Applied Intelligence (2019) 49:3221–3234
https://doi.org/10.1007/s10489-019-01449-w

Incremental regularized Data Density-Based Clustering neural networks to aid in the construction of effort forecasting systems in software development

Paulo Vitor de Campos Souza[1] · Augusto Junio Guimaraes[1] · Vanessa Souza Araujo[1] · Thiago Silva Rezende[1] · Vinicius Jonathan Silva Araujo[1]

Published online: 22 March 2019

Abstract

The challenge of reducing complexity, failures and time in software development are tasks found in the vast majority of information technology companies. The professionals seek to structure the development of applications through agile methodologies, but in reality, there are several difficulties in planning the time to make applications development tasks. To help in this situation, this paper proposes the use of fuzzy neural network composed by fuzzy rules to assist in the construction of a specialist system based on interpretable rules, facilitating the prediction of software development hours according to the complexity of the elements present in the project. To support in the data fuzzification process, an incremental density technique is proposed for the first layer of the model. To improve the sensitivity of the neuron present in the neural network a Leaky-ReLU type activation function is used to obtain the results. The set of rules to be created, through tests in a real database based on the technique of use case point, can help in the development of future expert systems, to be used by these professionals. The results of the tests were efficient to generate predictability about the efforts to build the software.

Keywords Fuzzy neural network · Effort forecasting · Use case point · Expert systems · Data density

1 Introduction

Software development planning and construction is a complex task for software engineers, developers and project managers. With the help of concepts on business rules and side by side with agile methodologies, these professionals work to meet the needs and specifications of their clients. Regardless of the programming language to be implemented in the project, crucial factors need to be carefully considered, defined and documented. Besides, the initial phase is characterized by the feasibility study of the project where it is abstracted, directly by the client, everything that is impacting the development of the work. From then on, they follow the design, preparation, construction, testing, homologation, deployment and stabilization phases. They are performed iteratively and incrementally, where in each iteration a set of use cases is developed [1]. The problem encountered in large part of software developer companies are failures in predicting software efforts caused by various factors. Among them, omissions in the choice of methodologies, little time devoted to complex tasks, choice of disqualified professionals, and even human errors, due to delays or failures in the implementation of functionalities. They are thus causing significant impacts on software development time [2]. A study has been carried out to collect data referring to several multiple criteria present in software development. These studies motivated evaluations to understand the nature of this problem and to build mechanisms that allow managers to be able to close the deadline for delivering a software product to

✉ Paulo Vitor de Campos Souza
goldenpaul@informatica.esp.ufmg.br

Augusto Junio Guimaraes
augustojunioguimaraes@gmail.com

Vanessa Souza Araujo
v.souzaaraujo@yahoo.com.br

Thiago Silva Rezende
silvarezendethiago@gmail.com

Vinicius Jonathan Silva Araujo
vinicius.j.s.a22@hotmail.com

[1] University Center UNA- Betim, Av. Gov. Valadares, 640 - Centro, Betim - MG, 32510-010, Brazil

their client where the main evaluation technique was the use case point [3]. This paper proposes the usage of a real database collected in Turkish software development companies [4] for a hybrid model of fuzzy neural networks aiming to assist the construction of expert systems for time estimation of software development [5]. The fuzzy rules are constructed through the first layer of the hybrid model that uses concepts from the Incremental Data Density-Based Clustering (IDDC) algorithm [6] to create membership functions based on the density of the training data. This approach allows creating Takagi Sugeno rules of the If / THEN type [7], allowing the database to be interpreted more straightforwardly by the managers involved in the process. The second layer uses fuzzy logic neurons, which perform the aggregation of the neurons of the first layer. Because it is a problem where the number of neurons generated can be high due to the number of characteristics evaluated, a bootstrap-based smoothing technique is applied to define the essential neurons of the problem. The final weights of the second layer will be used by a neural aggregation network that has a single artificial neuron [5]. The synaptic weights will be generated using the concept of extreme learning machine [8], where there is no need to update the internal parameters of the fuzzy neural network continually. The paper is organized as follows: Section 2 presents the central concepts that guide the research, such as the definitions of fuzzy neural networks, concepts related to the complexity of software construction and fuzzification techniques. Section 3 have as its central focus the presentation of aspects related to the database and how it has already been worked in the literature, evidencing the contribution of this work. In it will also be present concepts and forms with which the model of fuzzy neural networks will act in the resolution of the problems. Section 4 presents the methodology used in the tests and the methods, parameters, and configurations of the tests and their respective results will be presented. Finally, in Section 5 the conclusions of the work are presented.

2 Literature review

2.1 Software development

Software development allowed companies and individuals to obtain higher quality digital services, allowing manual tasks to be performed in more dynamic ways through computerized devices. The use of mobile phones, tablets and direct access to the internet made business decision making more and more dynamic and cohesive to meet the emerging needs of this market. Currently, countries that excel in software development like Japan, China, the United States, and India import several electronic products with different software. Other countries like Brazil invest in several software factories to produce applications to serve the domestic and foreign markets [1]. However, there is a specific barrier in software production and customer satisfaction, especially as it is a new area of economic interest. In surveys carried out by [9] and [10] it is noted that most of the products are not delivered as they should, especially after the deadline and without meeting the specifications defined in the initial phases of the project. Software development suffers from immeasurable external variables, such as the lack of technological resources to develop a specific task requested by the contractor, a lack of skilled labor, high volatility of software requirements during the execution phase of the project, and especially the lack of time to develop quality applications. In many cases, the software tests are left aside so that the deliveries happen in the established deadlines, but without taking into account possible bugs that may appear. In general, software engineers are already planning the high-time application development schedule for post-install corrections. The time for the production of a use case varies according to several external agents, but the complexity of the case interferes directly with this type of evaluation [11]. As well as the use cases, the complexity of the actors involved in the problem, the connection between internal and external systems, the use of techniques such as refactoring and reusing is highlighted. Based on these contexts, several researchers tried to understand the direct relationship between the complexities involved in various factors and the time of software construction. To improve these agile technical indices have been developed and are widely diffused in software factories, but there is still a shortage of experienced human resources to handle all the characteristics of these techniques. These alternatives can improve delivery time, but their prediction even becomes a complex factor, depending only on the experience of the project manager should know very well the potential of his team or the reasonableness of the developer, who must weight their productive capacity and the main demands requested by customers [2].

2.2 Software effort estimation

Diverse models have happened proposed for software development effort estimation, e.g., COCOMO, SLIM, Estimacs, Function Point Analysis (FPA). The software effort estimation methods can be grouped according to the characteristics used to perform the predictions, standing out in three categories: specialist knowledge, analogy, and regression analysis [12].

The first strategies to estimate software development effort was approached in the late 1960s and relied on expert knowledge. In these circumstances, a domain expert applies

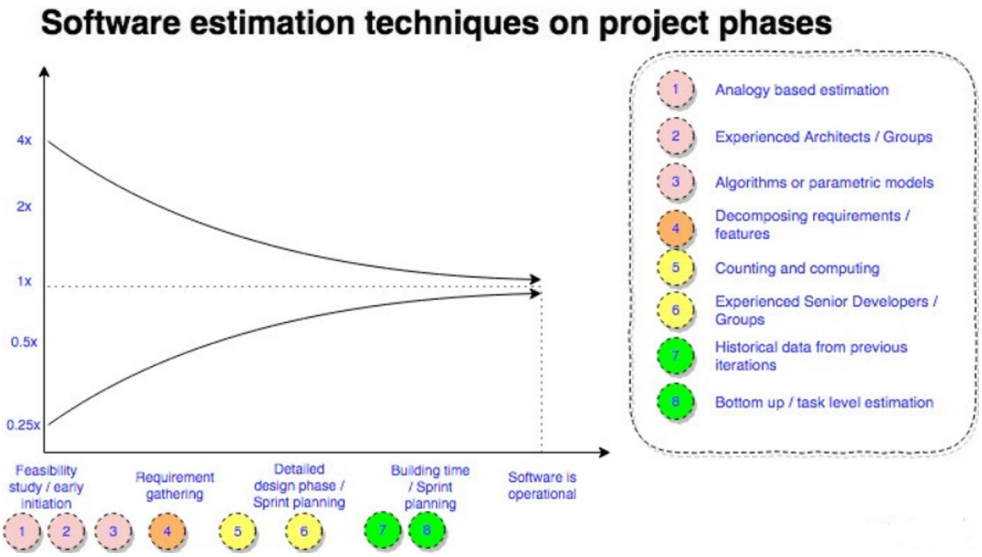

Fig. 1 Software Estimation Techniques on project phases. Available in: https://medium.com/@pablo.iorio/which-software-development-estimation-techniques-work–better-depending-the-project-phase-e1b2738287dd

his or her prior experience to come up with an estimation of the required effort. Some different variations exist, e.g., Delphi expert estimation, in which several expert developers formulate an independent estimate and the median of these estimates are used as the final effort estimation [13]. Expert judgment relies on the experience of experts allowing a more significant number of variables, weights, and activities to be representative of the company context.

Estimation by analogy is the procedure that has been proposed for a long period as a legitimate alternative to expert analysis and algorithmic cost estimation. The uses of analogy-based software effort estimation models have been determined and validated in many studies. Estimation by analogy is case-based reasoning (CBR) approach [14]. CBR is based on the psychological concepts of analogical reasoning, dynamic memory and the role of previous situations in learning and problem-solving. A CBR processing cycle is composed of four stages:

(1) Retrieve the most similar project (case);
(2) Reuse the project to attempt to determine the problem;
(3) Revise the suggested solution if necessary;
(4) Retain the solution and the new problem as a different project [15].

Figure 1 presents relevant features on software efforts according to the phase that the software development project is.

Effort estimation by analogy has several benefits. Users are more prepared to accept results from analogy-based methods as deprecated to formal model-based methods because it is a similar form of reasoning as human problem-solving. In general, they can be categorized as the composite method or as a machine learning method [16].

Analogy-based estimation can deal with poorly understood domains that are difficult to model, since it relies on historical data of similar projects to the target project rather than rules in the case of rule-based systems [16].

The models exist based on regression analysis of some set of preceding cases. Independent variables introduce an estimate of system size (in lines of code alternatively, function points) [17].

There are several comparative studies between the techniques of definition or estimation of efforts in software construction, as in [17]. Empirical studies using artificial neural networks were also applied in the prediction of UCP [18]. It should be noted that studies related to software testing efforts were also applied using UCP [19].

2.3 Use case points

Several techniques have been proposed to facilitate the management of software construction, such as the *SLOC* (Source Lines of Code), [11] and *FP* (Function Points) [20] technique. These methods seek to evaluate comprehension according to features present in code lines or function points, making a proportional relation between them. In this paper, we highlight the approach proposed by [21] that take in account the evidence of the software complexity linking it to the characteristics present in the system usage. Some factors are relevant to this approach: At first, the actors of the system are assigned graded labels, such as simple, medium and complex. This assessment depends on the synergy between business and systems analysis team to not measure undue influence on the actors who will be using the system. The next step is to set the Unweighted Weight of Actors (*UAW*) which is calculated by summing the weights of all actors [21]. The same procedure is performed with the use cases, characterizing them as simple, medium and complex, depending on the number of steps to run on your mainstream and alternate flows. In short, the fewer steps

Technical Factor	Weight	Relevance 0-5	Score
T1 Distributed System	2	1	2
T2 Performance	1	1	1
T3 End User Efficiency	1	1	1
T4 Complex Internal Processing	1	1	1
T5 Reusability	1	2	2
T6 Easy to Install	0,5	1	0,5
T7 Easy to Use	0,5	1	0,5
T8 Portability	2	3	6
T9 Easy to Change	1	2	2
T10 Concurrency	1	1	1
T11 Special Security Features	1	1	1
T12 Provides Direct Access for Third Parties	1	1	1
T13 Special User Training Facilities Are Required	1	1	1
Sum			20
Complexity Factor			0,6
C2			0,01
TCF			0,8

Fig. 2 Example UCP

that are performed to complete an activity in a use case, the less complex it will be for the system and consequently to be developed. In the same way, as in the actors, the Unadjusted Use-Case Weight (*UUCW*) is calculated as the sum of the weights of all use cases. Then the *UAW* is added to the *UUCW* to produce the Unadjusted Use-Case Points (*UUCP*) [21]. Finally, the Evaluation Points are then adjusted according to the Technical Complexity Factors (*TCF*), such as the number of people involved, the client engagement, the level of knowledge of the team, among others, and environmental factors (*EF*) that external threats, such as crises, lack of corporate redemption, socio-economic or political events, are linked. Environmental factors and their weights were imported from Function Point theory and technical complexity factors. Thus, $UCP = UUCP * TCF * EF$ [21]. Figure 2 shows an applied example of any evaluation of UCP. Note that the weights and values will vary according to the nature of the problem being solved and the experts involved in this analysis.

2.4 Intelligence models used in predicting use-case points

Fuzzy rules-based fuzzy systems were used in [22] to perform the estimation of effort in the production of software using the UCP concepts. Already in the work of [23] the concepts of linear regression and perceptron were combined to solve the problems of estimation of efforts. Already in the works of [24] the focus was the same because the concepts of a cascade conceived the artificial neural network used. Finally, the model proposed by [25] worked to find the software effort with a hybrid two-layer model, where the first layer was composed of fuzzy neurons and the second layer composed of an artificial neuron. It should be noted that in the model addressed by [22], fuzzy rules were also generated, but the training approach proposed in this article, as it was used with few dimensions, differs from the proposal of this work that intends to use all dimensions of the problem. The following are presented intelligent model architectures that were used to support this type of effort prediction, with emphasis on the models proposed in [22], and [23]. Also noteworthy are the models proposed in [4, 26–37]. These works generally deal with intelligent techniques to assist in the prediction of hours for the execution of software activities through techniques of artificial intelligence, genetic algorithms, evolutionary algorithms, among others. Figure 3 presents examples of intelligent model architectures used to predict effort in software development. The work developed in [37] is different from the current work due to the fuzzification approach and how the output of the fuzzy neural network is treated using different activation functions.

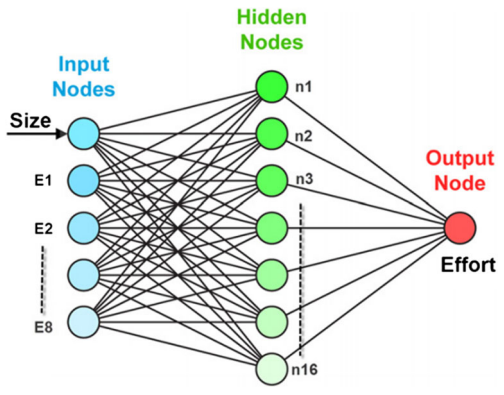

Fig. 3 Effort model architecture [23]

2.5 Fuzzy neural network

Fuzzy neural networks are characterized by neural networks formed by fuzzy neurons [38]. Thus, a Fuzzy neural network can be defined as an intelligent model composed of a fuzzy inference system and a neural aggregation network, trained by an algorithm provided by a neural network. The use of hybrid systems allows the joint use of advantageous properties that each of the approaches can promote to solve the problem. Fuzzy systems are known for their ability to transform data into more coherent information for human understanding, through techniques that change the input space into linguistic features. In the case of artificial neural networks, their ability to adapt to various types of training and learning techniques can carry out activities commonly practiced by the human mind in solving and learning everyday situations. Given this analogy, the union of the neural network with the fuzzy logic comes with the intention of softening the deficiency of each of these systems, making us have a more efficient system, robust and easy to understand [39].

Currently, several research areas have used hybrid models to solve complex problems, we can find them in the pattern classification for children autism prediction [40], Improvement in the detection of breast cancer [41], In the time series prediction [5, 42, 43], In addition to helping voice-controlled robot systems [44]. Other health topics use fuzzy neural networks for immunotherapy treatments [45], diseases [46], and predictions about the heart [47]. The area of the projection of natural events is highlighted with the hybrid models proposed in [48–50]. Areas of the economy [51], pattern classification [52, 53], feature selection [54] and regression problems [55] have also been solved using fuzzy neural networks.

In the field of computing, we highlight works in the area of cyber attacks [56–58] and [59].

2.6 Incremental Data Density-Based Clustering - IDDC

The DDC algorithm (Density-Based Clustering) was proposed to use different concepts of algorithms such as fuzzy c-means [60]. He used the density of the actual data sample distribution to identify the clusters automatically. That is, it uses the organization of the data submitted to the problem to determine the possible groupings and correlated characteristics between the problem information. This approach is based on mountain-grouping techniques [61], where clusters are identified by the mountain function, which is a Gaussian function of distances between data samples. The proposal is influenced by subtractive group [62] allowing samples to be used for clustering centers rather than points in a grid without the need for [6]. This approach allows for a better interpretation of the data without the need to be based on the previous number of membership functions. Commonly, this fuzzification algorithm transfers to the nature of the input data of the problem the organization of the clusters of the problem.

For a long time, considered useful methods of extracting information from data sets, grouping methods are used to partition the input space of intelligent models to solve problems of different characteristics. In the DDC method, it is possible to identify the cluster centers based on the density of the samples in the region. The highest density point is chosen for the first center of the cluster. All examples within the first rays are included in the group. These points are then removed from the non-clustered data and recalculated densities [6].

The maximum density point can be found at multiple locations in the sample space, even within a cluster, in which case it is very likely that this method will find new cluster centers at the edge of a local group. At this stage, a significant part of the cluster may include low-density spaces between groups or overlap a nearby cluster. To overcome this, DDC moves the cluster center to the densest point of data and reallocates the data samples to the cluster. This allows new instances that are within the radius of the cluster to be added, allowing you to move the cluster closer to the center of the data group. The cluster spokes are now adjusted to match the data. Any previously grouped samples that are no longer within the radius will be returned to the non-clustered data [6].

In the offline case (uses all data), you can fine-tune the details of the cluster to better match the data spread. When analyzing data assigned to a group, you can remove discrepant values and improve *radii* to match the size of the cluster in each data dimension. In traditional clustering methods, densities would be recalculated using the distances between each point and all other points. In the DDC approach, each remaining sample requires only a single calculation to update its density, thus reducing the complexity of the estimation for the data-grouping model to be found. The difference for computational load becomes more apparent for large data sizes and high dimensions of variables so that they can be adapted for massive data problems [6].

To perform the data mean (μ_0), a mean of the data N is calculated recursively. By (1) the mean of all data points x_i for i, ...N being analyzed, defined as Global for all residual data or locations for the data in a cluster [6].

$$\mu_0 = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{1}$$

This is the scalar product X_o also calculated recursively by (2) is used to calculate the sample density recursively and is defined as Global for all remaining data or local for the data inside a cluster [6].

$$X_o = \frac{1}{N}\sum_{i=1}^{N} x_i^2 \tag{2}$$

The sample density D_i (3) determined recursively by applying the results of (1) and (2), is also described as Global for all remaining data or local for the data within a cluster [6].

$$D_i = \frac{1}{1 + \|x_i - \mu_0\|^2 + X_0 - \|\mu_0\|^2} \tag{3}$$

Lastly, the *Radii* learning equation used to modernize the cluster radii according to the data range is show in (Eq. 4). A value for α= 1 uses the data spread while α=0 uses the user input radii [6].

$$r_j^2 = \alpha r_0^2 + (1-\alpha)\frac{1}{N_j}\sum_{i=1}^{N} \|x_i = \mu_j\|^2 \tag{4}$$

The IDDC algorithm incremental approach performs the grouping of each new training sample data and makes a complete iteration of the DDC algorithm offline each time. The algorithm seeks to use the DDC speed to update the previous cluster results without delay associated with Subtractive Clustering. Algorithm 1 summarizes the steps to be considered for the measurement of clusters [6].

Algorithm 1 IDDC- Data density based clustering-Incremental.

1- Read first data sample and assign as first cluster center
2- Calculate the global mean, global scalar product and sample densities using (1), (2), (3)
3- Read the next data sample
4- Calculate the global mean, global scalar product and sample densities using (1), (2), (3)
5- Assign global densest point to be first cluster center
6- Copy global values to local values
7- Find all points within the cluster radii and assign to the cluster
8- Update the global mean and global scalar product using (1), (2)
9- Calculate new densities using (3)
10- Assign local densest point as cluster center
11- Assign samples within radii to the cluster
12- Remove samples from un-clustered data
13- If $\leq$ 2 samples remain to go to 8
14- While data stream continues to go to 3

3 Fuzzy Neural Network - software effort for prediction using concepts of points of use cases.

3.1 Network architecture

The fuzzy neural network described in this section is composed of three layers, moreover, derives from the work of [63]. In the first layer, fuzzification is used through the concept of IDDC model. The membership functions adopted in the first layer can be of the Gaussian type. Already in the second layer the logical neurons of the andneuron type, different from the orneuron and unineuron adopted in [5] and [55]. These neurons have weights and activation functions determined at random and through t-norms (product) and s-norms (probabilistic sum) to aggregate the neurons of the first layer. To define the weights that connect the second layer with the output layer, the concept of an extreme learning machine [8] is used to act on the neuron with a leaky ReLU activation function.

Andneuron is used to construct fuzzy neural networks in the second layer to solve pattern recognition problems and bring interpretability to the model. Figure 4 illustrates the feedforward topology of the fuzzy neural networks considered in this paper. The first layer is composed of neurons whose activation functions are membership functions of fuzzy sets defined for the input variables using IDDC. For each input variable x_{ij}, L clouds are defined A_{lj}, l = 1 ...L whose membership functions are the activation functions of the corresponding neurons. Thus, the outputs of the first layer are the membership degrees associated with the input values, i.e., $a_{jl} = \mu_l^A$ for j = 1... N and l = 1 ... L, where N is the number of inputs and L is the number of fuzzy sets for each input results by IDDC [5, 55]. The second layer is composed by L fuzzy andneurons. Each neuron performs a weighted aggregation of some of the first layer outputs. This aggregation is performed using

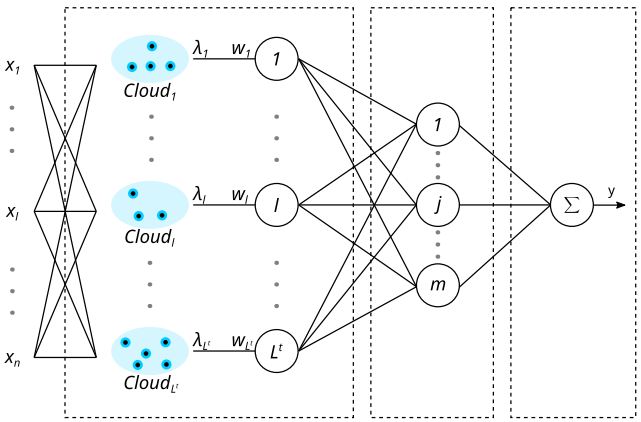

Fig. 4 FNN architecture

the weights w_{il} (for $i = 1...N$ and $l = 1...L$). For each input variable j, only one first layer output a_{jl} is defined as input of the l-th neuron. So that $\mathbf{w}$ is sparse (that is, with many weights equal to zero), each neuron of the second layer is associated with an input variable. Finally, the output layer is composed of one neuron whose activation functions are leaky ReLU [64]. The output of the model is:

$$y = \sum_{j=0}^{l} f_{LeakyReLU}(z_l v_l) \tag{5}$$

where $z_0 = 1$, v_0 is the bias, andz_j and v_j, $j = 1, ..., l$ are the output of each fuzzy neuron of the second layer and their corresponding weight, respectively. Figure 4 presents an example of FNN architecture proposed in this paper.

This is an improved function of the ReLU function [65] because a small linear component is inserted at the input of the neuron. This type of change allows small changes to be noticed and features that would be relevant to the model are not discarded. Its function is expressed by [64]:

$$f_{LeakyReLU}(x, \alpha) = max(\alpha x, x) \tag{6}$$

The logical neurons used in the second layer of the model are of the andneuron type, where the input signals are individually combined with the weights through *t-norm* and *s-norm* operators. The andneuron used in this work can be expressed as [66]:

$$z = AND(w; x) = T_{i=1}^{n}(w_i \; s \; x_i) \tag{7}$$

where T are *t-norm*, s is a *s-norm*. Fuzzy rules (Eq. 8) can be extracted from andneurons according to the following example:

$$\begin{aligned}
&Rule_1: \; If \; x_{i1} \; is \; A_1^1 \; with \; certainty \; w_{11}... \\
&and \; x_{i2} \; is \; A_1^2 \; with \; certainty \; w_{21}... \\
&Then \; y_1 \; is \; v_1 \\
&Rule_2: \; If x_{i1} \; is \; A_2^1 \; with \; certainty \; w_{12}... \\
&and \; x_{i2} \; is \; A_2^2 \; with \; certainty \; w_{22}... \\
&Then \; y_2 \; is \; v_2 \\
&Rule_3: \; If \; x_{il} \; is \; A_3^1 \; with \; certainty \; w_{13}... \\
&Then \; y_3 \; is \; v_3 \\
&Rule_4: \; If \; x_{i2} \; is \; A_3^2 \; with \; certainty \; w_{23}... \\
&Then \; y_4 \; is \; v_4
\end{aligned} \tag{8}$$

These rules allow the creation of a building base for expert systems [39].

3.2 Training fuzzy neural network

The membership functions in the first layer of the FNN are adopted as Gaussian. The number of neurons created with the input data partition is exponential between the number of membership functions and the number of features present in the problem database. The number of neurons L in the first layer is defined according to the input data and by the number of membership functions M), defined parametrically. The second layer performs the aggregation of the L neurons from the first layer through the andneurons.

After the construction of the L andneurons, the bolasso algorithm [67] is executed to select LARS (a regression algorithm for high-dimensional data that is proficient in measuring exactly the regression coefficients but also a subset of candidate regressors to be incorporated in the final model) using the most significant neurons (called L_β). The final network architecture is defined through a feature extraction technique based on l_1 regularization and resampling. The learning algorithm assumes that the output second layer composed of the candidate neurons can be written as [55]:

$$f(x_i) = \sum_{i=0}^{L_\beta} v_i z_i(x_i) = z(x_i)v \tag{9}$$

where $\mathbf{v} = [v_0, v_1, v_2, ..., v_{L\beta}]$ is the weight vector of the output layer and $\mathbf{z}\ (x_i) = [z_0, z_1(x_i), z_2(x_i)...z_{L\beta}(x_i)\ ]$ the output vector of the second layer, for $z_0 = 1$. In this context, $\mathbf{z}\ (x_i)$ is considered as the non-linear mapping of the input space for a space of fuzzy characteristics of dimension L_β [55]. The non-linear mapping allows the model to determine the predictability relationship between the andneuron neurons and the outputs of the second layer.

The LARS algorithm can be used to perform the model selection since for a given value of λ only a fraction (or none) of the regressors have corresponding nonzero weights. If $\lambda = 0$, the problem becomes unrestricted regression, and all weights are nonzero. As λ_{max} increases from 0 to a given valueλ_{max}, the number of nonzero weights decreases to zero. For the problem considered in this paper, the z_l regressors are the outputs of the significant neurons. [67]. Bolasso procedure is summarized in Algorithm 2.

Algorithm 2 Bolasso- bootstrap-enhanced least absolute shrinkage operator.

(b1) Let n be the number of examples, (lines) in X:
(b2) Show n examples of (**X, Y**), uniformly and with substitution, called here ($Xsamp$, $Ysamp$).
(b3) Determine which weights are nonzero given a λ value.
(b4) Repeat steps b1: b3 for a specified number of bootstraps bt.
(b5) Take the intersection of the non-zero weights indexes of all bootstrap replications. Select the resulting variables.
(b6) Revise using the variables selected via non-regularized least squares regression (if requested).
(b7) Repeat the procedure for each value of bt bootstraps and λ (actually done more efficiently by collecting interim results).
(b8) Determine "optimal" values for λ and bt.

Subsequently, following the determination of the network topology, the predictions of the evaluation of the vector of weights' output layer are performed. In this paper, this vector is considered by the Moore-Penrose pseudo Inverse [55]:

$$\mathbf{v} = \mathbf{Z}^{+}\mathbf{y} \tag{10}$$

Z^{+} is the Moore-Penrose pseudo Inverse of z, which is the minimum norm of the least squares solution for the output weights. synthesized as demonstrated in Algorithm 3. It has three parameters:

1- the number of grid size, ρ;
2- the number of bootstrap replications, bt;
3- the consensus threshold, λ.

4 Regression models of use case point problems

4.1 Assumptions and initial test configurations

The tests performed in this paper seek to find a predictor model for the definition of the effort in hours for the construction of software using the fuzzy neural network. The accuracy of the training and the test of the model will be realized by checking the values obtained by the model, comparing them with the expected result. In this context, they are evaluated through the root mean square error (RMSE). The formula for defining your calculation is shown below:

$$RMSE = \frac{1}{N}\left(\sum_{k=0}^{n} y^{k} - y'^{k}\right)^{\frac{1}{2}} \tag{11}$$

, where y^{k} is the response provided by the model and y'^{k} is the expected output for the test in question. To perform the training, 30 repetitions were conducted with the samples made available through the software construction effort study. The percentage is defined as 70% of the samples allocated for training and the remaining 30% for the test phase of the model. To avoid trends in the characteristics of each of the examples, a proposal was made where all the samples destined to the training and testing of the fuzzy neural network were randomly sampled. This ensures that there will be no dependencies of the data stream for the model results. All samples involved in the test were normalized with mean zero and variance 1. The activation functions of the third layer neuron are of the leaky ReLU type. The values of the bootstrap replicate, the decision consensus for the use of the bolasso are, respectively, bt=16 and λ= 0.7. These values were found to be optimal values using a 10 k-fold technique in preliminary tests. For fuzzy neural networks using Gaussian membership functions, a number of grid sizes (ρ) it was defined as 3 and 5 by the same 10-k-fold process that the configuration parameters of the bolasso method were found.

Algorithm 3 Forecasting effort - FNN training.

(1) Define the number of grid sizes, ρ.
(2) Define bootstrap replications, bt.
(3) Define consensus threshold, λ
(4) Calculate L neurons in the first layer using IDDC.
(5) Construct L fuzzy neurons with Gaussian membership functions constructed with center derived from IDDC and sigma values defined randomly.
(6) Define the weights and bias of the fuzzy neurons randomly.
(7) Construct L andneurons with random weights and bias on the second layer of the network by welding the L fuzzy neurons of the first layer.
(8) **For all** K input do
(8.1) Calculate the mapping $z_k(x_k)$ using andneurons **end for**
(9) Select significant L_β using the lasso bootstrap according to the settings of bt and λ.
(10) Estimate the weights (**v**) of the output layer (5)
(11) Calculate output **y** using leaky ReLU. (6)

4.2 Database used in the tests

Data set was collected by [4] from three software companies that provided the data (D1, D2, and D3) and is based on the following problem areas: Insurance, Government, Banks, and other domains. This database provides data such as the methodology used in software production, the complexity of weights for cases of uses and actors, and other metrics relevant to the evaluation of efforts.

The previous UCP effort estimation models were tested over a limited number of projects, thereby reducing the credibility of the models. To avoid this pitfall, the dataset used in this study is a combination from 3 datasets: [68], [22], and [4]. All datasets share the same set of features that facilitate the process of merging them into one large dataset. The first dataset came from Ochodek et al. [68], which contains 14 projects, which were developed in companies targeted to business. Some of the projects were developed from scratch, while others were customized for new customers. The second dataset came from a company which contains 110 projects developed from information systems projects such as chains of hotels, multibranch universities, and multiwarehouse book stores.11 The architectures used to develop these projects are 2-tier desktop application and 3-tier Web architecture. The programming languages used in this company are object-oriented languages such

as Java, C++, and PHP5. The third dataset came from Silhavy et al. [4], which contains 71 projects. The projects were developed for different government, health, and business sectors; most of them were written by Java and C# programming languages. The whole dataset has been preprocessed using boxplot to exclude outliers based on the effort variable. We found nine projects that are categorized as an outlier based on effort variable; therefore, we excluded them, which resulted in 186 projects in the final dataset [35].

To be applied in the fuzzy neural network, the columns referring to the textual information were transformed into numerical values. The principal dimensions used in this evaluation are donator, simple actor methodology, average actors, complex actors, actor weight, simple use case, medium use case, complicated use case, use case weight, technical complexity, the real effort 20 hours. The methodology was defined as 0 for traditional methods and 1 for agile methodologies.

For the programming languages used were used numbers from 1 to 5 according to their complexity. The same occurred with the areas where software was developed. One approach has transformed the areas of operation of systems concerning their complications. The two methods were made with specialists.

4.3 Prediction tests on efforts in software construction

Table 1 presents the results of the model proposed in this paper.

After the tests, it was verified that the model behaves very efficiently in the prediction of efforts related to software construction. In the comparison between the two models, the one that uses 3 Gaussian membership functions obtained the best result with which it used 5 Gaussian membership functions. Different from the study by [4] that evaluated with subgroups of characteristics, this work performed an efficient evaluation with the group of all the characteristics involved.

This approach worked on the final network with an adequate number of neurons.

The Figs. 5 and 6 show the results of the prediction performed by the fuzzy neural network in train and test. This example demonstrates through graphs, relevant characteristics of the answers obtained by the model In Fig. 5 results of training. Figure 6 the results of tests.

Table 1 Accuracies of the FNN

ρ	L	L_β	RMSE Train.	RMSE Test
3	100.00 (0)	27.83 (18.62)	156.85 (28.42)	**146.13 (62.12)**
5	100.00 (0)	33.42 (17.92)	161.28 (21.12)	149.18 (75.01)

Bold entries signify the best test results

In the second test of the fuzzy neural network model, the approaches proposed in this paper were compared with approximation models commonly used in the literature. To do this, the weka [69] software was used to perform the tests using the Multilayer Perceptron (MLP) [70] model and the Linear Regression (LIN-R) [71] model. The configurations used in the two models follow those established by Weka, differing only by the initial number of neurons, which is the same as that used in the whole paper.

To compare with fuzzy neural network models, the model proposed in [55] was chosen. It differs from the model proposed in this paper because of its first layer that uses the Anfis concept [72] to generate equally spaced Gaussian membership functions (A-FNN). The model proposed in [5] uses a regularized fuzzy neural network structure based on Or neurons and linear activation function (OR-FNN). Representing the models based on organic chemistry, we used the model proposed by [73] called Artificial Hydrocarbon network (AHN). The model has presented percussive results in the prediction and approximation of values in linear and non-linear functions [74–76]. Its structure differs from regular models of artificial neural networks because it is composed of carbon and hydrogen molecules. Further details on the model can be found at [73]. In the tests, the values of membership functions (M) for A-FNN and OR-FNN and number of molecules of AHN are the same as those used as grid size ρ by the model proposed in this paper. The bootstrap and decision consensus values for the method in the models A-FNN and OR-FNN were also the same to which the model proposed in this article was submitted.

A-FNN and OR-FNN were developed in Matlab. AHN was used in the R language. The code was obtained from the address of the references.[1]

For the regression tests about effort estimation, the CARET package [77] for R was used. In this experiment, we compared the model proposed in this paper, as well with other machine learning methods: the k-nearest neighbor regression (KNNR) with a value of $k = 3$; the independent component regression (ICR) with $c.comp$=2; and, artificial neural networks with a principal component step (NPC) with $size = 10$ and attributes $linout = false$

It is noted that the data density approach performed better than the work of [37] when compared to the model that uses equally spaced membership functions through Anfis [72]. Therefore, the data cloud approach improves the predictability of the model, making the partitioning of the input space more coherent with the reality of the evaluated problem.

[1] https://github.com/jroberayalas/ahnr

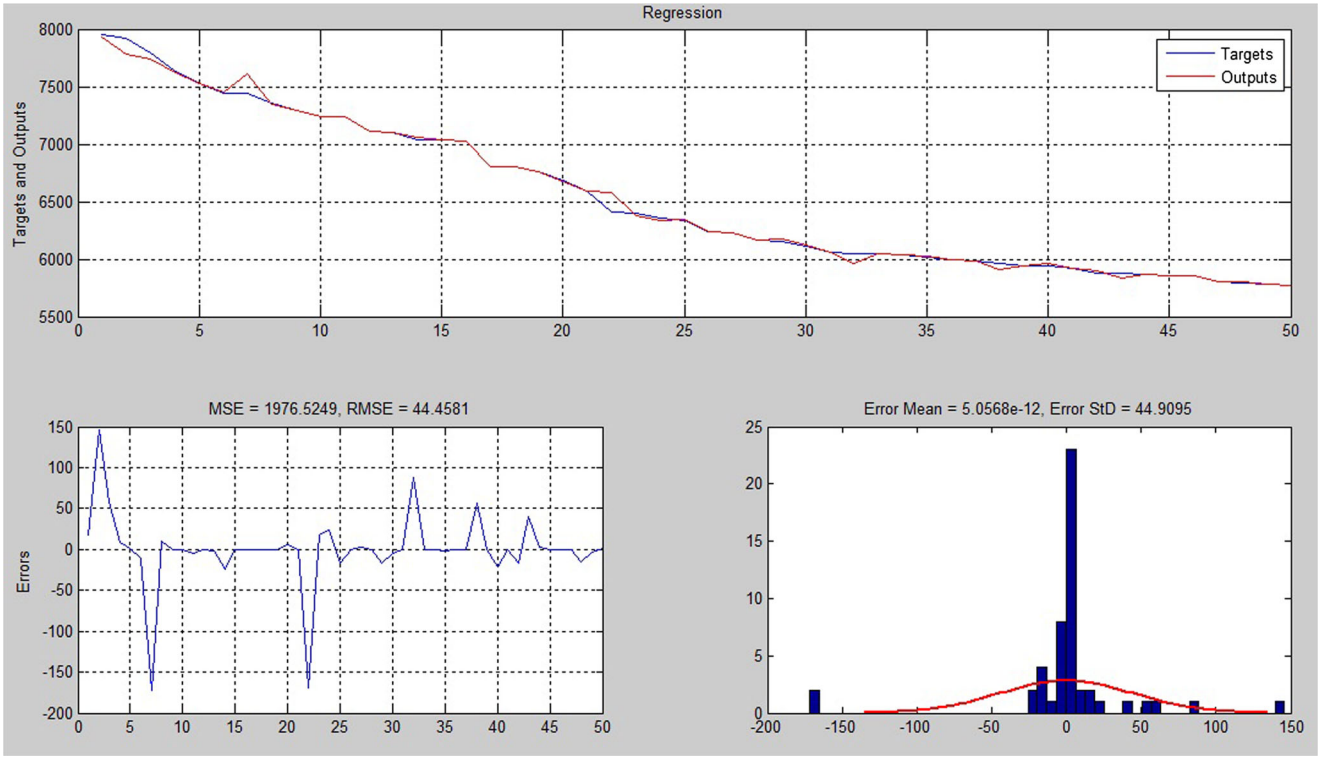

Fig. 5 Predict train results

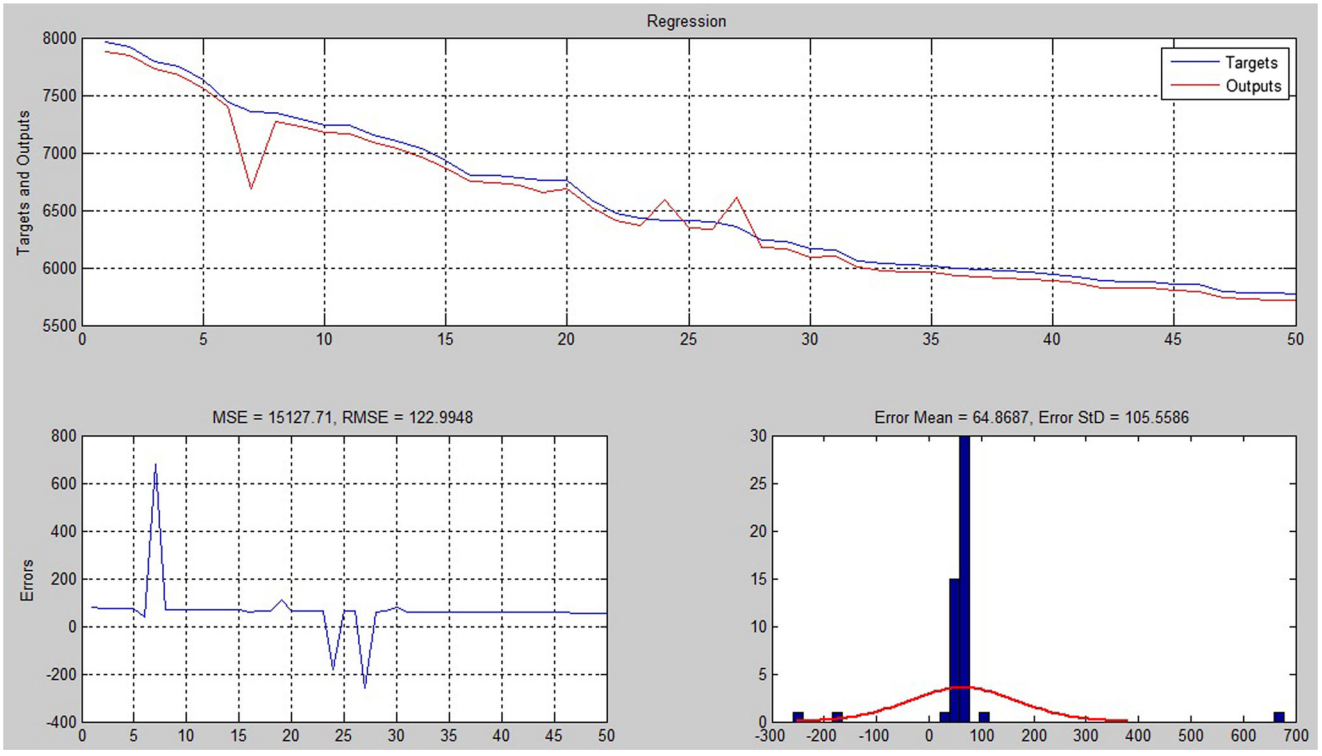

Fig. 6 Predict Test Results

We can verify that in a matter of execution time artificial neural networks achieve excellent performance, however, on their answers can not say the same. In comparison with fuzzy neural network models, the time was significantly reduced when we compare the proposals that use functions of pertinence with the proposal in this paper. The hydrocarbon-based regression model maintained good results but was no better than fuzzy neural networks.

Another factor observed is that the approach proposed in this paper obtained better test results when compared to traditional models in the literature.

4.4 Interpretability of the problem based on fuzzy rules

The following rules are examples of rules with the bases used. Each membership function can receive values to be consistent with the analyzed context. The system generated numerous rules as can be confirmed in Table 2, but here some relationships were highlighted to demonstrate the viability of the technique.

1. If (Donator is simple) and (Methodology is structured) and (SimpleActor is small) and (MediumActor is small) and (Complexactor is small) and (WeightActor is high) and (usecaseSimple is large) and (usecaseMedium is small) and (usecaseComplex is large) and (WeightUseCase is high) and (Technical complexity is low) and (complexity factors is low) and (realeffort20hours is high) with certainty −0.0442 then (effort is 1256.54)
2. If (Donator is simple) and (Methodology is structured) and (SimpleActor is small) and (MediumActor is small) and (Complexactor is small) and (WeightActor is high) and (usecaseSimple is large) and (usecaseMedium is medium) and (usecaseComplex is large) and (WeightUseCase is low) and (Technical complexity is low) and (complexity factors is low) and (realeffort20hours is high) with certainty 1.0365 then (effort is 568.21)
3. If (Donator is medium) and (Methodology is traditional) and (SimpleActor is medium) and (MediumActor is medium) and (Complexactor is small) and (WeightActor is high) and (usecaseSimple is large) and (usecaseMedium is small) and (usecaseComplex is large) and (WeightUseCase is high) and (Techinicalcomplexity is medium) and (complexityfactors is medium) and (realeffort20hours is high) with certainty -9.3365 then (effort is 2854.74)
4. If (Donator is medium) and (Methodology is traditional) and (SimpleActor is medium) and (MediumActor is medium) and (Complexactor is small) and (WeightActor is high) and (usecaseSimple is large) and (usecaseMedium is small) and (usecaseComplex is large) and (WeightUseCase is high) and (Techinicalcomplexity is low) and (complexityfactors is high) and (realeffort20hours is high) with certainty 2.2125 then (effort is 2985.66)
5. If (Donator is complex) and (Methodology is Agile) and (SimpleActor is large) and (MediumActor is medium) and (Complexactor is small) and (WeightActor is high) and (usecaseSimple is large) and (usecaseMedium is big) and (usecaseComplex is large) and (WeightUseCase is high) and (Techinicalcomplexity is low) and (complexityfactors is medium) and (realeffort20hours is high) with certainty −0.0117 then (effort is 986.74)
6. If (Donator is complex) and (Methodology is Agile) and (SimpleActor is large) and (MediumActor is small) and (Complexactor is small) and (WeightActor is low) and (usecaseSimple is large) and (usecaseMedium is big) and (usecaseComplex is small) and (WeightUseCase is high) and (Technical complexity is medium) and (complexity factors is high) and (realeffort20hours is low) with certainty −0.2482 then (effort is 1126.53)

Table 2 Accuracies of the FNN and another methods

Model	L	L_s	RMSE Train.	RMSE Test	time[2]
$FNN_{\beta 3}$	100.00 (0)	27.83 (18.62)	156.85 (18.42)	**145.13 (35.55)**	19.16 (2.55)
$FNN_{\beta 5}$	100.00 (0)	23.17 (8.99)	176.14 (2.29)	165.18 (25.89)	29.14 (5.69)
LIN-R	100.00 (0)	100.00 (0)	168.25 (14.29)	152.85 (27.18)	14.11 (1.15)
MLP	100.00 (0)	100.00 (0)	178.28 (16.71)	162.44 (55.13)	39.41 (2.57)
A-FNN	100.00 (0)	32.17 (6.57)	**59.98 (21.32)**	206.48 (29.13)	49.25 (9.65)
OR-FNN	100.00 (0)	39.54 (5.58)	178.28 (19.25)	199.54 (71.24)	36.77 (6.98)
AHN	5.0 (0)	5.0 (0)	196.39 (31.04)	182.11 (14.18)	16.22 (2.21)
KNNR	-	-	194.28 (16.91)	162.44 (55.13)	**0.09 (0.01)**
ICR	-	-	305.26 (84.21)	209.44 (12.17)	21.52 (2.25)
NPC	-	-	252.29 (96.71)	175.44 (42.29)	0.18 (0.03)

Bold entries signify the best test results

5 Conclusion

We can conclude that the approach using data density significantly reduces the execution time of fuzzy neural networks, maintains the ability to approximate the prediction functions of software efforts and at the same time can generate fuzzy rules to aid in the construction of expert systems. The models and tests prove that the approach is feasible to help managers and information technology workers to develop their applications with the predictability of effort closer to the real.

These creative modeling system elements allow managers, ordinary people, and developers to estimate more accurately or even have an exit point to define the amount of time to design software.

Of course, the prior knowledge about the qualitative characteristics of the factors involved in the process is that they determine the assertiveness of the model.

Other tests can be carried out to build more compact specialist systems so that even the construction of specialist systems is more consistent with the routine of the companies. Feature selection techniques to filter out many correlated elements in the database, other models and other algorithms such as the evolving version of the data density algorithm can foster the continuity of this work.

Acknowledgments The thanks of this work are destined to CEFET-MG and UNA.

Compliance with Ethical Standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. Krusche S, Scharlau B, Cajander Å, Hughes J (2018) 50 years of software engineering: challenges, results, and opportunities in its education. In: Proceedings of the 23rd annual ACM conference on innovation and technology in computer science education. ACM, pp 362–363
2. Ghezzi C, Jazayeri M, Mandrioli D (2002) Fundamentals of software engineering. Prentice Hall PTR, Englewood Cliffs
3. Harman M (2007) The current state and future of search based software engineering. In: 2007 future of software engineering. IEEE Computer Society, pp 342–357
4. Silhavy R, Silhavy P, Prokopova Z (2017) Analysis and selection of a regression model for the use case points method using a stepwise approach. J Syst Softw 125:1–14
5. de Campos Souza PV, Torres LCB (2018) Regularized fuzzy neural network based on or neuron for time series forecasting. In: Barreto GA, Coelho R (eds) Fuzzy information processing. Springer International Publishing, Cham, pp 13–23
6. Hyde R, Angelov P (2014) Data density based clustering. In: 2014 14th UK workshop on computational intelligence (UKCI), pp 1–7
7. Takagi T, Sugeno M (1983) Derivation of fuzzy control rules from human operator's control actions. IFAC Proc 16(13):55–60
8. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1-3):489–501
9. Melnik G, Maurer F (2006) Comparative analysis of job satisfaction in agile and non-agile software development teams. In: International conference on extreme programming and agile processes in software engineering. Springer, pp 32–42
10. El Emam K, Koru AG (2008) A replicated survey of it software project failures. IEEE Softw 25(5):84–90
11. Humphrey WS (1995) A discipline for software engineering. Addison-wesley Longman Publishing Co. Inc, Reading
12. Mall R (2018) Fundamentals of software engineering. PHI Learning Pvt Ltd.
13. Dejaeger K, Verbeke W, Martens D, Baesens B (2012) Data mining techniques for software effort estimation: a comparative study. IEEE Trans Softw Eng 38(2):375–397
14. Shepperd M, Schofield C (1997) Estimating software project effort using analogies. IEEE Trans Softw Eng 23(11):736–743
15. Huang S-J, Chiu N-H (2006) Optimization of analogy weights by genetic algorithm for software effort estimation. Inf Softw Technol 48(11):1034–1045
16. Li J, Ruhe G, Al-Emran A, Richter MM (2007) A flexible method for software effort estimation by analogy. Empir Softw Eng 12(1):65–106
17. Finnie GR, Wittig GE, Desharnais J-M (1997) A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. J Syst Softw 39(3):281–289
18. Park H, Baek S (2008) An empirical validation of a neural network model for software effort estimation. Expert Syst Appl 35(3):929–937
19. Nageswaran S (2001) Test effort estimation using use case points. Qual Week 6:1–6
20. Vazquez CE, Simões GS, Albert RM (2003) Análise de pontos de função: medição, estimativas e gerenciamento de projetos de software, vol 3. Editora Érica, São Paulo
21. Karner G (1993) Resource estimation for objectory projects, vol 17. Objective Systems SF AB
22. Nassif AB, Capretz LF, Ho D (2011) Estimating software effort based on use case point model using sugeno fuzzy inference system. In: 2011 23rd IEEE international conference on tools with artificial intelligence (ICTAI). IEEE, pp 393–398
23. Nassif AB, Ho D, Capretz LF (2013) Towards an early software estimation using log-linear regression and a multilayer perceptron model. J Syst Softw 86(1):144–160
24. Nassif AB, Capretz LF, Ho D (2012) Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model. In: 2012 13th ACIS international conference on software engineering, artificial intelligence, networking and parallel & distributed computing (SNPD 2012). IEEE, pp 589–594
25. Azzeh M, Nassif AB (2016) A hybrid model for estimating software project effort from use case points. Appl Soft Comput 49:981–989
26. Silhavy R, Silhavy P, Prokopova Z (2015) Algorithmic optimisation method for improving use case points estimation. PloS One 10(11):e0141887
27. Satapathy SM (2016) Effort estimation methods in software development using machine learning algorithms. Ph.D. Dissertation, National Institute of Technology Rourkela
28. Urbanek T, Prokopova Z, Silhavy R, Kuncar A (2016) Using analytical programming for software effort estimation. In: Software engineering perspectives and application in intelligent systems. Springer, pp 261–272
29. Prokopová Z, Silhavy R, Silhavy P (2017) The effects of clustering to software size estimation for the use case points methods. In: Computer science on-line conference. Springer, pp 479–490

30. Fellir F, Nafil K, Touahni R, Chung L (2018) Improving case based software effort estimation using a multi-criteria decision technique. In: Computer science on-line conference. Springer, pp 438–451
31. Azzeh M, Nassif AB (2018) Project productivity evaluation in early software effort estimation. J Softw Evol Process 30:e2110
32. Azzeh M, Nassif AB, Banitaan S (2017) Comparative analysis of soft computing techniques for predicting software effort based use case points. IET Softw 12(1):19–29
33. Bagheri S, Shameli-Sendi A (2018) Software project estimation using improved use case point. In: 2018 IEEE 16th international conference on software engineering research, management and applications (SERA). IEEE, pp 143–150
34. Silhavy P, Silhavy R, Prokopova Z (2018) Stepwise regression clustering method in function points estimation. In: Proceedings of the computational methods in systems and software. Springer, pp 333–340
35. Azzeh M, Nassif AB (2018) Project productivity evaluation in early software effort estimation. J Soft Evol Process 0(0):e2110. JSME-18-0078.R2. [Online]. Available: https://doi.org/10.1002/smr.2110
36. Azzeh M (2017) Analyzing the relationship between project productivity and environment factors in the use case points method. J Softw: Evol Process 29(9):e1882. e1882 JSME-15-0206.R2. [Online]. Available: https://doi.org/10.1002/smr.1882
37. Souza PVdC, Guimaraes AJ, Araujo VS, Rezende TS, Araujo VJS (2018) Regularized fuzzy neural networks to aid effort forecasting in the construction and software development. Int J Artif Intell Appl 9(6):13–26
38. Pedrycz W, Gomide F (2007) Fuzzy systems engineering: toward human-centric computing. Wiley, New York
39. Caminhas WM, Tavares H, Gomide FA, Pedrycz W (1999) Fuzzy set based neural networks: structure, learning and application. JACIII 3(3):151–157
40. de Campos Souza PV, Guimaraes AJ (2018) Using fuzzy neural networks for improving the prediction of children with autism through mobile devices. In: IEEE symposium on computers and communications (ISCC). IEEE, pp 01 086–01 089
41. Guimarães AJ, Araújo VJ, de Oliveira Batista L, Souza PVC, Araújo V, Rezende TS (2018) Using fuzzy neural networks to improve prediction of expert systems for detection of breast cancer. In: Anais do XV Encontro Nacional de Inteligência Artificial e Computacional. SBC, Porto Alegre, RS, Brasil, pp 799–810. [Online]. Available: http://portaldeconteudo.sbc.org.br/index.php/eniac/article/view/4468
42. Kasabov NK, Song Q (2002) Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans Fuzzy Syst 10(2):144–154
43. Gao Y, Er MJ (2005) Narmax time series model prediction: feedforward and recurrent fuzzy neural network approaches. Fuzzy Sets Syst 150(2):331–350
44. Chatterjee A, Pulasinghe K, Watanabe K, Izumi K (2005) A particle-swarm-optimized fuzzy-neural network for voice-controlled robot systems. IEEE Trans Ind Electron 52(6):1478–1489
45. Guimarães AJ, Araujo VJS, de Campos Souza PV, Araujo VS, Rezende TS (2018) Using fuzzy neural networks to the prediction of improvement in expert systems for treatment of immunotherapy. In: Ibero-American conference on artificial intelligence. Springer, pp 229–240
46. Ramirez-Rodriguez C, Hernandez-Silveira M (2001) Multi-thread implementation of a fuzzy neural network for automatic ecg arrhythmia detection. In: Computers in cardiology 2001. IEEE, pp 297–300
47. Lim JS (2009) Finding features for real-time premature ventricular contraction detection using a fuzzy neural network system. IEEE Trans Neural Netw 20(3):522–527
48. Ma X, Jin Y, Dong Q (2017) A generalized dynamic fuzzy neural network based on singular spectrum analysis optimized by brain storm optimization for short-term wind speed forecasting. Appl Soft Comput 54:296–312
49. Chang F-J, Chang K-Y, Chang L-C (2008) Counterpropagation fuzzy-neural network for city flood control system. J Hydrol 358(1-2):24–34
50. de Campos Souza PV, Guimarães AJ (2018) Detecção de pulsares utilizando redes neurais nebulosas baseadas em uninormas. In: Quinto Congresso Brasileiro de Sistemas Fuzzy, vol 1. Sociedade Brasileira de Matemática Aplicada e Computacional, pp 41–52
51. Lam K, Hu T, Thomas Ng S, Skitmore M, Cheung S (2001) A fuzzy neural network approach for contractor prequalification. Constr Manag Econ 19(2):175–188
52. Vitor de Campos Souza P (2018) Pruning fuzzy neural networks based on unineuron for problems of classification of patterns. J Intell Fuzzy Syst 35(2):2597–2605
53. de Campos Souza PV, Torres LCB, Guimaraes AJ, Araujo VS, Araujo VJS, Rezende TS (2019) Data density-based clustering for regularized fuzzy neural networks based on nullneurons and robust activation function. Soft Comput 1–15, preprint
54. Li R-P, Mukaidono M, Turksen IB (2002) A fuzzy neural network for pattern classification and feature selection. Fuzzy Sets Syst 130(1):101–108
55. de Campos Souza PV, Guimaraes AJ, Araújo VS, Rezende TS, Araújo VJS (2018) Fuzzy neural networks based on fuzzy logic neurons regularized by resampling techniques and regularization theory for regression problems. Intel Artif 21(62):114–133
56. Demertzis K, Iliadis L (2015) A bio-inspired hybrid artificial intelligence framework for cyber security. In: Computation, cryptography, and network security. Springer, pp 161–193
57. Demertzis K (2013) A hybrid network anomaly and intrusion detection approach based on evolving spiking neural network classification. In: International conference on e-democracy. Springer, pp 11–23
58. Batista LO, de Silva GA, Araújo VS, Araújo VJS, Rezende TS, Junio A, Guimarães PVdCS (2018) Utilização de redes neurais nebulosas para criação de um sistema especialista em invasões cibernéticas. In: The tenth international conference on forensic computer science and CYBER LAW-ICOFCS 2018, vol 10. BrasÍLia Chapter of the High Technology Crime Investigation Association (HTCIA), pp 12–22
59. Batista LO, de Silva GA, Araújo VS, Araújo VJS, Rezende TS, Guimarães AJ, Souza PVdC (2019) Fuzzy neural networks to create an expert system for detecting attacks by sql injection. Int J Forensic Comput Sci 13(1):8–21
60. Bezdek JC, Ehrlich R, Full W (1984) Fcm: the fuzzy c-means clustering algorithm. Comput Geosci 10(2-3):191–203
61. Szabo A, de França FO (2015) The proposal of dynamic thresholds in an immune algorithm for fuzzy clustering. In: IEEE international conference on fuzzy systems (FUZZ-IEEE) 2015. IEEE, pp 1–8
62. Angelov P (2014) Anomaly detection based on eccentricity analysis. In: 2014 IEEE symposium on evolving and autonomous learning systems (EALS). IEEE, pp 1–8
63. Souza PVC (2018) Regularized fuzzy neural networks for pattern classification problems. Int J Appl Eng Res 13(5):2985–2991
64. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. Proc ICML 30:3

65. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
66. Pedrycz W (1991) Neurocomputations in relational systems. IEEE Trans Pattern Anal Mach Intell 13(3):289–297
67. Bach FR (2008) Bolasso: model consistent lasso estimation through the bootstrap. In: Proceedings of the 25th international conference on machine learning. ACM, pp 33–40
68. Ochodek M, Nawrocki J, Kwarciak K (2011) Simplifying effort estimation based on use case points. Inf Softw Technol 53(3):200–213
69. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. ACM SIGKDD Explor Newslett 11(1):10–18
70. Ruck DW, Rogers SK, Kabrisky M, Oxley ME, Suter BW (1990) The multilayer perceptron as an approximation to a bayes optimal discriminant function. IEEE Trans Neural Netw 1(4):296–298
71. Neter J, Wasserman W, Kutner MH (1989) Applied linear regression models. Irwin Homewood, IL
72. Jang J-S (1993) Anfis: adaptive-network-based fuzzy inference system. IEEE Trans Syst Man Cybern 23(3):665–685
73. Ponce-Espinosa H, Ponce-Cruz P, Molina A (2013) Artificial organic networks: artificial intelligence based on carbon networks, vol 521. Springer
74. Ponce H, González-Mora G., Martínez-Villaseñor L (2018) A reinforcement learning method for continuous domains using artificial hydrocarbon networks. In: International joint conference on neural networks (IJCNN). IEEE, pp 1–6
75. Ponce H, Ponce P, Molina A (2014) Adaptive noise filtering based on artificial hydrocarbon networks: an application to audio signals. Expert Syst Appl 41(14):6512–6523
76. Ponce H, Martínez-Villaseñor MdL, Miralles-Pechuán L (2016) A novel wearable sensor-based human activity recognition approach using artificial hydrocarbon networks. Sensors 16(7):1033
77. Kuhn M (2008) Building predictive models in r using the caret package. J Stat Softw Artic 28(5):1–26