Genetic interval neural networks for granular data regression

Mario G.C.A. Cimino [a,*], Beatrice Lazzerini [a], Francesco Marcelloni [a], Witold Pedrycz [b,c]

[a] Dipartimento di Ingegneria dell'Informazione, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy
[b] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada T6G 2G7
[c] Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

Keywords:
Function approximation
Genetic algorithm
Granular computing
Interval analysis
Interval order relation
Neurocomputing

A B S T R A C T

Granular data and granular models offer an interesting tool for representing data in problems involving uncertainty, inaccuracy, variability and subjectivity have to be taken into account. In this paper, we deal with a particular type of information granules, namely interval-valued data. We propose a multilayer perceptron (MLP) to model interval-valued input–output mappings. The proposed MLP comes with interval-valued weights and biases, and is trained using a genetic algorithm designed to fit data with different levels of granularity. In the evolutionary optimization, two implementations of the objective function, based on a numeric-valued and an interval-valued network error, respectively, are discussed and compared. The modeling capabilities of the proposed MLP are illustrated by means of its application to both synthetic and real world datasets.

1. Introduction and background

Humans exploit abilities of processing non-numeric information clumps (*granules*) rather than individual numeric values [29]. Granulation is a suitable way for managing situations characterized by excess or a lack of data [13]. The first situation occurs, for instance, when there are collections of many objects that exhibit some similarity in terms of their properties or functional appearance [24]. Here, information granulation provides a vehicle to abstract the complexity of the data set that one can organize into hierarchies and convert the original problem into manageable subtasks. The second situation occurs, for instance, when there are noisy data or we encounter qualitative assessments provided by human experts. Here, granulation of information allows modeling the precision of indirect measurements, providing a computationally appealing view of knowledge [13].

There are a number of formal models of information granules including intervals, sets, rough sets, fuzzy sets, and shadowed sets just to name a few existing alternatives. The type of representation formalism is an essential issue to be tackled. Its choice depends on the available domain knowledge. In [27,28] the authors claim that the implementation of information granules in terms of *interval-valued* data is the easiest to comprehend and express by a domain expert, and the simplest to process when there is a great variability of granule sizes. In interval-valued data, all elements included in the same interval lose their identity in the sense they become fully indistinguishable. In a multidimensional domain, each granule is modeled by a hyperbox, which is a simple geometrical structure fully defined by its boundaries.

In the literature, two different areas are concentrated on interval-valued data, namely Symbolic Data Analysis and Interval Analysis [22]. Symbolic Data Analysis is a new development related to statistics, data mining and computer science. This paradigm offers a comprehensive approach that consists of summarizing a dataset by means of symbolic variables, e.g.

* Corresponding author. Tel.: +39 050 2217455; fax: +39 050 2217600.
E-mail addresses: m.cimino@iet.unipi.it (M.G.C.A. Cimino), b.lazzerini@iet.unipi.it (B. Lazzerini), f.marcelloni@iet.unipi.it (F. Marcelloni), wpedrycz@ualberta.ca (W. Pedrycz).

interval variables, so as to transform the data set into a smaller and more manageable set of symbolic data. Symbolic data preserve the essential information and can be processed by means of symbolic methods. Interval Analysis introduces intervals as a fundamental means of representing real data, thus providing methods for numeric processing of these intervals.

In real-world scenarios, interval-valued data arise in several situations, such as recording monthly interval temperatures at meteorological stations, daily interval stock prices, inaccuracy of the measurement instruments, and range of variation of a variable through time [16].

The use of information granulation requires developing appropriate learning algorithms. Given this objective in mind, in the paper we propose a neural architecture to process information granules consisting of interval-valued data. Our focus is on an interval regression problem.

The first conceptualization of neural networks for processing granular data was introduced by Pedrycz and Vukovich [28]. Here, several design approaches are discussed, together with a number of architectures of granular neural networks and associated training methods. Also, the authors tackle a number of fundamental issues of these networks, such as specificity of information granules, learning complexity and generalization capabilities. Neural architectures based on interval arithmetic have been proposed in [7,10,11,22,24,26]. In particular, the model developed in [22] uses a standard multilayer perceptron (MLP) with numeric weights and biases, and a neuron transfer function able to operate with interval-valued inputs and outputs. Here, the training process uses an error function based on a weighted Euclidean distance between intervals, and a Quasi Newton method for the minimization of the error function. Other minimization methods such as genetic algorithms (GA) and evolutionary strategies have been also discussed [10].

In its general architecture proposed in the literature, an MLP that processes interval-valued data is characterized by weights and biases expressed in terms of intervals, and maps an interval-valued input vector to an interval-valued output. However, very often, in the design of the training algorithms some simplifying assumptions are being made, e.g., it is assumed that inputs, weights and biases may be real numbers, or the error function formed for the intervals is not compliant with the rules of the interval arithmetic. As regards the error function, other strategies exploit a modified quadratic error function, based on upper and lower bounds [10,14]. In [11] we proposed a new genetic algorithm-based learning method for a general interval-valued neural architecture. In this study, we adopted a numeric-valued error function derived from basic properties of interval arithmetic. However, due to the interval nature of the network output, the error of the model should be viewed as an interval to quantify the precision of the model (in addition to the quantification of its accuracy [4]). Indeed, to assess the extent to which a set of outputs of the model satisfy the requirements is not sufficient to know the proximity of a representative output to a desired value. We should also know how much other outputs are close to the representative. Such information is often important to guarantee an effective optimization, especially when the model exhibits low resolution. Hence, a numeric-valued error represents a design constraint which strongly limits the capabilities of the GA optimization.

Unlike the approaches proposed in the literature, in our MLP architecture, weights and biases are intervals, and each operation performed in the network is based on interval arithmetic. Furthermore, the parameters of the neural network are learned by using a GA which optimizes an error function expressed in terms of intervals. The use of network errors expressed as intervals requires defining an ordering relation between the chromosomes based on the interval arithmetic.

The resulting network allows forming mappings at different levels of granularity and therefore at different model resolutions. Resolution concerns the smallest change in the data that produces a modification in the model. Since the level of granularity is problem-oriented and user-dependent, it is a parameter of our neural architecture. We show the effectiveness of the method by using various interval-valued datasets.

The paper is organized as follows. In Section 2, we introduce some basic notions of interval arithmetic. The architecture of the interval neural network and the problem requirements are formally defined in Sections 3 and 4. Sections 5 and 6 describe the design of the GA, especially its search space representation issues. A procedure for parameter setting and experimental results are shown in Section 7. Finally, Section 8 draws some conclusions and discusses some future work.

2. Interval arithmetic: some definitions

We employ a generic implementation of information granules in terms of conventional interval-valued schemes. An interval-valued variable $\ddot{X}$ is defined as:

$$\ddot{X} = [\underline{x}, \bar{x}] \in \mathbb{IR}, \quad \underline{x}, \bar{x} \in \mathbb{R}, \tag{1}$$

where $\mathbb{IR}$ is the set of all closed and bounded intervals in the real line, and $\underline{x}$ and $\bar{x}$ are the boundaries of the intervals. An F-dimensional granule (hyperbox [15]) is then represented by a vector of interval-valued variables as follows:

$$\ddot{\mathbf{X}} = \left[\ddot{X}_1, \ldots, \ddot{X}_F\right] \in \mathbb{IR}^F, \quad \ddot{X}_i \in \mathbb{IR}. \tag{2}$$

Alternatively, an interval variable can be described in terms of its midpoint $\dot{x}$ and half-width $\widehat{x}$, as follows [10]:

$$\widehat{\dot{X}} = \langle \dot{x}, \widehat{x} \rangle \in \mathbb{IR}, \quad \dot{x}, \ \widehat{x} \in \mathbb{R}, \quad \dot{x} = (\underline{x} + \bar{x})/2, \quad \widehat{x} = (\bar{x} - \underline{x})/2. \tag{3}$$

The notation of the midpoint and half-width is common to express the result of a physical measurement $\dot{x}$ with an associated accuracy $\widehat{x}$, i.e.,$\dot{x} \pm \widehat{x}$, related to a measuring instrument with unknown probability of errors [16].

Numbers can be considered as special intervals, called *degenerated* intervals, with $\widehat{x} = 0$, i.e., $\underline{x} = \bar{x}$ [21]. The four algebraic operations obey the following rule [15]:

$$\ddot{X} \ \mathbf{op} \ \ddot{Y} = \{x \ \mathbf{op} \ y | x \in \ddot{X}, \ y \in \ddot{Y}\} \quad \text{for } \ddot{X}, \ddot{Y} \in \ \mathbb{IR}, \quad \mathbf{op} \in \{+, -, \times, \div\}. \tag{4}$$

In the case of division, the condition $0 \notin \ddot{Y}$ is also assumed. Thus, the image of each of the four basic interval operations is the range of the corresponding real operation. Unary operations obey the following rule [1]:

$$u(\ddot{X}) = \{u(x) | x \in \ddot{X}\} \quad \text{for } \ddot{X} \in \ \mathbb{IR}, \tag{5}$$

where $u(\cdot)$ is a real continuous function defined on $\mathbb{R}$ (e.g., *sqr*, *sqrt*, *sine*, *log*).

Although rule (4) characterizes these operations in a formal way, the usefulness of the interval arithmetic is due to direct expressions of the boundaries of the result. Table 1 summarizes some basic operations of interval arithmetic that have been used in this study. The reader can refer to a detailed discussion in [1,2,9,15,17]. In interval arithmetic any generic operator $\diamondsuit$ must satisfy the following condition:

$$\mathbf{If} \ \ddot{Z} = \ddot{X} \diamondsuit \ddot{Y} \quad \mathbf{then} \ x \diamondsuit y = z \in \ddot{Z}, \quad \forall (x, y) | x \in \ddot{X}, \quad y \in \ddot{Y}. \tag{6}$$

Interval arithmetic does not follow the same rules as the "standard" arithmetic for $\mathbb{R}^2$ [1]. For instance, $\ddot{X} - \ddot{X} \neq [0, 0]$ and $\ddot{X}/\ddot{X} \neq [1, 1]$, for $\ddot{X} \in \mathbb{IR}$.

It is worth noting that the implementations (vii) and (viii) presented in Table 1 do not obey the condition (6). Indeed, such implementations return a real value instead of an interval. Fig. 1a shows an example of computing based on the implementation (vii) of real distance between two intervals. More specifically, here the distance is equal to δ, which is the maximum distance between corresponding boundaries of the two intervals. Fig. 1b shows the computation of the interval distance between the same intervals of Fig. 1a, based on the implementation (iii) and (ix). Here, the resulting interval effectively satisfies the condition (6), representing the minimum and maximum possible distance between two corresponding elements of the two intervals, respectively.

3. The adopted interval-valued neural architecture

Let $f : \mathbb{IR}^F \rightarrow \mathbb{IR}$ be an F-dimensional interval-valued regression model:

$$f(\ddot{\mathbf{X}}) = f\left(\ddot{X}_1, \ldots, \ddot{X}_F\right) = \ddot{Y}. \tag{7}$$

Fig. 2 shows the MLP we adopt to deal with the regression problem modeled by (7). This architecture has been already proposed by some authors (for instance, in [10]). The novelty of the approach here concerns the training process, which allows for an effective and efficient sensitivity analysis (i.e., an ability to quantify the effect of input variability on the outputs).

The hidden layer comprises N nonlinear hidden units and the output layer consists of a single linear output unit.

The activation of each hidden unit j is formed as a sum between the weighted linear combination, with weights $\ddot{\Omega}_{ij}, \ i = 1, \ldots, F, \ j = 1, \ldots, N$, of the F interval-valued inputs $\ddot{X}$ and the bias $\ddot{\Omega}_{0j}$. Since both weights and biases are intervals, this linear combination results in a new interval. The output of each hidden unit is then obtained by transforming its activation interval using a hyperbolic tangent (sigmoid) function. According to the formula (vi) in Table 1, since the function is monotonic, this transformation yields a new interval [10]. Finally, the output of the network, $\ddot{Y}$, is obtained as the sum between the weighted linear combination, with weights $\ddot{\Omega}_j, \ j = 1, \ldots, N$, of the outputs of the hidden layer, and the bias $\ddot{\Omega}_0$. The overall processing method is based on the fundamental arithmetic operations on $\mathbb{IR}$ as shown in Table 1. The resulting model

Table 1
Some basic interval arithmetic operations used in neural network.

	Operation	Implementation
(i)	Addition	$[\underline{x}, \bar{x}] + \left[\underline{y}, \bar{y}\right] = \left[\underline{x} + \underline{y}, \bar{x} + \bar{y}\right]$
(ii)	Real multiplication	$k[\underline{x}, \bar{x}] = [k\underline{x}, k\bar{x}]$ if $k > 0$; $[k\bar{x}, k\underline{x}]$ if $k < 0$
(iii)	Subtraction	$[\underline{x}, \bar{x}] - \left[\underline{y}, \bar{y}\right] = [\underline{x}, \bar{x}] + \left(-\left[\underline{y}, \bar{y}\right]\right) = \left[\underline{x} - \bar{y}, \bar{x} - \underline{y}\right]$
(iv)	Multiplication	$[\underline{x}, \bar{x}] \times \left[\underline{y}, \bar{y}\right] = \left[\min\left\{\underline{x}\underline{y}; \underline{x}\bar{y}; \ \bar{x}\underline{y}; \bar{x}\bar{y}\right\}, \ \max\left\{\underline{x}\underline{y}; \underline{x}\bar{y}; \bar{x}\underline{y}; \bar{x}\bar{y}\right\}\right]$
(v)	Division	$[\underline{x}, \bar{x}] \div \left[\underline{y}, \bar{y}\right] = [\underline{x}, \bar{x}] \times \left[1/\bar{y}, 1/\underline{y}\right]$
(vi)	Function evaluation	$F([\underline{x}, \bar{x}]) = [F(\underline{x}), \ F(\bar{x})]$, F monotonically increasing
(vii)	Real distance [1]	$dist\left([\underline{x}, \bar{x}], \ \left[\underline{y}, \bar{y}\right]\right) = \max\left\{\left\lvert\underline{x} - \underline{y}\right\rvert, \ \left\lvert\bar{x} - \bar{y}\right\rvert\right\}$
(viii)	Real Absolute value [1]	$\lvert\underline{x}, \bar{x}\rvert = dist([x, \bar{x}], \ [0, 0]) = \max\{\lvert\underline{x}\rvert, \lvert\bar{x}\rvert\}$
(ix)	Absolute value [17]	$\lvert\underline{x}, \bar{x}\rvert = [\max\{\underline{x}, -\bar{x}, 0\}, \max\{-\underline{x}, \bar{x}\}]$

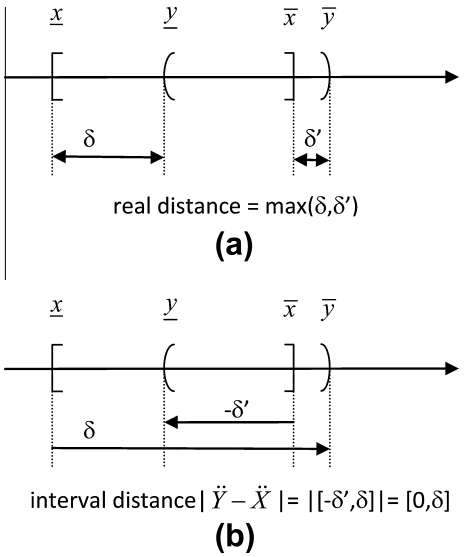

Fig. 1. Example of computation of real distance (a) and interval distance (b).

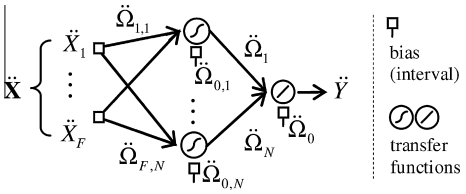

Fig. 2. The proposed MLP architecture for interval-valued data.

can be used in two ways [22]: (i) as a granular function approximation model, whose granular weights can be adjusted through supervised learning by minimizing an error function and (ii) as a tool to evaluate the prediction of a pre-adjusted MLP model subject to variable uncertainty associated with its input variables. Such input uncertainty can be characterized using interval inputs of different lengths.

4. The problem requirements and the granularity interval

As pointed out in [10], the width of the predicted output regions for an interval MLP is affected by the width of the weight intervals. Wide widths cause the propagation of large ranges of intermediate values through the network, thus generating wide output intervals. This is known as the "bound explosion" effect. To control this effect, we adopt the following procedure. Let $\left\{\ddot{\mathbf{X}},\ \ddot{Y}\right\} \in \mathbb{IR}^{F+1}$ be a set of T input–output interval-valued samples, represented as midpoint and half-width. First, we use the midpoints of T to train a "conventional" MLP which has the same structure as the interval-valued MLP to be developed. We adopt the Levenberg-Marquadt method. In this way, we create a reference model which solves the regression problem for reference points of the interval-valued data. When we tackle the regression problem for the interval-valued data, we expect that the interval weights and biases contain the numerical weights and biases ($\dot{\omega}_{i,j}^{(init)}$ and $\dot{\omega}_{j}^{(init)}$) of the reference model, respectively. Further, we expect that the widths of these intervals are constrained by the level of granularity of the mapping that is determined by the problem itself and by the desired resolution at which the user is interested in working with the data.

The first requirement is satisfied by enforcing the following relationship:

$$\dot{\omega}_{i,j}^{(init)} \in \ddot{\Omega}_{i,j} \quad \text{and} \quad \dot{\omega}_{j}^{(init)} \in \ddot{\Omega}_{j} \quad \forall i,j. \tag{8}$$

With regard to the second requirement, we enforce that the half-widths of weights and biases are bounded by an interval-valued percentage of the initial values:

$$\widehat{\omega}_{i,j} \in \left|\dot{\omega}_{i,j}^{(init)}\right| \cdot \ddot{G} \quad \text{and} \quad \widehat{\omega}_{j} \in \left|\dot{\omega}_{j}^{(init)}\right| \cdot \ddot{G} \quad \forall i,j, \tag{9}$$

where $\ddot{G} = \left[\underline{g}, \bar{g}\right] \in \mathbb{IR}^{+}$, with $\underline{g},\ \bar{g} \in \mathbb{R}^{+}$, is a granularity interval expressed in percentage which allows adapting the granularity of the mapping to the granularity level of the information. The choice of $\ddot{G}$ captures a subjective aspect of the problem:

there is no a universal value of it, since it is problem-oriented and user-dependent [29,34]. In case of a numeric-valued $\ddot{G}$, i.e. $\underline{g} = \bar{g} = g$, the midpoints and half-widths of weights and biases are prefixed by (8), (9). At the limit, for $g = 0$ the MLP is forced to be a conventional MLP, which is generally suitable only if data are numeric-valued. In general, the user provides an interval-valued $\ddot{G}$ so as to allow an optimization of weights and biases with respect to data. For instance, with very limited uncertainty in data, $\ddot{G} \to \ddot{0}$ may produce a detailed mapping. In such case, higher values of both $\underline{g}$ and $\bar{g}$ may produce a coarse-grained description of the data, by omitting irrelevant details. Hence, the granularity interval produces models of different *resolution*. The resolution of the model relates to the smallest change in the data that produces a different model. It is a different concept with respect to the *accuracy* (average error) of the model and its *precision* (its variance).

To learn the interval-valued weights and biases, standard error back propagation is likely to give poor results [10]. Indeed, the network prediction error surface is expected to be nonlinear with several local minima. A global search method is much more desirable. GAs and evolutionary strategies are, in general, effective examples of such methods.

The literature on interval-valued objective function optimization is entirely based on GA [2,8,18,30,31,35]. To the best of our knowledge, only in [10], differential evolution has been used to train an interval-valued network for a regression problem, but the fitness function is real-valued. Thus, we decided to adopt a GA as optimization tool.

5. The genetic training of the interval neural network

The chromosome coding used in the GA is shown in Fig. 3. Each gene encodes an interval weight or interval bias in terms of its midpoint and half-width. Such representation is very efficient in enforcing constraints (8) and (9).

Ranking of individuals is made according to the network error, whose computation will be discussed in the next section. First, the individuals are sorted in a descending order. The worst and best individuals are placed in positions 1 and *Nind*, respectively, where *Nind* is the number of individuals in the population. A fitness value is then assigned to each individual, depending on its position *Pos* in the sorted population, according to the linear ranking rule [32]:

$$Fitn(Pos) = 2 - SP + 2(SP - 1)\frac{Pos - 1}{Nind - 1}, \tag{10}$$

where *SP* represents the selective pressure, commonly set to 2.

The initial population is randomly generated by satisfying the constraints expressed in formulas (8) and (9). Chromosomes are selected for mating by using the linear ranking selection: the individuals are sorted according to their fitness values and the rank *Nind* is assigned to the best individual and the rank 1 to the worst individual. For each individual, the probability to be selected is linearly proportional to its rank [6].

We perform both crossover and mutation on the selected individuals. As regards the crossover, we have experimented two different crossover operators, i.e., the two-point (TP) crossover and the intermediate recombination (IR) operators. The former has the advantage of preserving the constraints fixed in formulas (8) and (9). The latter is considered more suitable for real coding chromosome [3]. Since the IR operator does not preserve the constraints, it has been adapted accordingly to generate offspring pursuant to formulas (8) and (9). The TP crossover operator consists in choosing two crossover points randomly in the two chromosomes (parents) selected for mating and exchanging the contents within these points between the two parents. In the IR operator, the values of the offspring are chosen around and between the corresponding values of the parents. More specifically, each value $\omega_{i,j}^{C}$ of the offspring C is the result of a linear combination of the corresponding values $\omega_{i,j}^{A}$ and $\omega_{i,j}^{B}$ of the parents A and B, according to the following expression: $\omega_{i,j}^{C} = \omega_{i,j}^{A} + \alpha\left(\omega_{i,j}^{B} - \omega_{i,j}^{A}\right)$, where the scaling factor $\alpha \in [-0.25, 1.25]$ is chosen for each pair of parents. In geometric terms, IR is capable of producing new values within a slightly larger hypercube, determined by the scaling factor, than that defined by parents. As mentioned above, the IR crossover operator does not preserve the constraints fixed in formulas (8) and (9). Thus, we have reformulated the above expression of $\omega_{i,j}^{C}$ as follows:

$$\widehat{\omega}_{i,j}^{C} = \min\left(\max\left(\left|\dot{\omega}_{i,j}^{(init)}\right| \cdot \underline{g},\ \widehat{\omega}_{i,j}^{A} + \alpha\left(\widehat{\omega}_{i,j}^{B} - \widehat{\omega}_{i,j}^{A}\right)\right),\ \left|\dot{\omega}_{i,j}^{(init)}\right| \cdot \bar{g}\right),$$
$$\dot{\omega}_{i,j}^{C} = \min\left(\max\left(\dot{\omega}_{i,j}^{(init)} - \widehat{\omega}_{i,j}^{C}, \dot{\omega}_{i,j}^{A} + \alpha\left(\dot{\omega}_{i,j}^{B} - \dot{\omega}_{i,j}^{A}\right)\right),\ \dot{\omega}_{i,j}^{(init)} + \widehat{\omega}_{i,j}^{C}\right).$$

Both the crossover operators are applied with some probability P_C.

The mutation operator replaces the current values of a gene by randomly extracting two values $\widehat{\omega}_{i,j}$ and $\dot{\omega}_{i,j}$ in the intervals $\left[\left|\dot{\omega}_{i,j}^{(init)}\right| \cdot \underline{g},\ \left|\dot{\omega}_{i,j}^{(init)}\right| \cdot \bar{g}\right]$ and $\left[\dot{\omega}_{i,j}^{(init)} - \widehat{\omega}_{i,j},\ \dot{\omega}_{i,j}^{(init)} + \widehat{\omega}_{i,j}\right]$, respectively. The first interval is directly related to the definition of $\ddot{G}$. Once provided $\widehat{\omega}_{i,j}$, from formula (8) we derive that the maximum distance from $\dot{\omega}_{i,j}^{(init)}$ can be $\widehat{\omega}_{i,j}$ so as to allow $\dot{\omega}_{i,j}^{(init)} \in \ddot{\Omega}_{i,j}$. Fig. 4 shows the relation between $\dot{\omega}_{i,j}^{(init)}$ and $\dot{\omega}_{i,j}$. Here $\ddot{\Omega}_{i,j}$ has been put in evidence by a shadowed strip. More specifically,

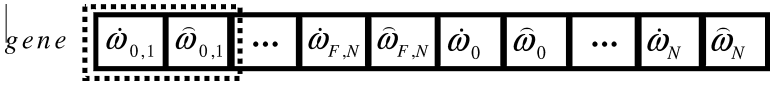

Fig. 3. The chromosome coding.

Fig. 4a and b shows, respectively, an example of value of $\dot{\omega}_{i,j}$ that obeys formula (8), i.e., such that $\dot{\omega}_{i,j}^{(init)} \in \ddot{\Omega}_{i,j}$, and an example of value of $\dot{\omega}_{i,j}$ that does not obey formula (8), i.e., such that $\dot{\omega}_{i,j}^{(init)} \notin \ddot{\Omega}_{i,j}$. The mutation operator is applied to each gene with probability γ_m/L, where γ_m is a user-defined mutation coefficient and $L = 2N(F+2)+2$ is the chromosome length.

As a termination criterion for the GA, the algorithm stops if a maximum number N_G of generations has been reached.

6. Objective function: numeric and interval valued network error

As regards the objective function, the network error functions proposed in the literature implicitly assume an isomorphism between $\mathbb{IR}$ and $\mathbb{R}^2$, which does not obey condition (6) and distorts the search space. In this paper we study two different error functions directly derived from the interval arithmetic operations shown in Table 1:

$$E = \frac{1}{T}\sum_{i=1}^{T} E_i, \quad E_i = dist\left(\ddot{Y}_i, \ddot{Y}'_i\right) \in \mathbb{R}^+, \tag{11}$$

$$\ddot{E} = \frac{1}{T}\sum_{i=1}^{T} \ddot{E}_i, \quad \ddot{E}_i = \left|\ddot{Y}_i - \ddot{Y}'_i\right| \in \mathbb{IR}^+, \tag{12}$$

where $\ddot{Y}_i$ and $\ddot{Y}'_i$ are the desired and network outputs. To minimize the interval error in (12) via the GA described in the previous section, an order relation between intervals has to be exploited for computing the fitness in (10) [8]. In the literature, there are a few studies on order relations of intervals [18,20,25]. A common strategy is to establish an order on the basis of boundaries, midpoints and half-widths. The method described in [25] uses midpoints and half-widths to obtain a membership function of the fuzzy relation "the interval $\ddot{E}$ is more/less than interval $\ddot{F}$". In [18] the authors show how the order relations between two intervals depend on the type of the problem and the decision maker's point of view. For this reason, we define the order relation between intervals with respect to decision maker's point of view for the minimization problem and under the pessimistic principle "more uncertainty is worse than less uncertainty" [19].

Given two error intervals $\ddot{E} = [\underline{e}, \bar{e}]$ and $\ddot{F} = \left[\underline{f}, \bar{f}\right]$, let us define the order relation between them as follows: $\ddot{E} \leqslant \ddot{F}$ if and only if the area covered by the pairs $(e,f) \in \ddot{E} \times \ddot{F}$ such that $e \leqslant f$ is larger than the area covered by the pairs $(e,f) \in \ddot{E} \times \ddot{F}$ such that $e > f$. $\ddot{E} \times \ddot{F}$ is the Cartesian product of the two intervals on $\mathbb{IR}^2$. Let us denote with $\mathcal{A}(S)$ the area covered in the Cartesian product by the set S.

More specifically, the two intervals $\ddot{E} = [\underline{e}, \bar{e}]$ and $\ddot{F} = \left[\underline{f}, \bar{f}\right]$ can be in the following three different reciprocal relations, shown in Fig. 5.

(a) *Disjoining*: $\ddot{E} \cap \ddot{F} = \varnothing$. If $\bar{e} < \underline{f}$ then $e < f \forall e \in \ddot{E}, \quad f \in \ddot{F}$, i.e., all the elements of $\ddot{E}$ are lower than the elements of $\ddot{F}$. Hence $\ddot{E} < \ddot{F} \iff \bar{e} < f$.

(b) *Including*: $\ddot{E} \subseteq \ddot{F}$ i.e., $\ddot{E}$ is a subset of $\ddot{F}$. Let $\ddot{F}_{LEFT}$ and $\ddot{F}_{RIGHT}$ be the subset of $\ddot{F}$ at the left and right of $\ddot{E}$, respectively. The following relations hold:

(b.1) $\mathcal{A}\{(e,f) : e \leqslant f\} = \mathcal{A}\{(e,f) \in \ddot{E} \times \ddot{E} : e \leqslant f\} + \mathcal{A}\left\{(e,f) \in \ddot{E} \times \ddot{F}_{RIGHT}\right\} = \mathcal{A}_{\cap} + (\bar{e} - \underline{e})(\bar{f} - \bar{e})$.

(b.2) $\mathcal{A}\{(e,f) : e > f\} = \mathcal{A}\left\{(e,f) \in \ddot{E} \times \ddot{E} : e > f\right\} + \mathcal{A}\left\{(e,f) \in \ddot{E} \times \ddot{F}_{LEFT}\right\} = \mathcal{A}_{\cap} + (\bar{e} - \underline{e})\left(\underline{e} - \underline{f}\right)$.

From b.1 and b.2 we can conclude that $\ddot{E} \leqslant \ddot{F} \iff \widehat{F}_{LEFT} \leqslant \widehat{F}_{RIGHT}$.

(c) *Overlapping*: $\ddot{E} \cap \ddot{F} \neq \varnothing$, $\ddot{E} \setminus \ddot{F} \neq \varnothing$ and $\ddot{F} \setminus \ddot{E} \neq \varnothing$. The following relations hold:

(c.1) $\mathcal{A}\{(e,f) : e \leqslant f\} = \mathcal{A}\{(e,f) \in \left(\ddot{E} \cap \ddot{F}\right) \times \left(\ddot{E} \cap \ddot{F}\right) : e \leqslant f\} + \mathcal{A}\left\{(e,f) \in \left(\ddot{E} \setminus \ddot{F}\right) \times \left(\ddot{F} \setminus \ddot{E}\right)\right\} + \mathcal{A}\left\{(e,f) \in \left(\ddot{E} \setminus \ddot{F}\right) \times \left(\ddot{E} \cap \ddot{F}\right)\right\}$
$+\mathcal{A}\left\{(e,f) \in \left(\ddot{E} \cap \ddot{F}\right) \times \left(\ddot{F} \setminus \ddot{E}\right)\right\} = \mathcal{A}_{\cap} + \left(\underline{f} - \underline{e}\right)(\bar{f} - \bar{e}) + \left(\underline{f} - \underline{e}\right)\left(\bar{e} - \underline{f}\right) + \left(\bar{e} - \underline{f}\right)(\bar{f} - \bar{e}) = \mathcal{A}_{\cap} + \left(\underline{f} - \underline{e}\right)\left(\bar{f} - \underline{f}\right)$
$+\left(\bar{e} - \underline{f}\right)(\bar{f} - \bar{e})$.

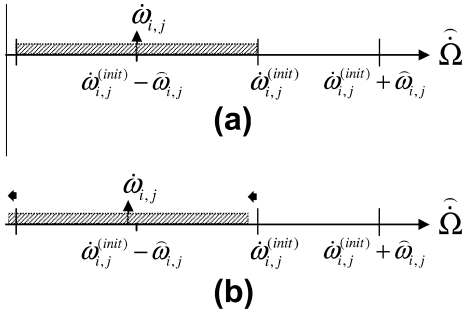

Fig. 4. Two scenarios of value for $\dot{\omega}_{i,j}$: (a) $\dot{\omega}_{i,j}^{(init)} \in \ddot{\Omega}_{i,j}$ and (b) $\dot{\omega}_{i,j}^{(init)} \notin \ddot{\Omega}_{i,j}$.

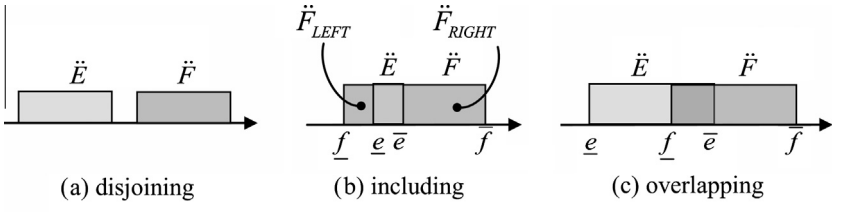

Fig. 5. Types of intervals: (a) disjoining; (b) including; and (c) overlapping.

(c.2) $\mathcal{A}\{(e,f) : e > f\} = \mathcal{A}\left\{(e,f) \in \left(\ddot{E} \cap \ddot{F}\right) \times \left(\ddot{E} \cap \ddot{F}\right) : e > f\right\} = \mathcal{A}_{\cap}$.

Fig. 6 shows an example of the area $\mathcal{A}_{\cap}$. From c.1 and c.2 we can conclude that $\ddot{E} \leqslant \ddot{F} \iff \left(\underline{f} - \underline{e}\right)\left(\overline{f} - \underline{f}\right) + \left(\overline{e} - \underline{f}\right)\left(\overline{f} - \overline{e}\right) \geqslant 0$, which is true in the scenario shown in Fig. 5c. It is worth noting that if $\overline{e} = \underline{f}$ holds, $A_{\cap} = \varnothing$, and then $A\{(e,f) : e \leqslant f\} = (\overline{e} - \underline{e})\left(\overline{f} - \underline{f}\right)$, i.e. the total area covered by the two intervals, and obviously $\ddot{E} \leqslant \ddot{F}$.

7. Experimental studies

Although several works have been published in the field of interval-valued data, unfortunately there is still a lack of significant benchmark datasets for interval-valued data regression. In this section, we discuss the application of our interval-valued neural architecture to four real world and two synthetic datasets. In all experiments, the data are normalized between 0 and 1 (by subtracting the minimum value and dividing the data by the difference between the maximum and the minimum values). In what follows, we consider two GAs, which adopt the numeric-valued error E (formula (11)) and the interval-valued error $\ddot{E}$ (formula (12)), respectively. We denote these GAs as GA-NE and GA-IE, respectively.

7.1. Setting user parameters and crossover

In this subsection we present the methodology for setting the parameters of the system and then evaluate the performances of the two crossover operators. First of all, we observe that the values of parameters of the GA scheme and the MLP depend on the type of dataset.

Table 2 briefly characterizes the interval datasets used in the experiments.

We adopt the following heuristic procedure for determining the main parameters which affect the performance of our approach, namely, granularity interval, number of neurons in the hidden layer in the interval MLP and number of individuals in the population of the GA. First of all, we determine the granularity interval. As discussed in Section 4, the choice of the granularity interval is basically problem-oriented and user-dependent. Indeed, the lower bound of the interval determines the level of resolution with which the user wants to analyze the data: the higher this level, the lower the resolution. Thus, if the user would like to use the maximum resolution, he fixes the lower bound to 0; otherwise, he tries different increasing values until the desired resolution level is determined. To perform this analysis on the dataset considered in the paper, we adopted a simple interval MLP with 5 neurons in the hidden layer and 12 individuals in the population of the GA. The width of the granularity interval imposes a constraint on the variation of weights and biases during the optimization process so as

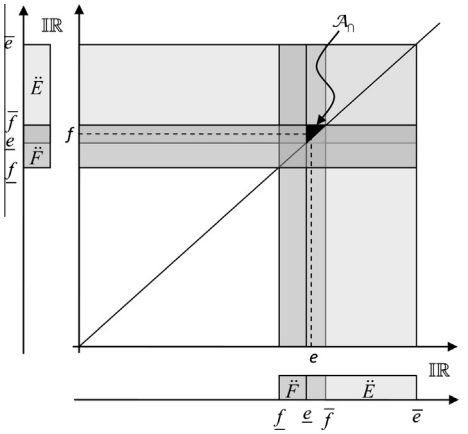

Fig. 6. An example of the area $\mathcal{A}_{\cap}$ of the Cartesian product in the *Overlapping* scenario.

to limit the search space. A small width can constrain excessively the exploration of the GA, preventing it from achieving satisfactory results. On the other hand, a large width increases the search space, thus requiring the execution of a high number of generations of the GA for producing an optimal interval MLP. We have verified that variations of this width, which do not constrain too much the search space, do not affect significantly the performance of the GA. Once we have determined the granularity interval, we choose the other parameters by following a procedure which is similar to the one typically adopted in the design of conventional MLPs. We first determine the lower and upper bounds of the number of neurons in the hidden layer which do not determine underfitting and overfitting. Underfitting and overfitting are detected by the low accuracy on the training set and by the high difference in accuracy between training and testing sets, respectively. Within these bounds, we test three different values. In general, the number of neurons in the hidden layer needed by the interval MLP to achieve a good accuracy is not high. This is mainly due to the characteristics of the interval-valued data. Indeed, intervals provide a means for abstracting the complexity and the noise in the data. Hence, a paradigm shift from numeric-valued to interval-valued data (via granulation or human induction) brings to a reduction of both the data complexity and noise. It follows that a low number of neurons in the hidden layer is already sufficient to fit the interval-valued dataset. We consider the number of neurons as the first parameter to be analyzed because this value is strongly related to the complexity of the dataset and therefore considerably affects the performance of the interval MLP. The analysis is performed by executing the GA for 100 generations with a population of 12 individuals. This starting setting has been determined as follows. With regard to the generations, we have verified that 100 generations are sufficient to achieve a stable interval-valued error, that is, the error does not significantly decrease with the increase of the number of generations. Hence, the parameter N_G has been definitively set to 100. With regard to the population size, we have verified that a value of about ten individuals is a lower bound to avoid quality issues when setting the other parameters. Finally, we analyze if the increase of the number of the individuals in the population produces a decrease of the interval-valued error which can justify the increase of the computational time. We have verified in practice that, even with a non-optimal parameters setting, populations larger than about twenty individuals are not beneficial to the optimization. For this reason, we test values from 12 to 24.

The parameters P_C and γ_m have been set to 0.4, and 0.7, respectively. These values have been easily determined by means of the common practice adopted in GAs. Indeed, they are typical values, and are largely independent of the dataset.

To numerically assess the model performance on each combination of parameters, we adopt a 10-fold cross-validation.

At the end of this procedure, we choose the combination of the parameters which generates the best interval MLP in terms of interval-valued error.

For the sake of brevity, we do not discuss all the experiments performed on all the datasets. Just to give a glimpse to the reader, we show in Table 3 and 4 the most significant results obtained on the parameterization of the GA-IE on the Salary dataset. We show the data obtained by applying the two different crossover operators experimented in the paper: the TP crossover (Table 3) and the IR (Table 4) operators, respectively [3].

In the tables, N, I and $\ddot{G}$ denote the number of neurons in the hidden layer, the number of individuals in the population and the granularity interval, respectively. In the tables, we show the combination of the parameters in the same sequence in which they have been evaluated following the procedure described above. First of all, we have assessed the granularity interval. Since we were interested in using the maximum resolution, we set the lower bound to 0. In the tables, we note that, when the width of the interval is too small, the GA cannot explore the search space adequately (high interval-valued error in the training set). On the other hand, when the width is large, the GA has difficulties to find the optimal interval MLP. We adopted the intermediate width 4% for evaluating the other parameters.

We can observe that when N is too low ($N = 2$), the resulting interval MLP suffers from underfitting (high interval-valued error on the training set). On the contrary, when N is high ($N = 8$), the interval MLP suffers from overfitting (high difference between the inter-valued errors computed on the training and testing sets, respectively). Once fixed the number of neurons to $N = 5$, we analyze how the number of individuals affects the final result. We observe from the tables that just with 12 individuals the GA achieves the best and also the most stable results. Thus, we choose the combination $N = 5$, $I = 12$, $\ddot{G} = [0, 4]\%$ (shown in boldface in the two tables).

In Table 5 we show the results obtained by the GAs executed with IR and TP crossover operators for all the datasets. We executed the GAs for the same number of fitness evaluations. The values in the table are obtained by applying a 10-fold cross-validation. We note that the interval MLPs generated by using the TP crossover operator generally achieve lower interval-valued errors than the ones generated by using the IR operator. This result shows that the choice of the GA with the TP crossover operator as optimization tool is a suitable option for the specific problem. Also, the TP crossover operator preserves

Table 2
A brief summary of the interval-valued datasets used in the experiments.

Data set name	No. of samples	Description	Type
Salary	30	US Salary by years of experience (1989)	Real
Peak	189	Dataset with a high density of intervals	Synthetic
Micro-Spread	50	Spread Germany-Portugal in the last decade	Real
Macro-Spread	20	Spread Germany-Portugal in the last decade	Real
Heart	90	Time series of heartbeats of a man with CHF	Real
Wave	400	Three dimensional dataset	Synthetic

the constraints determined by formulas (8) and (9). Thus, we can conclude that the TP crossover operator is preferred to the IR operator in this specific context.

7.2. The Salary dataset

The Salary dataset [23] shown in Fig. 7 consists of 30 interval-valued samples which represent the range of salaries by years of experience for American males with degree in 1989.

The original data samples are not granular, and are subject to significant sampling error. We performed a granulation process as follows. First fuzzy information granules have been generated via Fuzzy C-Means (FCM) clustering [5]. Hence, an alpha-cut of 0.05 has been applied to the resulting fuzzy partition. Finally, interval-valued data have been derived considering, for each alpha-cut, the smallest containing rectangle.

We determined $\ddot{G} = [0, 4]\%,\ N = 5$, and $I = 12$.

In order to highlight the structural quality of the system, Fig. 8 shows a scenario of function fit for the conventional MLP trained by using the midpoints of the intervals. For the scenario, the Mean Squared Error (MSE) is 3.7×10^{-6}. The effectiveness of this process is crucial to provide the neurons in the hidden layer with reliable reference weights and biases $\left(\dot{\omega}_{i,j}^{(init)}\right)$. This stage is typically very fast and accurate.

Table 6 shows the mean values ± the standard deviations of the interval-valued error $\ddot{E}$ on training and testing sets achieved by the GA-NE and GA-IE. With the aim of comparing the two approaches, we have computed and reported the interval-valued error $\ddot{E}$ also for the best solution generated by the GA-NE. Due to the definition of the absolute value operator, the lower bound of $\ddot{E}$ is 0 when 0 belongs to the interval-valued error. This condition has been always true in our experiments. For this reason, the variability of the interval-valued error $\ddot{E}$ with lower bound $\underline{e}$ equal to 0 has been determined only by the variation of the upper bound $\bar{e}$. Thus, we show the standard deviation of $\bar{e}$ in the table. We can observe that GA-IE outperforms GA-NE in terms of interval-valued error $\ddot{E}$. Further, the interval-valued errors $\ddot{E}$ computed on training and testing sets are very similar, thus confirming the good generalization features of the network.

Fig. 9 shows the weights and biases for the hidden and output layers. We observe that many weights and biases are really interval-valued, thus reflecting an amount of uncertainty existing in data. Furthermore, there is no individual weight/bias that may be responsible for the majority of the uncertainty, thus confirming a good balancing of the tuning process. Since the scenario visualized in Fig. 9 is common to all the experiments, for the sake of brevity we do not show the corresponding plots for the other datasets.

Finally, Fig. 10 shows the target output and the network output of an MLP trained by using GA-IE. We observe that the network approximates very well the target intervals.

7.3. The Peak dataset

The Peak dataset presented in Fig. 11 consists of 189 synthetic interval-valued samples, generated from a noisy sampling of the function $K_0 + K_1 \sin(x_1)$ with high distortion in the ordinate. We determined $\ddot{G} = [0.4, 4]\%,\ N = 9$, and $I = 12$.

Table 7 shows the mean values ± the standard deviations of the interval error $\ddot{E}$ on training and testing sets. Again, we observe that GA-IE outperforms GA-NE in terms of the interval-valued error $\ddot{E}$.

In order to highlight the overall quality of the training process, Fig. 12 shows the plot of the MSE versus the number of epochs for the conventional MLP trained by using the midpoints of the intervals. We can observe that the MSE gets stable after a few epochs, with a good function fit. Hence, the conventional MLP achieves convergence in a very few epochs and therefore is not computationally demanding.

Table 3
Salary dataset, interval-valued error $\ddot{E}$ on training and testing sets, produced by different interval MLPs, generated by GA-IE after the same number of fitness evaluations, using different combinations of parameters and the two-point crossover operator.

Parameters	Training set	Testing set
$N:5, I:12, \ddot{G}:[0,1]\%$	[0, 0.116 ± 0.049]	[0, 0.104 ± 0.050]
$N:5, I:12, \ddot{G}:[0,4]\%$	[0, 0.097 ± 0.017]	[0, 0.098 ± 0.037]
$N:5, I:12, \ddot{G}:[0,7]\%$	[0, 0.112 ± 0.017]	[0, 0.110 ± 0.045]
$N:2, I:12, \ddot{G}:[0,4]\%$	[0, 0.117 ± 0.041]	[0, 0.113 ± 0.048]
$\boldsymbol{N:5, I:12, \ddot{G}:[0,4]\%}$	**[0, 0.097 ± 0.017]**	**[0, 0.098 ± 0.037]**
$N:8, I:12, \ddot{G}:[0,4]\%$	[0, 0.092 ± 0.029]	[0, 0.244 ± 0.127]
$N:5, I:12, \ddot{G}:[0,4]\%$	[0, 0.097 ± 0.017]	[0, 0.098 ± 0.037]
$N:5, I:15, \ddot{G}:[0,4]\%$	[0, 0.101 ± 0.010]	[0, 0.111 ± 0.054]
$N:5, I:18, \ddot{G}:[0,4]\%$	[0, 0.105 ± 0.017]	[0, 0.120 ± 0.069]

Table 4
Salary dataset, interval-valued error $\ddot{E}$ on training and testing sets, produced by different interval MLPs, generated by GA-IE after the same number of fitness evaluations, using different combinations of parameters and the intermediate recombination operator.

Parameters	Training set	Testing set
$N:5, I:12, \ddot{G}:[0,1]\%$	[0, 0.108 ± 0.014]	[0, 0.108 ± 0.051]
$N:5, I:12, \ddot{G}:[0,4]\%$	[0, 0.100 ± 0.007]	[0, 0.103 ± 0.045]
$N:5, I:12, \ddot{G}:[0,7]\%$	[0, 0.115 ± 0.021]	[0, 0.121 ± 0.052]
$N:2, I:12, \ddot{G}:[0,4]\%$	[0, 0.118 ± 0.049]	[0, 0.121 ± 0.067]
$\boldsymbol{N:5, I:12, \ddot{G}:[0,4]\%}$	**[0, 0.100 ± 0.007]**	**[0, 0.103 ± 0.045]**
$N:8, I:12, \ddot{G}:[0,4]\%$	[0, 0.095 ± 0.023]	[0, 0.203 ± 0.075]
$N:5, I:12, \ddot{G}:[0,4]\%$	[0, 0.100 ± 0.007]	[0, 0.103 ± 0.045]
$N:5, I:15, \ddot{G}:[0,4]\%$	[0, 0.102 ± 0.009]	[0, 0.113 ± 0.067]
$N:5, I:18, \ddot{G}:[0,4]\%$	[0, 0.108 ± 0.007]	[0, 0.115 ± 0.055]

Table 5
Comparison in terms of interval-valued error $\ddot{E}$ on training and testing sets between the IR and TP crossover operators.

Data set	Parameters	Training set	Testing set
Salary	$N:5, I):12, \ddot{G}:[0,4]\%, C:$ IR	[0, 0.100 ± 0.007]	[0, 0.103 ± 0.045]
	$N:5, I:12, \ddot{G}:[0,4]\%, C:$ TP	[0, 0.097 ± 0.017]	[0, 0.098 ± 0.037]
Heart	$N:8, I:12, \ddot{G}:[0,3]\%, C:$ IR	[0, 0.128 ± 0.026]	[0, 0.135 ± 0.038]
	$N:8, I:12, \ddot{G}:[0,3]\%, C:$ TP	[0, 0.124 ± 0.033]	[0, 0.129 ± 0.038]
Peak	$N:9, I:12, \ddot{G}:[0.4,4]\%, C:$ IR	[0, 0.062 ± 0.0098]	[0, 0.064 ± 0.0013]
	$N:9, I:12, \ddot{G}:[0.4,4]\%, C:$ TP	[0, 0.037 ± 0.0027]	[0, 0.035 ± 0.0025]
Micro-Spread	$N:12, I:12, \ddot{G}:[0,0.1]\%, C:$ IR	[0, 0.065 ± 0.0161]	[0, 0.075 ± 0.0364]
	$N:12, I:12, \ddot{G}:[0,0.1]\%, C:$ TP	[0, 0.048 ± 0.0080]	[0, 0.053 ± 0.0010]
Macro-Spread	$N:7, I:12, \ddot{G}:[0,1]\%, C:$ IR	[0, 0.097 ± 0.0033]	[0, 0.12 ± 0.0047]
	$N:7, I:12, \ddot{G}:[0,1]\%, C:$ TP	[0, 0.092 ± 0.0031]	[0, 0.10 ± 0.0032]
Wave	$N:8, I:12, \ddot{G}:[0,0.3]\%, C:$ IR	[0, 0.16 ± 0.02]	[0, 0.16 ± 0.02]
	$N:8, I:12, \ddot{G}:[0,0.3]\%, C:$ TP	[0, 0.15 ± 0.02]	[0, 0.16 ± 0.02]

Fig. 7. The Salary dataset.

Figs. 13 and 14 show the numeric-valued and interval-valued training errors versus generations, for GA-NE and GA-IE respectively. In the case of GA-IE, we plotted only the upper bound $\bar{e}$ of $\ddot{E}$ since the lower bound is always zero. We can observe that both the errors become stable after 100 generations. It is worth noting that the two figures are not directly comparable with each other because they refer to the two different definitions of the error. However, to allow a comparative analysis of the final networks, in Table 7 the errors are calculated by using the same performance index, i.e., the interval-valued error.

Finally, Fig. 15 shows the target output and the network output of an MLP trained by using GA-IE. We observe that the network approximates very well the target intervals.

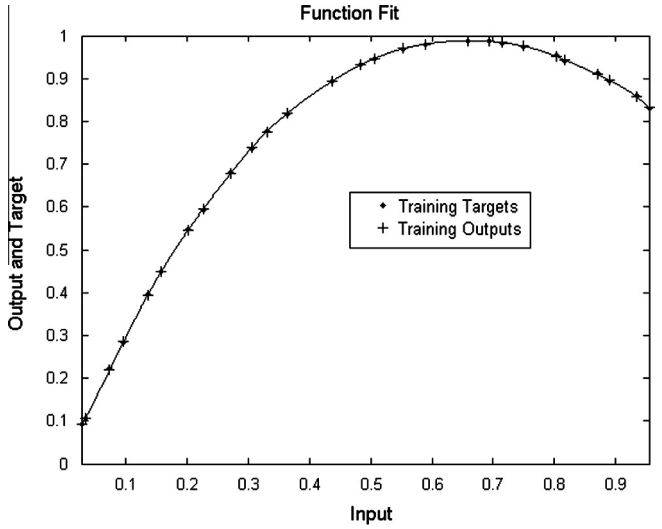

Fig. 8. Salary dataset: the function fit for the conventional MLP trained with the midpoints.

Table 6
The Salary dataset: interval-valued error $\ddot{E}$ on training and testing sets.

GA	Training set	Testing set
GA-NE	[0, 0.17 ± 0.029]	[0, 0.167 ± 0.049]
GA-IE	[0, 0.097 ± 0.017]	[0, 0.098 ± 0.037]

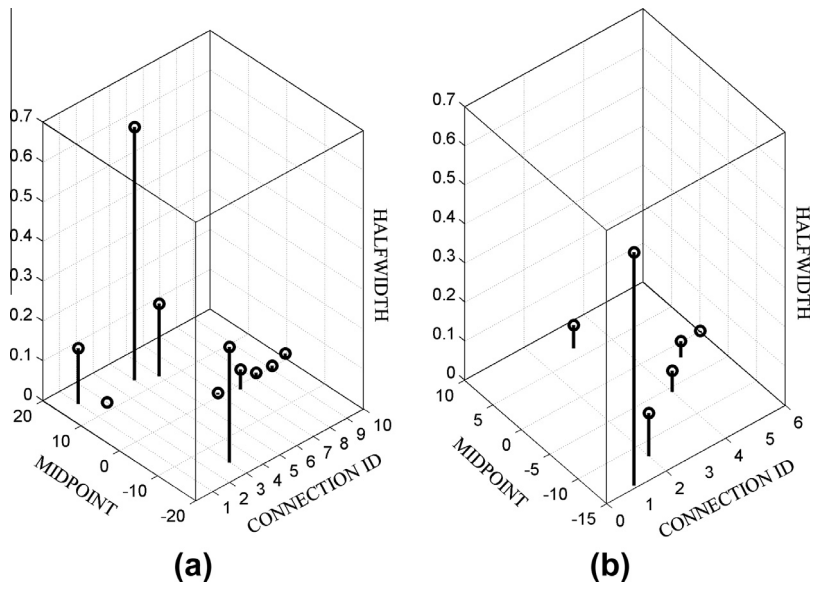

Fig. 9. Salary dataset: weights and biases in the hidden (a) and output (b) layers of an interval MLP obtained in a trial of GA-IE.

7.4. The Spread dataset

The Spread dataset represents the spread Germany–Portugal in the last decade.[1] As most macroeconomic indicators, the original data samples are subject to local uncertainty. Typically, monthly general economic statistics are more volatile and subject to higher revisions than lower frequency data. For instance, quarterly rates are usually quite stable, with low revisions in terms of percentage points.

[1] www.bloomberg.com.

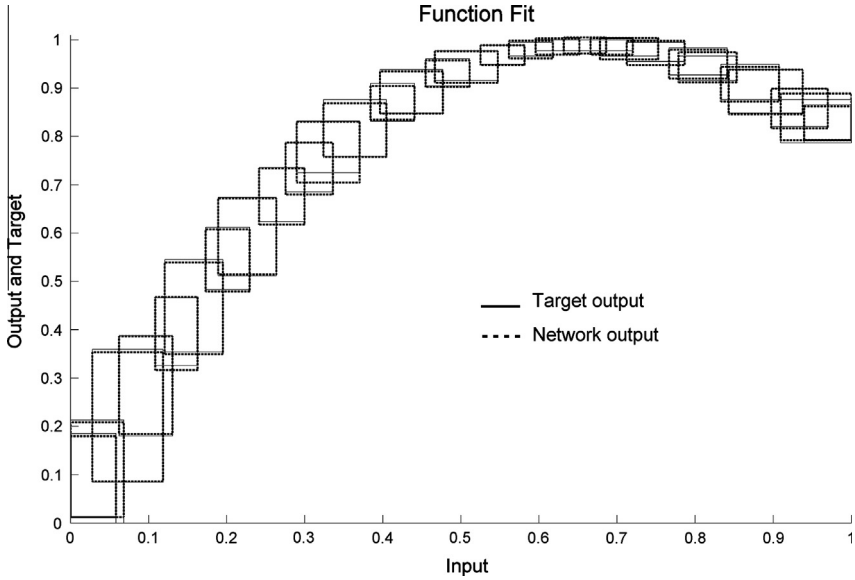

Fig. 10. Salary dataset: outputs of an interval MLP obtained in a trial of GA-IE versus the target outputs.

We performed the same granulation process employed for the Salary dataset, generating two interval-valued datasets with different resolutions: (a) the Micro-Spread dataset, using 50 clusters and alpha-cut equal to 0.4; (b) the Macro-Spread dataset, using 20 clusters and alpha-cut equal to 0.3. Both datasets are shown in Fig. 16. The granularity intervals used to observe the mapping are $\ddot{G}_a = [0, 0.1]\%$ and $\ddot{G}_b = [0, 1]\%$ for the Micro-Spread and the Macro-Spread datasets, respectively. We also determined a different number N of neurons in the hidden layer, 12 and 7 neurons, respectively, according to the complexity of the mapping.

Tables 8 and 9 show the mean values ± the standard deviations of the interval-valued error $\ddot{E}$ on training and testing sets, for the Micro-Spread and Macro-Spread datasets, respectively. We can observe that GA-IE outperforms GA-NE in terms of interval-valued error $\ddot{E}$.

Fig. 17a and b shows the granular outputs of two MLPs trained by using GA-IE.

7.5. The Heart dataset

The *c5nn* Heart dataset [12] comes from a time series of heartbeats of a 61-year-old man with congestive heart failure (CHF), whose chaotic signature is known in the literature [33]. We performed the same granulation process employed for the Salary and the Spread datasets. Fig. 18 shows 650 samples of the original dataset, with a granulation process made

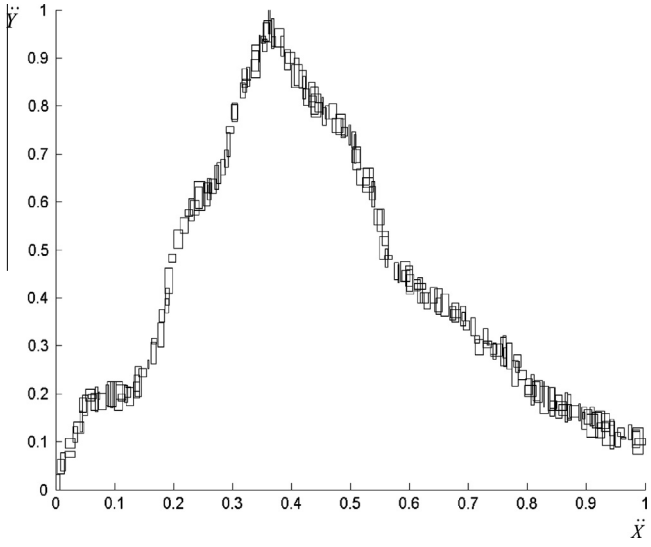

Fig. 11. The Peak dataset.

Table 7
The Peak dataset: interval-valued error $\ddot{E}$ on training and testing sets.

GA	Training set	Testing set
GA-NE	[0, 0.045 ± 0.0024]	[0, 0.049 ± 0.0028]
GA-IE	[0, 0.037 ± 0.0027]	[0, 0.035 ± 0.0025]

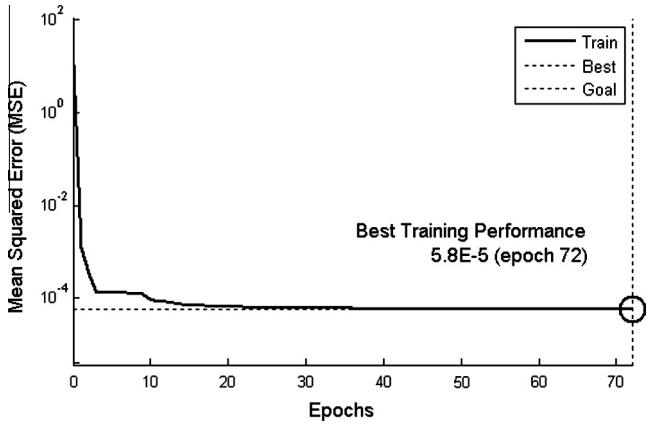

Fig. 12. Peak dataset: MSE on the training set versus the number of epochs for the conventional MLP trained with the midpoints.

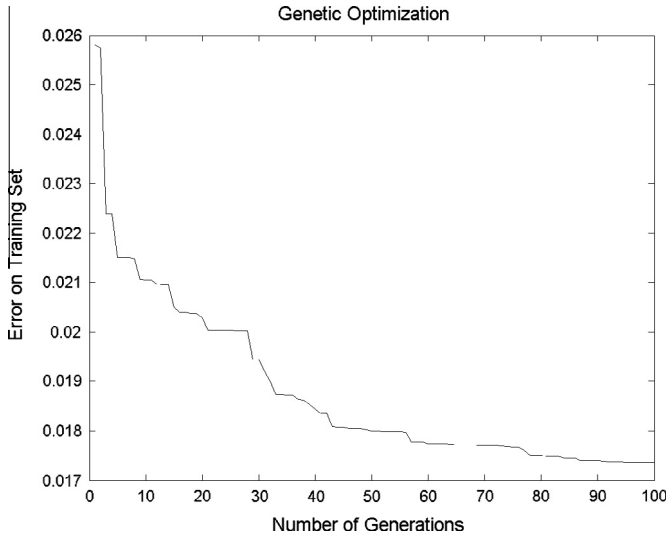

Fig. 13. Peak dataset: numeric-valued training error versus number of generations in a trial for GA-NE.

by using 90 clusters and alpha-cut equals to 0.15. These values are adequate for hiding the micro fluctuations, so as to allow the analysis to focus on the macro heart rate variability.

We determined $\ddot{G} = [0, 3]\%$, $N = 8$ neurons in the hidden layer, and $I = 12$ individuals in the GA population. Table 10 shows the mean values ± the standard deviation of the interval-valued error $\ddot{E}$ on training and testing sets. We can observe that GA-IE outperforms GA-NE in terms of interval-valued error $\ddot{E}$.

Fig. 19 shows the granular outputs of an interval MLP trained using GA-IE. We note that the MLP approximates well the target output intervals.

7.6. The Wave dataset

The Wave dataset shown in Fig. 20 consists of 400 synthetic interval-valued samples in the three-dimensional space, generated from a noisy sampling of the function $K_0 + K_1 \sin(x_1)\cos(x_2)$. We use this dataset for analyzing the differences between the mappings with different granularity intervals on the same dataset.

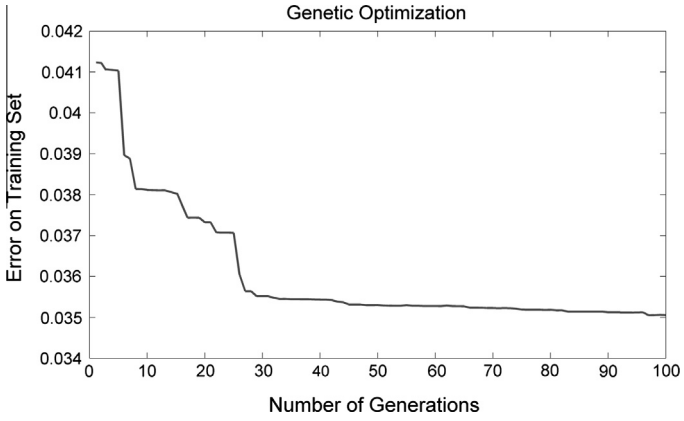

Fig. 14. Peak dataset: interval-valued training error versus number of generations in a trial for GA-IE.

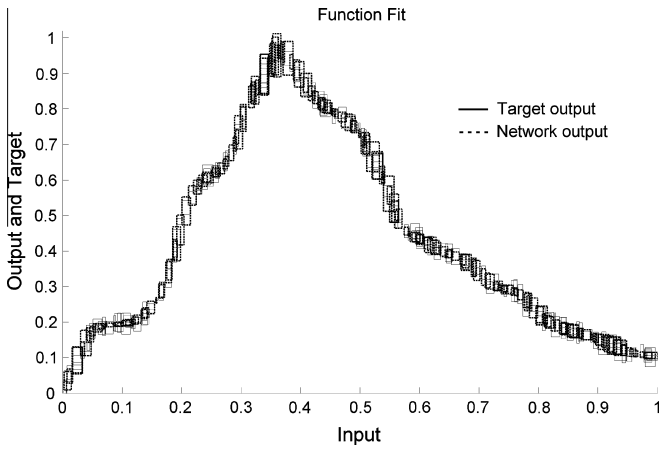

Fig. 15. Peak dataset: outputs of an interval MLP obtained in a trial of GA-IE versus the target outputs.

Fig. 21 shows a high resolution mapping of an MLP obtained in a trial of GA-IE. The MLP contains 8 neurons in the hidden layer and is trained with $\ddot{G} = [0, 0.3]\%$ in 30 generations. For this scenario, the final upper bounds of the interval-valued error $\ddot{E}$ on training and test sets are 0.15561 and 0.15714, respectively. Table 11 shows the mean values ± the standard deviations of the interval-valued error $\ddot{E}$ on training and testing sets. We can observe that GA-IE considerably outperforms GA-NE.

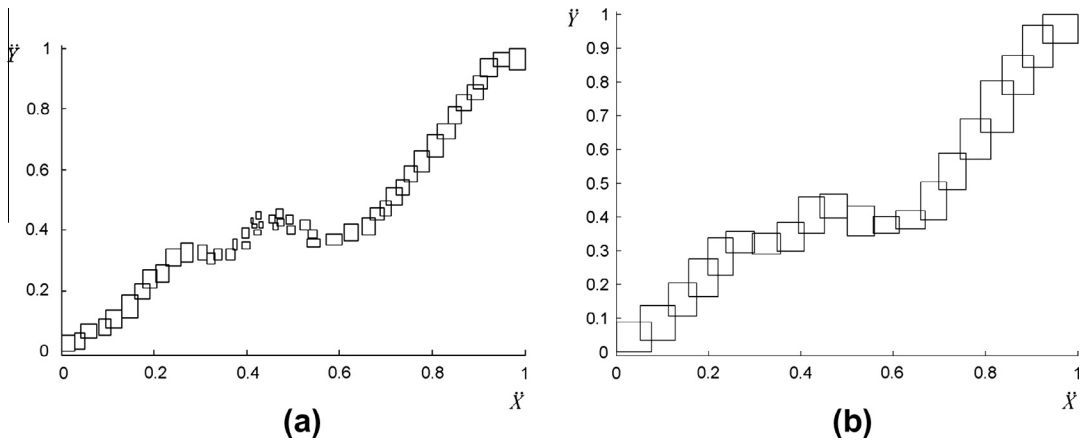

Fig. 16. The Micro-Spread (a) and the Macro-Spread (b) datasets.

Table 8
The Micro-Spread dataset: interval-valued error $\ddot{E}$ on training and testing sets.

GA	Training set	Testing set
GA-NE	[0, 0.063 ± 0.0080]	[0, 0.080 ± 0.0027]
GA-IE	[0, 0.048 ± 0.0080]	[0, 0.053 ± 0.0010]

Table 9
The Macro-Spread dataset: interval-valued error $\ddot{E}$ on training and testing sets.

GA	Training set	Testing set
GA-NE	[0, 0.11 ± 0.0029]	[0, 0.11 ± 0.0041]
GA-IE	[0, 0.092 ± 0.0031]	[0, 0.10 ± 0.0032]

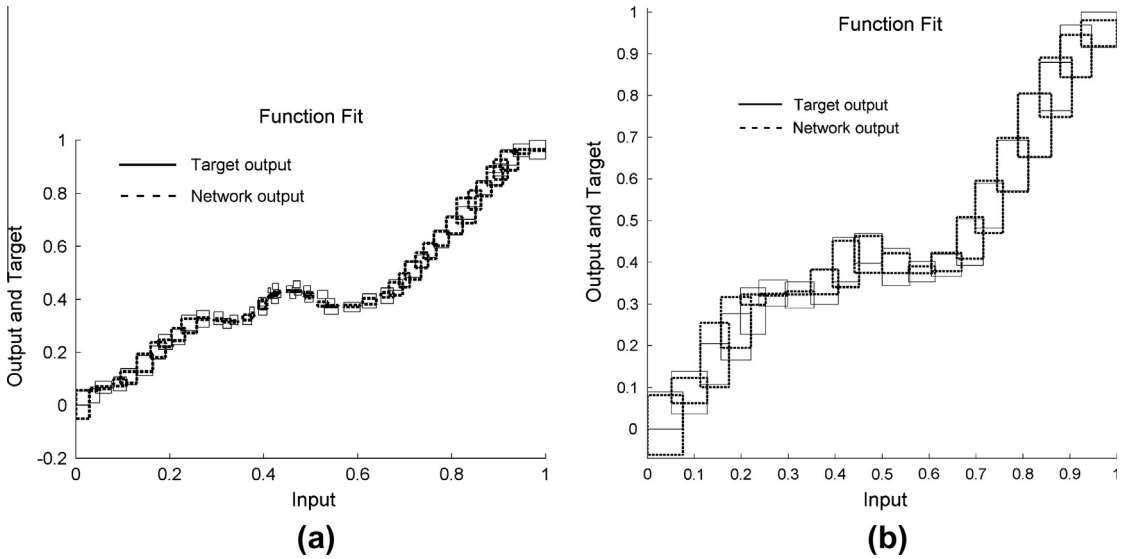

Fig. 17. Outputs of an interval MLP obtained in a trial of GA-IE versus the target output: (a) Micro-Spread dataset and (b) Macro-Spread dataset.

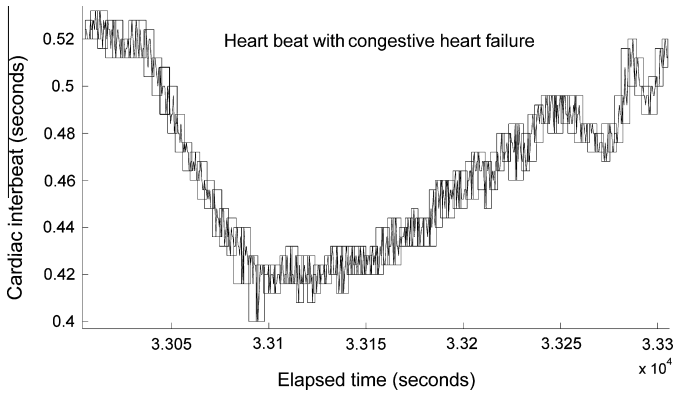

Fig. 18. The Heart dataset.

Fig. 22 shows a low resolution mapping of an MLP, obtained in a trial of GA-IE by raising the granularity interval with respect to the high resolution model. The MLP contains 5 neurons and is trained with $\ddot{G} = [0.3, 0.6]\%$ for 10 generations. For this scenario, the final upper bounds of the interval-valued error $\ddot{E}$ on training and testing sets are 0.25189 and

Table 10
The Heart dataset: interval-valued error $\ddot{E}$ on training and testing sets.

GA	Training set	Testing set
GA-NE	[0, 0.186 ± 0.071]	[0, 0.194 ± 0.063]
GA-IE	[0, 0.124 ± 0.033]	[0, 0.129 ± 0.038]

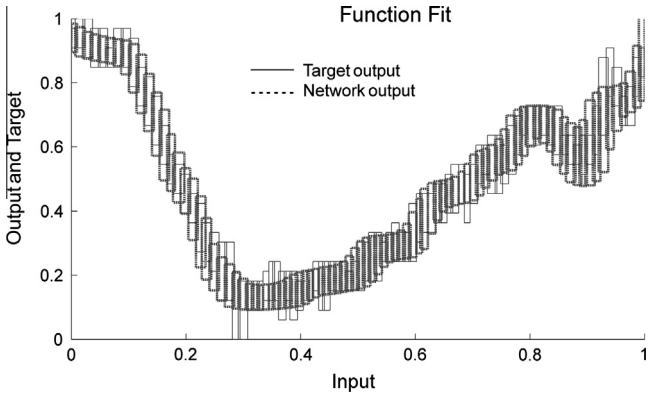

Fig. 19. The Heart dataset: outputs of an interval MLP obtained in a trial of GA-IE versus the target output.

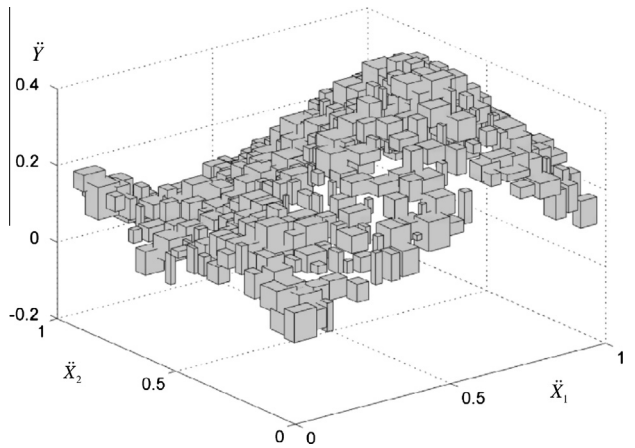

Fig. 20. The *Wave* dataset.

0.26489, respectively. We can observe that, due to the lower complexity, the low resolution mapping can be efficiently achieved by reducing the number of neurons and the number of generations with respect to the high resolution mapping.

8. Conclusions and future work

We have proposed a new learning method based on genetic algorithms for a general interval-valued MLP architecture. The training process is mainly managed by a user-centric granulation process and by an interval-valued objective function. In particular, we have introduced the notion of granularity interval, which specifies the resolution of the mapping with respect to the granularity level of the information, and have designed the GA according to the constraints fixed by this user parameter. In the design of the GA, we have also compared two different exploitation strategies. Further, we have introduced a new interval-valued network error, which allows us to perform an optimization coherent with the interval semantics. Finally, we have introduced an order relation between intervals to be used by the genetic algorithm. The effectiveness of the approach, in terms of generalization capabilities and multi-resolution mapping, has been quantified by reporting a numeric-valued and an interval-valued network error on six datasets.

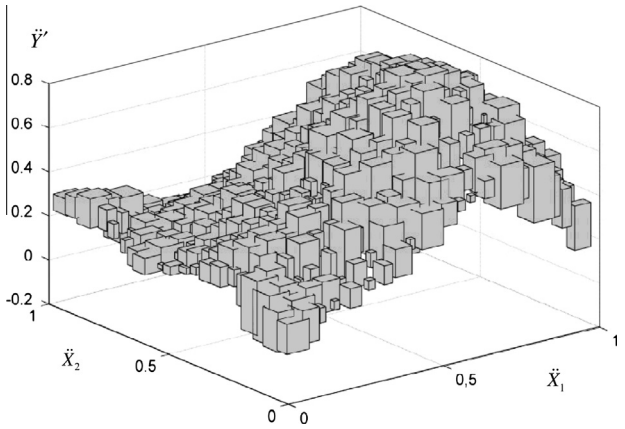

Fig. 21. *Wave* dataset: a high resolution mapping of an interval MLP obtained in a trial of GA-IE.

Table 11
The Wave dataset: interval-valued error $\ddot{E}$ on training and testing sets.

GA	Training set	Testing set
GA-NE	[0, 0.22 ± 0.11]	[0, 0.21 ± 0.09]
GA-IE	[0, 0.15 ± 0.02]	[0, 0.16 ± 0.02]

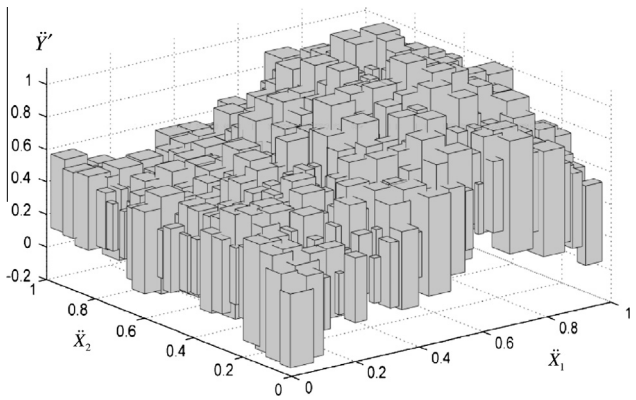

Fig. 22. *Wave* dataset: a low resolution mapping of an interval MLP obtained in a trial of GA-IE.

Although the experimental results have shown that the GA has been an adequate choice, other evolutionary strategies may be effective in solving the type of optimization problem tackled in this work. Thus, as future research, we plan to investigate the use of differential evolution and particle swarm optimization strategies, and to make a comparative analysis with the use of GA. Further, we are going to experiment other order relations between intervals in the computation of the fitness function, so as to take different decision maker's points of view into account. Finally, we would like to extend the interval-valued input–output mapping approach to other neural network architectures.

References

[1] G. Alefeld, D. Claudio, The basic properties of interval arithmetic, its software realizations and some applications, Computers & Structures 67 (1998) 3–8.
[2] J.-M. Alliot, N. Durand, D. Gianazza, J.-B. Gotteland, Finding and proving the optimum: cooperative stochastic and deterministic search, in: Proceedings of 20th European Conference on Artificial Intelligence (ECAI'12), Montpellier, France, 2012, pp. 55–60.
[3] T. Back, A.E. Eiben, Generalizations of intermediate recombination in evolution strategies, in: Proceedings of the Congress on Evolutionary Computation (CEC'99), vol. 2, 1999, pp. 1566–1573.
[4] A. Bernat, V. Kreinovich, T.J. McLean, G.N. Solopchenko, What are interval computations and how are they related to quality in manufacturing? in: Proceedings of the International Workshop on Applications of Interval Computations (APIC'95), El Paso, Texas, 1995, pp. 10–12.
[5] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York, 1981.

[6] T. Blickle, L. Thiele, A comparison of selection schemes used in evolutionary algorithms, Evolutionary Computation 4 (4) (1996) 361–394.
[7] M.R. Baker, Universal approximation theorem for interval neural network, Reliable Computing 4 (1998) 235–239.
[8] A.K. Bhunia, L. Sahoo, Genetic algorithm based reliability optimization in interval environment, Innovative Computing Methods, vol. 357, Springer-Verlag, Berlin, SCI, 2011, pp. 13–36.
[9] R. Callejas-Bedregal, B.R.C. Bedregal, R.H.N. Santiago, Generalizing the real interval arithmetic, TEMA – Trends in Computational and Applied Mathematics 3 (1) (2002) 61–70.
[10] D. Chetwynd, K. Worden, G. Manson, An application of interval-valued neural networks to a regression problem, Proceedings of the Royal Society A 462 (2006) 3097–3114.
[11] M.G.C.A. Cimino, B. Lazzerini, F. Marcelloni, W. Pedrycz, Granular data regression with neural networks, in: A. Petrosino et al. (Eds.), Proceedings of LNCS 9th International Workshop on Fuzzy Logic and Applications, WILF 2011, LNAI 6857, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 172–179.
[12] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.Ch. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.-K. Peng, H.E. Stanley, PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000) e215–e220.
[13] J. Han, T.Y. Lin, Granular computing: models and applications, International Journal of Intelligent Systems 25 (2) (2009) 111–117.
[14] H. Ishibuchi, H. Tanaka, H. Okada, Architecture of neural networks with interval weights and its application to fuzzy regression analysis, Fuzzy Sets and Systems 57 (1993) 27–39.
[15] R.B. Kearfott, Interval computations: introduction, uses, and resources, Euromath Bulletin 2 (1996) 95–112.
[16] V. Kreinovich, Data processing beyond traditional statistics: applications of interval computations. a brief introduction, in: Proceedings of International Workshop on Applications of Interval Computations (APIC'95), El Paso, Texas, 1995, pp. 13–21.
[17] B. Lambov, Interval arithmetic using SSE-2, in: P. Hertling, C.M. Hoffmann, W. Luther, N. Revol (Eds.): Reliable Implementation of Real Number Algorithms: Theory and Practice, 08.01.–13.01.2006, Dagstuhl Seminar Proceedings 06021, Dagstuhl, Germany, 2006.
[18] J. Majumdar, A.K. Bhunia, Elitist genetic algorithm for assignment problem with imprecise goal, European Journal of Operational Research 177 (2007) 684–692.
[19] J. Majumdar, A.K. Bhunia, Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times, Journal of Computational and Applied Mathematics 235 (2011) 3063–3078.
[20] J. Majumdar, A.K. Bhunia, Solving airline crew-scheduling problem with imprecise service time using genetic algorithm, Advanced Modeling and Optimization 12 (2) (2010) 141–159.
[21] M. Marino, F. Palumbo, Interval linear regression: an application to soil permeability analysis, in: Proceedings of Intermediate SIS Conference – Multivariate Analysis for Economic and Social Sciences, Earth Sciences and Technology, Naple, June, 2003.
[22] A. Muñoz San Roque, C. Maté, J. Arroyo, Á. Sarabia, iMLP: applying multi-layer perceptrons to interval-valued data, Neural Processsing Letters 25 (2007) 157–169.
[23] K.M. Murphy, F. Welch, The structure of wages, Quarterly Journal of Economics 1 (1992) 285–326.
[24] A.V. Nandedkar, P.K. Biswas, A reflex fuzzy min max neural network for granular data classification, in: Proceedings of the 18th International Conference on Pattern Recognition, ICPR'06, Vol. 2, IEEE Computer Society, Washington, DC, USA, 2006, pp. 650–653.
[25] A. Niewiadomski, J. Ochelska, P.S. Szczepaniak, Interval-valued linguistic summaries of databases, Control and Cybernetics 35 (2) (2006) 415–443.
[26] R.E. Patiño-Escarcina, B.R. Callejas Bedregal, A. Lyra, Interval computing in neural networks: one layer interval neural networks, Intelligent Information Technology 3356 (2005) 68–75.
[27] W. Pedrycz, B. Russo, G. Succi, Knowledge transfer in system modeling and its realization through an optimal allocation of information granularity, Applied Soft Computing 12 (8) (2012) 1985–1995.
[28] W. Pedrycz, G. Vukovich, Granular neural networks, Neurocomputing 36 (2001) 205–224.
[29] W. Pedrycz, Knowledge-Based Clustering: From Data to Information Granules, John Wiley & Sons, Hoboken, NJ, 2005.
[30] L. Sahoo, A.K. Bhunia, D. Roy, A genetic algorithm based reliability redundancy optimization for interval valued reliabilities of components, Journal of Applied Quantitative Methods 5 (2) (2010) 270–287.
[31] L. Sahoo, A.K. Bhumia, D. Roy, An application of genetic algorithm in solving reliability optimization problem under interval component Weibull parameters, Mexical Journal of Operators Research 1 (1) (2012) 2–19.
[32] D. Whitley, The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best, in: Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA'89), Morgan Kaufmann Publishers, San Fransisco, CA, USA, 1989, pp. 116–121.
[33] G.-Q. Wu, N.M. Arzeno, L.-L. She, D.-K. Tang, D.A. Zheng, et al, Chaotic signatures of heart rate variability and its power spectrum in health, aging and heart failure, PLoS ONE 4 (2) (2009) e4323–e4332.
[34] Y.Y. Yao, Granular computing, Computer Science 31 (2004) 4–10.
[35] X. Zhang, S. Liu, A new interval-genetic algorithm, Proceedings of the Third International Conference on Natural Computation (ICNC'07), vol. 4, IEEE Computer Society, Washington, DC, USA, 2007, pp. 193–197.