Knowledge-Based Systems 199 (2020) 105972

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Interpretable neural networks based on continuous-valued logic and multicriteria decision operators

Orsolya Csiszár [a,b,*], Gábor Csiszár [c], József Dombi [d]

[a] Faculty of Basic Sciences, University of Applied Sciences Esslingen, Esslingen, Germany
[b] Institute of Applied Mathematics, Óbuda University, Budapest, Hungary
[c] Institute of Materials Physics, University of Stuttgart, Stuttgart, Germany
[d] Institute of Informatics, University of Szeged, Szeged, Hungary

ARTICLE INFO

Article history:
Received 9 February 2020
Received in revised form 20 April 2020
Accepted 23 April 2020
Available online 28 April 2020

Keywords:
Explainable artificial intelligence
Continuous logic
Nilpotent logic
Neural network
Adversarial problems

ABSTRACT

Combining neural networks with continuous logic and multicriteria decision-making tools can reduce the black-box nature of neural models. In this study, we show that nilpotent logical systems offer an appropriate mathematical framework for hybridization of continuous nilpotent logic and neural models, helping to improve the interpretability and safety of machine learning. In our concept, perceptrons model soft inequalities; namely membership functions and continuous logical operators. We design the network architecture before training, using continuous logical operators and multicriteria decision tools with given weights working in the hidden layers. Designing the structure appropriately leads to a drastic reduction in the number of parameters to be learned. The theoretical basis offers a straightforward choice of activation functions (the cutting function or its differentiable approximation, the squashing function), and also suggests an explanation to the great success of the rectified linear unit (ReLU). In this study, we focus on the architecture of a hybrid model and introduce the building blocks for future applications in deep neural networks.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

AI techniques, especially deep learning models are revolutionizing the business and technology world. One of the greatest challenges is the increasing need to address the problem of interpretability and to improve model transparency, performance, and safety. Although deep neural networks have achieved impressive experimental results e.g. in image classification, they may surprisingly be unstable when it comes to adversarial perturbations, that is, minimal changes to the input image that cause the network to misclassify it [1–4]. Interpretability becomes more and more important these days when it comes to predictive modeling. In a high-risk environment it is inevitable to know why a decision was made, it is not enough to know that the predictive performance on a test dataset was good. This knowledge can help you learn more about the problem, the data and the reason why a model might fail. In black-box models, less is known about what influencing variables are actually driving the final decision. The relationship between the input and output is often limited in complexity and local interpretations. White-box models such as linear regression and decision trees are significantly easier to explain and interpret, on the other hand, provide less predictive capacity and are not always capable of modeling the inherent complexity of the dataset (i.e. feature interactions).

Combining deep neural networks with structured logical rules and multicriteria decision tools, where logical operators are applied on clusters created in the first layer, contributes to the reduction of the black-box nature of neural models. Aiming at interpretability, transparency and safety, implementing continuous-valued logical operators offers a promising direction.

Although Boolean units and multilayer perceptrons have a long history, to the best of our knowledge there has been little attempt to combine neural networks with continuous logical systems so far. The basic idea of continuous logic is the replacement of the space of truth values $\{T, F\}$ by a compact interval such as $[0, 1]$. This means that the inputs and the outputs of the extended logical gates are real numbers of the unit interval, representing truth values of inequalities. Quantifiers $\forall x$ and $\exists x$ are replaced by $\sup_x$ and $\inf_x$, and logical connectives are continuous functions. Based on this idea, human thinking and natural language can be modeled in a sophisticated way.

Among other families of many-valued logics, t-norm fuzzy logics are broadly used in applied fuzzy logic and fuzzy set theory as a theoretical basis for approximate reasoning. In fuzzy logic,

* Corresponding author at: Institute of Applied Mathematics, Óbuda University, Budapest, Hungary.
E-mail addresses: csiszar.orsolya@nik.uni-obuda.hu (O. Csiszár), Gabor.Csiszar@mp.imw.uni-stuttgart.de (G. Csiszár), dombi@inf.u-szeged.hu (J. Dombi).

https://doi.org/10.1016/j.knosys.2020.105972
0950-7051/© 2020 Elsevier B.V. All rights reserved.

the membership function of a fuzzy set represents the degree of truth as a generalization of the indicator function in classical sets. Both propositional and first-order (or higher-order) t-norm fuzzy logics, as well as their expansions by modal and other operators, have been studied thoroughly. Important examples of t-norm fuzzy logics are monoidal t-norm logic of all left-continuous t-norms, the basic logic of all continuous t-norms, product fuzzy logic of the product t-norm, or the nilpotent minimum logic of the nilpotent minimum t-norm. Some independently motivated logics belong among t-norm fuzzy logics as well, like Łukasiewicz logic (which is the logic of the Łukasiewicz t-norm) and Gödel–Dummett logic (which is the logic of the minimum t-norm).

Recent results [5–10] show that in the field of continuous logic, nilpotent logical systems are the most suitable for neural computation, mainly because of their bounded generator functions. Moreover, among other preferable properties, the fulfillment of the law of contradiction and the excluded middle, and the coincidence of the residual and the S-implication [11,12] also make the application of nilpotent operators in logical systems promising. In [5–10] a rich asset of operators was examined thoroughly: in [5], negations, conjunctions and disjunctions, in [7] implications, and in [6] equivalence operators. In [8], a parametric form of a general operator o_ν was given by using a shifting transformation of the generator function. Varying the parameters, nilpotent conjunctive, disjunctive, aggregative (where a high input can compensate for a lower one) and negation operators can all be obtained. Moreover, as it was shown in [9], membership functions, which play a substantial role in the overall performance of fuzzy representation, can also be defined by means of a generator function.

In this study, we introduce a nilpotent neural model, where nilpotent logical operators and multicriteria decision tools are implemented in the hidden layers of neural networks (see Fig. 3). Only the weights of the first layer (parameters of hyperplanes separating the decision space) are to be learned, and the architecture needs to be designed. In a more sophisticated version, left for future work, the type of the operators in the hidden layers can also be learned by the network, or e.g. by a genetic algorithm. Moreover, in [9] the authors showed that the most important logical operators can be expressed as a composition of a parametric unary operator and the arithmetic mean. This means that the neural network only needs to learn the parameters of the first layer and (if not initially given) the parameters of these unary operators in the hidden layers.

In the nilpotent neural model, the activation functions in the first layer are membership functions representing truth values of inequalities, normalizing the inputs. At the same time, the activation functions in the hidden layers model the cutting function (or to avoid the vanishing gradient problem, its differentiable approximation, the so-called squashing function) in the nilpotent logical operators. The theoretical background offers a straightforward choice of activation functions: the squashing function, which is an approximation of the rectifier. The fact that the squashing function, in contrast to the rectifier, is bounded from above, makes the continuous logical concept applicable.

The article is organized as follows. After summarizing the most important related work in Section 2, we revisit the relevant preliminaries concerning nilpotent logical systems in Section 3. The nilpotent neural concept is described in Section 4. In Section 5, the model is illustrated with some extended tensorflow playground examples. Finally, the main results are summarized in Section 6.

2. Related work

Combinations of neural networks and logic rules have been considered in different contexts. Neuro-fuzzy systems [13] were examined thoroughly in the literature. These hybrid intelligent systems synergize the human-like reasoning style of fuzzy systems with the learning structure of neural networks through the use of fuzzy sets and a linguistic model consisting of a set of IF-THEN fuzzy rules. These models were the first attempts to combine continuous logical elements and neural computation.

KBANN [14], Neural-symbolic systems [15], such as CILP++ [16], constructed network architectures from given rules to perform knowledge acquisition. Kulkarni et al. [17] used a specialized training procedure to obtain an interpretable neural layer of an image network. In [18], Hu et al. proposed a general framework capable of enhancing various types of neural networks (e.g., CNNs and RNNs) with declarative first-order logic rules. Specifically, they developed an iterative distillation method that transfers the structured information of logic rules into the weights of neural networks. With a few highly intuitive rules, they obtained substantial improvements and achieved state-of-the-art or comparable results to previous best-performing systems. In [19], Xu et al. developed a novel methodology for using symbolic knowledge in deep learning by deriving a semantic loss function that bridges between neural output vectors and logical constraints. This loss function captures how close the neural network is to satisfying the constraints on its output. In [20], Fischer et al. presented DL2, a system for training and querying neural networks with logical constraints. Using DL2, one can declaratively specify domain knowledge constraints to be enforced during training, as well as pose queries on the model to find inputs that satisfy a set of constraints. DL2 works by translating logical constraints into a loss function with desirable mathematical properties. The loss is then minimized with standard gradient-based methods [21].

Last but not least, we would like to mention modular neural networks, as an improvement on the conventional artificial neural networks, where a task can be divided into subtasks, and each of these sub-tasks is learned by an expert sub-module. This technique has been successfully used e.g. in pattern recognition, particularly to human recognition using different biometric measures [22–26]. Granual computing, where a granule is defined as one of the numerous small particles forming a larger unit, has also been successfully combined with modular neural networks by granulating the information that the network is going to learn. This approach defines a granule as one of the numerous small particles forming a larger unit. In [25], the advantages of this type of networks were widely demonstrated. Using the granual approach, the smaller sub-tasks, which are easily described with the model proposed in this work, can be combined to solve a more complex problem.

All of these promising approaches point towards the desirable mathematical framework that nilpotent logical systems can offer. Our general aspiration here is to provide a general mathematical framework in order to benefit from tight integration of machine learning and continuous logical methods. The model is well aligned not only with black-box models, but also with white-box models such as regression or decision trees [27].

3. Nilpotent logical systems and multicriteria decision tools

In this Section, we show why a specific logical system, the nilpotent logical system is well-suited to the neural environment. First, we provide some basic preliminaries.

The most important operators in classical logic are the conjunction, the disjunction, and the negation operator. These three basic operators together form a so-called connective system.

When extending classical logic to continuous logic, compatibility and consistency are crucial. The negation should also be involutive; i.e. $n(n(x)) = x$, for $\forall x \in [0,1]$. Involutive negations are called strong negations.

Definition 1. The triple (c, d, n), where c is a t-norm, d is a t-conorm, and n is a strong negation, is called a connective system.

As mentioned in the Introduction, numerous continuous logical systems have been introduced and studied in the literature. In this study, we will show how nilpotent logical systems relate to neural networks.

Definition 2 (*[5]*)**.** A connective system is nilpotent, if the conjunction c is a nilpotent t-norm, and the disjunction d is a nilpotent t-conorm.

In the nilpotent case, the generator functions of the disjunction and the conjunction (denoted by $t(x)$ and $s(x)$ respectively) are bounded functions, being determined up to a multiplicative constant. This means that they can be normalized the following way:

$$f_c(x) := \frac{t(x)}{t(0)}, \qquad f_d(x) := \frac{s(x)}{s(1)}. \tag{1}$$

Note that the normalized generator functions are now uniquely defined.

Next, we recall the definition of the cutting function, to simplify the notations used. The differentiable approximation of this cutting function, the squashing function $S(x)$ introduced and examined in [28], will be a ReLu-like bounded activation function in our model. In [8], the authors showed that all the nilpotent operators can be described by using one generator function $f(x)$ and the cutting function.

Definition 3. Let us define the cutting operation $[\ ]$ by

$$[x] = \begin{cases} 0 & if \quad x < 0 \\ x & if \quad 0 \le x \le 1 \\ 1 & if \quad 1 < x \end{cases}$$

Remark 1. Note that the cutting function has the same values as ReLu (rectified linear unit) for $x \le 1$, but it remains bounded for $\forall x \in \mathbb{R}$.

Proposition 1. *With the help of the cutting operator, we can write the conjunction and disjunction in the following form, where f_c and f_d are decreasing and increasing normalized generator functions respectively.*

$$c(x,y) = f_c^{-1}[f_c(x) + f_c(y)], \tag{2}$$

$$d(x,y) = f_d^{-1}[f_d(x) + f_d(y)]. \tag{3}$$

Remark 2. For the natural negations to coincide, as shown in [5], $f_c(x) + f_d(x) = 1$ must hold for $\forall x \in [0,1]$, which means that only one generator function, e.g. $f_d(x)$ is needed to describe the operators. Henceforth, f_d is represented by $f(x)$.

Remark 3. Note that the min and max operators (often used as conjunction and disjunction in applications) can also be expressed by $[\ ]$ in the following way:

$$\min(x,y) = [x + [y - x + 1] - 1], \tag{4}$$

$$\max(x,y) = [x + [y - x]], \tag{5}$$

where $x, y \in [0,1]$.

The associativity of t-norms and t-conorms permits us to consider their extensions to the multivariable case. In [8], the authors examined a general parametric operator $o_\nu(\underline{x})$ of nilpotent systems.

Definition 4. Let $f : [0,1] \to [0,1]$ be an increasing bijection, $\nu \in [0,1]$, and $\underline{x} = (x_1, \dots, x_n)$, where $x_i \in [0,1]$ and let us define the general operator by

$$\begin{aligned} o_\nu(\underline{x}) &= f^{-1}\left[\sum_{i=1}^{n} (f(x_i) - f(\nu)) + f(\nu)\right] = \\ &= f^{-1}\left[\sum_{i=1}^{n} f(x_i) - (n-1)f(\nu)\right]. \end{aligned} \tag{6}$$

Remark 4. Note that the general operator for $\nu = 1$ is conjunctive, for $\nu = 0$ it is disjunctive and for $\nu = \nu^* = f^{-1}\left(\frac{1}{2}\right)$ it is self-dual.

On the basis of Remark 4, the conjunction, the disjunction and the aggregative operator can be defined in the following way.

Definition 5. Let $f : [0,1] \to [0,1]$ be an increasing bijection, $\underline{x} = (x_1, \dots, x_n)$, where $x_i \in [0,1]$. Let us define the conjunction, the disjunction and the aggregative operator by

$$c(\underline{x}) := o_1(\underline{x}) = f^{-1}\left[\sum_{i=1}^{n} f(x_i) - (n-1)\right], \tag{7}$$

$$d(\underline{x}) := o_0(\underline{x}) = f^{-1}\left[\sum_{i=1}^{n} f(x_i)\right], \tag{8}$$

$$a(\underline{x}) := o_{\nu^*}(\underline{x}) = f^{-1}\left[\sum_{i=1}^{n} f(x_i) - \frac{(n-1)}{2}\right], \tag{9}$$

respectively, where $\nu^* = f^{-1}\left(\frac{1}{2}\right)$.

A conjunction, a disjunction and an aggregative operator differ only in one parameter of the general operator in (6). The parameter ν has the semantic meaning of the level of expectation: maximal for the conjunction, neutral for the aggregation, and minimal for the disjunction. Next, let us recall the weighted form of the general operator:

Definition 6. Let $\underline{w} \in \mathbb{R}^n, f : [0,1] \to [0,1]$ an increasing bijection with $\nu \in [0,1], \underline{x} = (x_1, \dots, x_n)$, where $x_i \in [0,1]$. The weighted general operator is defined by

$$o_{\nu,\underline{w}}(\underline{x}) := f^{-1}\left[\sum_{i=1}^{n} w_i(f(x_i) - f(\nu)) + f(\nu)\right]. \tag{10}$$

Note that if the weight vector is normalized; i.e. for $\sum_{i=1}^{n} w_i = 1$,

$$o_{\nu,\underline{w}}(\underline{x}) = f^{-1}\left(\sum_{i=1}^{n} w_i f(x_i)\right). \tag{11}$$

For future application, we introduce a threshold-based operator in the following way.

Definition 7. Let $\underline{w} \in \mathbb{R}^n, \underline{x} = (x_1, \dots, x_n) \in [0,1]^n$, $\underline{\nu} = (\nu_1, \dots \nu_n) \in [0,1]^n$ and let $f : [0,1] \to [0,1]$ be a strictly increasing bijection. Let us define the threshold-based nilpotent operator by

$$o_{\underline{\nu},\underline{w}}(\underline{x}) = f^{-1}\left[\sum_{i=1}^{n} w_i\,(f(x_i) - f\,(\nu_i)) + f(\nu)\right] =$$

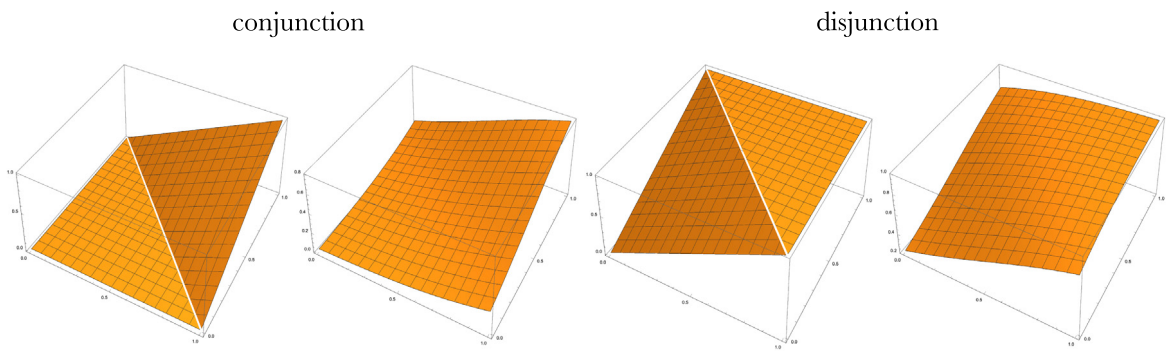

Fig. 1. Nilpotent conjunction and disjunction followed by their approximations using the squashing function.

Table 1
The most important two-variable operators $o_{\underline{w}}(x)$.

	w_1	w_2	C	$o_{\underline{w}}(x,y)$	for $f(x)=x$	Notation
Logical operators						
Disjunction	1	1	0	$f^{-1}[f(x)+f(y)]$	$[x+y]$	$d(x,y)$
Conjunction	1	1	−1	$f^{-1}[f(x)+f(y)-1]$	$[x+y-1]$	$c(x,y)$
Implication	−1	1	1	$f^{-1}[f(y)-f(x)+1]$	$[y-x+1]$	$i(x,y)$
Multicriteria decision tools						
Arithmetic mean	0.5	0.5	0	$f^{-1}\left[\frac{f(x)+f(y)}{2}\right]$	$\frac{x+y}{2}$	$m(x,y)$
Preference	−0.5	0.5	0.5	$f^{-1}\left[\frac{f(y)-f(x)+1}{2}\right]$	$\frac{y-x+1}{2}$	$p(x,y)$
Aggregative operator	1	1	−0.5	$f^{-1}\left[f(x)+f(y)-\frac{1}{2}\right]$	$\left[x+y-\frac{1}{2}\right]$	$a(x,y)$

$$= f^{-1}\left[\sum_{i=1}^{n} w_i f(x_i) + C\right], \tag{12}$$

where

$$C = f(\nu) - \sum_{i=1}^{n} w_i f(\nu_i). \tag{13}$$

Note that for $f(x) = x$, (12) gives the functions modeled by perceptrons in neural networks:

$$\left[\sum_{i=1}^{n} w_i x_i + C\right]. \tag{14}$$

Based on Eqs. (7) to (9), it is easy to see that the conjunction, the disjunction and also the aggregative operator can be expressed in this form. The most commonly used operators for $n = 2$ and for special values of w_i and C, also for $f(x) = x$, are listed in Table 1.

Now let us focus on the unary (1-variable) case, examined in [9], which also plays an important role in the nilpotent neural model. The unary operators are mainly used to construct modifiers and membership functions by using a generator function. The membership functions can be interpreted as modeling an inequality [29]. Note that non-symmetrical membership functions can also be constructed by connecting two unary operators with a conjunction [9,10].

Definition 8. Let $x \in [0,1]$, $\alpha, \gamma \in \mathbb{R}$ and let $f : [0,1] \to [0,1]$, a strictly increasing bijection. Then

$$o_{\alpha,\gamma}(x) := f^{-1}[\alpha f(x) + \gamma]. \tag{15}$$

Remark 5. Note that as shown in [9], Eq. (15) composed by the (weighted) arithmetic mean operator as an inner function, yields to Eq. (12).

Table 2
The most important unary operators $o_{\alpha,\gamma}(x)$.

	γ	$o_{\alpha,\gamma}(x)$	for $f(x)=x$	Notation
Possibility	0	$f^{-1}[\alpha f(x)]$	$[\alpha x]$	$\tau_P(x)$
Necessity	$1-\alpha$	$f^{-1}[\alpha f(x)-(\alpha-1)]$	$[\alpha x-(\alpha-1)]$	$\tau_N(x)$
Sharpness	$\frac{\alpha-1}{2}$	$f^{-1}[\alpha f(x)-\frac{(\alpha-1)}{2}]$	$[\alpha x-\frac{(\alpha-1)}{2}]$	$\tau_S(x)$

The most important unary operators for special γ values are listed in Table 2.

Our attention can now be turned to the cutting function. The main drawback of the cutting function in the nilpotent operator family is the lack of differentiability, which would be necessary for numerous practical applications. Although most fuzzy applications (e.g. embedded fuzzy control) use piecewise linear membership functions owing to their easy handling, there are areas where the parameters are learned by a gradient-based optimization method. In this case, the lack of continuous derivatives makes the application impossible. For example, the membership functions have to be differentiable for each input in order to fine-tune a fuzzy control system by a simple gradient-based technique. This problem could be easily solved by using the so-called squashing function, which provides a solution to the above-mentioned problem by a continuously differentiable approximation of the cutting function.

The squashing function given in Definition 9 is a continuously differentiable approximation of the generalized cutting function by means of sigmoid functions (see Fig. 2).

Definition 9. The squashing function [9,28,30] is defined as

$$S^{(\beta)}_{a,\lambda}(x) = \frac{1}{\lambda\beta}\ln\frac{1+e^{\beta(x-(a-\lambda/2))}}{1+e^{\beta(x-(a+\lambda/2))}} = \frac{1}{\lambda\beta}\ln\frac{\sigma^{(-\beta)}_{a+\lambda/2}(x)}{\sigma^{(-\beta)}_{a-\lambda/2}(x)}.$$

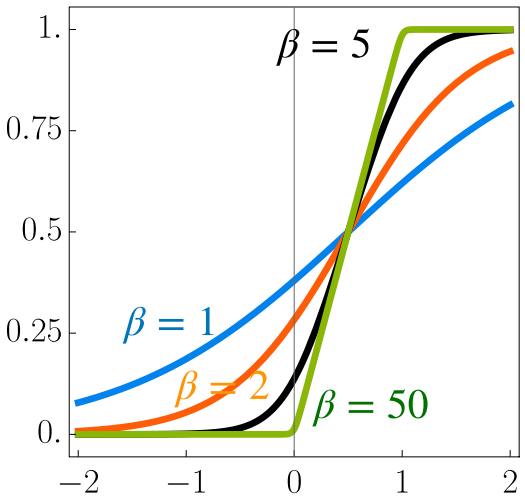

Fig. 2. Squashing functions for $a = 0.5$, $\lambda = 1$, for different β values ($\beta_1 = 1$, $\beta_2 = 2$, $\beta_3 = 5$, and $\beta_4 = 50$).

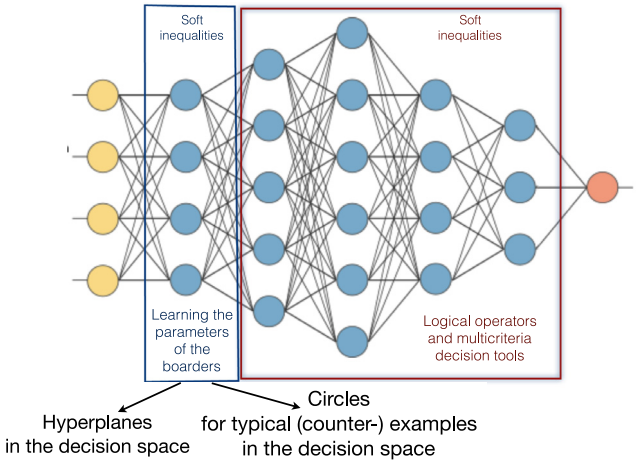

Fig. 3. Nilpotent neural model.

where $x, a, \lambda, \beta \in \mathbb{R}$ and $\sigma_d^{(\beta)}(x)$ denotes the logistic function:

$$\sigma_d^{(\beta)}(x) = \frac{1}{1 + e^{-\beta\cdot(x-d)}}. \tag{16}$$

By increasing the value of β, the squashing function approaches the generalized cutting function. In other words, β shows the accuracy of the approximation, while the parameters a and λ determine the center and width. The error of the approximation can be upper bounded by c/β, which means that by increasing the parameter β, the error decreases by the same order of magnitude. The derivatives of the squashing function are easy to calculate and can be expressed by sigmoid functions and itself:

$$\frac{\partial S_{a,\lambda}^{(\beta)}(x)}{\partial x} = \frac{1}{\lambda}\left(\sigma_{a-\lambda/2}^{(\beta)}(x) - \sigma_{a+\lambda/2}^{(\beta)}(x)\right) \tag{17}$$

$$\frac{\partial S_{a,\lambda}^{(\beta)}(x)}{\partial a} = \frac{1}{\lambda}\left(\sigma_{a+\lambda/2}^{(\beta)}(x) - \sigma_{a-\lambda/2}^{(\beta)}(x)\right) \tag{18}$$

$$\frac{\partial S_{a,\lambda}^{(\beta)}(x)}{\partial \lambda} = -\frac{1}{\lambda}S_{a,\lambda}^{(\beta)}(x) + \frac{1}{2\lambda}\left(\sigma_{a+\lambda/2}^{(\beta)}(x) + \sigma_{a-\lambda/2}^{(\beta)}(x)\right) \tag{19}$$

The squashing function defined above is an approximation of the rectifier (rectified linear unit, ReLU) for $x \leq 1$, with the benefit of having an upper bound. Being bounded from above makes the use of continuous logic possible. Also note the significant difference between the properties of the squashing function and the sigmoid. Using sigmoids, nilpotent logic can never be modeled. The fact that on the other hand, ReLu can approximate the cutting function, may offer an interpretation to its effectiveness and success. The fact that the squashing function is differentiable and its derivatives can be expressed by sigmoids improves efficiency in applications. An illustration of the nilpotent conjunction and disjunction operators with their soft approximations using the squashing function is shown in Fig. 1. Note that not only logical operators but also multicriteria decision tools, like the preference operator can be described similarly. This means that our model offers a unified framework, in which logic and multicriteria decision tools cooperate and supplement each other.

4. Nilpotent logic-based interpretation of neural networks

The results on nilpotent logical systems discussed in Section 3 offer a new approach to designing neural networks using continuous logic, since membership functions (representing the truth value of an inequality), and also nilpotent logical operators can be modeled by perceptrons. Whether for image classification or for multicriteria decision support, structured logical rules can contribute to the performance of a deep neural network. Given that the network has to find a region in the decision space or in an image, after designing the architecture appropriately, the network only has to find the parameters of the boundary. Here, we propose creating basic building blocks by applying the nilpotent logical concept in the perceptron model and also in the neural architecture.

Boolean units and multilayer perceptrons have a long history. Logical gates (such as the AND, NOT and OR gates) are the basis of any modern-day computer. It is well known that any Boolean function can be composed using a multi-layer perceptron. As examples, the conjunction and the disjunction are illustrated in Fig. 6. Note that for the XOR gate, an additional hidden layer is also required. It can be shown that a network of linear classifiers that fires if the input is in a given area with arbitrary complex decision boundaries can be constructed with only one hidden layer and a single output. This means that if a neural network learns to separate different regions in the n-dimensional space having n input values, each node in the first layer can separate the space into two half-spaces by drawing one hyperplane, while the nodes in the hidden layers can combine them using logical operators.

In Fig. 5, some basic types of neural networks are shown with two input values, finding different regions of the plane. Generally speaking, each node in the neural net represents one threshold and therefore it can draw one line in the picture. The line may be diagonal if the nodes receives both of the inputs i_1 and i_2. The line has to be horizontal or vertical if the node only receives one of the inputs. The deeper hidden levels are responsible for the logical operations.

From several perspectives, as mentioned in the Introduction, a continuous logical framework can provide a more sophisticated and effective approach to this problem than a Boolean can.

Among continuous logical systems, the nilpotent logical framework described above is well-suited for the neural concept architecture when it comes to implementing logical rules. For the sake of simplicity, henceforth we assume that for the generator function $f(x) = x$ holds and we design the neural network architecture in the following way. In the first layer, the perceptrons model membership functions as truth values of inequalities, such as

$$\sum_{i=1}^{n} w_i x_i - b > 0; \tag{20}$$

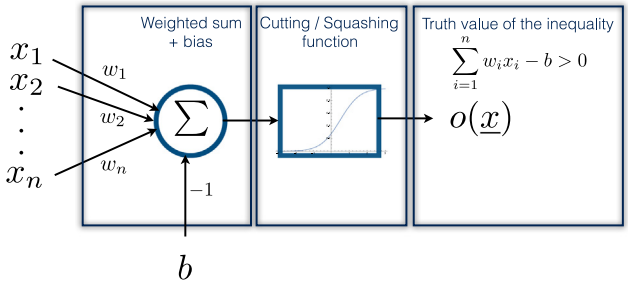

Fig. 4. Nilpotent perceptron model.

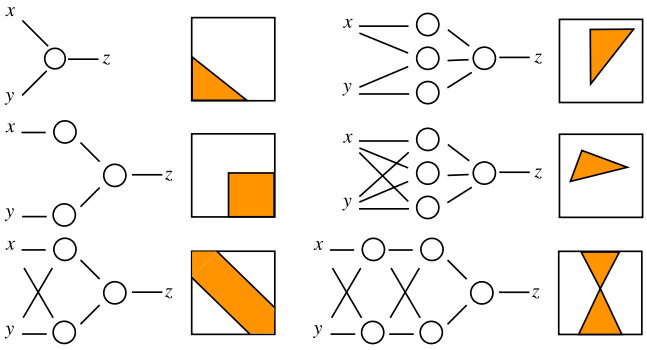

Fig. 5. Basic types of neural networks with two input values using logical operators in the hidden layer used to find different regions of the plane.

representing a half-space bounded by a hyperplane in the decision space (see Fig. 4). Here, the weights w_i and the bias b are to be learned. The truth value of this inequality can be modeled by

$$\left[\sum_{i=1}^{n} w_i x_i - b\right]; \tag{21}$$

or to avoid the vanishing gradient problem, the cutting function can be approximated by the so-called squashing function, by the differentiable approximation of the cutting function

$$S\left(\sum_{i=1}^{n} w_i x_i - b\right). \tag{22}$$

The parameters of the squashing function in (22) now have a context-dependent semantic meaning.

Since the nilpotent logical operators also represent inequalities and therefore have the same structure (compare with Eq. (14), and see also Table 1 and Fig. 4), in the hidden layers, we can apply them on the clusters created in the first layer (see Fig. 3). Here, the weights and biases characterize the type of the logical operator. As an illustration, the perceptron models of the conjunction and of the disjunction can be seen in Fig. 6. This means that for a given logical operator, the weights and the bias can be frozen. The squashing function plays the role of the activation function in all of the layers. The backpropagation algorithm needs to be adjusted: the error function is calculated based on all of the weights and biases (frozen and learnable), but the backpropagation leaves the frozen layers out. Moreover, in this nilpotent model, the conjunction, the disjunction, and the aggregation differ only in a translation parameter; i.e. the weights are equal for all of them and only the biases are different. This fact makes it possible for the network to learn the type of logical operators just by learning the bias.

To illustrate the model, two basic examples are given.

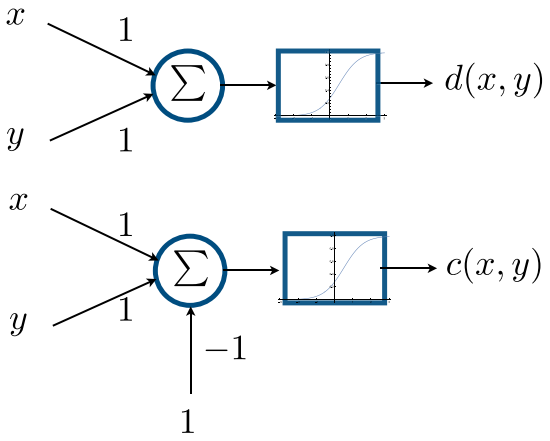

Fig. 6. Perceptron model of the conjunction and the disjunction.

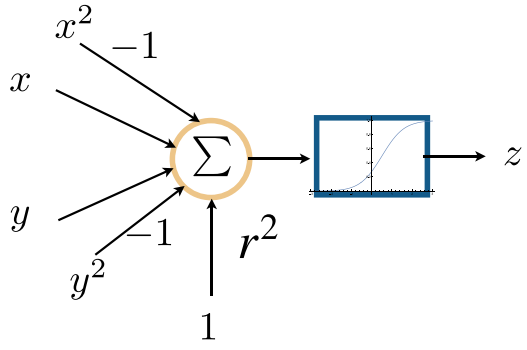

Fig. 7. Perceptron model classifying a circle with radius r.

Example 1. As an example, let us assume that a network needs to find positive examples that lie inside a triangular region. This means that we should design the network to conjunct three half-planes, and to find the parameters of the boundary lines. The output values for a triangular domain using nilpotent logic and its continuous approximation are illustrated in Fig. 8.

Example 2. The flow chart and the model for the logical expression

"$((x > 0)$ AND $(y > 0))$ OR $((x < 0)$ AND $(y < 0))$"

can be seen in Fig. 10. Here, $x < 0$ is modeled by NOT $(x > 0)$. Note that

$$[[x + y - 1] + [(1 - x) + (1 - y) - 1]] = \\ = [[x + y - 1] + [-x - y + 1]],$$

therefore the bias for the negated inputs is $+1$. The weights and biases for the logical operators are listed in Table 3.

Additionally, taking into account the fact that the area inside or outside a circle is described by an inequality containing the squares of the input values, it is also possible to construct a novel type of unit by adding the square of each input into the input layer (see Fig. 7). This way, the polygon approximation of the circle can be eliminated. For an illustration, see Fig. 9. Note that by modifying the weights, an arbitrary conic section can also be described.

Choosing the right activation function for each layer is crucial and may have a significant impact on metric scores and the training speed of the neural model. In the model introduced in this Section, the smooth approximation of the cutting function is a natural choice for the activation function in the first layer as

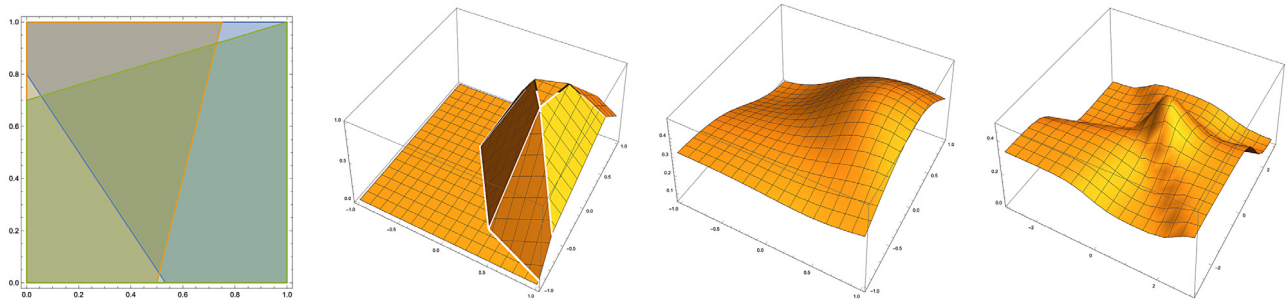

Fig. 8. Output values for a triangular domain using nilpotent logic and its continuous approximation for different parameter values.

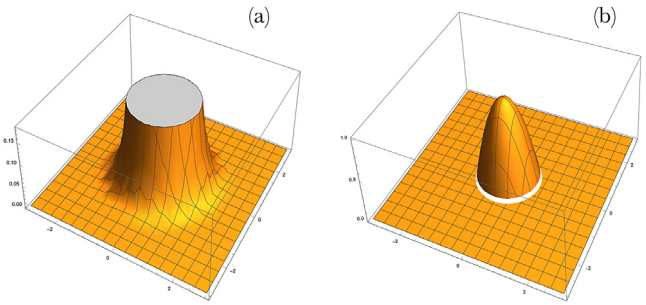

Fig. 9. Output values for a circular region using nilpotent logic (a), and its differentiable approximation (b).

well as in the hidden layers, where the logical operators work. Although there are a vast number of activation functions (e.g. linear, sigmoid, tanh, or the recently introduced Rectified Linear Unit (ReLU) [31], exponential linear unit (ELU) [32], sigmoid-weighted linear unit (SiLU) [33]) considered in the literature, most of them are introduced based on some desired properties, without any theoretical background. The parameters are usually fitted only on the basis of experimental results. The squashing function stands out of the other candidates by having a theoretical background thanks to the nilpotent logic which lies behind the scenes.

To sum up, on the one hand, this structure leads to a drastic reduction in the number of parameters to be learned, and on the other hand, it supports the interpretation, making the debugging process manageable. Given the logical structure, the parameters to be learned are located in the first layer. The choice of the activation functions in the first layer as well as in the hidden, logical layers, have a sound theoretical background. Designing the network architecture appropriately, arbitrary regions can be described as intersections and unions of polyhedra in the decision space. Moreover, multicriteria decision tools can also be integrated with given weights and thresholds. Note that the weights and biases in the hidden layers define the type of operator to be used. These parameters can also be learned in a more sophisticated model to be examined in future work.

5. Playground examples

To illustrate our model with some simple examples, we extended the Tensorflow Playground with the squashing function ($\beta = 50, \lambda = 1, a = 0.5$) as activation function and modified the backpropagation algorithm according to the frozen weights in the hidden layers.

5.1. XOR

Let us first consider an example of a particular data set based on Example 2. An image of a generated set of data is shown in

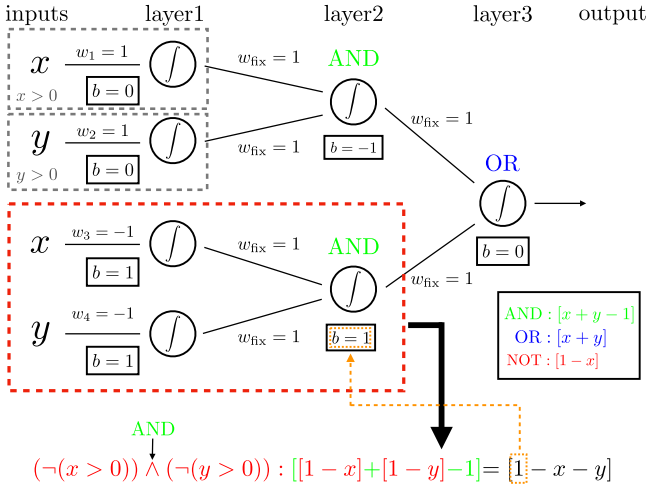

Fig. 10. Nilpotent neural structure representing the expression ($x > 0$) AND ($y > 0$) OR ($x < 0$) AND ($y < 0$).

Table 3
Weights and biases for modeling the XOR logical gate.

$x > 0$	$w_1 = 1$	$w_2 = 0$	$b = 0$	$[x]$
$y > 0$	$w_1 = 0$	$w_2 = 1$	$b = 0$	$[y]$
x AND y	$w_1 = 1$	$w_2 = 1$	$b = -1$	$[x + y - 1]$
x OR y	$w_1 = 1$	$w_2 = 1$	$b = 0$	$[x + y]$
NOT (x)	$w_1 = -1$	$w_2 = 0$	$b = 1$	$[1 - x]$
NOT (y)	$w_1 = 0$	$w_2 = -1$	$b = 1$	$[1 - y]$
(NOT (x)) AND (NOT (y))	$w_1 = -1$	$w_2 = -1$	$b = 1$	$[-x - y + 1]$

Fig. 11. Orange data points have a value of -1 and blue points have a value of $+1$. Here, the target variable is positive when x and y are both positive or both negative. In a logical network:

- If ($x_1 > 0$) AND ($x_2 > 0$) THEN predict $+1$
- If ($x_1 < 0$) AND ($x_2 < 0$) THEN predict $+1$
- Else predict -1

An efficient neural network can be built to make predictions for this logical expression even without using the cross feature $x * y$. For the structure and for the frozen weights and biases, see Table 3.

According to our model, the smooth approximation of the cutting function called the squashing function is a natural choice for the activation function in the first layer as well as in the hidden layers, where the logical operators are used. If we design this logical structure before training, an interpretation of the network naturally emerges.

Table 4
Weights and biases for modeling the preference operator.

$x > y$	$w_1 = 0.5$	$w_2 = -0.5$	$b = 0.5$	$\left[\frac{x-y+1}{2}\right]$
$y > -x$	$w_1 = 0.5$	$w_2 = 0.5$	$b = 0.5$	$\left[\frac{x+y+1}{2}\right]$
$x < y$	$w_1 = -0.5$	$w_2 = 0.5$	$b = 0.5$	$\left[\frac{-x+y+1}{2}\right]$
$y < -x$	$w_1 = -0.5$	$w_2 = -0.5$	$b = 0.5$	$\left[\frac{-x-y+1}{2}\right]$

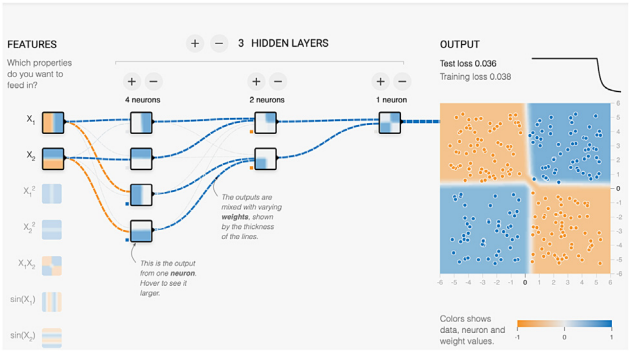

Fig. 11. Nilpotent neural network block designed for modeling XOR. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Notice how the neurons in the hidden layer reveal the logical structure of the network (Fig. 11), assisting the interpretability of the neural model.

5.2. Preference

Another image of a generated set of data is shown in Fig. 12. Orange data points have a value of -1 and blue points have a value of $+1$. Here, the network has to learn the parameters of the straight lines separating the different regions. The target variable is positive when both $x > y$ and $y > -x$ hold or where both $x < y$ and $y < -x$ hold. In a logical network:

- If $(x > y)$ AND $(y > -x)$ THEN predict $+1$
- If $(x < y)$ AND $(y < -x)$ THEN predict $+1$
- Else predict -1

The network structure is illustrated in Fig. 12. Here, the expression $x > y$ is modeled by the preference operator $p(x, y)$ (see Tables 1 and 4). Notice how the neurons in the hidden layer reveal the logical structure of the network (see Fig. 12), assisting the interpretability of the neural model.

Remark 6. Networks can also be readily designed for finding concave regions. For example, see Fig. 13.

Remark 7. Note that the frequently used min and max operators can also be modeled by a similar network, based on Eqs. (4) and (5).

6. Conclusions

In this study, we suggested interpreting neural networks by using continuous nilpotent logic and multicriteria decision tools to reduce the black-box nature of the neural models, aiming at the interpretability and improved safety of machine learning. We introduced the main concept and the basic building blocks of the model to lay the foundations for the future steps of the application. In our model, membership functions (representing truth values of inequalities), and also nilpotent operators are modeled by perceptrons. The network architecture is designed prior to training. In the first layer, the parameters of the membership functions are needed to be learned, while in the hidden layers, the nilpotent logical operators work with given weights and biases. Based on previous results, a rich asset of logical operators with rigorously examined properties is available. A novel type of neural unit was also introduced by adding the square of each input to the input layer (see Fig. 7) to describe the inside or the outside of a circle without polygon approximation.

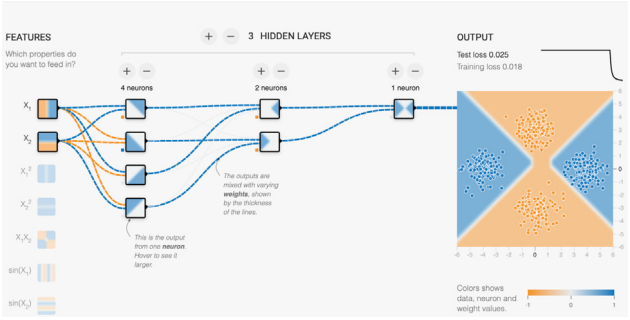

Fig. 12. Nilpotent neural network block designed for modeling preference. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

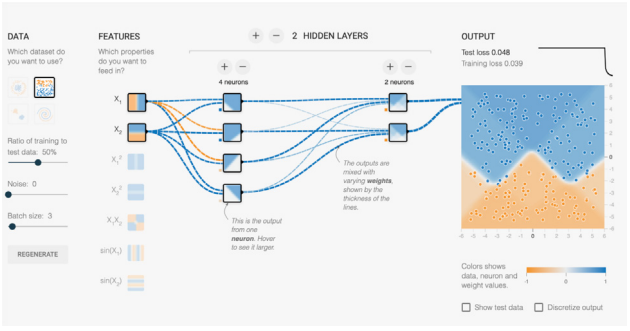

Fig. 13. Nilpotent neural network block designed for finding a concave region.

The theoretical basis offers a straightforward choice of activation functions: the cutting function or its differentiable approximation, the squashing function. Both functions represent truth values of soft inequalities, and the parameters have a semantic meaning. Our model also seems to provide an explanation for the great success of the rectified linear unit (ReLU).

Note that in case of deep neural networks, the computational complexity of the nilpotent neural model corresponds to the complexity of the traditional architecture, in other words, better interpretability can be achieved without an increase of computation time. Moreover, thanks to the simple structure of the model, more complex tasks can be described by using significantly less parameters and layers, which can result in a drastic decrease of computational complexity. To give an illustrative example, consider a classification problem, where the positive examples are represented in the area between two concentric circles. The network has to find the parameters of these circles. By designing the network using the model introduced in our work, there are only a few parameters to be found (the two radii and the coordinates of the center), whereas using a traditional neural network, the computation is much more demanding.

There is still some limitations of the proposed model. Handling overlapping regions in classification problems still remains a major difficulty, as well as the exponential increase of test data

needed in 3- or more dimensional learning problems. Another limitation is of course, that a-priori information is needed to design the architecture in an appropriate way.

The concept was illustrated with some toy examples taken from an extended version of the tensorflow playground. The implementation of this hybrid model in deeper networks (by combining the building blocks introduced here) and its application e.g. in multicriteria decision making or image classification, illustrated with simulations and formal comparison of results is left for future work.

Beyond classification, this model is also well aligned with white-box machine learning, linear regression and decision trees. To further our research, we are planning to provide a detailed discussion on this topic as well.

CRediT authorship contribution statement

Orsolya Csiszár: Writing - original draft, Writing - review & editing, Formal analysis, Methodology, Visualisation. **Gábor Csiszár:** Software, Visualization, Writing - original draft, Writing - review & editing. **József Dombi:** Conceptualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This study was partially supported by grant TUDFO/47138-1/2019-ITM of the Ministry for Innovation and Technology, Hungary.

References

[1] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, F. Roli, Evasion Attacks Against Machine Learning At Test Time, in: Lecture Notes in Computer Science, 2013, pp. 387–402.
[2] C. Szegedy, Z. Wojciech, I. Sutskever J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2013, arXiv:1312.6199.
[3] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv:1412.6572.
[4] S. Thys, W. Van Ranst, T. Goedemé, Fooling automated surveillance cameras: adversarial patches to attack person detection, 2019, arXiv:1904.08653.
[5] J. Dombi, O. Csiszár, The general nilpotent operator system, Fuzzy Sets and Systems 261 (2015) 1–19.
[6] J. Dombi, O. Csiszár, Equivalence operators in nilpotent systems, Fuzzy Sets Syst. (2015) http://dx.doi.org/10.1016/j.fss.2015.08.012, available online.
[7] J. Dombi, O. Csiszár, Implications in bounded systems, Inform. Sci. 283 (2014) 229–240.
[8] J. Dombi, O. Csiszár, Self-dual operators and a general framework for weighted nilpotent operators, Int. J. Approx. Reason. 81 (2017) 115–127.
[9] J. Dombi, O. Csiszár, Operator-dependent modifiers in nilpotent logical systems, operator-dependent modifiers in nilpotent logical systems, in: Proceedings of the 10th International Joint Conference on Computational Intelligence, IJCCI 2018, 2018, pp. 126–134.
[10] O. Csiszár, J. Dombi, Generator-based modifiers and membership functions in nilpotent operator systems, in: IEEE International Work Conference on Bioinspired Intelligence, iwobi 2019, July 3–5, 2019, Budapest, Hungary, 2019.
[11] D. Dubois, H. Prade, Fuzzy sets in approximate reasoning. Part 1: Inference with possibility distributions, Fuzzy Sets Syst. 40 (1991) 143–202.
[12] E. Trillas, L. Valverde, On some functionally expressable implications for fuzzy set theory, in: Proc. of the 3rd International Seminar on Fuzzy Set Theory, Linz, Austria, 173-190, 1981.
[13] C.T. Lin, C.S.G. Lee, Neural Fuzzy Systems: A Neuro-Fuzzy Synergism To Intelligent Systems, Prentice Hall, Upper Saddle River, NJ, 1996.
[14] G.G. Towell, J.W. Shavlik, M.O. Noordewier, Refinement of approximate domain theories by knowledge-based neural networks, in: Proceedings of the eighth National Conference on Artificial Intelligence, Boston, MA, 861-866, 1990.
[15] A.S.d. Garcez, K. Broda, D.M. Gabbay, Neural-Symbolic Learning Systems: Foundations and Applications, Springer Science & Business Media, 2012.
[16] M.V. Franca, G. Zaverucha, A.S.d. Garcez, Fast relational learning using bottom clause propositionalization with artificial neural networks, Mach. Learn. 94 (1) (2014) 81–104.
[17] T.D. Kulkarni, W.F. Whitney, P. Kohli, J. Tenenbaum, Deep convolutional inverse graphics network, in: Proc. of NIPS, 2015, pp. 2530–2538.
[18] Z. Hu, X. Ma, Z. Liu, E. Hovy, E.P. Xing, Harnessing deep neural networks with logic rules, arXiv:1603.06318v5.
[19] J. Xu, Z. Zhang, T. Friedman, Y. Liang, G.V. den Broeck, A semantic loss function for deep learning with symbolic knowledge, in: Proceedings of the 35th International Conference on Machine Learning, Vol. 80, ICML 2018, Stockholm, 2018, pp. 5498–5507.
[20] M. Fisher, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, M. Vechev, DL2: Training and querying neural networks with logic, in: Proceedings of the 36 th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.
[21] Y. Yao, Perspectives of granular computing, in: IEEE International Conference on Granular Computing, GrC, 2005, pp. 85–90.
[22] D. Sánchez, P. Melin, O. Castillo, A grey wolf optimizer for modular granular neural networks for human recognition, Comput. Intell. Neurosci. (2017) 4180510:1–4180510:26.
[23] D. Sánchez, P. Melin, O. Castillo, Comparison of particle swarm optimization variants with fuzzy dynamic parameter adaptation for modular granular neural networks for human recognition, J. Intell. Fuzzy Systems 38 (3) (2020) 3229–3252.
[24] P. Melin, D. Sánchez, Multi-objective optimization for modular granular neural networks applied to pattern recognition, Inform. Sci. 460–461 (2018) 594–610.
[25] D. Sánchez, P. Melin, Optimization of modular granular neural networks using hierarchical genetic algorithms for human recognition using the ear biometric measure, Eng. Appl. Artif. Intell. 27 (2014) 41–56.
[26] D. Sánchez, P. Melin, O. Castillo, Optimization of modular granular neural networks using a firefly algorithm for human recognition, Eng. Appl. of Artif. Intell. 64 (2017) 172–186.
[27] C. Molnar, Interpretable machine learning. a guide for making black box models explainable, 2019, https://christophm.github.io/interpretable-ml-book/.
[28] J. Dombi, Z.s. Gera, The approximation of piecewise linear membership functions and Łukasiewicz operators, Fuzzy Sets and Systems 154 (2005) 275– 286.
[29] J. Dombi, Membership function as an evaluation, Fuzzy Sets and Systems 35 (1990) 1–21.
[30] J. Dombi, Z.s. Gera, Fuzzy rule based classifier construction using squashing functions, J. Intell. Fuzzy Syst. 19 (2008) 3–8.
[31] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, 2014.
[32] D. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (ELUs), 2015, arXiv:1511.07289.
[33] S. Elfwing, E. Uchibe, K. Doya, Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, Neural Netw. 107 (2018) 3–11.