Viaña *et al*. *Complex Eng Syst* 2022;2:8
DOI: 10.20517/ces.2022.14

Complex Engineering Systems

Research Article

Open Access

Explainable fuzzy cluster-based regression algorithm with gradient descent learning

Javier Viaña[1,2], Stephan Ralescu[1], Anca Ralescu[1], Kelly Cohen[1], Vladik Kreinovich[3]

[1]College of Engineering and Applied Sciences, University of Cincinnati, Cincinnati, OH 45219, USA.
[2]Kavli Institute for Astrophysics and Space Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
[3]Computer Science Department, University of Texas at El Paso, El Paso, TX 79968, USA.

Correspondence to: Javier Viaña, College of Engineering and Applied Sciences, University of Cincinnati, 2850 Campus Way, Baldwin Hall - Room 745, Cincinnati, OH 45219, USA. E-mail: vianajr@mail.uc.edu

How to cite this article: Viaña J, Ralescu S, Ralescu A, Cohen K, Kreinovich V. Explainable fuzzy cluster-based regression algorithm with gradient descent learning. *Complex Eng Syst* 2022;2:8. http://dx.doi.org/10.20517/ces.2022.14

Received: 27 Apr 2022 **First Decision:** 9 May 2022 **Revised:** 11 May 2022 **Accepted:** 17 May 2022 **Published:** 31 May 2022

Academic Editor: Hamid Reza Karimi **Copy Editor:** Fanglin Lan **Production Editor:** Fanglin Lan

Abstract

We propose an algorithm for n-dimensional regression problems with continuous variables. Its main property is explainability, which we identify as the ability to understand the algorithm's decisions from a human perspective. This has been achieved thanks to the simplicity of the architecture, the lack of hidden layers (as opposed to deep neural networks used for this same task), and the linguistic nature of its fuzzy inference system. First, the algorithm divides the joint input-output space into clusters that are subsequently approximated using linear functions. Then, we fit a Cauchy membership function to each cluster, therefore identifying them as fuzzy sets. The prediction of each linear regression is merged using a Takagi-Sugeno-Kang approach to generate the prediction of the model. Finally, the parameters of the algorithm (those from the linear functions and Cauchy membership functions) are fine-tuned using gradient descent optimization. In order to validate this algorithm, we considered three different scenarios: The first two are simple one-input and two-input problems with artificial data, which allow visual inspection of the results. In the third scenario, we use real data for the prediction of the power generated by a combined cycle power plant. The results obtained in this last problem (3.513 RMSE and 2.649 MAE) outperform the state of the art (3.787 RMSE and 2.818 MAE).

Keywords: Clustering algorithms, explainable artificial intelligence, fuzzy logic, gradient methods, Takagi-Sugeno model

1. INTRODUCTION

Over the last few years, artificial ntelligence (AI) and Machine Learning (ML) have become ubiquitous in human society. Riding primarily on the empirical success of the deep neural network (DNN), AI and ML have expanded to sectors where safety and accountability must be guaranteed (e.g., medicine and national security) with an increasing emphasis on data privacy and protection. Decisions derived from AI-powered systems in such sectors directly affect people's lives, motivating a need for transparency: explainability and interpretability. Models should make sense to the human observer, and decisions must be justified and legitimate, with detailed explanations that promote trust. Human observers should understand how these decisions are made (the actions or procedures taken by the model) and why they work when they do (or fail when they don't).

The excellent performance of DNNs, however, relies on opaque abstractions in the (often hundreds) of hidden layers and millions of parameters that obscure their decision-making process, leading to the development of black box models which lack a clear understanding of how the model works. This performance – transparency trade-off was historically acceptable since AI-powered systems were deployed primarily for scientific and limited commercial work. The widespread expansion of AI and DNNs outside of academia thus motivates the need for modified machine learning techniques that learn explainable features while maintaining performance, as well as more interpretable, structured causal models.

Measuring explainability is an additional concern. Montavon et al.[1], Vedaldi et al.[2], and Letham et al.[3] are summaries of current techniques used by DNNs. There are seven major ways to achieve interpretability in a model:

1. Globally: understanding the entire logic of a model. For example, using rule sets generated from Bayesian models[4], similarly to how the fuzzy if-then rules behave, or through activation maximization[5,6], synthesizing the preferred inputs for neurons in neural networks.
2. Locally: understanding each individual decision or prediction separately[7–12].
3. Through visualization: representing the weights of a neural unit, by means of surrogate models[13], partial dependence plot (PDP)[14–16] and individual conditional expectation (ICE)[17].
4. Extracting rules: deriving comprehensive descriptions and approximations of the decision-making process[18,19].
5. Distilling a model: transforming complex models (e.g., deep networks) into more transparent models (shallow networks)[20–22].
6. Analyzing Sensitivity: studying how the output of the model is affected by its input and/or weight perturbations[23,24].
7. Analyzing feature importance: quantifying the contribution of each input variable[25].

The previous mechanisms could be understood as a top-down addition of mathematical tools to increase the interpretability. A less explored approach is the bottom-up redefinition of model architectures such that the algorithms are inherently transparent. In fact, most of the methods that seek to provide the desired transparency in neural networks elaborate on top of the fundamentals and rarely provide a novel algorithmic architecture. For example, Yang et al.[26] introduced an enhanced explainable neural network (ExNN) that decomposes a complex relationship into additive components, where the explainability is obtained by the addition of orthogonality constraints and the sparsity of the generated subnetworks. Similarly, Tran et al.[27] used variational autoencoders to generate interpretable features, and Wolf et al.[28] suggested the visualization of intermediate models to improve the trustworthiness of the system. While these techniques provide more clarity on the decision-making process, they are not easy to generalize and, in fact, are less explainable than a decision tree or a linear regression.

Additionally, minor changes to a DNN's input can negatively impact performance, causing misclassifications or false predictions, leading to the use of fragile models easily fooled by noise. The expansion of AI to mission

critical fields thus also motivates the need for robust solutions resilient to noise. Al-Mahasneh et al.[29] introduced a novel evolutionary algorithm that leverages competitive learning strategies to obtain the most optimal network that can handle noisy data. Autoencoders have been very successfully employed for denoising tasks, especially in medical procession[30], and speech enhancement[31]. Nevertheless, all the aforementioned methods hinge on the use of uncorrupted data points in the learning process. Both the noisy and clean instances are shown to the system until it learns to identify noisy patterns and develops its noise suppression strategy – information that is unavailable in most real-world applications. "Truth" often refers to the output values in the data, which in fact, may have a certain percentage of noise from the data acquisition process. Thus, the ground truth remains unknown.

This paper proposes a novel noise-resilient, explainable learning algorithm for n-dimensional regression problems with continuous variables. The salient features of the algorithm include initialization by (1) clustering the input data using a hierarchical approach which is (2) subsequently approximated using linear functions and (3) fitted to a Cauchy membership function, identifying the clusters as fuzzy sets. A Takagi-Sugeno-Kang (TSK) fuzzy system[32–36] merges the prediction of each linear regression to generate the prediction of the model. Parameters are fined tuned using gradient descent (GD) optimization. We use a GD-based optimization because it is computationally efficient; it produces a stable error gradient and a stable convergence. We demonstrate that our approach yields superior performance while providing explainability and interpretability as demonstrated for a meaningful industrial AI benchmark and comparing it with other state-of-the-art techniques.

Section II is a description of the algorithm's phases and the corresponding mathematical formulation. Section III describes the data sets used for empirical evaluation, including both synthetic and real-world data. Section IV discusses our empirical results. Section V concludes this paper and offers ideas for future work.

2. PROPOSED ALGORITHM

First, a clustering algorithm divides the joint input-output space into C groups of data points (C is user-defined). Despite referring to them as clusters, they do not necessarily represent isolated and well defined bulks of data, but rather convex regions of a tessellated work-space with observations inside. Therefore, we use an agglomerative MAX-linkage/complete-linkage hierarchical clustering algorithm[37–39] to get non-overlapping and non-intersecting globular rounded clusters (with convex boundaries). Hierarchical clustering algorithms have been widely used in combination with fuzzy logic[40–42].

Then, in each cluster, the dependence of the output y on the input $\mathbf{x}$ is approximated by a linear function, similar to how an ensemble of systems works[43], where each model is an expert in a certain region of the space. We use the notation $\mathbf{x_i}$ to denote the ith input, $i = 1, \ldots, Q$. Each $\mathbf{x_i}$ is a row vector of dimension N, $\mathbf{x_i}(j) = x_{ij}$, $j = 1, \ldots, N$. So, the data matrix can be expressed as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \vdots \\ \mathbf{x_Q} \end{bmatrix}. \tag{1}$$

For a generic vector $\mathbf{x}$, the linear function of cluster c is,

$$r_c(\mathbf{x}) = n_c + \mathbf{m_c} \cdot \mathbf{x}, \tag{2}$$

where $\mathbf{m_c}$ denotes the row vector of the slopes of the function and n_c denotes the intercept.

The output of the model is generated by merging the linear functions of the clusters through a Takagi-Sugeno-Kang approach,

$$\hat{y}(\mathbf{x}) = \frac{\sum_{c=1}^{C} \mu_c(\mathbf{x})\, r_c(\mathbf{x})}{\sum_{c=1}^{C} \mu_c(\mathbf{x})}, \tag{3}$$

where $\mu_c(\mathbf{x})$ is the membership function of each cluster. Appendix A provides an overview of fuzzy learning and the importance of the membership function's choice. We decided to use Cauchy membership functions for this algorithm and we provide three explanations for this choice. The first two are explained in Viaña et al.[44,45]. The third is covered in Appendix B, where we proof that this membership function is the fastest to compute. In the multidimensional space, the definition of the Cauchy membership function is defined as

$$\mu_c(\mathbf{x}) = \left[1 + \left\| \frac{\mathbf{x} - \mathbf{a_c}}{\mathbf{b_c}} \right\| \right]^{-1}, \tag{4}$$

where $\mathbf{a_c}$ and $\mathbf{b_c}$ are row vectors that determine the location of the center, and the amplitude, of the function, respectively, in each of the dimensions of the input space, $\mathbf{a_c}(j) = a_{cj}$, $\mathbf{a_c}(j) = a_{cj}$, $j = 1, \ldots, N$. As usual, $\|\mathbf{a_c}\|$ is the Euclidean norm $\sqrt{|a_{c1}|^2 + \ldots + |a_{cN}|^2}$. Here for vectors $\mathbf{x_i}$, $\mathbf{a_c}$ and $\mathbf{b_c}$, the operation $(\mathbf{x_i} - \mathbf{a_c})/\mathbf{b_c}$ means component-wise division, i.e., $(x_{ij} - a_{cj})/b_{cj}$. The values of $\mathbf{a_c}$ are initialized with the means of the data points from the cluster c in each of the N dimensions. Similarly, the values of $\mathbf{b_c}$ are initialized with the standard deviations of the cluster.

Once the parameters of both the membership functions and the linear functions have been initialized, the model is trained using GD learning. We define the loss function for the data vector as

$$J(\mathbf{x_i}) = [y_i - \hat{y}(\mathbf{x_i})]^2, \tag{5}$$

and the loss over the whole the training set X as

$$J(\mathbf{X}) = \frac{1}{2} \sum_{i=1}^{Q} [y_i - \hat{y}(\mathbf{x_i})]^2, \tag{6}$$

where the factor $\frac{1}{2}$ is added to ease the differentiation of (6).

The matrix expression of the formulation for the update of the different parameters is

$$\begin{bmatrix} \Delta a_{kj} \\ \Delta b_{kj} \\ \Delta m_{kj} \\ \Delta n_k \end{bmatrix} = \eta \sum_{i=1}^{m} \left\{ \frac{\mu_k(\mathbf{x_i})\ [y_i - \hat{y}(\mathbf{x_i})]}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \begin{bmatrix} \alpha_{kij} \\ \beta_{kij} \\ x_{ij} \\ 1 \end{bmatrix} \right\}, \tag{7}$$

where η denotes the learning rate in each epoch of the training, Δ denotes the increment required in each parameter,

$$\alpha_{kij} = 2\, \mu_k(\mathbf{x_i})\ [r_k(\mathbf{x_i}) - \hat{y}(\mathbf{x_i})]\ (x_{ij} - a_{kj}), \tag{8}$$

and

$$\beta_{kij} = \alpha_{kij}\, (x_{ij} - a_{kj}). \tag{9}$$

Equation (7) has been developed minimizing (6), as shown in the Appendix C.

For the update of the parameters, one can think of two different approaches, synchronously and asynchronously. The synchronous approach requires visiting all the observations of the training sample before training the

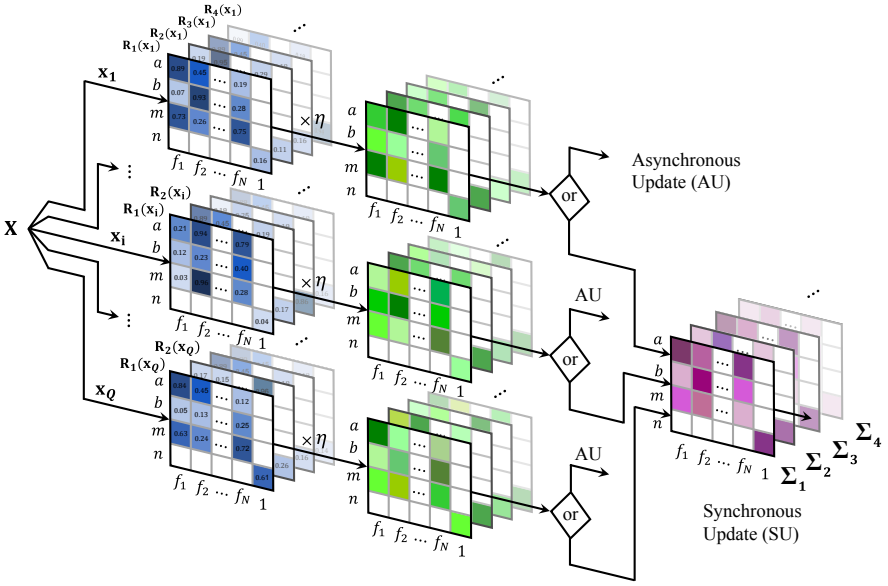

Figure 1. Synchronous and asynchronous approaches for the update of the system's parameters. Each observation of the dataset has a different set of matrices that together constitute the gradient of the loss function (transparent matrices of the back refer to different clusters). The entries of the blue matrices represent the derivatives with respect to the different parameters of the system; the rows determine the type of the parameter, a, b, m, or n, organized in a top-down fashion, and the columns identify the input feature (denoted with letter f, and 1 for the independent term). The green matrices are obtained after weighting the derivatives with the learning rates, which do not necessarily need to be identical for all the parameters. Finally, the resulting matrix could be added to those obtained for the other observations, creating the purple matrices, or it can be used before moving to the next instance of the training sample. The first approach is labeled as a synchronous update and considers a single update of the parameters in every epoch after all the observations have been visited. We identify the second approach as an asynchronous update, which requires, within the same epoch, as many updates of the parameters as observations are in the sample.

parameters. Thus, in this configuration, there is only one update performed at the end of each epoch. In the asynchronous version, we update the parameters of every cluster after studying an instance of the sample. In this case, we would have as many updates as observations per epoch. The latter requires a higher computational cost, but the evolution of the parameters is more stable and smooth than the synchronous approach. In the results section of this paper, we considered the asynchronous update. These two optimization versions can be visualized in Figure 1.

3. RESULTS

We tested this algorithm in three different problems; 1 input 1 output, 2 inputs 1 output, and 4 inputs 1 output. In the first two cases, we created the data artificially, and in the third application, we used real data obtained from the University of California Irvine (UCI) Machine Learning repository[46]. In this section, we provide a brief revision of the lessons learned from the first two, which were covered in Viaña et al.[47,48], and a detailed explanation of the results obtained in the last scenario.

3.1. Single-input problem

We studied 20 different functions. In order to measure the noise resilience of the method, we injected random noise in the output variable of the training dataset (from a uniform distribution), but the testing data had no noise (the algorithm was never exposed to the ground truth). In each case, we used a ranging number of clusters to evaluate the differences in the approximations obtained. As a benchmark, we utilized a selection of neural networks of one-hidden layer with varying numbers of hidden neurons. The proposed algorithm obtained significantly better results than all the neural network configurations studied[47]. In Figures 2-3, we

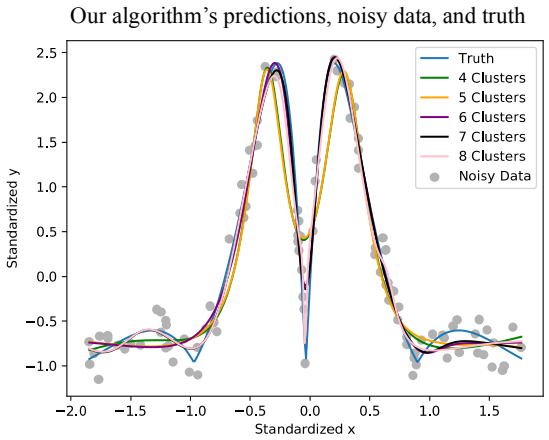

Figure 2. Single input function approximation with the algorithm presented in this paper considering 4, 5, 6, 7 and 8 clusters. The noisy data and the ground truth is also visible. All the systems were trained till the achievement of convergence in the training error.

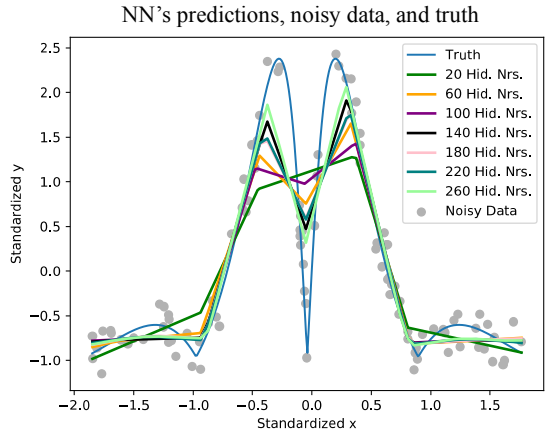

Figure 3. Single input function approximation benchmark with neural networks of architectures ranging from 20 to 260 hidden neurons in steps of 40. The noisy data and the ground truth are also visible. All the systems were trained till the achievement of convergence in the training error.

show the results obtained with both approaches for the function $y_i = \frac{|\sin(x_{i1})|}{1+x_{i1}^2}$.

For the training data, we considered the same 100 noisy observations for both approaches. All the systems were trained using a learning rate of 0.01, and the learning stopped when the loss converged. In all the cases studied, our algorithm converged in 1,000 epochs, whereas the neural networks required 10,000.

3.2. Double-input problem

The purpose of this second problem was to prove the applicability of the algorithm to n-dimensional inputs. Similarly to what it was done for the previous section, we trained the system with noisy data (artificially created from different functions and noise injected randomly from uniform distributions), but we tested it with the ground truth (non-noisy). In Viaña et al.[48], we discussed the results obtained for a variety of functions. Figure 4 shows the comparison for the function $y_i = \sin(x_{i1}) \cdot \sin(x_{i2}) \cdot \left(x_{i1}^2 + x_{i2}^2\right)$ when a system of 7 clusters is trained for 400 epochs, with 0.01 learning rate in all the parameters, and with 160 training points.

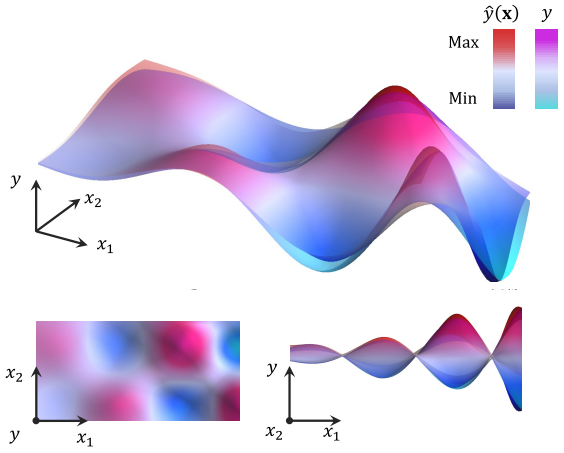

Figure 4. Double input function approximation with the algorithm presented in this paper considering 7 clusters. Both the ground truth and the approximation are visible. Three views are displayed for clarity. The system was trained till we achieved convergence in the training error.

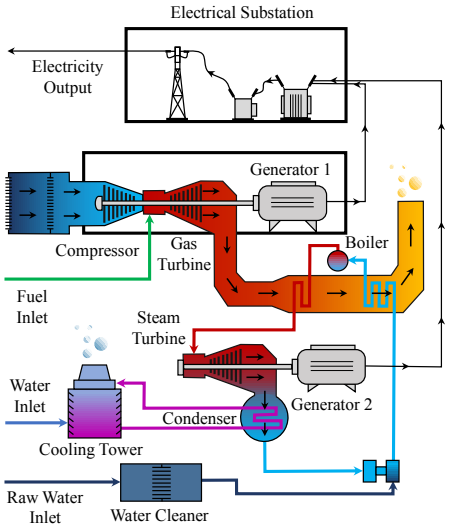

Figure 5. Schematic representation of the Combined Cycle Power Plant layout.

3.3. Four-input problem

In order to prove the applicability of the algorithm with real multidimensional problems, we considered the regression data of a combined cycle power plant (CCPP)[49] from the UCI machine learning repository. A CCPP consists of two types of turbines that generate electricity (gas and steam turbines). Gas turbines take the air and natural gas fuel inlets for the production of electricity at the cost of generating an outlet of exhaust gases at a high temperature. The heat of these gases is recycled with a secondary circuit of water that is evaporated for further extraction of energy in the water turbine. Figure 5 shows the schematic layout of a typical CCPP system, simplified for clarity purposes.

The popularity of such power plants has increased over the last years, particularly in areas with natural gas resources[50]. The CCPP considered for this study has a nominal generating capacity of 480 MW, two 160 MW ABB 13E2 Gas Turbines, two dual pressure Heat Recovery Steam Generators, and one 160 MW ABB Steam Turbine.

Table 1. Variables of the Combined Cycle Power Plant problem (publicly available dataset) [49]

Variable	Min	Max	Mean
Ambient Temperature	1.8°C	37.1°C	19.6°C
Atmospheric Pressure	99289 Pa	103330 Pa	101326 Pa
Relative Humidity	25.5%	100%	73.3%
Exhaust Steam Pressure	3381 Pa	10874 Pa	7241 Pa
Full Load Power Output	420.26 MW	495.76 MW	454.37 MW

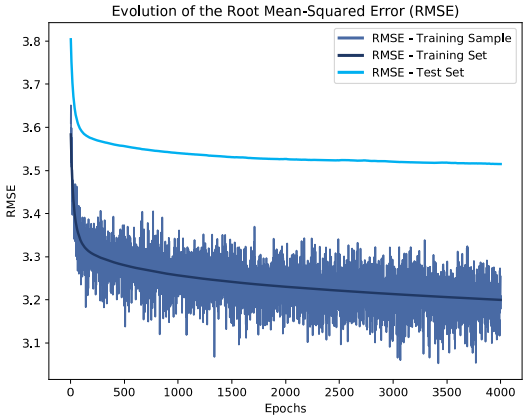

Figure 6. Evolution over epochs of the Root Mean Squared Error's for the training sample, the entire training set, and the test set for the CCPP problem.

The task chosen focuses on the prediction of the electric power produced by the plant when it operates at full load. The ambient temperature, atmospheric pressure, and relative humidity have an impact on the efficiency of the gas turbine, and so does the exhaust steam pressure affect the steam turbine performance. These four variables are considered inputs of the problem, and the power output of the plant, the value we want to predict, is the target. All the observations in the data correspond to the average hourly measurements of the sensors installed in the plant. Table 1 contains some basic statistics of these measures for the dataset selected.

The dataset consists of 10,000 observations. We considered 1,600 randomly chosen instances for training and 800 for testing (0.8 split). We chose a model of 6 clusters, and we trained for 4,000 epochs with a 0.0001 learning rate (for all the parameters). In order to avoid overtraining, we selected a random sample of 960 training instances randomly chosen from the training set in every epoch (samples of 60% from the training points). We used the root mean squared error (RMSE) and the mean absolute error (MAE) as figures of merit to evaluate the performance of our algorithm. Once the learning process finished, the final RMSE and MAE values of the training set were 3.200 and 2.450, respectively. Similarly, the RMSE and MAE values obtained for the testing set were 3.513 and 2.649. Figures 6-7 show the evolution of the RMSE and MAE over the epochs of the training process while using the GD optimization in our algorithm.

In Tufekci [50], a variety of different methods were studied for this particular CCPP problem. We provide a sum up of the RMSE values obtained in all those approaches together with our results in Table 2.

Table 3 shows the figures of merit for the top four methods studied by Tufekci [50] and the proposed algorithm.

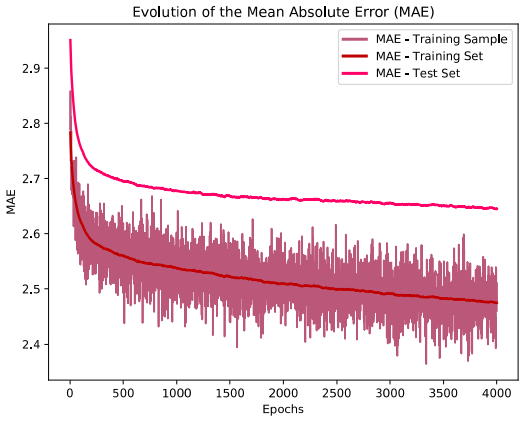

Figure 7. Evolution over epochs of the Mean Absolute Error for the training sample, the entire training set, and the test set for the CCPP problem.

Table 2. Comparison of test set RMSE for the different regression algorithms

Category	Regression Algorithm	Acronym	Reference	RMSE
Functions	This Paper's Algorithm	-	-	3.513
	Simple Linear Regression	SLR	[51]	5.426
	Linear Regression	LR	[52]	4.561
	Least Median Square	LMS	[53]	4.572
	Multilayer Perceptron	MLP	[54]	5.399
	Radial Basis Function NN	RBF	[55]	8.487
	Pace Regression	PR	[51]	4.561
	Support Vector Poly Kernel	SVR	[55]	4.563
Lazy-learning	IBk Linear NN Search	IBk	[56]	4.656
	KStar	K*	[57]	3.861
	Locally Weighted Learning	LWL	[56]	8.221
Meta-learning	Additive Regression	AR	[58]	5.556
	Bagging REP Tree	BREP	[59]	3.787
Rule-based	Model Trees Rules	M5R	[51]	4.128
Tree-based	Model Trees Regression	M5P	[60]	4.087
	REP Trees	REP	[61]	4.211

Table 3. Sorted list of the best regression algorithms using the test set performances

Category	Regression Algorithm	Performance	
		RMSE	MAE
Functions	This Paper's Algorithm	3.513	2.649
Meta-learning	Bagging REP Tree	3.787	2.818
Lazy-learning	KStar	3.861	2.882
Tree-based	Model Trees Regression	4.087	3.140
Rule-based	Model Trees Rules	4.128	3.172

4. DISCUSSION

Using the GD optimization provides a significant advantage in computational efficiency when it is compared to other classical bio-inspired evolutionary algorithms. The latter has been widely used to fine-tune fuzzy inference systems in a variety of applications. In the aerospace sector, for example, genetic fuzzy systems have a demonstrated success. We can see their usage in aerial vehicle controls for combat missions [62], multi-agent UAV routing [63,64], or autonomous collaborative operations [65–67]. On the contrary, GD has not been used

with fuzzy logic as much as nature-inspired meta-heuristics. In part because GD requires a tailored learning formulation for the parameters of the model, although it provides faster learning.

In terms of performance, the results obtained for the scenarios we tested are excellent. For the case of the single-input problem, it can be seen in Figure 2 that all the output curves are smooth and very similar despite the noise of the data. Conversely, the curves of the neural network benchmark shown in Figure 3 are significantly different and sharp, with abrupt changes that do not capture the essence of the ground truth. The smoothness property of our method is obtained thanks to the information merging of every cluster. Indeed, the joint contribution and weighting of each linear regression provide robustness and resilience to noise. We perceive that same smoothness in the double-input problem of Figure 4. As it occurred in the single-input case, our model is able to learn the pattern and not the noisy data, capturing the essence of the ground truth.

The parameters of the model we have introduced have a displayable mathematical meaning which eases their visualization. Each cluster can be plotted as linear regression and its associated membership function. Actually, a membership function identifies the region where the influence of each regression is greater, similarly to how an ensemble of expert systems would perform. This allows for easier interpretability and comprehension of the predictions made, which ultimately grants an explainable nature to our algorithm. On the other hand, the neurons of the neural network architecture considered cannot be uniquely associated with a certain region of the joint input-output space. Thus, it makes it harder to understand the meaning of each weight, which results in greater opacity and misunderstanding.

For the CCPP dataset, the evolution of both the RMSE and MAE of Figure 6-7 have a significant drop in the first 200 epochs. This initial phase of the learning process is crucial for an accurate reorganization of the membership functions and the model's parameters. Although from epoch 2,000 on, there is no significant improvement in the figures of merit of the test set, we decided to continue the training to proof the robustness to overtraining of our algorithm. This can be perceived from the fact that we do not see any increasing trends in the curves for the test set, despite the prolonged training.

Compared to other meta-heuristic approaches, such as evolutionary algorithms which are often combined with fuzzy systems[66,67], the proposed algorithm is significantly faster, and so it provides a big advantage in the training stages.

The superiority of the proposed algorithm is clearly demonstrated in Table 3, where we show its performance in comparison to the mentioned state-of-the-art techniques.

5. CONCLUSIONS

We have introduced a cluster-based algorithm for regression tasks with three key components:

1. A hierarchical clustering for initialization of its parameters.
2. A Takagi-Sugeno-Kang fuzzy inference system with Cauchy membership functions for prediction.
3. A gradient descent optimization for learning and fine-tuning.

We provide a theoretical development of the formulation for the parameter update by deriving the total loss of the training set. We also tested the algorithm in three different scenarios, two of them with synthetic data and a third with real data from a CCPP. The results we obtained outperformed the benchmarks. In the case of the CCPP, we were able to reduce the value of the RMSE from 3.787 (best score obtained from a wide variety of methods,[50]) to 3.513, and the MAE from 2.818 to 2.649. Additionally, this algorithm did not only show performance but also:

- Robustness to overtraining and noise resilience due to the merged contribution of the clusters.
- Transparency as a direct consequence of the interpretable nature of its parameters, the dominance of a cluster in every region of the input-output space, the lack of complexity, and the linguistic nature of its fuzzy if-then rules.

DECLARATIONS

Acknowledgments
The authors would like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript.

Authors' contributions
Conceptualization: Viaña J, Cohen K
Algorithm development: Viaña J
Membership function justification: Kreinovich V
Manuscript drafting: Viaña J
Manuscript edition and review: Viaña J, Cohen K, Ralescu A, Ralescu S

Availability of data and materials
Not applicable.

Financial support and sponsorship
The project that generated these results was supported by a grant from the "la Caixa" Banking Foundation (ID 100010434), whose code is LCF/BQ/AA19/11720045.

Conflicts of interest
All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

APPENDIX

A. Formulation of the problem
One of the main reasons why Lotfi Zadeh invented fuzzy techniques was to translate expert rules that use imprecise ("fuzzy") natural-language properties like "small", "medium", etc., into a precise control strategy. For this purpose, to each such property P, Zadeh proposed to assign a function $\mu_P(x)$ (known as *membership function*) that describes, for each possible value x of the corresponding quantity, the degree to which, according to the expert, an object with this value satisfies the property P – e.g., to what extent the amount x is small. This degree is usually assumed to be from the interval $[0, 1]$.

This is how the first applications of fuzzy techniques emerged: researchers elicited rules and membership functions from the experts, and used fuzzy methodology to design a control strategy. The resulting control was often reasonably good, but not perfect. So, a natural idea was proposed: to use the original fuzzy control

as a first approximation, and then tune its parameters based on the practical behavior of the resulting system.

This "fuzzy learning" idea was first used in situations when we have expert rules that provide a reasonable first approximation. However, it turned out that this learning algorithm leads to a reasonable control even when we do not have any expert rules, i.e., when we only have data.

When we start with expert knowledge, we elicit membership functions from the experts. But when we use fuzzy learning in situations where there is no expert knowledge, a natural question is: which membership functions should we use?

B. How to select membership functions: analysis and conclusion

To speed up the learning process, a natural idea is to select a membership function that would be the easiest to compute.

The main goal of any learning is to optimize the corresponding objective function – a function that describes which outputs are better and which are worse. For example, if we have examples of desired outputs, then the objective is to minimize the discrepancy between the values produced by the system and the values that we want to obtain.

Since the invention of calculus, the most efficient optimization techniques are based on computing the derivatives: one of the main uses of calculus is to identify points where a function attains its maximum or minimum as points where its derivative is 0, and the fastest ways to reach these points is to use the derivatives of the objective function. There are many optimization techniques, from the simplest gradient descent to more complex methods; all these techniques use differentiation.

From this viewpoint, it is desirable to have a differentiable (smooth) membership function. So, we are looking for easiest-to-compute smooth functions.

In a computer, the only hardware supported operations with numbers are arithmetic operations: addition, subtraction (which, for the computer, is, in effect, the same as addition), multiplication, and taking an inverse (division is implemented as $a/b = a \cdot (1/b)$). There are also operations min, max, and absolute value, but they are not everywhere differentiable, so we will not consider them.

We need the inverse operation in order to obtain a bounded function. If we do not use the inverse, then we get functions which are compositions of additions, subtractions, and multiplications – and thus, polynomials, since a polynomial can be defined as any function that can be obtained from variables and constants by using addition, subtraction, and multiplication. But a polynomial is not bounded – unless it is a constant. So, we need to use the inverse.

The inverse $1/x$ is not bounded – and for $x = 0$ it is not even defined, so we need at least one other operation.

We can have the additional operation before the inversion or after the inversion. If we perform the additional operation before the inverse, we have the following options: $1/(x + c)$ for some constant c, $1/(x + x)$, $1/(c \cdot x)$, $1/(x \cdot x)$, and $1/(1/x)$. The last option just gives x, and the other ones lead to unbounded functions which are not even everywhere defined.

If we have an additional operation after the inversion, we get $c + 1/x$, $x + 1/x$, $c \cdot 1/x$, and $x \cdot (1/x)$. The last option is simply a constant 1, and the others lead to unbounded functions. So, one additional operation is not sufficient, and we need at least two additional arithmetic operations.

One can show that if we have an additional operation after the inversion, we will still not get a bounded everywhere defined function. So, the only remaining option is to have two operations followed by inversion. These operations can be addition or multiplication, and they can operate on the variable x and some constant c.

If both operations are additions, then we get $1/(x + c_1 + c_2)$, $1/(x + x + c)$, and $1/(x + x + x)$ – all unbounded. If both are multiplications, we get $1/(x \cdot c_1 \cdot c_2)$, $1/(x \cdot x \cdot c)$, and $1/(x \cdot x \cdot x)$ – all unbounded too.

So, one of the two operations must be addition, and another one must be multiplication. We can have two subcases: when addition is performed first, and when multiplication is performed first. Let us consider them one by one.

- When addition is performed first: As a result of addition, we can have $c_1 + c_2$ – which is the same as a single constant (so it was already covered before), $x + c_1$, and $x + x$. We can multiply this result either by a constant c_2, or by x. So, we get the following options: $1/(c_2 \cdot (x + c_1))$, $1/(c_2 \cdot (x + x))$, $1/(x \cdot (x + c_1))$, and $1/(x \cdot (x + x))$. In all these cases, we have unbounded functions.
- When multiplication is performed first: As a result of multiplication, we can have $c_1 \cdot x$ or $x \cdot x$. We can add to each such result either a constant c_2 or a variable x. So, we get the following options: $1/(c_1 \cdot x + c_2)$, $1/(c_1 \cdot x + x)$, $1/(x \cdot x + c_2)$, and $1/(x \cdot x + x)$. In the first, second, and fourth cases, the function is unbounded and not everywhere defined. The only case when the function is bounded and everywhere defined is the third case $1/(x^2 + \text{const})$, which is exactly what we called a Cauchy membership function.

Let us obtain the resulting membership functions. A membership function is usually defined in such a way that its largest value is 1. For the function $1/(x^2 + c)$, the largest possible value is $1/c$, so we should take $c = 1$ and consider the membership function

$$\mu(x) = \frac{1}{1 + x^2}. \tag{10}$$

We also need to take into account that the numerical value of a physical quantity depends on the choice of the measuring unit and the choice of the starting point. If we change a measuring unit and/or a starting point, then we get new numerical values X which can be obtained from previous values x by a linear transformation $X = k \cdot x + a$, where k is the ratio of the measuring units and a is the difference in starting points. A classical example is the relation between temperature t_C in Celsius and temperature t_F in Fahrenheit: $t_F = 1.8 \cdot t_C + 32$.

When the original values x are described by the membership function (10), then, to get the membership function for the new numerical values X, we need to substitute, into the formula (10), the expression $x = \dfrac{X - a}{k}$ that describes x (the old value) in terms of X (the new value). As a result, for the new values, we get the following membership function

$$\mu_X(X) = \frac{1}{1 + \dfrac{(X - a)^2}{k^2}}. \tag{11}$$

Thus, the membership functions for which the computation is the simplest are Cauchy membership functions (11).

C. Proof of the learning formulas

Let us consider a generic cluster k, such that $\mathbf{a_k}$ is a row vector of dimension N, $\mathbf{a_k}(j) = a_{kj}$, $j = 1, \ldots, N$. The same applies to $\mathbf{b_k}$ and $\mathbf{m_k}$, but not to the independent term n_k. We can build the following matrix that

considers the gradient of the loss function for each of the different parameters, of this cluster.

$$\mathbf{R_k}(\mathbf{x_i}) = \begin{bmatrix} \frac{\partial J(\mathbf{x_i})}{\partial a_{k1}} & \cdots & \frac{\partial J(\mathbf{x_i})}{\partial a_{kj}} & \cdots & \frac{\partial J(\mathbf{x_i})}{\partial a_{kN}} & 0 \\ \frac{\partial J(\mathbf{x_i})}{\partial b_{k1}} & \cdots & \frac{\partial J(\mathbf{x_i})}{\partial b_{kj}} & \cdots & \frac{\partial J(\mathbf{x_i})}{\partial b_{kN}} & 0 \\ \frac{\partial J(\mathbf{x_i})}{\partial m_{k1}} & \cdots & \frac{\partial J(\mathbf{x_i})}{\partial m_{kj}} & \cdots & \frac{\partial J(\mathbf{x_i})}{\partial m_{kj}} & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \frac{\partial J(\mathbf{x_i})}{\partial n_k} \end{bmatrix}. \tag{12}$$

Then, the gradient of J is

$$\nabla \mathbf{J}(\mathbf{x_i}) = \{\mathbf{R_1}(\mathbf{x_i}), \ldots, \mathbf{R_k}(\mathbf{x_i}), \ldots, \mathbf{R_C}(\mathbf{x_i})\}. \tag{13}$$

Let us calculate each of the derivatives of $\mathbf{R_k}$, to do so, we use p_{kj} as one of the four possible parameters,

$$p_{kj} \in \{a_{kj}, b_{kj}, m_{kj}, n_k\}. \tag{14}$$

Then,

$$\frac{\partial J(\mathbf{x_i})}{\partial p_{kj}} = \frac{1}{2}\sum_{i=1}^{Q} \frac{\partial [y_i - \hat{y}(\mathbf{x_i})]^2}{\partial p_{kj}} == \sum_{i=1}^{Q} [y_i - \hat{y}(\mathbf{x_i})] \frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}}. \tag{15}$$

We study the last derivative expression separately, and substitute the definition of $\hat{y}(\mathbf{x_i})$,

$$\frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}} = \frac{\partial y_i}{\partial p_{kj}} - \frac{\frac{\partial}{\partial p_{kj}}\left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\ r_c(\mathbf{x_i})\right]}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} - \left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\ r_c(\mathbf{x_i})\right] \frac{\partial \left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\right]^{-1}}{\partial p_{kj}}. \tag{16}$$

The term $\frac{\partial y_i}{\partial p_{kj}}$ is null because no parameter has an influence over the observation. Thus,

$$\frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}} = -\frac{\frac{\partial}{\partial p_{kj}}\left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\ r_c(\mathbf{x_i})\right]}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} + \left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\ r_c(\mathbf{x_i})\right] \frac{\frac{\partial}{\partial p_{kj}}\left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\right]}{\left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\right]^2}. \tag{17}$$

Again, we resort to the definition of $\hat{y}(\mathbf{x_i})$, to simplify the second term,

$$\frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}} = -\frac{\frac{\partial}{\partial p_{kj}}\left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\ r_c(\mathbf{x_i})\right]}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} + \frac{\hat{y}(\mathbf{x_i})\ \frac{\partial}{\partial p_{kj}}\left[\sum_{c=1}^{C} \mu_c(\mathbf{x_i})\right]}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})}. \tag{18}$$

The parameter p_{kj} has only influence over $\mu_c(\mathbf{x_i})$ or $r_c(\mathbf{x_i})$ when $k = c$. Thus,

$$\frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}} = \frac{-\frac{\partial}{\partial p_{kj}}[\mu_k(\mathbf{x_i})\ r_k(\mathbf{x_i})] + \hat{y}(\mathbf{x_i})\ \frac{\partial \mu_k(\mathbf{x_i})}{\partial p_{kj}}}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})}. \tag{19}$$

Solving,

$$\frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}} = \frac{[\hat{y}(\mathbf{x_i}) - r_k(\mathbf{x_i})]\ \frac{\partial \mu_k(\mathbf{x_i})}{\partial p_{kj}} - \mu_k(\mathbf{x_i})\ \frac{\partial r_k(\mathbf{x_i})}{\partial p_{kj}}}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})}. \tag{20}$$

What follows is replacing p_{kj} for each one of the four parameters it represents. For the case of m_{kj}, the term $\frac{\partial \mu_k(\mathbf{x_i})}{\partial m_{kj}}$ is null, so

$$\frac{\partial \left[y_i - \hat{y}(\mathbf{x_i})\right]}{\partial m_{kj}} = \frac{-\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \frac{\partial r_k(\mathbf{x_i})}{\partial m_{kj}} = \frac{-\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} x_{ij}. \tag{21}$$

Similarly, the derivative with respect to n_k is

$$\frac{\partial \left[y_i - \hat{y}(\mathbf{x_i})\right]}{\partial n_k} = \frac{-\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \frac{\partial r_k(\mathbf{x_i})}{\partial n_k} = \frac{-\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})}. \tag{22}$$

For the case of a_{kj}, the term $\frac{\partial r_k(\mathbf{x_i})}{\partial a_{kj}}$ is null, so

$$\frac{\partial \left[y_i - \hat{y}(\mathbf{x_i})\right]}{\partial a_{kj}} = \frac{\hat{y}(\mathbf{x_i}) - r_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \frac{\partial \mu_k(\mathbf{x_i})}{\partial a_{kj}} \tag{23}$$

Let us calculate $\frac{\partial \mu_k(\mathbf{x_i})}{\partial a_{kj}}$,

$$\frac{\partial \mu_k(\mathbf{x_i})}{\partial a_{kj}} = \frac{\partial}{\partial a_{kj}} \left[1 + \left\| \frac{\mathbf{x_i} - \mathbf{a_k}}{\mathbf{b_k}} \right\|^2 \right]^{-1} = -\frac{\frac{\partial}{\partial^j a_k} \left\| \frac{\mathbf{x_i} - \mathbf{a_k}}{\mathbf{b_k}} \right\|^2}{\left[1 + \left\| \frac{\mathbf{x_i} - \mathbf{a_k}}{\mathbf{b_k}} \right\|^2 \right]^2} = \frac{2\left[\mu_k(\mathbf{x_i})\right]^2 \left(x_{ij} - a_{kj}\right)}{{b_{kj}}^2}. \tag{24}$$

Thus,

$$\frac{\partial \left[y_i - \hat{y}(\mathbf{x_i})\right]}{\partial a_{kj}} = \frac{\hat{y}(\mathbf{x_i}) - r_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \frac{2\left[\mu_k(\mathbf{x_i})\right]^2 \left(x_{ij} - a_{kj}\right)}{{b_{kj}}^2}. \tag{25}$$

Similarly, the derivative with respect to b_{kj} is

$$\frac{\partial \left[y_i - \hat{y}(\mathbf{x_i})\right]}{\partial b_{kj}} = \frac{\hat{y}(\mathbf{x_i}) - r_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \frac{2\left[\mu_k(\mathbf{x_i})\right]^2 \left(x_{ij} - a_{kj}\right)^2}{{b_{kj}}^3}. \tag{26}$$

Substituting the expressions obtained of $\frac{\partial [y_i - \hat{y}(\mathbf{x_i})]}{\partial p_{kj}}$ for each parameter in (15), we obtain the derivatives

$$\frac{\partial J(\mathbf{x_i})}{\partial m_{kj}} = -\sum_{i=1}^{Q} \left[y_i - \hat{y}(\mathbf{x_i})\right] \frac{\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} x_{ij}, \tag{27}$$

$$\frac{\partial J(\mathbf{x_i})}{\partial n_k} = -\sum_{i=1}^{Q} \left[y_i - \hat{y}(\mathbf{x_i})\right] \frac{\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})}, \tag{28}$$

$$\frac{\partial J(\mathbf{x_i})}{\partial a_{kj}} = -\sum_{i=1}^{Q} \left[y_i - \hat{y}(\mathbf{x_i})\right] \frac{\mu_k(\mathbf{x_i})}{\sum_{c=1}^{C} \mu_c(\mathbf{x_i})} \times \frac{2\,\mu_k(\mathbf{x_i}) \left[r_k(\mathbf{x_i}) - \hat{y}(\mathbf{x_i})\right] \left(x_{ij} - a_{kj}\right)}{{b_{kj}}^2}, \tag{29}$$

$$\frac{\partial J(\mathbf{x_i})}{\partial b_{kj}} = -\sum_{i=1}^{Q} \left[y_i - \hat{y}\left(\mathbf{x_i}\right)\right] \frac{\mu_k\left(\mathbf{x_i}\right)}{\sum_{c=1}^{C} \mu_c\left(\mathbf{x_i}\right)} \times \frac{2\,\mu_k\left(\mathbf{x_i}\right)\left[r_k\left(\mathbf{x_i}\right) - \hat{y}\left(\mathbf{x_i}\right)\right]\left(x_{ij} - a_{kj}\right)^2}{{b_{kj}}^3}. \tag{30}$$

We call

$$\gamma_{kij} = \frac{2\,\mu_k\left(\mathbf{x_i}\right)\left[r_k\left(\mathbf{x_i}\right) - \hat{y}\left(\mathbf{x_i}\right)\right]\left(x_{ij} - a_{kj}\right)}{{b_{kj}}^2}, \tag{31}$$

and

$$\delta_{kij} = \gamma_{kij} \frac{x_{ij} - a_{kj}}{b_{kj}}. \tag{32}$$

We have seen in the results that for small values of b_{kj} the variables γ_{kij} and δ_{kij} become too big. Thus, we advise either to normalize b_{kj} for the given clusters, or to get rid of the denominator in the learning process. Thus, if

$$\alpha_{kij} = 2\,\mu_k\left(\mathbf{x_i}\right)\left[r_k\left(\mathbf{x_i}\right) - \hat{y}\left(\mathbf{x_i}\right)\right]\left(x_{ij} - a_{kj}\right), \tag{33}$$

and

$$\beta_{kij} = \alpha_{kij}\left(x_{ij} - a_{kj}\right), \tag{34}$$

then, the resulting learning rules of the parameters are

$$\begin{bmatrix} \Delta a_{kj} \\ \Delta b_{kj} \\ \Delta m_{kj} \\ \Delta n_k \end{bmatrix} = -\eta \begin{bmatrix} \frac{\partial J}{\partial a_{kj}} \\ \frac{\partial J}{\partial b_{kj}} \\ \frac{\partial J}{\partial m_{kj}} \\ \frac{\partial J}{\partial n_k} \end{bmatrix} = \eta \sum_{i=1}^{Q} \left\{ \left[y_i - \hat{y}\left(\mathbf{x_i}\right)\right] \frac{\mu_k\left(\mathbf{x_i}\right)}{\sum_{c=1}^{C} \mu_c\left(\mathbf{x_i}\right)} \begin{bmatrix} \alpha_{kij} \\ \beta_{kij} \\ x_{ij} \\ 1 \end{bmatrix} \right\} \tag{35}$$

where η is the learning rate.

REFERENCES

1. Montavon G, Samek W, Müller KR. Methods for interpreting and understanding deep neural networks. *Digital signal processing* 2018 Feb;73:1–15. Available from: https://dx.doi.org/10.1016/j.dsp.2017.10.011. DOI
2. Vedaldi A, Montavon G, Hansen LK, Samek W, Muller KR. Explainable AI: interpreting, explaining and visualizing deep learning. Springer; 2019. Available from: https://www.vlebooks.com/vleweb/product/openreader?id=none&isbn=9783030289546.
3. Adadi A, Berrada M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE access* 2018;6:52138–60. Available from: https://ieeexplore.ieee.org/document/8466590. DOI
4. Letham B, Rudin C, McCormick TH, Madigan D. Interpretable Classifiers Using Rules And Bayesian Analysis: Building A Better Stroke Prediction Model. *The annals of applied statistics* 2015 Sep 1;9:1350–71. Available from: https://www.jstor.org/stable/43826424. DOI
5. Katzmann A, Taubmann O, Ahmad S, Mühlberg A, Sühling M, et al. Explaining clinical decision support systems in medical imaging using cycle-consistent activation maximization. *Neurocomputing (Amsterdam)* 2021 Oct 7;458:141–56. Available from: https://dx.doi.org/10.1016/j.neucom.2021.05.081. DOI
6. Xiao W, Kreiman G. Gradient-free activation maximization for identifying effective stimuli. *PLOS Comp Biol* 2019 May 1;16. DOI
7. Ribeiro MT, Singh S, Guestrin C. "Why Should I Trust You?". KDD '16. ACM; Aug 13, 2016. pp. 1135–44. Available from: http://dl.acm.org/citation.cfm?id=2939778. DOI
8. Ribeiro MT, Singh S, Guestrin C. Nothing Else Matters: Model-Agnostic Explanations By Identifying Prediction Invariance; 2016. Available from: https://explore.openaire.eu/search/publication?articleId=od_____18::5d14f874a3c1396f9cb09d48afc22423.
9. Lei J, G'Sell M, Rinaldo A, Tibshirani RJ, Wasserman L. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association* 2018 Jul 3;113:1094–111. Available from: http://www.tandfonline.com/doi/abs/10.1080/01621459.2017.1307116. DOI
10. Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, et al. How to explain individual classification decisions. *Journal of Machine Learning Research* 2010;11:1803–31. Available from: http://publica.fraunhofer.de/documents/N-143882.html.
11. Zeiler MD, Fergus R. In: Visualizing and Understanding Convolutional Networks. Computer Vision – ECCV 2014. Cham: Springer International Publishing;. pp. 818–33. Available from: http://link.springer.com/10.1007/978-3-319-10590-1_53. DOI
12. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning Deep Features for Discriminative Localization. IEEE; Jun 2016. pp. 2921–29. Available from: https://ieeexplore.ieee.org/document/7780688. DOI

13. Tripathy RK, Bilionis I. Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. *Journal of computational physics* 2018 Dec 15;375:565–88. Available from: https://dx.doi.org/10.1016/j.jcp.2018.08.036. DOI
14. Elith J, Leathwick JR, Hastie T. A Working Guide to Boosted Regression Trees. *The Journal of animal ecology* 2008 Jul 1;77:802–13. Available from: https://www.jstor.org/stable/20143253. DOI Pubmed
15. Chipman HA, George EI, Mcculloch RE. BART: Bayesian Additive Regression Trees. *The annals of applied statistics* 2010 Mar 1;4:266–98. Available from: https://www.jstor.org/stable/27801587. DOI
16. Green DP, Kern HL. Modeling Heterogeneous Treatment Effects In Survey Experiments With Bayesian Additive Regression Trees. *Public opinion quarterly* 2012 Oct 1;76:491–511. Available from: https://www.jstor.org/stable/41684581. DOI
17. Goldstein A, Kapelner A, Bleich J, Pitkin E. Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation; 2015. Available from: https://search.datacite.org/works/10.6084/m9.figshare.1006469.v2. DOI
18. Aung MSH, Lisboa PJG, Etchells TA, Testa AC, Calster BV, et al. In: Comparing Analytical Decision Support Models Through Boolean Rule Extraction: A Case Study of Ovarian Tumour Malignancy. Advances in Neural Networks – ISNN 2007. Berlin, Heidelberg: Springer Berlin Heidelberg;. pp. 1177–86. Available from: http://link.springer.com/10.1007/978-3-540-72393-6_139. DOI
19. Johansson U, Lofstrom T, Konig R, Sonstrod C, Niklasson L. Rule Extraction from Opaque Models– A Slightly Different Perspective. IEEE; Dec 2006. pp. 22–27. Available from: https://ieeexplore.ieee.org/document/4041465. DOI
20. Yashchenko AV, Belikov AV, Peterson MV, Potapov AS. Distillation of neural network models for detection and description of image key points. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* 2020 Jun 1;20:402–9. Available from: https://doaj.org/article/587a8589d29d48d8b0858eb10db8ae53. DOI
21. Tan S. Interpretable Approaches to Detect Bias in Black-Box Models. AIES '18. ACM; Dec 27, 2018. pp. 382–83. Available from: http://dl.acm.org/citation.cfm?id=3278802. DOI
22. Zhang W, Biswas G, Zhao Q, Zhao H, Feng W. Knowledge distilling based model compression and feature learning in fault diagnosis. *Applied soft computing* 2020 Mar;88:105958. Available from: https://dx.doi.org/10.1016/j.asoc.2019.105958. DOI
23. Cortez P, Embrechts MJ. Opening black box Data Mining models using Sensitivity Analysis. IEEE; Apr 2011. pp. 341–48. Available from: https://ieeexplore.ieee.org/document/5949423. DOI
24. Cortez P, Embrechts MJ. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information sciences* 2013 Mar 10;225:1–17. Available from: https://dx.doi.org/10.1016/j.ins.2012.10.039. DOI
25. Bien J, Tibshirani R. Prototype Selection For Interpretable Classification. *The annals of applied statistics* 2011 Dec 1;5:2403–24. Available from: https://www.jstor.org/stable/23069335. DOI
26. Yang Z, Zhang A, Sudjianto A. Enhancing Explainability of Neural Networks Through Architecture Constraints. *IEEE transaction on neural networks and learning systems* 2021 Jun;32:2610–21. Available from: https://ieeexplore.ieee.org/document/9149804. DOI Pubmed
27. Tran L, Dolph C, Zhao D. Enhancing Neural Network Explainability with Variational Autoencoders;. . DOI
28. Wolf L, Galanti T, Hazan T. A Formal Approach to Explainability. Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society; Jan 15, 2020. pp. 255–61.
29. Al-Mahasneh AJ, Anavatti SG, Garratt MA. Evolving General Regression Neural Networks for Learning from Noisy Datasets. IEEE; Dec 2019. pp. 1473–78. Available from: https://ieeexplore.ieee.org/document/9003073. DOI
30. Jifara W, Jiang F, Rho S, Cheng M, Liu S. Medical image denoising using convolutional neural network: a residual learning approach. *The Journal of supercomputing* 2019 Feb 6;75:704–18. Available from: https://search.proquest.com/docview/2187111783. DOI
31. Kao YY, Hsu HP, Hung KH, Lee SK, Lai YH, et al. A study on attention-based objective function in deep denoising autoencoder based speech enhancement. *The Journal of the Acoustical Society of America* 2019 Oct;146:2794. Available from: http://dx.doi.org/10.1121/1.5136680. DOI
32. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Single Hidden Layer CEFYDRA: Cluster-first Explainable FuzzY-based Deep self-Reorganizing Algorithm. In: Applications of Fuzzy Techniques: Proceedings of the 2022 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS; May 2022. .
33. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Multiple Hidden Layered CEFYDRA: Cluster-first Explainable FuzzY-based Deep self-Reorganizing Algorithm. In: Applications of Fuzzy Techniques: Proceedings of the 2022 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS; May 2022. .
34. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Initialization and Plasticity of CEFYDRA: Cluster-first Explainable FuzzY-based Deep self-Reorganizing Algorithm. In: Applications of Fuzzy Techniques: Proceedings of the 2022 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS; May 2022. .
35. Bede B, Williams A. In: Takagi-Sugeno Fuzzy Systems with Triangular Membership Functions as Interpretable Neural Networks. Explainable AI and Other Applications of Fuzzy Techniques. Cham: Springer International Publishing; 2021. p. 14–25. Available from: http://link.springer.com/10.1007/978-3-030-82099-2_42. DOI
36. GFS-TSK BCWDDAU. In: Allison Murphy. Explainable AI and Other Applications of Fuzzy Techniques. Cham: Springer International Publishing; 2021. Available from: http://link.springer.com/10.1007/978-3-030-82099-2_42. DOI
37. Murtagh F, Contreras P. Algorithms for hierarchical clustering: an overview. *Wiley interdisciplinary reviews Data mining and knowledge discovery* 2012 Jan;2:86–97. Available from: https://api.istex.fr/ark:/67375/WNG-ZSSZ1C7W-3/fulltext.pdf. DOI
38. Murtagh F, Contreras P. Algorithms for hierarchical clustering: an overview, II. *Wiley interdisciplinary reviews Data mining and knowledge discovery* 2017 Nov;7:n/a. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1219. DOI
39. Sharma N, Sharma P, Tiwari K. A Review on Clustering Method based on Unsupervised Learning Approach. *International journal of*

computer applications 2018 Sep 18,;181:20–23. DOI
40. Zhang X, Xu Z. Hesitant fuzzy agglomerative hierarchical clustering algorithms. *International journal of systems science* 2015 Feb 17;46:562–76. Available from: http://www.tandfonline.com/doi/abs/10.1080/00207721.2013.797037. DOI
41. Ciaramella A, Nardone D, Staiano A. Data integration by fuzzy similarity-based hierarchical clustering. *BMC bioinformatics* 2020 Aug 21;21:350. Available from: https://www.ncbi.nlm.nih.gov/pubmed/32838739. DOI Pubmed
42. Aliahmadipour L, Eslami E. GHFHC: Generalized Hesitant Fuzzy Hierarchical Clustering Algorithm. *International journal of intelligent systems* 2016 Sep;31:855–71. Available from: https://api.istex.fr/ark:/67375/WNG-3QPVTL6Q-T/fulltext.pdf. DOI
43. Polikar R. In: Ensemble Learning. Ensemble Machine Learning. Boston, MA: Springer US; 2012. pp. 1–34. Available from: http://link.springer.com/10.1007/978-1-4419-9326-7_1. DOI
44. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Why Cauchy Membership Functions: Efficiency. *Advances in Artificial Intelligence and Machine Learning* 2021;1:81–88.
45. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Why Cauchy Membership Functions: Reliability. *Advances in Artificial Intelligence and Machine Learning (To appear)* .
46. Dua D, Graff C. UCI Machine Learning Repository; 2017. University of California, Irvine, School of Information and Computer Sciences. Available from: http://archive.ics.uci.edu/ml.
47. Viaña J, Cohen K. In: Fuzzy-Based, Noise-Resilient, Explainable Algorithm for Regression. Explainable AI and Other Applications of Fuzzy Techniques. Cham: Springer International Publishing; 2021. pp. 461–72. Available from: http://link.springer.com/10.1007/978-3-030-82099-2_42. DOI
48. Viaña J, Ralescu S, Cohen K, Ralescu A, Kreinovich V. Extension to Multidimensional Problems of a Fuzzy- based Explainable & Noise-Resilient Algorithm. In: Constraint Programming and Decision Making (CoProd 2021); 2021. .
49. *Combined Cycle Power Plant Data Set, UCI Machine Learning Repository* 2012. Available from: https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant.
50. Tufekci P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International journal of electrical power & energy systems* 2014 Sep;60:126–40. Available from: https://dx.doi.org/10.1016/j.ijepes.2014.02.027. DOI
51. Ekinci S, Celebi UB, Bal M, Amasyali MF, Boyaci UK. Predictions of oil/chemical tanker main design parameters using computational intelligence techniques. *Applied soft computing* 2011;11:2356–66. Available from: https://dx.doi.org/10.1016/j.asoc.2010.08.015. DOI
52. Azuaje F. Witten IH, Frank E: Data Mining: Practical Machine Learning Tools and Techniques 2nd edition. *Biomedical engineering online* 2006 Sep 29;5. DOI
53. Simpson DG. In: Introduction to Rousseeuw (1984) Least Median of Squares Regression. Breakthroughs in Statistics. New York, NY: Springer New York;. pp. 433–61. Available from: http://link.springer.com/10.1007/978-1-4612-0667-5_18. DOI
54. Kubat M. Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7. *Knowledge engineering review* 1999 Feb;13:409–12. Available from: https://dx.doi.org/10.1017/S0269888998214044. DOI
55. Elish MO. A comparative study of fault density prediction in aspect-oriented systems using MLP, RBF, KNN, RT, DENFIS and SVR models. *Artificial Intelligence Review* 2014 Dec;42:695–703. Available from: https://search.proquest.com/docview/1624871931. DOI
56. Han J, Kamber M. Data mining: concepts and techniques. Morgan Kaufmann; 2005. Available from: http://cds.cern.ch/record/1991675.
57. Cleary JG, Trigg LE. In: K: An Instance-based Learner Using an Entropic Distance Measure. Machine Learning Proceedings 1995. Elsevier Inc; 1995. pp. 108–14. Available from: https://dx.doi.org/10.1016/B978-1-55860-377-6.50022-0. DOI
58. Friedman JH. Stochastic gradient boosting. vol. 38 of Computational Statistics & Data Analysis. Amsterdam: Elsevier Science; 2002. pp. 367–78. Available from: http://econpapers.repec.org/article/eeecsdana/v_3a38_3ay_3a2002_3ai_3a4_3ap_3a367-378.htm. DOI
59. D'Haen J, Poel DVD. Temporary Staffing Services: A Data Mining Perspective. IEEE; Dec 2012. pp. 287–92. Available from: https://ieeexplore.ieee.org/document/6406453. DOI
60. Wang Y, Witten IH. Induction of model trees for predicting continuous classes. vol. 96/23. Hamilton, N.Z: Dept. of Computer Science, University of Waikato; 1996. Available from: https://natlib-primo.hosted.exlibrisgroup.com/primo-explore/search?query=any,contains,992935923502836&tab=catalogue&search_scope=NLNZ&vid=NLNZ&offset=0.
61. Portnoy S, Koenker R. The Gaussian Hare and the Laplacian Tortoise: Computability of Squared- Error versus Absolute-Error Estimators. *Statistical science* 1997 Nov 1;12:279–96. Available from: https://www.jstor.org/stable/2246216. DOI
62. Ernest N, Carroll D, Schumacher C, Clark M, Cohen K, et al. Genetic Fuzzy based Artificial Intelligence for Unmanned Combat Aerial Vehicle Control in Simulated Air Combat Missions. *Journal of defense management* 2016;6. DOI
63. Sathyan A, Ernest ND, Cohen K. An Efficient Genetic Fuzzy Approach to UAV Swarm Routing. *Unmanned systems (Singapore)* 2016 Apr;4:117–27. Available from: http://www.worldscientific.com/doi/abs/10.1142/S2301385016500011. DOI
64. Ernest N, Cohen K, Kivelevitch E, Schumacher C, Casbeer D. Genetic Fuzzy Trees and their Application Towards Autonomous Training and Control of a Squadron of Unmanned Combat Aerial Vehicles. *Unmanned systems (Singapore)* 2015 Jul;3:185–204. Available from: http://www.worldscientific.com/doi/abs/10.1142/S2301385015500120. DOI
65. Sathyan A, Cohen K, Ma O. Comparison Between Genetic Fuzzy Methodology and Q-Learning for Collaborative Control Design. *International journal of artificial intelligence & applications* 2019 Mar 31;10:1–15. DOI
66. Sathyan A, Cohen K, Ma O. Genetic Fuzzy Based Scalable System of Distributed Robots for a Collaborative Task. *Frontiers in robotics and AI* 2020;7. Available from: https://www.ncbi.nlm.nih.gov/pubmed/33501362. DOI PubMed
67. Sathyan A, Ma J, Cohen K. Decentralized cooperative driving automation: a reinforcement learning framework using genetic fuzzy systems. *Transportmetrica (Abingdon, Oxfordshire, UK)* 2021 Jan 1;9:775–97. DOI