i}(t+1) = \bar{w}_{\,i}(t) + \alpha(t)\left(1 - \frac{d_{mi}}{d_{max}+1}\right)[\bar{X}_n(t) - \bar{w}_{\,i}(t)] \tag{2}$$

where α is the learning rate, d_{max} is the size of the neighborhood radius, that decreases during the training phase, and d_{mi} is the topological distance defined as the number of neurons that separates the considered m-th neuron and the winning neuron c_i. The learning rate changes during the training phase, as in Eq. (3)[25]:

$$\alpha(t) = (\alpha^{start} - \alpha^{final})\left(1 - \frac{t}{t_{tot}}\right) + \alpha^{final} \tag{3}$$

where t_{tot} is the total number of training epochs, α^{start} and α^{final} are the learning rate at the beginning and end of the training, usually in [0.1, 0.9] and in [0, α^{start}] respectively[25]. The training is usually performed in two phases: in the first phase, relatively large initial learning rate α^{start} and neighborhood radius d_{max} are used; in the second phase both learning rate and neighborhood radius are small right from the beginning. This procedure corresponds to first tuning the SOM approximately to the same space as the input data and, then, fine-tuning the map.

An additional input parameter to be set is the number of neurons M composing the map: this is usually set equal to:

$$M = 5 \cdot \sqrt{N} \tag{4}$$

Since different parameters and initializations give rise to different maps, it is important to know whether the map has properly adapted itself to the training data[30]. Two commonly used quality measures that can be used to determine the quality of the map and help choosing suitable learning parameters and map sizes are the Mean Quantization Error (MQE) and the Topographic Error (TE).

MQE is a measure of how good the map can fit the input data, and the best map is expected to yield the smallest average quantization error between the BMU $\bar{w}_i$ and the input vectors $\bar{X}_n$. MQE is calculated with Eq. (5):

$$MQE = \frac{1}{N}\sum_{n=1}^{N}\|\bar{X}_n - \bar{w}_i\| \tag{5}$$

where N is the number of the input vectors used to train the map. Practically, the lower the MQE, the better the map.

TE measures how well the topology is preserved by the map. Unlike the MQE, it considers the structure of the map. For each input vector, the distance of the BMU and the second BMU (the second weight vector closer to the input vector) on the map is considered; if the nodes are not neighbors, then, the topology is not preserved. TE is computed with Eq. (6):

$$TE = \frac{1}{N}\sum_{k=1} N \cdot u(\bar{X}_k) \quad (6)$$

In supervised training, the class of input vectors are known a priori and used as input itself. One of the most famous learning strategy is the Counterpropagation Neural Networks (CP-ANNs)[25], that is very similar to Kononen maps, but an output layer is added to the map, in order to handle supervised modeling. In this additional layer, neurons have as many weights as the number of classes to be modeled (G). As input for the algorithm also a class vector $\bar{Y} = [y_1, y_2, \dots, y_N]$ is needed (each n-th element of the class vector is associated with the corresponding input vector), which represents the supervised part of the network model. As a consequence, the map algorithm is fed with a set of data with $d + 1$ variables that also consider the class label. During the training, the weights of the m-th neuron $\bar{w}_m$ in the output layer are updated in a supervised manner, on the basis of the winning neuron selected in the Kohonen layer.

III.C. The semi-supervised SOM

Recently, modifications to CP-ANNs led to the introduction of new supervised neural network strategies as the XY-fused network (XY-Fs) and the Supervised Kohonen Networks (SKNs)[31]. The former is particularly interesting for classification models, since it exploits the similarities both in the Kohonen layer (X-map) and in the output layer (Y-map). We consider again a set of N input vectors $\bar{\bar{X}} = [\bar{X}_1, \bar{X}_2, \dots, \bar{X}_N]$, M neurons $C = [c_1, c_2, \dots, c_M]$ and weight vectors $\bar{w}_m = [w_1, w_2, \dots, w_d]$ associated to them. This time we add in input a vector $\bar{Y} = [y_1, y_2, \dots, y_N]$, called class vector and representing the G classes of each n-th input vector. The training is still based on the three phases of competition, cooperation and adaptation but in SSSOM the algorithm differs from the USOM one, since the fused similarity measure is based on a weighted combination distances between an object (vector) $\bar{X}_n$ and all units in the $Xmap$ ($S(\bar{X}, Xmap)$), and the distances between the corresponding output object y_n and the units in the $Ymap$ ($S(\bar{Y}, Ymap)$). By $S_{fused}(n, m)$ a common winning unit for both maps is determined:

$$S_{Fused}(n, m) = \varepsilon(t) S(\bar{X}_n, Xmap_m) + \big(1 - \varepsilon(t)\big) S(\bar{Y}_n, Ymap_m) \quad (7)$$

The location of the minimum of the above function is the common winning unit ε_i and $\varepsilon(t)$ regulates the relative weight between similarities in the X and Y maps, and it is still dependent on the number of training epochs, decreasing linearly during the training.

III.D. The semi-supervised SOM based on the Manhattan distance

The IDPSA scenarios to be post-processed consists of sequence vectors of MVL variables, as described in Section 2.2 (i.e., failure time, magnitude and order of components failures). For classifying the scenarios into the safe, failed, NMs and PIs classes with a SSSOM, the MVL formalism has to be accommodated within the similarity assessment between data and neurons. For this, we replace the d-dimensional Euclidean distance between the n-th input vector $\bar{X}_N$ and the weight vector $\bar{w}_m$, associated to the generic m-th neuron:

$$d_{Euclidean} = \sqrt{\sum_{k=1}^{d} (\bar{X}_k - \bar{w}_k)^2} \quad (8)$$

with the Manhattan distance:

$$d_{Manhattan}(\bar{X}_n, \bar{w}_{\,\mathrm{m}}) = \sum_{k=1}^{d} \|\bar{X}_k - \bar{w}_{\,\mathrm{k}}\| \quad (9)$$

where $\|\cdot\|$ is the absolute value of the difference between the two vectors in each dimension. By so doing, even the small differences between sequence vectors can be taken into account, that would be, otherwise, smoothed out with an Euclidean distance.

IV. POST-PROCESSING UTSG SCENARIOS

IV.A. Training of the SOM

We use the SSSOM described in Section 3.4 to analyze the $N = 100509$ scenarios of $d = 12$ variables and build the best 2-D representation of the $G = 4$ classes of safe, failed, NMs and PIs scenarios. We set the parameters of the map by trial and error:

- $M = 3025$ neurons;
- $t_{tot} = 15$ epochs;
- $\varepsilon(t = 0) = 0.01$ (see Eq. (7)).

This latter has been chosen to be small to initially favor the BMU assignment based on the class information rather than on the $d = 12$ MVL characteristics. In other words, the class G has an important role in the selection of the BMU. The obtained SSSOM is shown in Fig. 6. Different shades of color represent different classes: it is worth noticing the capability of the SOM to locate safe, failed, NMs and PIs scenarios in compact domains of neurons.

The confusion matrix of the classification is given in Table I: for each class, the numbers of correctly and wrongly assigned patterns are given. In general, most of the sequence vectors have been assigned to the right class. For example, the first row of the matrix describes how safe scenarios have been classified by the SSSOM: 50008 out of 64049 safe input vectors are correctly classified as safe, 5684 are classified as minterms, 5299 as NMs and 3056 as PIs. Looking at PIs, only a small fraction of scenarios are misclassified as safe (1), failed (2), and NMs (5) scenarios.

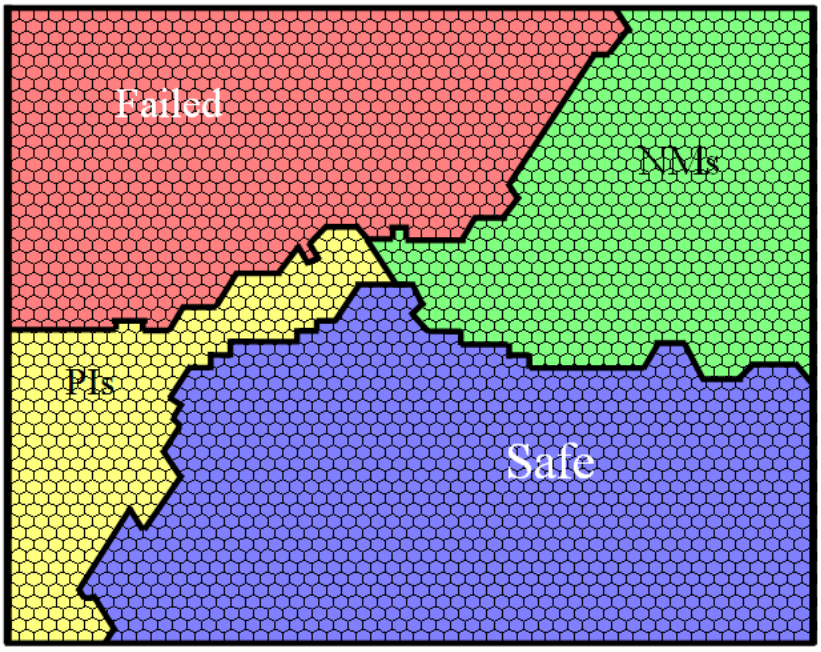


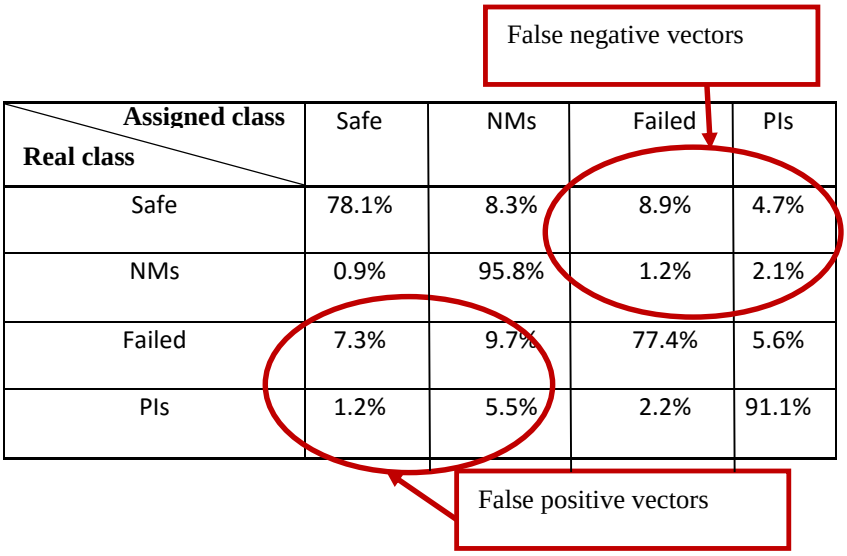
Fig. 6. The SSSOM for post-processing the UTSG accidental scenarios.

TABLE I. Confusion matrix of the SSSOM in Fig. 6.

Assigned class / Real class	Safe	NMs	Failed	PIs
Safe	50008	5299	5684	3056
NMs	3	318	4	7
Failed	2648	3521	27880	1989
PIs	1	5	2	82

Table II shows the confusion matrix in terms of percentages: the elements of the diagonal of the matrix are the percentages of scenarios correctly assigned. Overall 77.9% of the 100509 scenarios are correctly assigned. In Table II, we also highlight "false negative" and "false positive" assignments: the former are those vectors that are safe or NMs, but they are misclassified as failed or PIs, whereas the latter represent failed or PIs scenarios classified as safe or NMs. "False positive" are the scenarios we are concerned about because these have been misclassified. In particular, 17% (7.3% + 9.7%) of failed scenarios are classified as safe or NMs and a 6.7% (1.2% + 5.5%) of PIs are wrongly assigned to scenarios where the system does not fail.

TABLE II. Confusion matrix of the SSSOM in Fig. 6 (percentages).



The trained SSSOM performance is also evaluated with respect to (see Table III):

- Precision: the capability of the SSSOM to not include samples of other classes in the considered g-th class;
- Sensitivity: the capability of the SSSOM to correctly recognize samples belonging to the g-th class;
- Specificity: the capability of each g-th class of the SSSOM to reject the samples of all the others.

TABLE III. Performances of the SSSOM in Fig. 6.

Classes / Parameter	Safe	NMs	Failed	PIs
Precision	0.949	0.034	0.83	0.016
Sensibility	0.781	0.958	0.774	0.911
Specificity	0.927	0.911	0.911	0.949

As we can see, the SSSOM has high values of specificity for all the classes (i.e., it avoids misclassifying scenarios) even if precision is very small (at least for NMs and PIs).

The information provided by Tables I, II and III are useful for assessing the operational risk of the system under analysis. For this, we consider the product of the probability of occurrence of the scenario and the consequences that the developing scenario can cause to the system[3]. To compute the occurrence probability, we consider the component failures to occur with a small probability p. Obviously scenarios with more component failures are less probable to occur and we observe that safe scenarios have few component failures, whereas failed scenarios usually have a (relatively) large number of failures. In particular, considering the probabilities of occurrence of the "false negative" and "false positive" scenarios, we can see that: safe scenarios that are misclassified as failed have high probabilities of occurrence, because they are safe, but do not carry a large contribution to risk, even if not correctly identified, because of negligible consequences; failed scenarios, instead, may have large consequences, but occur with small probabilities.

IV.B. Test of the SSSOM

The Manhattan distance-based SSSOM is intended for post-processing the scenarios of an IDPSA of a dynamic system, classifying them in safe, failed, NMs and PIs under different MVL granularity, possibly also treating continuous variables.

We have created a new set of input data $\bar{\bar{X}}_{test}$ of 2000 sequences, in which components can fail randomly between 0 and the mission time of 4000 (s). Then, the trained SSSOM is used to classify $\bar{\bar{X}}_{test}$. Due to the discretization of the MVL framework, some of the safe sequences may be classified as failed and vice versa. Fig 7 shows, for example, the water level of a safe scenario (solid line) classified as failed (dashed line). From a continuous to a discretized representation, in fact, the failure time is shifted earlier and this is enough to change the end state of the system from safe to failed.

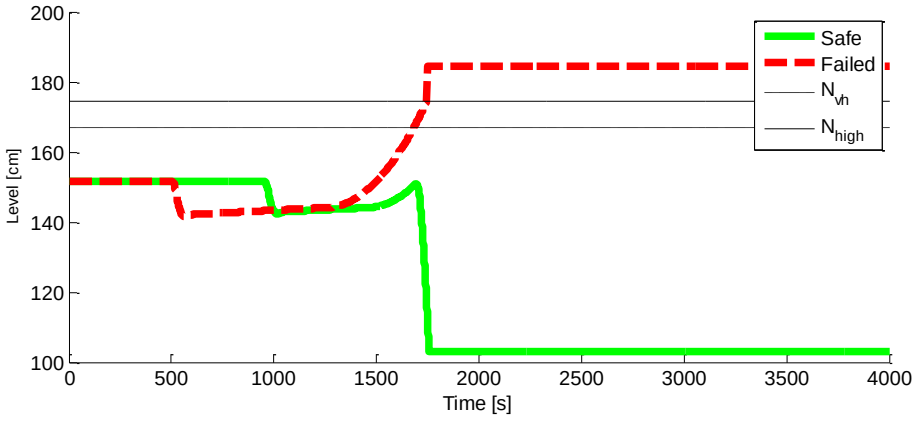


Fig. 7. UTSG water level for a safe scenario (solid line) that is classified as failed (dashed line).

Figure 8 shows a failed scenario (solid line) classified as safe (dashed line). Also in this case the failure time plays a fundamental role for the end state of the system: due to the MVL approximation, the failure time is delayed and the system does no longer exceeds N_{high}.

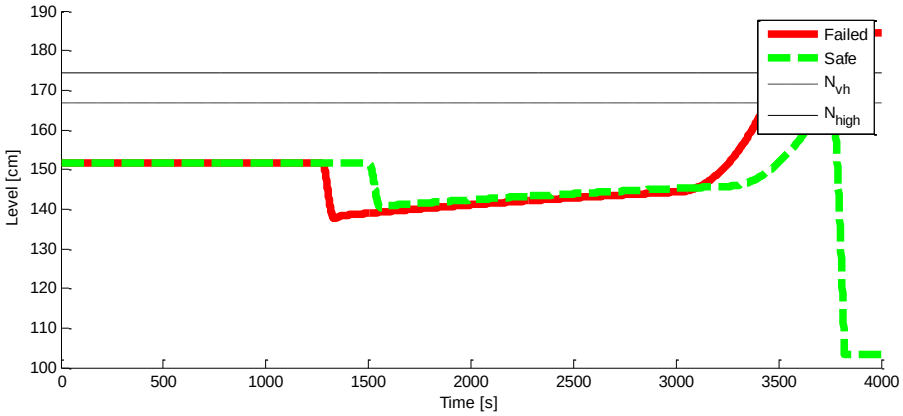


Fig. 8. UTSG water level for a failed scenario (solid line) that is classified as safe (dashed line).

On the other hand, in many cases the MVL approximation does not alter the scenarios characteristics because the time shift is not sufficient for a change of the end state of the scenario, as seen in Figures 9 and 10. Obviously, a finer MVL discretization would improve the SSSOM classification performances at the expenses of a longer computational burden for the MVL discretization, the increased number of scenarios to be post-processed and the complexity of the SSSOM to be built.

The results of the SSSOM test are shown in Table IV. Overall, it can be seen that 949 of the 2000 test scenarios considered (75.8%) are classified as safe or NMs and, among these, 941 out of 1244 are correctly classified as safe (75.64%), and all 8 NMs are correctly classified. However 542 of the 749 failed scenarios present in the test set are correctly classified as failed (563 i.e., 75.3% of failed scenarios) and PIs (9 i.e., 100% of PIs scenarios) classes.

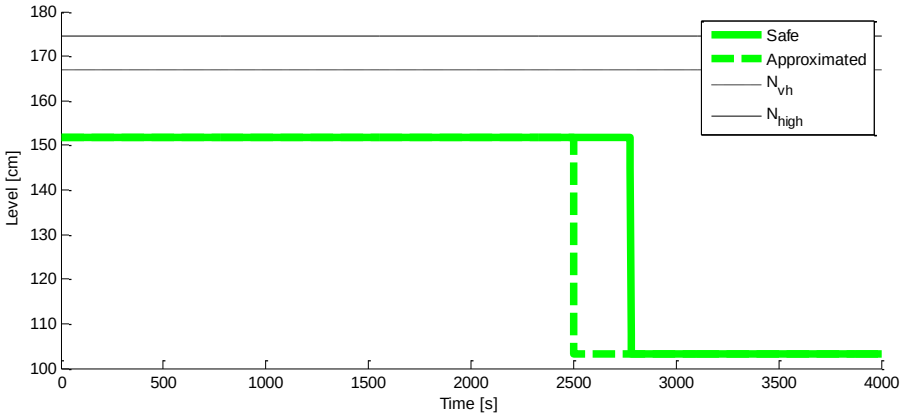


Fig. 9. UTSG water level for a safe scenario (solid line) that is classified as safe (dashed line).

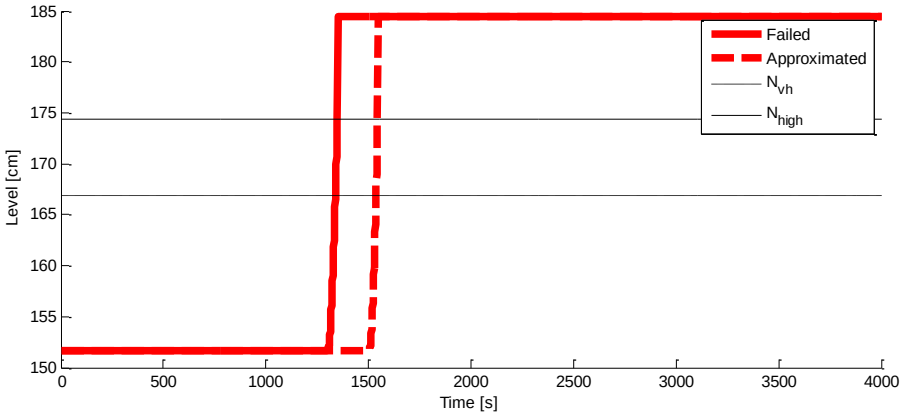


Fig. 10. UTSG water level for a failed scenario (solid line) that is classified as failed (dashed line).

	Number of correct assignments	%		Number of correct assignments	%
No fail	949	75.8%	**Safe**	941	75.6%
			NMs	8	100%
Fail	542	76.5%	**Failed**	563	75.3%
			PIs	9	100%

Table IV. Results of the testing phase.

V. CONCLUSIONS

The post-processing of scenarios collected during an IDPSA of a dynamic system can be difficult, due to the combinatorial explosion of the scenarios generated. In this paper, we have presented a new methodology for classifying IDPSA scenarios as safe, failed, NMs and PIs. The method is based on a Manhattan distance-based SSSOM, in order to treat the multi-valued dynamic scenarios. The application to an UTSG case study has shown the capability of grouping the scenarios in four distinct regions of the map and retrieving safety-relevant information.

REFERENCES

1. E. Zio, F. Di Maio, "The needs and dreams for methodologies of IDPSA", in *Proceedings of the Integrated Deterministic and Probabilistic Safety Analyses Workshop*, KTH, Stockholm, Sweden (2012).
2. Zio E. Integrated deterministic and probabilistic safety assessment: Concepts, challenges, research directions. *Nucl Eng Des*. 2014;280:413-419. doi:10.1016/j.nucengdes.2014.09.004.
3. F. Di Maio, M. Vagnoli, E. Zio, "Risk-Based Clustering for Near Misses Identification in Integrated Deterministic and Probabilistic Safety Analysis", *Sci Technol Nucl Install*., 1-29 (2015), doi:10.1155/2015/693891.

4. Aldemir T. A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants. *Ann Nucl Energy*. 2013;52:113-124. doi:10.1016/j.anucene.2012.08.001.
5. F. Di Maio, S. Baronchelli, E. Zio, "A visual interactive method for prime implicants identification", *IEEE Trans Reliab*., 64(2):539-549, (2015), doi:10.1109/TR.2014.2371015.
6. Vorobyev Y, Kudinov P. Development and application of a genetic algorithm based dynamic pra methodology to plant vulnerability search. In: *International Topical Meeting on Probabilistic Safety Assessment and Analysis 2011, PSA 2011*. Vol 1. ; 2011:559-573.
7. Labeau PE, Smidts C, Swaminathan S. Dynamic reliability: Towards an integrated platform for probabilistic risk assessment. *Reliab Eng Syst Saf*. 2000;68(3):219-254. doi:10.1016/S0951-8320(00)00017-X.
8. E. Zio, F. Di Maio, "Processing dynamic scenarios from a reliability analysis of a nuclear power plant digital instrumentation and control system", *Ann Nucl Energy*, 36(9):1386-1399 (2009), doi:10.1016/j.anucene.2009.06.012.
9. F. Di Maio, S. Baronchelli, E. Zio, "Hierarchical differential evolution for minimal cut sets identification: Application to nuclear safety systems", *Eur J Oper Res*., 238(2):645-652 (2014), doi:10.1016/j.ejor.2014.04.021.
10. F. Di Maio, S. Baronchelli, E. Zio, "A Computational Framework for Prime Implicants Identification in Noncoherent Dynamic Systems", *Risk Anal*., 35(1):142-156 (2015), doi:10.1111/risa.12251.
11. Quine W V. The Problem of Simplifying Truth Functions. *Am Math Mon*. 1952;59(8):521-531. doi:10.2307/2308219.
12. Mandelli D, Yilmaz A, Aldemir T, Metzroth K, Denning R. Scenario clustering and dynamic probabilistic risk assessment. *Reliab Eng Syst Saf*. 2013;115:146-160. doi:10.1016/j.ress.2013.02.013.
13. Galushin S, Kudinov P. An approach to grouping and classification of scenarios in integrated deterministic-probabilistic safety analysis. In: *PSAM 2014 - Probabilistic Safety Assessment and Management*. ; 2014. http://www.scopus.com/inward/record.url?eid=2-s2.0-84925067962&partnerID=tZOtx3y1.
14. F. Di Maio, S. Baronchelli, E. Zio, "A Computational Framework for Prime Implicants Identification in Noncoherent Dynamic Systems", *Risk Anal*., 35(1):142-156, (2015), doi:10.1111/risa.12251.
15. MacQueen J B., "Kmeans Some Methods for classification and Analysis of Multivariate Observations". *5th Berkeley Symp Math Stat Probab 1967*. 1967;1(233):281-297. doi:citeulike-article-id:6083430.
16. Kohonen T. The self-organizing map. *Proc IEEE*. 1990;78(9):1464-1480. doi:10.1109/5.58325.
17. Wu S, Chow TWS. Induction machine fault detection using SOM-based RBF neural networks. *Ind Electron IEEE Trans*. 2004;51(1):183-194. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1265797.
18. Yu H, Khan F, Garaniya V. Risk-based fault detection using Self-Organizing Map. *Reliab Eng Syst Saf*. 2015;139:82-96. doi:10.1016/j.ress.2015.02.011.
19. Aubry J.,F., Babykina G, Barros A, et al. *Project APPRODYN: APPROches de La Fiabilité DYNamique Pour Modéliser Des Systèmes Critiques*.; 2012.
20. Devooght J, Smidts C. Probabilistic reactor dynamics - I: Theory of continuous event trees. *Nucl Sci Eng*. 1992;111(3):229-240. http://www.scopus.com/inward/record.url?eid=2-s2.0-0026884986&partnerID=40&md5=cee7d51f4c102a1da1444041fa5f5fc8.
21. David E. Rumelhart, McClelland JL. *Parallel Distributing Processing; Exploration of the Microstructure of Cognition; Volume 1: Foundations*. Cambridge, MA, USA: MIT Press; 1986.
22. Siegelmann HT, Sontag ED. On the Computational Power of Neural Nets. *J Comput Syst Sci*. 1995;50(1):132-150. doi:10.1006/jcss.1995.1013.
23. Kandel ER, Schwartz JH, Jessell TM. *Principles of Neural Science*. Vol 3.; 2000. doi:10.1036/0838577016.
24. Kohonen T. The self-organizing map. *Neurocomputing*. 1998;21(1-3):1-6. doi:10.1016/S0925-2312(98)00030-7.
25. Ballabio D, Vasighi M. A MATLAB toolbox for Self Organizing Maps and supervised neural network learning strategies. *Chemom Intell Lab Syst*. 2012;118:24-32. doi:10.1016/j.chemolab.2012.07.005.
26. Astudillo CA, Oommen BJ. Self-organizing maps whose topologies can be learned with adaptive binary search trees using conditional rotations. *Pattern Recognit*. 2014;47(1):96-113. doi:10.1016/j.patcog.2013.04.012.
27. Zhong F. Z. F., Shi T.S.T., He T.H.T., "Fault diagnosis of motor bearing using self-organizing maps", *2005 Int Conf Electr Mach Syst*. 2005;3(50375047):2-5. doi:10.1109/ICEMS.2005.203004.
28. Appiah K, Hunter A, Meng H, et al. A binary Self-Organizing Map and its FPGA implementation. In: *Proceedings of the International Joint Conference on Neural Networks*. ; 2009:164-171. doi:10.1109/IJCNN.2009.5179001.
29. Vesanto J, Himberg J, Alhoniemi E, Parhankangas J. SOM Toolbox for Matlab 5. *Tech Rep A57*. 2000;2(0):59. doi:http://www.cis.hut.fi/somtoolbox/package/papers/techrep.pdf.
30. Gabrielsson S. The use of Self-Organizing Maps in Recommender Systems; A survey of the Recommender Systems field and a presentation of a State of the Art Highly Interactive Visual Movie Recommender System. 2006.
31. Melssen W, Wehrens R, Buydens L. Supervised Kohonen networks for classification problems. *Chemom Intell Lab Syst*. 2006;83(2):99-113. doi:10.1016/j.chemolab.2006.02.003.