Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 14, No. 1, 2026, pp. 337-347
DOI:10.22034/cmde.2025.65211.2984

A dynamic neuro-fuzzy approach for pattern classification

Erfan Veisi[1], **Bahram Sadeghi Bigham**[2,*], **and Mahdi Vasighi**[3]

[1]Continuous Improvement Department, Mirab Valves Company, Tehran, Iran.

[2]Department of Computer Science, Faculty of Mathematical Sciences, Alzahra University, Tehran, Iran.

[3]Department of Computer Science and Information Technology Institute for Advanced Studies in Basic Sciences, Zanjan, Iran.

Abstract

Nowadays, the application of neuro-fuzzy methods has been discovered more than ever for pattern recognition. These powerful tools are able to model the reality of data structure as it should be because, in the real world, datasets are defined in a fuzzy concept. In this research, we present a novel neuro-fuzzy method called Fuzzy Growing Map (FGM), combining the dynamic properties of the Growing Self-Organizing Map (GSOM) and fuzzy set theory. FGM is a dynamic neural fuzzy inference system based on if-then rules, which has the ability to generate fuzzy rules based on certain criteria during the learning phase. This approach can be used as a classifier and approximator. In addition, the trained FGM was used to visualize the fuzzy sets as a map, and the structure of the data can easily be revealed in the feature space.
To investigate the effectiveness of FGM, several benchmark datasets were analyzed, and the experimental results for classification show improvements in terms of accuracy and topographic error compared to classification algorithms Fuzzy Self-Organizing Map (FSOM)and Counter Propagation Neural Networks (CPNN).

Keywords. Pattern recognition, Growing self-Organizing map, Neuro-fuzzy methods, Fuzzy sets, Classification.
2010 Mathematics Subject Classification. 68T10, 03E72,62H30, 68T014.

1. Introduction

Neuro-fuzzy methods, as a vital subset of soft computing techniques, have consistently demonstrated their remarkable capabilities in addressing complex pattern classification problems. These challenges often arise in scenarios where overlapping classes do not have clear decision boundaries, making conventional approaches insufficient. By merging the adaptive learning and parallel processing capabilities of neural networks with the interpretive and approximate reasoning strengths of fuzzy systems, neuro-fuzzy methods surpass traditional non-fuzzy techniques in both flexibility and adaptability. Among these, neural fuzzy inference systems stand out as advanced frameworks, incorporating neural network characteristics such as learning algorithms and parallelism into fuzzy inference systems. This integration enhances their efficiency and robustness, paving the way for more sophisticated and precise decision-making [13].

A critical element in neuro-fuzzy systems is the Self-Organizing Map (SOM) algorithm [9], an unsupervised neural network model based on competitive learning. SOM plays a transformative role by visualizing complex datasets while preserving their inherent topological structure on a grid. This ability to reflect data relationships in an organized manner makes SOM particularly effective for integration into neural fuzzy inference systems. Within these systems, SOM is instrumental in determining the optimal positioning and number of fuzzy rules, a critical challenge in designing efficient fuzzy frameworks [6].

One of the pivotal advancements in this domain was introduced by Vuorimaa [19], who proposed the Fuzzy Self-Organizing Map (FSOM). This innovative method replaced traditional SOM neurons with fuzzy rules and organized

Received: 29 December 2024; Accepted: 02 March 2025.
∗ Corresponding author. Email: b_sadeghi_b@alzahra.ac.ir.

them using the SOM algorithm. This novel approach not only bridged the gap between neural and fuzzy paradigms but also laid the groundwork for applying neuro-fuzzy systems to a broader range of classification tasks.

Building upon such foundational work, recent studies have sought to refine and enhance neuro-fuzzy systems further, addressing emerging challenges in classification and prediction. For instance, the introduction of the Extended Self-Organizing Map with Ubiquitous Counter Propagation Network [16] showcased a significant improvement in handling noisy and imbalanced datasets, such as the Pima Indians Diabetes Dataset. By integrating fuzzy logic with counterpropagation techniques, this method achieved superior accuracy and resilience, setting a new standard in medical data classification. Similarly, a novel deep learning-based framework [7] demonstrated the potential of combining fuzzy logic with advanced neural architectures to classify sensory data, such as tastants, including sweet, bitter, and umami flavors. These approaches highlighted the versatility of neuro-fuzzy systems in tackling complex, multidimensional problems.

Further innovations have involved incorporating graph-based models into neuro-fuzzy systems. For example, modular-designed graph neural networks [3] have been employed for bitterness prediction, effectively capturing intricate data relationships and enhancing classification accuracy. In parallel, explainable machine learning frameworks [11] have emerged to address the dual objectives of interpretability and precision, providing reliable predictions for complex classification tasks like sweetener and bitterant identification. Additionally, a multi-objective framework [17] has been developed to predict multiple taste sensations, effectively balancing accuracy with interpretability in multi-class data scenarios.

In this paper, we build upon these advancements by introducing a novel dynamic neuro-fuzzy inference system, termed the Fuzzy Growing Map (FGM). This system leverages the Growing Self-Organizing Map (GSOM) and Learning Vector Quantization (LVQ) algorithms to establish a dynamic network structure that evolves. By utilizing the GSOM algorithm, FGM autonomously generates and self-organizes fuzzy rules, effectively addressing the limitations associated with static rule definitions. Experimental evaluations conducted on multiple datasets demonstrate that FGM outperforms existing approaches, including Fuzzy Self-Organizing Map (FSOM) [14] and Counter Propagation Neural Networks (CPNN) [10, 12], in terms of accuracy, Root-Mean-Square Error (RMSE), and topographic error.

Beyond its classification performance, FGM offers a distinctive advantage through its capability to visualize fuzzy sets via a membership function map. This unique feature provides an intuitive and interpretable representation of data, distinguishing FGM from other neuro-fuzzy methods and contributing to its broader applicability.

The structure of the paper is as follows: Section 2 details the structure and learning process of the proposed FGM. Section 3 presents the results of applying FGM to four datasets and compares the results with those of the FSOM and CPNN algorithms. Finally, section 4 provides the conclusion.

2. Fuzzy Growing Map

2.1. **Basic Structure.** The FGM is based on the GSOM, but the neurons of the original model are replaced by fuzzy rules and their fuzzy sets. The Growing Self-Organizing Map (GSOM), an extended version of Kohonen's SOM, is an unsupervised neural network that can dynamically grow and adapt its structure according to the data [1]. The structure of GSOM consists of a single layer of linear neurons. GSOM usually starts with four neurons arranged in a rectangular lattice (Figure 1(A)). The weight vectors of the starting neurons are randomly initialized, and the growth threshold (GT) is introduced as the maximum accumulated error a neuron can tolerate. For a given dataset, the GT can be obtained according to Eq. (2.1).

$$GT = -D \times \ln(SF), \tag{2.1}$$

where D refers to the dimensionality of the data vectors, and $SF$is the spreading factor, which is a growth control variable and needs to be chosen between 0 and 1, representing the minimum and maximum growth, respectively.

Then, in the next step, called the growing phase, which implements distance computation between input and weight vectors, the input signals are fed to all neurons. Then their distance (usually Euclidean distance) between the inputs and the connection weights of each neuron is computed. The output of the network is given by the most active neuron c (winner) according to Eq. (2.2).

$$\|x(t) - w_c(t)\| = \min_i \{\|x(i) - w_i(t)\|\}, \tag{2.2}$$

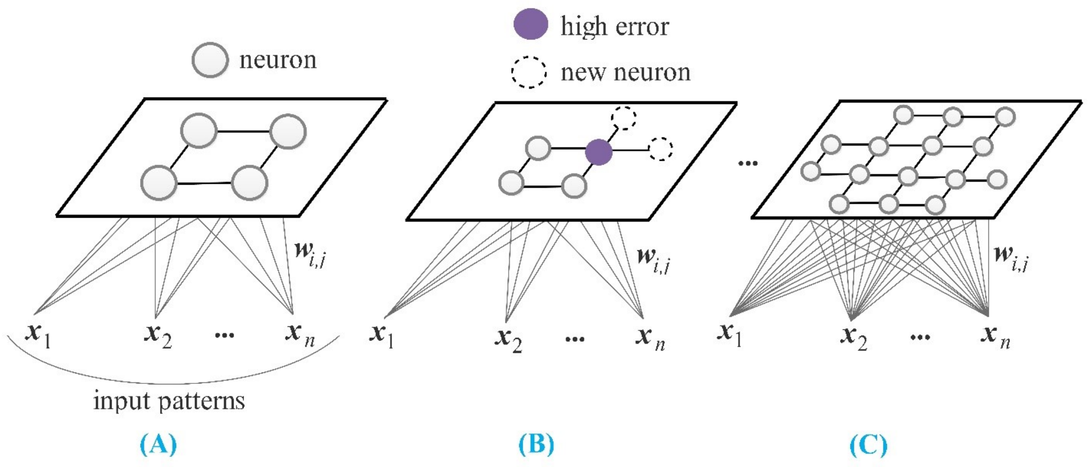

Figure 1. The structure of GSOM.

where x is an input vector, and w_i refers to the weight vector of neuron i. In the next step, known as the growing phase, the winning neuron c, along with other neurons belonging to its topological neighborhood, adjusts its weights towards the sample x according to Eq. (2.3).

$$w_i(t+1) = w_i(t) + \eta(t) \times h(t) \times [x(t) - w_i(t)]\,, \tag{2.3}$$

where η is the learning rate, and h is the neighborhood function, which could be a radial function such as a Gaussian. In the growing phase, the accumulative error (E) of the winner neuron is calculated and updated as follows:

$$E_c(t+1) = E_c(t) + \|x(t) - w_c(t)\|\ . \tag{2.4}$$

If a neuron has a high accumulative error, it is not a good representative of its Voronoi region in the data space. Therefore, new neurons can be added to distribute the accumulated error and better represent the data space. When the accumulative error of a neuron exceeds the Growth Threshold (GT), new neurons are added to available free positions around the winner neuron in the lattice (Figure 1(B)).

The weight vector of the new neurons is assigned through extrapolation based on the neighboring neurons to preserve the smoothness of the grid in the data space. If the winning neuron has no free neighboring position, half of E_c its cumulative error is equally distributed among the neighboring neurons, as defined by Equations (2.5) and (2.6):

$$E_c(t+1) = \frac{E_c(t)}{2}, \tag{2.5}$$

$$E_n(t+1) = E_n(t) + \frac{E_c(t)}{2N}, \tag{2.6}$$

where E_n represents the neighbors' error, and N is the number of adjacent neighbors of the winner neuron. One epoch is completed when all input vectors have been presented. The growing phase is then repeated for a specified number of learning epochs until the map reaches an appropriate size (Figure 1(C)).

In the final phase, known as the smoothing phase, no new neurons are added. The GSOM instead follows Kohonen's SOM learning rule, with a lower adaptation rate and a fixed, small neighborhood function width. The weight vectors of the winner and its neighbors are updated in the same manner as in the growing phase. There is also a batch version of GSOM, where all input data are presented to the network as one batch. The weight vectors are updated by the net effect of all the training samples in a single concurrent procedure, which accelerates the learning process of the network.

For a training dataset, the batch learning algorithm for SOMs can be performed by presenting an input vector x_j to all the neurons simultaneously to find the winner neuron whose weight vector $\mathbf{w}_c$ has the minimum distance to x_j.

The input vector x_j is then added to the Voronoi set of the winner neuron c. A set of samples with the same winner neuron constitutes the Voronoi set of that neuron [9].

After repeating these steps for all input vectors, the weight vectors of the neurons are updated with the average of the Voronoi sets, weighted by the neighborhood function, as described by:

$$w_i^{new} = \frac{\sum_{j=1}^{k} h_{c_j,i} x_j}{\sum_{j=1}^{k} h_{c_j,i}}, \tag{2.7}$$

where $h_{c_j,i}$ is a Gaussian neighborhood function described as follows:

$$h_{c_j,i} = \exp\left(-\frac{\|w_i - w_{cj}\|^2}{2\sigma^2(t)} \right), \tag{2.8}$$

where $\mathbf{w}_i$ is the weight vector of the i-th neuron, and w_{cj} is the weight vector of the winning neuron c for i-th input vector. $\|w_i - w_{cj}\|$ represents the distance between these two prototypes on the grid, and σ is the width of the Gaussian function, which controls the cooperation of neighboring neurons in the learning process. The value of $\sigma(t)$ decreases with time. This procedure can be repeated a number of times specified by the user.

In the FGM, each neuron of the GSOM is defined as a fuzzy rule as follows:

- if x_1 is $U_{i,1}$ and x_2 is $U_{i,2}$ and x_n is $U_{i,n}$,
- then y_1 is $a_{i,1}$ and y_2 is $a_{i,2}$ and y_p is $U_{i,p}$.

where each condition (x_j *is* $U_{i,j}$) is interpreted as the membership value $\mu_{U_{i,j}}(x_j)$ of the input signal x_j in the fuzzy set $U_{i,j}$ The consequence $a_{i,j}$ of the fuzzy rule is a real-valued singleton.

The membership functions used in FGM are triangular. The triangular shape has the advantage of being computationally efficient. The triangular membership function is defined as:

$$\begin{cases} \mu_{U_{i,j}}(x_j) = \frac{x_j - sl_{i,j}}{c_{i,j} - sl_{i,j}}, & sl_{i,j} \le x_j \le c_{i,j}, \\ \mu_{U_{i,j}}(x_j) = \frac{x_j - sr_{i,j}}{c_{i,j} - sr_{i,j}}, & c_{i,j} \le x_j \le sr_{i,j}, \\ \mu_{U_{i,j}}(x_j) = 0, & Otherwise. \end{cases} \tag{2.9}$$

The firing strength α_i of each fuzzy rule is computed by combining the membership values $\mu_{U_{i,j}}$ using the union connective operation. In the FGM, the minimum operation is used as the connective. Therefore, the firing strengths are computed as follows:

$$\alpha_i = \min\left\{ \mu_{U_{i,1}}(x_1), \mu_{U_{i,2}}(x_2), ..., \mu_{U_{i,n}}(x_n) \right\}. \tag{2.10}$$

The consequences of the fuzzy rules are singletons, and the output of the neuro-fuzzy system is calculated using a weighted average (WA) defuzzification method, defined as follows:

$$y^* = \frac{\sum_{i=0}^{m} \alpha_i a_{i,k}}{\sum_{i=0}^{m} \alpha_i}, \tag{2.11}$$

where m is the number of fuzzy rules. The values of the singletons $a_{i,k}$ are specified separately for each output and fuzzy rule.

The basic structure of the FGM for three rules is illustrated in Figure 2.

2.2. **Learning Schema.**

The learning schema of the FGM, which involves tuning the membership functions, is carried out using a supervised algorithm named modified LVQ2.1 [19]. The system is trained on a dataset containing input-output data samples.

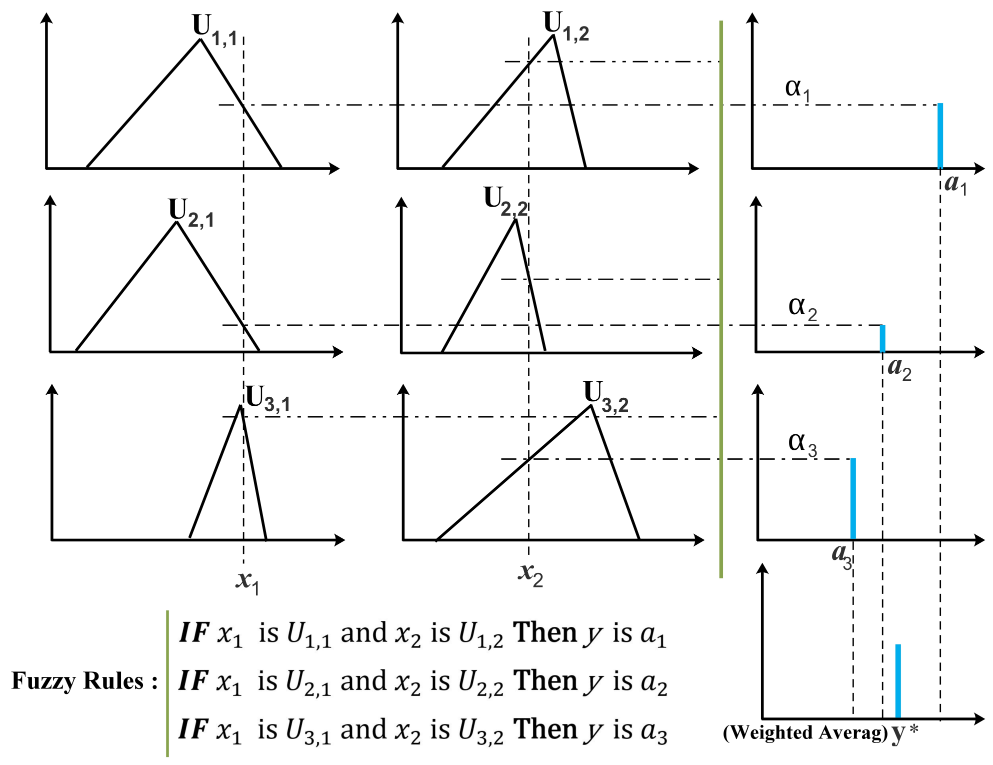
$U_{1,1}$
$U_{1,2}$
α_1
a_1
$U_{2,1}$
$U_{2,2}$
α_2
a_2
$U_{3,1}$
$U_{3,2}$
α_3
a_3
x_1
x_2
Fuzzy Rules :
IF x_1 is $U_{1,1}$ and x_2 is $U_{1,2}$ Then y is a_1
IF x_1 is $U_{2,1}$ and x_2 is $U_{2,2}$ Then y is a_2
IF x_1 is $U_{3,1}$ and x_2 is $U_{3,2}$ Then y is a_3
(Weighted Averag) y^*

FIGURE 2. Structure of FGM for three rules.

The antecedent membership functions in the FGM are defined by three parameters, while the consequent singleton is described by one parameter. Three updating laws are employed: two updating laws modify the centers and spreads of the antecedent fuzzy sets, and the third updating law adjusts the values of the output singletons.

Updating spreads. The modified LVQ2.1 algorithm relies on the concept of the window. The window is defined as the overlapping area of the two most active fuzzy rules. The winning rule w has the highest firing strength, and the rule r is the first runner-up. When at least two fuzzy rules fire simultaneously, one of the spreads of a fuzzy set is updated. Specifically, when the input sample falls within the window, one spread of the first runner-up rule is chosen to be updated. The spread $s_{r,k}$ is determined by computing the distances between the winner and the first runner-up rule along each input dimension and by choosing the input dimension and its spread where the distance between the winner and the first runner-up is the largest, as follows:

$$|c_{w,k} - c_{r,k}| = \max_{j} \{|c_{w,k} - c_{r,k}|\}, \tag{2.12}$$

where $c_{w,j}$ and $c_{r,j}$ for $j = 1, 2, ..., n$ are the centers of the winner rule and the first runner-up, respectively. The actual updating is performed by moving the spread $s_{r,k}$ ($sl_{r,k}$ or $sr_{r,k}$) either towards the center $c_{r,k}$ or the spread $s_{w,k}$ of the winner rule w. This adjustment either increases or decreases the influence of the first runner-up rule on the output of the FGM. The updating is performed according to:

$$\begin{cases} s_{r,k}(t+1) = s_{r,k}(t) + g_{U,r}(t)\left[c_{w,k}(t) - s_{r,k}(t)\right], & \operatorname{sgn}(y - y*) = \operatorname{sgn}(a_r - a_w), \\ s_{r,k}(t+1) = s_{r,k}(t) + g_{U,r}(t)\left[s_{w,k}(t) - s_{r,k}(t)\right], & Otherwise, \end{cases} \tag{2.13}$$

where $c_{w,k}$ and $c_{r,k}$ are the centers, $s_{w,k}$ and $s_{r,k}$ are the spreads of the fuzzy rules, and g_U is the learning rate of the fuzzy sets. The variable y represents the desired output of the training dataset, y^* is the output of FGM, a_w is the output of the winner rule, and a_r is the output of the first runner-up rule. sgn(t) is the signum operator.

2.3. Automatic generation of fuzzy rules.

Before the fuzzy sets can be updated, the fuzzy rules must exist. For this purpose, the centers of the fuzzy sets are initialized and self-organized using the GSOM algorithm. Fuzzy sets are formed around the centers c_i, with the left and right sr_i spreads and a constant width w_0, and then the fuzzy sets are tuned by the modified LVQ2.1.

When the original scheme is used, the structure of the rule base cannot change during the learning procedure. This usually restricts the approximation accuracy or can lead to the use of many unnecessary fuzzy rules. Thus, the self-generation (self-construction) of fuzzy rules is a convenient property. It enables the creation of new fuzzy rules only when needed.

The self-generating procedure in our proposed method is performed using the GSOM algorithm. This means that the system starts with four fuzzy rules, and neurons or fuzzy rules are added to the system by GSOM laws as needed. The algorithm of FGM is as follows:

Algorithm 1 Fuzzy growing map (FGM).

```
Initialize the weight vectors of four neurons with random weights,
Calculate the growth threshold (GT) and specify the number of training epochs,
repeat
    Feeding all the input samples to the neurons,
    Calculate distances between input data and all weight vectors,
    Determine the winners,
    Update all the weight vectors,
    Forming the centers of fuzzy sets around the neurons,
    Feeding all the input samples to the fuzzy sets,
    repeat
        Tuning of fuzzy sets by LVQ2.1 algorithm,
    until End of epoch number,
    Calculate TE (total error) for all centers,
    if $GT \leq TE_i$ then
        if The winner node $i$ is a boundary node, then
            Grow nodes,
            Initialize the new node weight vector to match the neighbouring node weights,
        else
            distribute weights to neighbors,
        end if
    end if
until End of epoch number.
```

3. Experimental Results

To evaluate the efficiency of the FGM method, several benchmark datasets with varying dimensions were utilized. The number of classes and dimensions for each dataset is detailed in Table 1.

The Iris dataset comprises 150 samples distributed across three classes, each representing a distinct subspecies of Iris: setosa, virginica, and versicolor, with 50 samples in each class.

The sweet-tasteless and sweet-bitter datasets contain 566 samples (433 sweet and 133 tasteless molecules) and 508 samples (427 sweet and 81 bitter molecules), respectively.

Lastly, the Sediment dataset includes 1,413 samples with 16 features categorized into two classes, containing 1,218 and 195 samples in the first and second classes, respectively. Sediments play a critical role in marine ecosystems, serving

Table 1. Datasets description.

Dataset	Reference	Number of Samples	Number of Features	Number of Classes
Iris	[4]	150	4	3
Sweet bitter	[15]	508	4	2
Sweet tasteless	[15]	566	9	2
Sediment	[2]	1413	16	2

Table 2. The values of the FGM parameters.

Datasets	w_0	Max iteration of LVQ2.1	SF
Iris	0/5	12	0/8
Sweet bitter	1/5	12	0/6
Sweet tasteless	0/6	8	0/4
Sediment	0/5	10	0/4

as reservoirs for various chemicals that accumulate over time and potentially pose risks to both the environment and human health.

In the classification of these datasets, the proposed method generally demonstrates superior performance in terms of average accuracy and topographic error (TE) compared to the FSOM and CPNN methods.

Classification accuracy is defined as the average percentage of correctly predicted samples out of the total samples in the test data. In other words, it represents the proportion of accurate predictions to the total number of predictions.

Topographic error (TE) measures how well the structure of the input space is preserved by the map. TE (Eq. (3.1)) is calculated by identifying the best-matching (B) and second-best-matching (S) neurons for each input and evaluating their positions on the map. If these nodes are adjacent, the topology is considered preserved for the input. Otherwise, it is counted as an error. The TE of the map is obtained by dividing the total number of errors by the total number of data points [6].

$$TE(M) = \frac{1}{n}\sum_{i=1}^{n} t(x_i), \qquad t(x) = \begin{cases} 0, & if B\, and\, S\;\; areneighbors \\ 1, & Otherwise. \end{cases} \tag{3.1}$$

To evaluate our model, the FGM learning process was repeated 10 times for all datasets, and the mean accuracy values from 5-fold cross-validation [8] were reported. In each fold of cross-validation, the dataset was randomly split into a training set (80% of the samples) and a test set (20% of the samples). The training set was used in the learning process, while the test set was reserved for performance evaluation. The number of training epochs was fixed at 100 for all methods.

The classification results are summarized in Table 3, and the parameter settings for the FGM method are detailed in Table 2. This evaluation strategy ensures a robust assessment of the model's performance, as it minimizes the influence of random splits and provides a more reliable estimate of its generalization capability across different datasets.

As shown in the Table 3, the classification performance of the methods on the benchmark datasets demonstrates that the FGM, with a nearly equal number of neurons, significantly outperforms the others in terms of classification accuracy. Furthermore, a comparison of the topographic error across methods clearly indicates that the proposed FGM achieves superior performance in all cases.

3.1. **Visualization of FGM.**

One of the key advantages of the proposed method, compared to other similar neuro-fuzzy approaches, is its ability to visualize and represent fuzzy membership functions in a two-dimensional space. This enables a clearer understanding of the data structure within the feature space.

To illustrate this, we apply the FGM method to the Iris dataset in this study. Figure 3 displays the membership functions of the fuzzy sets for the Iris dataset in a 2D feature space with a spread factor (SF) of 0.2. The membership functions are represented as polygons, and since the Iris dataset is four-dimensional, an octagon is used for visualization,

Table 3. Comparison of the proposed method with two other methods based on topographic error and average accuracy.

Dataset	Method	Accuracy (%)	Topographic Error	Number of Neurons
Iris	CPNN	94.33	0.13	$(13 \times 14)182$
	FSOM	94.32	0.13	$(13 \times 14)182$
	FGM	**96.1**	**0.08**	≈ 179
Sweet Bitter	CPNN	90.1	0.23	$(22 \times 22)484$
	FSOM	89.31	0.22	$(22 \times 22)484$
	FGM	**94.85**	**0.14**	≈ 480
Sweet Tasteless	CPNN	82.3	0.14	$(14 \times 14)196$
	FSOM	77.08	0.13	$(14 \times 14)196$
	FGM	**92.56**	**0.06**	≈ 191
Sediment	CPNN	89.95	0.12	$(21 \times 21)441$
	FSOM	88.47	0.11	$(21 \times 21)441$
	FGM	**93**	**0.10**	≈ 445

where each diameter corresponds to one feature. The length of each diameter reflects the width of the membership function in that dimension, with a larger diameter indicating greater importance of that dimension.

This visualization in Figure 3 aligns with the class label positions shown in Figure 4, which further enhances our understanding of the dataset's structure.

It is worth noting that at the start of the learning process, the fuzzy membership functions are symmetric within the octagon due to identical widths. However, as the learning progresses, this symmetry may be disrupted because the left and right spreads are updated independently. This is evident in Figure 3, where slight deviations from symmetry become noticeable. This deviation highlights the adaptive capability of the proposed method, illustrating how the model dynamically adjusts the membership functions to more accurately reflect the data distribution and enhance classification performance over time.

4. Conclusion

This paper presents a novel dynamic algorithm for pattern classification and data visualization, combining the Generalized Self-Organizing Map (GSOM) with fuzzy systems. The proposed approach integrates the GSOM algorithm into the design of a dynamic neuro-fuzzy model, aiming to enhance classification accuracy and reduce topographic error across several benchmark datasets. The GSOM algorithm is utilized to automate the generation of fuzzy rules and adaptively grow the system. By strategically determining the placement of fuzzy rules, GSOM ensures they are generated in optimal locations while effectively controlling their number, thereby avoiding the creation of redundant or irrelevant rules.

Experimental evaluations on benchmark datasets demonstrate that the proposed method outperforms existing algorithms, including FSOM and CPNN, in terms of classification accuracy and topographic error. Furthermore, the proposed approach offers the unique advantage of visualizing fuzzy membership functions as a two-dimensional map, providing deeper insights into the data structure and enhancing interpretability. These results highlight the effectiveness and versatility of the proposed method in addressing challenges in dynamic neuro-fuzzy modeling and pattern classification. The method's ability to both improve accuracy and provide intuitive visualizations suggests its potential for wide-ranging applications in fields that require complex, high-dimensional data analysis. The method's ability to both improve accuracy and provide intuitive visualizations suggests its potential for wide-ranging applications in fields that require complex, high-dimensional data analysis.

5. Limitations and Future Work

While the proposed Fuzzy Growing Map (FGM) demonstrates significant improvements in classification accuracy and interpretability, it is not without limitations. One potential challenge is the computational complexity associated

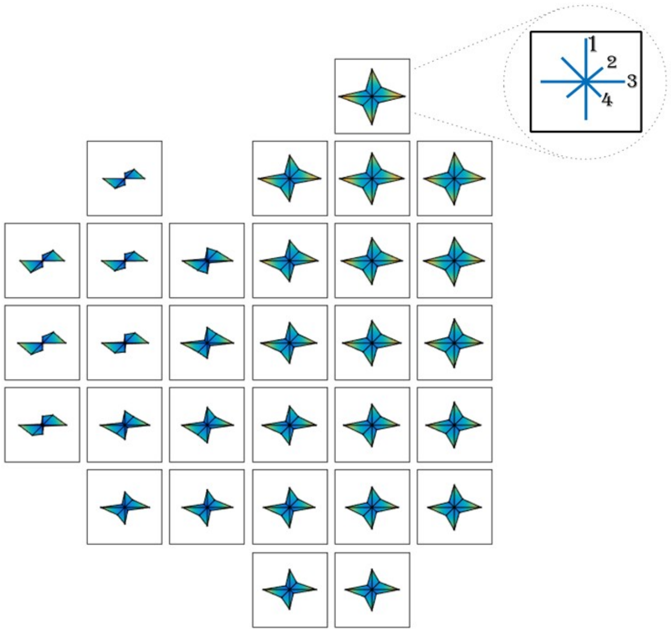

Figure 3. Representation of the membership functions of the Iris dataset in 2D space with $SF = 0.2$.

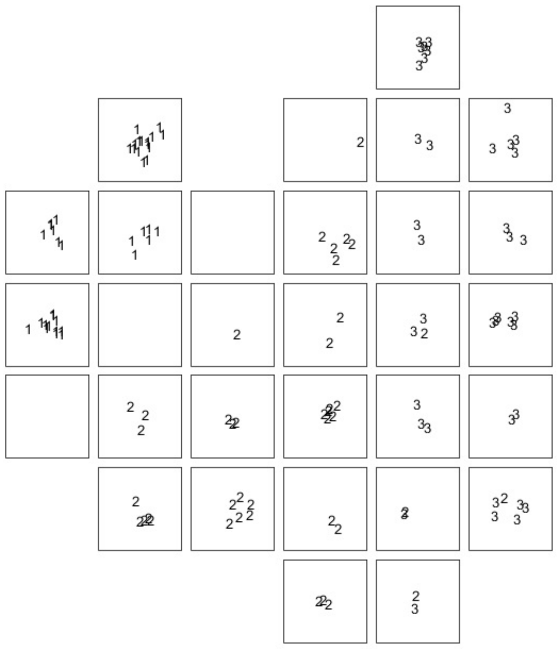

Figure 4. The predicted labels for the Iris dataset with a factor of 0.2 in 2D space.

with the dynamic growth and self-organization processes, especially when applied to large-scale datasets. The growth of the map and the continuous adaptation of fuzzy rules can be resource-intensive, which may limit their practical use for real-time applications or systems with resource constraints. Moreover, the performance of FGM can be sensitive to the choice of initial parameters, such as the learning rate and the growth threshold, which may require fine-tuning for different datasets or domains. These factors might affect the consistency and reliability of the system across various applications.

Future work could focus on optimizing the algorithm to reduce computational overhead and improve its scalability. Exploring hybrid approaches that combine FGM with other advanced machine learning techniques, such as deep learning or reinforcement learning, may further enhance its performance, especially in environments characterized by complex and high-dimensional data. Additionally, integrating FGM with transfer learning could help overcome issues related to data scarcity in specific domains.

Despite these challenges, FGM holds significant potential for various applications that remain relatively underexplored. Three promising fields for FGM application include:

- **Medical Diagnosis Systems:** While FGM has shown effectiveness in classifying medical datasets, its potential for real-time diagnostics in healthcare is still underexplored. The ability of FGM to visualize fuzzy sets could aid in explaining complex medical conditions, making it a valuable tool for clinical decision support systems.
- **Environmental Monitoring:** FGM could be applied to environmental data classification, such as predicting pollution levels or identifying climate patterns. The self-organizing aspect of FGM allows it to dynamically adapt to changing environmental data, making it suitable for continuous monitoring systems.
- **Smart Grid Optimization:** FGM can be applied in the optimization of energy usage in smart grids. By classifying and analyzing energy consumption data, FGM could help in predicting demand patterns and suggesting optimal energy distribution strategies, contributing to more efficient energy management systems.

By addressing the computational challenges and expanding its applications into these new domains, FGM could further solidify its position as a versatile and powerful tool in the machine learning and artificial intelligence landscape.

References

[1] D. Alahakoon, S. K. Halgamuge, and B. Srinivasan, *Dynamic self-organizing maps with controlled growth for knowledge discovery*, IEEE Transactions on neural networks, *11*(3) (2000), 601-614.

[2] M. Alvarez Guerra, D. Ballabio, J. M. Amigo, J. R. Viguri, and R. Bro, *A chemometric approach to the environmental problem of predicting toxicity in contaminated sediments*, Journal of Chemometrics, *24*(7-8) (2010), 379–386.

[3] L. Androutsos, L. Pallante, A. Bompotas, F. Stojceski, G. Grasso, D. Piga, G. D. Benedetto, C. Alexakos, A. Kalogeras, K. Theofilatos, M. A. Deriu, and S. Mavroudi, *Predicting multiple taste sensations with a multiobjective machine learning method*, NPJ Science of Food, *8*(1) (2024), 47.

[4] C. Armanino, R. Leardi, S. Lanteri, and G. Modi, *Chemometric analysis of Tuscan olive oils*, Chemometrics and Intelligent Laboratory Systems, *5*(4) (1989), 343-354.

[5] T. Ayadi, T. M. Hamdani, and A. M. Alimi, *A new data topology matching technique with multilevel interior growing self-organizing maps*, 2010 IEEE International Conference on Systems, Man and Cybernetics, IEEE, (2010).

[6] G. T. Breard, *Evaluating self-organizing map quality measures as convergence criteria*, Open Access Master's Theses, (2017), Paper 1033.

[7] P. Dutta, D. Jain, R. Gupta, and B. Rai, *Classification of tastants: A deep learning based approach*, Molecular Informatics, *42*(12) (2023), e202300146.

[8] C. George and L. B. Roger, *Statistical Inference, volume 2 Duxbury*, Pacific Grove, CA, (2002).

[9] S. G. Kong and B. Kosko, *Adaptive fuzzy systems for backing up a truck-and-trailer*, IEEE Transactions on Neural Networks, *3*(2) (1992), 211-223.

[10] I. Kuzmanovski and M. Novič, *Counter-propagation neural networks in Matlab*, Chemometrics and Intelligent Laboratory Systems, *90*(1) (2008), 84-91.

[11] G. Maroni, L. Pallante, G. D. Benedetto, M. A. Deriu, D. Piga, and G. Grasso, *Informed classification of sweeteners/bitterants compounds via explainable machine learning*, Current Research in Food Science, *5* (2022), 2270-2280.
[12] W. Melssen, R. Wehrens, and L. Buydens, *Supervised Kohonen networks for classification problems*, Chemometrics and Intelligent Laboratory Systems, *83*(2) (2006), 99-113.
[13] T. Ojala, *Neuro-fuzzy systems in control*, Master of Science Thesis, Department of Electrical Engineering, Tampere University of Technology (1994).
[14] P. Vuorimaa, *Use of the fuzzy self-organizing map in pattern recognition*, Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, IEEE, (1994).
[15] C. Rojas, D. Ballabio, V. Consonni, P. Tripaldi, A. Mauri, and R. Todeschini, *Quantitative structure–activity relationships to predict sweet and non-sweet tastes*, Theoretical Chemistry Accounts, *135* (2016), 1-13.
[16] S. Sutha, N. Gnanambigai, and P. Dinadayalan, *Extended Self-Organizing Map With Ubiquitous Counter Propagation Network In Classification For Diabetic Database*, Solid State Technology, *63*(6) (2020), 11102-11118.
[17] R. Tuwani, S. Wadhwa, and G. Bagler, *BitterSweet: Building machine learning models for predicting the bitter and sweet taste of small molecules*, Scientific Reports, *9*(1) (2019), 7155.
[18] P. Vuorimaa, *Use of the fuzzy self-organizing map in pattern recognition*, Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, IEEE (1994).
[19] P. Vuorimaa, *Fuzzy self-organizing map*, Fuzzy Sets and Systems, *66*(2) (1994), 223-231.
[20] C. C. Yang and N. K. Bose, *Generating fuzzy membership function with self-organizing feature map*, Pattern Recognition Letters, *27*(5) (2006), 356-365.