gsaINknn: A GSA Optimized, Lattice Computing knn Classifier

Yazdan Jamshidi [a], Vassilis G. Kaburlasos [b,*]

[a] *Dept. of Computer Engineering, Kermanshah Science and Research Branch. Islamic Azad University, Kermanshah, Iran*

[b] *Human-Machines Interaction (HMI) Lab, Dept. of Computer & Informatics Engineering. Eastern Macedonia and Thrace Institute of Technology, Kavala, 65404 Greece*

[*] *Corresponding author E-mail*: *vgkabs@teikav.edu.gr*

Abstract: This work proposes an effective synergy of the Intervals' Number k- nearest neighbor (INknn) classifier, that is a granular extension of the conventional knn classifier in the metric lattice of Intervals' Numbers (INs), with the gravitational search algorithm (GSA) for stochastic search and optimization. Hence, the gsaINknn classifier emerges whose effectiveness is demonstrated here on twelve benchmark classification datasets. The experimental results show that the gsaINknn classifier compares favorably with alternative classifiers from the literature. The far-reaching potential of the gsaINknn classifier in computing with words is also delineated.

Keywords: Computing with words (CWW), Classification, Gravitational search algorithm (GSA), Intervals' number (IN), k-nearest neighbor (knn), Lattice computing (LC)

1. Introduction

This work introduces a stochastically optimized, granular k- nearest neighbor (knn) classifier based on INs, where IN stands for Intervals' Number.

An IN is a mathematical object, which may represent either a fuzzy interval or a probability distribution [44]. In any case, an IN can be interpreted as an information granule. INs have been studied in a series of publications. In particular, as explained in [31], it has been shown that the set $\mathfrak{F}_1$ of INs is a metric lattice with cardinality $\aleph_1$, where $\aleph_1$ is the cardinality of the set $\mathfrak{R}$ of real numbers; moreover, $\mathfrak{F}_1$ is a cone in a linear space. Note that previous work [20, 21] has employed the term FIN (i.e., fuzzy interval number) instead of the term IN because it stressed a fuzzy interpretation. Likewise, the term CALFIN, proposed previously for an algorithm which induces a FIN from a population of measurements, was later replaced by the term CALCIN [44]. Recall that an IN computed by algorithm CALCIN retains all-order data statistics [31]. In the aforementioned context, the capacity as well as the rich potential of INs, especially in industrial applications, has been demonstrated [24, 26, 44].

INs have been used in an array of computational intelligence applications regarding clustering, classification and regression [25, 26, 27, 30, 31, 44, 45]. There is experimental evidence that a parametric, IN-based scheme can be optimized toward clearly improving performance. More specifically, optimization has been pursued by stochastic search techniques including genetic algorithms [27, 30, 31, 44] and particle swarm optimization [45] since, currently, there are no analytic optimization methods available in the lattice of INs.

Previous works have frequently employed an inclusion measure (σ) function as an instrument for decision making in the lattice of INs [26, 30, 31, 45]. The interest of this work is in (fuzzy) nearest neighbor classification [10]. Note that, lately, a number of knn classifiers based on INs, namely INknn classifiers, have been introduced [32, 43, 61]; nevertheless, none of the latter classifiers was optimized. On the grounds of compelling evidence, as explained above, this work proposes an optimized INknn classifier toward improving performance. We remark that various heuristic optimization methods have been proposed in machine learning including Simulated Annealing (SA) [6], Ant Colony [7], Particle Swarm Optimization (PSO) algorithms [17, 40], Differential Evolution (DE) [37, 54], Genetic Algorithms (GAs) [46], etc. Lately, Rashedi and colleagues have proposed the gravitational search algorithm (GSA) [47], that is a swarm based meta-heuristic search algorithm based on the Newtonian laws of gravity. The GSA has already been successfully applied to numerous problems [2, 36, 48, 49, 50, 51, 53, 60, 66]. This work proposes a synergy of GSA with the INknn classifier toward improving the capacity of the latter classifier. Hence, the (granular) gsaINknn classifier emerges whose capacity is demonstrated here comparatively on twelve benchmark datasets.

In a more general context, the proposed gsaINknn classifier is a scheme of the emerging lattice computing (LC) paradigm. Note that LC was originally defined as “the collection of Computational Intelligence tools and techniques that either make use of lattice operators inf and sup for the construction of the computational algorithms or exploit Lattice Theory for language representation and reasoning" [12]. Recent work has extended the meaning of LC as "an evolving collection of tools and methodologies that process lattice ordered data *per se* including logic values, numbers, sets, symbols, graphs, etc" [24, 31, 33]. The LC paradigm provides instruments for granular computing, where uncertainty/ambiguity is accommodated in partially/lattice-ordered information granules [19, 22, 39, 57].

A number of LC models have already been proposed in the context of mathematical morphology. For instance, morphological neural networks (MNN) including both morphological perceptrons and fuzzy morphological associative memories (FMAMs) [56, 57, 58, 62] can be classified as LC models. In particular, Sussner and colleagues have employed a FMAM to implement a fuzzy inference system based on the complete lattice structure of the class of fuzzy sets [59, 63, 64]. Furthermore, Graña and colleagues have applied LC techniques to image analysis applications of mathematical morphology [14, 15, 16]. Of particular interest in LC is the notion of a *fuzzy lattice*, which has been proposed by Nanda toward fuzzifying a partial order relation [42]. Working independently Kaburlasos and colleagues, inspired from the adaptive resonance theory (ART) for neural computation [4, 5], have proposed a number of *fuzzy lattice neural networks* for clustering and classification [21] operating on fuzzy lattice reasoning (FLR) principles. The FLR classifier was introduced in [29] for inducing descriptive decision-making knowledge (rules) in a mathematical lattice data domain, including the space $\mathfrak{R}^N$ as a special case; moreover, the FLR classifier has been successfully applied to a variety of problems such as ambient ozone estimation as well as air quality assessment [1]. Recent trends in lattice computing appear in [13, 23, 28].

The layout of this paper is as follows. Section 2 outlines the mathematical background. Section 3 presents the INknn classifier including an explanatory application example. Section 4 describes the GSA optimization algorithm. Section 5 details the gsaINknn classifier. Section 6 presents comparatively experimental results regarding twelve benchmark classification datasets. Finally, section 7 concludes by both summarizing our contribution and delineating future work.

2. Mathematical background

This section outlines general lattice notions followed by a hierarchy of lattices ending up to Intervals' Numbers, or INs for short.

2.1 General lattices

A set P with a *partial order* (binary) relation $\sqsubseteq$ is called *partially ordered set* or *poset* for short, symbolically $(P,\sqsubseteq)$ [3, 21, 24, 26, 52]. A function φ: $P\to Q$ from a poset $(P,\sqsubseteq)$ to a poset $(Q,\sqsubseteq)$ is called *isomorphic* iff $x\sqsubseteq y \Leftrightarrow \varphi(x)\sqsubseteq\varphi(y)$. It is well known that the inverse $\sqsupseteq$, namely *dual* (order), of an order relation $\sqsubseteq$ is itself an order relation. A *lattice* $(\mathfrak{L},\sqsubseteq)$ is a poset with the additional property that any two of its elements $a,b\in\mathfrak{L}$ have both an *infimum* denoted by $a\sqcap b = \inf\{a,b\}$ and a *supremum* denoted by $a\sqcup b = \sup\{a,b\}$. The lattice operations $\sqcap$ and $\sqcup$ are called *meet* and *join*, respectively. A lattice $(\mathfrak{L},\sqsubseteq)$ is called *complete* when each of its subsets has a supremum as well as an infimum in $\mathfrak{L}$. A non-void complete lattice has both a least element and a greatest element denoted by o and i, respectively. A lattice $(\mathfrak{L},\sqsubseteq)$ is called *totally-ordered* iff for $a,b\in\mathfrak{L}$ it is either $a\sqsupseteq b$ or $a\sqsubset b$. In this work we use ''square symbols'' such as $\sqcup$, $\sqcap$ and $\sqsubseteq$ with general lattice elements, ''straight symbols'' $\vee$, $\wedge$ and $\leq$ with real numbers, and symbols $\cup$, $\cap$ and $\subseteq$ with sets.

An *aggregate* lattice $(\mathfrak{L},\sqsubseteq)$ is the Cartesian product of N *component* lattices $\mathfrak{L}_1,\ldots,\mathfrak{L}_N$; i.e. $(\mathfrak{L},\sqsubseteq) = (\mathfrak{L}_1,\sqsubseteq_1)\times\ldots\times(\mathfrak{L}_N,\sqsubseteq_N)$. The product lattice $\mathfrak{L}$ operations join and meet are defined as

$$(a_1,\ldots,a_N)\sqcup(b_1,\ldots,b_N) = (a_1\sqcup_1 b_1,\ldots,a_N\sqcup_1 b_N) \quad \text{and} \quad (a_1,\ldots,a_N)\sqcap(b_1,\ldots,b_N) = (a_1\sqcap_N b_1,\ldots,a_N\sqcap_N b_N)$$

A *valuation* on a lattice $(\mathfrak{L},\sqsubseteq)$ is a real function v: $\mathfrak{L}\rightarrow\mathfrak{R}$ which satisfies $v(a)+v(b) = v(a\sqcap b)+v(a\sqcup b)$. A valuation v is called *positive* iff $a\sqsubset b$ implies $v(a)<v(b)$. A positive valuation function v: $\mathfrak{L}\rightarrow\mathfrak{R}$ implies a metric function d: $\mathfrak{L}\times\mathfrak{L}\rightarrow\mathfrak{R}_0^+$ given by $d(x,y) = v(x\sqcup y)-v(x\sqcap y)$.

Generalized interval is an element of the product lattice $(\mathfrak{L},\sqsupseteq)\times(\mathfrak{L},\sqsubseteq) = (\mathfrak{L}\times\mathfrak{L},\sqsupseteq\times\sqsubseteq)$. The latter lattice may simply be denoted by $(\Delta,\sqsubseteq)$. A generalized interval is denoted by $[a,b]$. The ordering $(\sqsubseteq)$, join $(\sqcup)$ and meet $(\sqcap)$ operations in lattice $(\Delta,\sqsubseteq)$ are given as follows:

$$[a,b] \sqsubseteq [c,d] \Leftrightarrow (c\sqsubseteq a \text{ and } b\sqsubseteq d), \qquad [a,b] \sqcup [c,d] = [a\sqcap c,b\sqcup d], \quad \text{and} \quad [a,b] \sqcap [c,d] = [a\sqcup c,b\sqcap d]$$

Here, we are interested in a *dual isomorphic* function θ: $\mathfrak{L}\rightarrow\mathfrak{L}$ on a general lattice $(\mathfrak{L},\sqsubseteq)$ such that $x\sqsubset y \Leftrightarrow \varphi(x)\sqsupset\varphi(y)$. Based on both a positive valuation function v: $\mathfrak{L}\rightarrow\mathfrak{R}$ and a dual isomorphic function θ: $\mathfrak{L}\rightarrow\mathfrak{L}$ on a general lattice $(\mathfrak{L},\sqsubseteq)$, a positive valuation function v_Δ is defined on lattice $(\Delta,\sqsubseteq)$ as follows: $v_\Delta([a,b]) = v(\theta(a)) + v(b)$.

2.2 A hierarchy of complete lattices

Next, we constructively develop a hierarchy of lattices from a *reference set* $\mathfrak{L}\subseteq\overline{\mathfrak{R}}$, where $\overline{\mathfrak{R}} = \mathfrak{R}\cup\{-\infty,+\infty\}$ is the totally-ordered set of *extended real numbers*. In particular, we choose $\mathfrak{L}$ so that $(\mathfrak{L},\leq)$ is a complete lattice. For example, $\mathfrak{L}$ can be $\overline{\mathfrak{R}}$ itself or it might be an interval $[a,b]\subset\overline{\mathfrak{R}}$. In particular, for $\mathfrak{L}=\overline{\mathfrak{R}}$ it is o=-∞ and i=+∞, whereas for $\mathfrak{L}=[a,b]$ it is o=a and i=b.

Any strictly increasing real function v: $\mathfrak{L}\rightarrow\mathfrak{R}$ is a positive valuation on lattice $(\mathfrak{L},\leq)$. Moreover, any strictly decreasing function θ: $\mathfrak{L}\rightarrow\mathfrak{L}$ is a dual isomorphic function on $(\mathfrak{L},\leq)$. Note that choosing a suitable valuation function is problem dependent [18, 29, 38].

Consider the lattice $(\Delta,\sqsubseteq)$ of generalized intervals stemming from lattice $(\mathfrak{L},\leq)$. A metric distance function d_Δ: $\Delta\times\Delta\rightarrow\mathfrak{R}_0^+$ is defined on $(\Delta,\sqsubseteq)$ as follows:

$$d_\Delta([a,b],[c,d]) = v_\Delta([a,b]\sqcup[c,d]) - v_\Delta([a,b]\sqcap[c,d]) = v_\Delta([a\wedge c,b\vee d]) - v_\Delta([a\vee c,b\wedge d]) \qquad (1)$$

We define the set of *conventional intervals* as $\mathfrak{J}(\mathfrak{L}) = \{[a,b]: a,b\in\mathfrak{L} \text{ and } a\sqsubseteq b\}$. Augmenting $\mathfrak{J}(\mathfrak{L})$ by the empty interval, denoted by O, there follows the complete lattice $(\mathfrak{J}_1 = \mathfrak{J}_1(\mathfrak{L}) = \mathfrak{J}(\mathfrak{L})\cup\{O\},\subseteq)$, namely lattice of *Type-1 intervals*, or simply *intervals*, ordered by the set inclusion relation ($\subseteq$). The least and greatest element in lattice $(\mathfrak{J}_1,\subseteq)$ are denoted by $O=[i,o]$ and $I=[o,i]=\mathfrak{L}$, respectively.

Consider the following definition.

Definition 1: A Type-1 *Intervals' Number*, or (*Type-1*) *IN* for short, is a (Type-1) interval function $F_{.}$: $[0,1]\rightarrow\mathfrak{J}_1$ which satisfies 1) $h_1 \le h_2 \Rightarrow F_{h_1} \supseteq F_{h_2}$ and 2) $\forall P \subseteq [0,1]$: $\bigcap_{h\in P} F_h = F_{\vee P}$.

Let $\mathfrak{F}_1$ denote the set of INs. It turns out that $(\mathfrak{F}_1,\preceq)$ is a complete lattice with order $F \preceq G \Leftrightarrow (\forall h\in[0,1]: F_h\subseteq G_h) \Leftrightarrow (\forall x\in\mathfrak{L}: F(x)\le G(x))$. We remark that there are two equivalent representations for an IN F, namely the *interval-representation* $F_{(.)}$: $[0,1]\rightarrow\mathfrak{J}_1$ and the *membership-function-representation* $F(.)$: $\mathfrak{R}\rightarrow[0,1]$. An IN F is called *trivial* iff F_h is a *trivial interval*, i.e. $F_h = [a,a]$, $\forall h\in[0,1]$ and $a\in\mathfrak{L}$. The following proposition introduces a metric on lattice $(\mathfrak{F}_1,\preceq)$.

Proposition 1: Let F and G be INs in lattice $(\mathfrak{F}_1,\preceq)$. Assuming that the following integral exists, a metric function $d_{\mathfrak{F}_1}: \mathfrak{F}_1\times\mathfrak{F}_1\rightarrow\mathfrak{R}_0^+$ is given by $d_{\mathfrak{F}_1}(F,G) = \int_0^1 d_\Delta(F_h,G_h)dh$.

Given two N-tuple INs $\bar{F} = (F_1,\ldots,F_N)\in\mathfrak{F}_1^N$ and $\bar{G} = (G_1,\ldots,G_N)\in\mathfrak{F}_1^N$, a Minkowski metric $d_p: \mathfrak{F}_1^N\times\mathfrak{F}_1^N\rightarrow\mathfrak{R}_0^+$ is defined as $d_p(\bar{F},\bar{G}) = \left[\left(d_{\mathfrak{F}_1}(F_1,G_1)\right)^p + \ldots + \left(d_{\mathfrak{F}_1}(F_N,G_N)\right)^p\right]^{1/p}$.

A *Type-2 interval* is defined as an *interval of Type-1 intervals*; moreover, a *Type-2 IN* is defined as an *interval of Type-1 INs* [30, 31]. It turns out that the aforementioned Minkowski metric can extend to N-tuple Type-2 INs.

An IN F can be induced from a set $\{x_1,\ldots,x_n\}$ of data samples by defining a piecewise linear, strictly increasing, cumulative real function c: $\mathfrak{R}\rightarrow\mathfrak{R}_0^+$ such that $c(x_i) = \frac{1}{n}|\{x_j: j\in\{1,\ldots,n\}$ and x_j

$\leq x_i\}|$, $i\in\{1,\ldots,n\}$; where $|S|$ denotes the cardinality of the set S. Let x_0 be such that $c(x_0) = 0.5$. Finally, IN F is defined such that for $x \leq x_0$ it is $F(x) = 2c(x)$, whereas for $x > x_0$ it is $F(x) = 2(1-c(x))$ as illustrated in [30] (section 4.1). In practice, the membership function $F(.)$: $\mathfrak{L}\rightarrow[0,1]$ is specified by two vectors *pts* and *val* which hold the corresponding abscissa values and ordinate values, respectively, of the membership function F. In the context of this work, we define the *weight* of an IN F as the number of data samples used to induce IN F.

3. The INknn classifier

In the following we present an extension, namely INknn, of the conventional knn classifier for k=1 in the metric lattice $(\mathfrak{F}_1,\preceq)^{\mathrm{N}}$ of INs (see in **The INknn algorithm**).

In words, the INknn classifier operates by calculating the (metric) distance of an unlabeled testing datum $\overline{F}_0 \in \mathfrak{F}_1^{\mathrm{N}}$ from all the labeled training data. In conclusion, $\overline{F}_0$ is assigned the label of its nearest training datum $\overline{F}_{\mathrm{J}}$ provided that the corresponding distance is less than a user-defined threshold μ_0. Otherwise, if $d_2(\overline{F}_0,\overline{F}_{\mathrm{J}}) > \mu_0$, then $\overline{F}_0$ is of "unknown" class. Note that the proposed INknn classifier uses the Minkowski metric $d_2 : \mathfrak{F}_1^{\mathrm{N}} \times \mathfrak{F}_1^{\mathrm{N}} \rightarrow \mathfrak{R}_0^{+}$.

The following example illustrates the relation between IN *weight* (the latter was defined at the very end of the previous section) and *Classification Accuracy*; the latter is the percentage of correctly classified data and it can be computed by Eq.(8).

The INknn algorithm

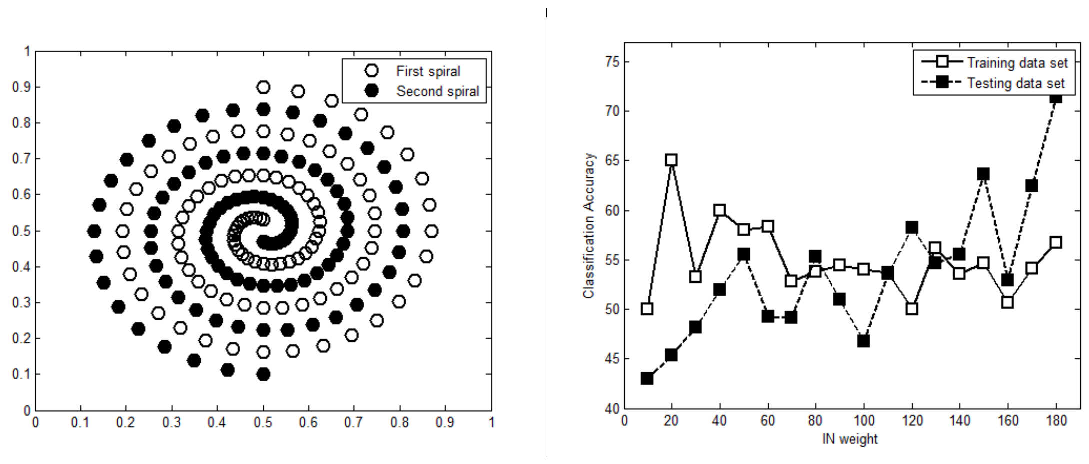
1. Consider labeled, training data $\{(\overline{F}_1,c_1),\ldots,(\overline{F}_n,c_n)\}$, where $\overline{F}_i \in \mathfrak{F}_1^{\mathrm{N}}$, $c_i\in L$ for $i\in\{1,\ldots,n\}$ and L is a set of class labels.
2. Consider a new, unlabeled (testing) datum $\overline{F}_0 \in \mathfrak{F}_1^{\mathrm{N}}$ for classification.
3. For each training datum $\overline{F}_i$, $i\in\{1,\ldots,n\}$ compute the Minkowski distance $d_2(\overline{F}_0,\overline{F}_i)$.
4. Let $J = \arg\min_{i\in\{1,\ldots,n\}}\{d_2(\overline{F}_0,\overline{F}_i)\}$.
5. The class label of $\overline{F}_0$ is defined to be $c_J\in L$ given that $d_2(\overline{F}_0,\overline{F}_{\mathrm{J}})\leq\mu_0$, where μ_0 is a user-defined threshold; otherwise, if $d_2(\overline{F}_0,\overline{F}_{\mathrm{J}}) > \mu_0$, then the class of $\overline{F}_0$ is "unknown".

An application example: The spiral problem involves a nontrivial benchmark dataset for testing the capacity of a learning scheme [5, 35]. The corresponding dataset includes sample points (x_n,y_n) from two 2D spirals given by (x_n,y_n)= ($r_n\sin a_n$+0.5, $r_n\cos a_n$+0.5), where $r_n = 0.4\left(\frac{105-n}{104}\right)$, $a_n = \frac{\pi(n-1)}{16}$. In particular, here we employed sample points (x_n,y_n), n=1,…,97 from the first spiral/class as well as sample points (1-x_n,1-y_n), n=1,…,97 from the second spiral/class shown in Fig.1(a); hence, we employed a total of 97+97 = 194 sample points or, equivalently, 194 data.

We carried out a series of INknn classification experiments such that, after shuffling randomly the 194 data before an experiment, we engaged 10,20,…,170 and 180 data, respectively, for training. In particular, half of the engaged data originated from each spiral class. In conclusion, in each experiment we used all the training data available from a class in order to compute a single (2-tuple) IN per class. Hence, two (2-tuple) INs were calculated per experiment with (sum of) weights equal to 10,20,…,170 and 180, respectively. The previous completes step 1 of the INknn algorithm. In the remaining steps of the INknn algorithm, a testing datum (x_n,y_n) was represented by a pair ([x_n,x_n],[y_n,y_n]) of trivial INs for h$\in$[0,1].

Fig.1(b) displays the Classification Accuracy regarding the training/testing datasets versus the (sum of) IN weights per experiment. In particular, Fig.1(b) shows that accommodating ever more data into the two INs, first, typically keeps the Classification Accuracy on the training dataset in the range 50%-60% due to the intermingling of the two classes and, second, it progressively increases Classification Accuracy on the testing dataset due to an improvement of INknn's capacity for generalization especially as the number of the testing data decreases.

The optimal number of INs per class depends on the data. For instance, were all the data of a class both near each other and apart from the data in any other class then one (N-tuple) IN per class would suffice; otherwise, more than one INs are required per class. The optimal number of INs in a specific application is not known "a priori" but rather it is to be induced from the data.



(a) (b)

Fig. 1. (a) The two-classes Spiral dataset. **(b)** Training/testing Classification Accuracy versus IN weight.

4. The gravitational search algorithm

This section delineates the Gravitational Search Algorithm, or GSA for short, that is a population-based stochastic search heuristic [47, 48], where a population member is called *agent*. An agent represents (part of) the training data with k (N-tuple) INs per class in N data dimensions. The mechanics of GSA is inspired from the Newtonian laws of gravity and motion. More specifically, a GSA agent is dealt with as a physical object with mass. The position of an agent represents a solution to an optimization problem, whereas an agent's mass represents the corresponding Classification Accuracy such that larger masses correspond to better solutions. All agents interact with one another with gravity forces resulting in motion. During the life time of GSA, agents with large masses grow larger and move more slowly toward computing better solutions. The basic equations of the GSA are presented next.

Given a user-defined number N_a of agents, we define the position X_i of agent i as

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^N),\ i = 1,\ldots,N_a, \tag{2}$$

where N is the search space dimension. A *normalized* mass for agent i is computed as follows

$$M_i(t) = \frac{fit_i(t) - worst(t)}{\sum_{j=1}^{N_a}\left(fit_j(t) - worst(t)\right)},\ i = 1,\ldots,N_a, \tag{3}$$

where *fit*$_i$(*t*) is the *fitness* value (calculated by Eq.(8)) of agent i at time $t \in \{1,\ldots,\mathrm{T}\}$, $worst(t)$ for a maximization (resp. minimization) optimization problem is the minimum (resp. maximum) fitness of all agents.

We compute the next position $x_i^d(t+1)$ of agent $i \in \{1,\ldots,N_a\}$ in dimension d at time *t*+1 as follows. First, we use Eq.(4) to compute the force $F_i^d(t)$ at time *t* applied by the K largest masses in the set "Kbest"; second, we compute the corresponding acceleration $a_i^d(t)$ by Eq.(5); third, we compute the velocity $v_i^d(t+1)$ at time *t*+1 as a fraction of the velocity at time *t* increased by the acceleration at time *t* according to Eq.(6); finally, we compute $x_i^d(t+1)$ using Eq.(7).

$$F_i^d(t) = \sum_{j \in \mathrm{Kbest}, j \neq i} (rand_j) G(t) \frac{M_j(t) M_i(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t) \right) \tag{4}$$

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} = = \sum_{j \in \mathrm{Kbest}, j \neq i} (rand_j) G(t) \frac{M_j(t)}{R_{ij}(t) + \varepsilon} \left(x_j^d(t) - x_i^d(t) \right) \tag{5}$$

$$v_i^d(t+1) = (rand_i) v_i^d(t) + a_i^d(t) \tag{6}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{7}$$

where

- both "*rand*$_i$" and "*rand*$_j$" are random numbers, uniformly distributed over the interval [0,1],
- "$R_{ij}(t)$" is the Euclidean distance between agents i and j,
- "ε" is a small, positive number (to avoid dividing by zero),
- "Kbest" is the set of the K largest masses, where K is a decreasing function of time *t* with a user-defined initial value "K(*t*=0)=K0=N$_a$" and final value "K(*t*=T)=1", and
- "$G(t) = G_0 e^{-\alpha \frac{t}{T}}$" is the gravitational constant with initial value G_0, α is a user-defined constant, $t \in \{1,\ldots,\mathrm{T}\}$ is the iteration no., and T is the total number of iterations.

Note that Eq.(4) uses "$R_{ij}(t)$" instead of the square "$R_{ij}^2(t)$" in Newton's laws because it was confirmed empirically that "$R_{ij}(t)$" typically produces better results than "$R_{ij}^2(t)$".

5. The gsaINknn classifier

Given a dataset for training in the N-dimensional Euclidean space with a number of C classes, the objective is to compute an optimal INknn classifier using GSA heuristics.

Using a random training dataset partition (including N_a parts) we assume an initial population of N_a agents such that a different agent corresponds to a different partition part. In conclusion, an agent is represented by a (C×k×N×S)-dimensional vector, where C is the number of classes, k denotes the (user-defined) maximum number of INs per class, N is the dimension of the data (feature) space, moreover S equals the (user-defined) number of abscissa/ordinate samples used to represent an IN membership function. The objective is to compute an agent that produces as large classification accuracy as possible. Note that, during training, only the S = 2L abscissa (interval ends) values of an IN membership function update, whereas the corresponding L ordinate values remain constant spaced uniformly from 0 to 1 included.

The fitness function, for computing the mass $M_i(t)$ of agent $i \in \{1,\ldots,N_a\}$ at iteration time $t \in \{1,\ldots,T\}$, was calculated as follows.

$$fit_i = \frac{CC}{CC + IC} \times 100 \qquad (8)$$

where *CC* and *IC* denote the number of correctly and incorrectly classified data, respectively. The objective is to maximize fit_i for an agent $i \in \{1,\ldots,N_a\}$. A cycle of computing the next agent position repeats until the *stop condition* "t > T" is met as shown in **The gsaINknn classifier algorithm**. Recall the final value "K(t=T)=1" mentioned in the previous section; that is, only a single agent remains in the set Kbest at the termination of the gsaINknn classifier algorithm. Since there is no guarantee that the single agent at time t=T corresponds to the best solution, we keep storing the best solution calculated during the life time of the gsaINknn classifier algorithm.

The gsaINknn classifier algorithm

```
For a classification problem with a number of C classes in the N-dimensional feature space:
 1. A user-defines: The number $N_a$ of agents in the population; the maximum number k of INs per
    class; the initial number $K(t=0)=K0=N_a$ of largest masses; the total number T of iterations; the
    number S of abscissa values of an IN membership function;
 2. Randomly initialize a $(C\times k\times N\times S)$-dimensional swarm of $N_a$ agents;
 3. while $t \leq T$ do
 4.   Decrease the integer number K; increase the time $t$ by 1;
 5.   Identify the set Kbest of the largest K masses;
 6.     for each agent $i=1,\ldots,N_a$,
 7.         Evaluate the fitness function using Eq. (8);
 8.         Calculate the normalized mass using Eq. (3);
 9.         Calculate the acceleration using Eq. (5);
10.         Update the velocity using Eq. (6);
11.         Update the next position using Eq. (7);
12.     end for
13. end while
14. Best solution found.
```

6. Experiments and results

This section demonstrates the application of the gsaINknn classifier comparatively with alternative classifiers from the literature on eleven benchmark datasets from the University of California Irvine repository of machine learning databases [11]. In addition, we used the Ripley's benchmark dataset from Web site http://www.stats.ox.ac.uk/pub/PRNN/.

6.1. Benchmark datasets

Table 1 displays selected characteristics of the twelve benchmark datasets used in this work. In particular, note that the original Ecoli benchmark dataset includes 336 instances partitioned in 8 classes. Nevertheless, since 3 of the classes contain only 2, 2 and 5 instances, respectively, we decided to omit them. Hence, we employed a modified Ecoli dataset with 5 classes including 327 instances. Furthermore, since one of the 10 classes of the original Yeast dataset includes only 5 instances, we decided to omit it. In conclusion, we employed a modified Yeast benchmark dataset with 9 classes including 1479 instances.

Table 1 Selected characteristics of the twelve benchmark datasets used in this work.

Benchmark dataset name	# of instances	Missing value	# training data	# testing data	# attributes	# of classes
1. Dermatology	366	Yes	258	108	34	6
2. Ecoli	327	No	237	90	7	5
3. Haberman's Survival	306	No	216	90	3	2
4. Iris	150	No	105	45	4	3
5. Pima Indians Diabetes	768	No	543	225	8	2
6. Ripley's	1250	No	250	1000	2	2
7. Wine	178	No	130	48	13	3
8. Credit Approval	690	Yes	483	207	14	2
9. Statlog (Heart)	270	No	189	81	13	2
10. Thyroid	215	No	155	60	5	3
11. Yeast	1479	No	1038	441	8	9
12. Breast Cancer Wisconsin (Diagnostic)	569	No	401	168	30	2

6.2. Data preprocessing and classifier initialization

In a data preprocessing step, N-dimensional instances, where N is the corresponding number of attributes/dimensions, were normalized in the unit hypercube $[0,1]^N$ by replacing an attribute value x by the normalized value $x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$, where $x_{\min}$ and $x_{\max}$ stand for the corresponding minimum and maximum attribute values, respectively.

We employed a number of alternative classifiers from the literature including the GSA [2], fuzzy-ART [4], (conventional) knn [8], SOM [34], INknn [43], GRNN [55] and Support Vector Machine (SVM) [65] all implemented in the C++ programming language. For a fair comparison we used identical datasets for training/testing. When a training/testing dataset was not given explicitly for a benchmark dataset, we engaged a random 70% of the instances for training; whereas the remaining instances were used for testing. Care was taken so that each class is represented fairly in the dataset for training as well as in the dataset for testing; that is, each class was represented in the training/testing dataset in proportion to its corresponding total number of instances in the dataset. A missing value in an instance attribute was replaced by the corresponding attribute average. The initialization of different classifiers is described next.

Computational experiments with the SOM classifier were carried out using a 4×4 grid of units for 100 epochs. Since the results by SOM depend on the initial values of the weights, we chose the weights that yielded the best results on the training dataset in 10 random initializations. The GRNN classifier was considered with values of the variance parameter between 0 and 0.5 in steps of 0.001. For the fuzzy-ART we assumed a choice parameter value equal to 0.01;

furthermore, the values of the vigilance parameter were between 0 and 1 in steps of 0.01. For both knn and INknn classifiers, k=1 was user-defined; that is, only the (one) nearest neighbor was considered. Regarding the SVM classifier, we employed the "one versus all" binary classification method; moreover, for the SVM classifier here we employed Gaussian radial basis function kernels with a default scaling factor sigma of 1; furthermore, we used the Sequential Minimal Optimization (SMO) method to compute the separating hyperplanes of the SVM classifier. The parameters for the GSA classifier were tuned as described in [2]. More specifically, a population of N_a agents was user-defined so as to cover the training data. Moreover, the gsaINknn classifier ran for T=100 iterations with an initial value of the G parameter in the range [0.5,5.0]. The value of "k" in the product "C×k×N×S" was (user-defined to) k=1. In addition, "S" was user-defined to S=2L=2*32=64. We point out that the number of data employed for inducing an IN by either classifier INknn or gsaINknn depended on the specific (benchmark) classification dataset. In addition, the functions $\theta(x)=1-x$ and $v(x)=x$ were employed by both classifiers INknn and gsaINknn.

6.3. Comparative experimental results and discussion

Table 2 and Table 3 display (a) the average classification accuracy in 10 computational experiments for different (random) data permutations, and (b) the ranking (within parentheses) of each classifier for each benchmark dataset regarding training and testing, respectively. In other words, each Table cell displays the average (percentage) correct classification of a specific classifier on a specific dataset in 10 computational experiments. The best result for each benchmark dataset is displayed in bold. Note that the training data classification accuracy of the knn classifier on the Haberman's Survival benchmark (Table 2) is less than 100% because some training data are erroneously repeated with different class labels.

Both Tables 2 and 3 demonstrate the well-known fact that there is no "universally optimal classifier", in the sense that a classifier may perform well on some datasets and poorly on other ones [9]. On the one hand, Table 2 demonstrates the very good accuracy of both GRNN and knn classifiers: their ranking is either 1 or 2. Nevertheless, both GRNN and knn need to store the entire training dataset. Both classifiers SOM and fuzzy-ART have demonstrated a moderate performance. The INknn classifier typically achieved one of the worst rankings on most benchmark datasets. On the other hand, Table 3 demonstrates that the GSA and/or gsaINknn classifiers achieved some of the best recognition results on the testing datasets. Table 2 and especially Table 3 demonstrate the typically poor performance of the SVM classifier.

Table 4 summarizes the overall (average) classification accuracy of every classifier over the twelve benchmark classification datasets considered in this work. More specifically, each cell in Table 4 displays the average of a column of either Table 2 or Table 3 regarding the training datasets and the testing datasets, respectively. In addition, each cell in Table 4 displays (within parentheses) the corresponding overall ranking of each classifier. Table 4 indicates that the knn classifier has demonstrated the best overall accuracy (ranking 1) on the training datasets followed by the GRNN (ranking 2) and the GSA (ranking 3) classifiers. The INknn classifier has demonstrated the worst overall accuracy (ranking 8) on the training datasets and a bad accuracy (ranking 7) on the testing datasets. The proposed gsaINknn classifier performed rather poorly on the training datasets (ranking 5); nevertheless, it outperformed all other classifiers on the testing datasets (ranking 1). In other words, the proposed gsaINknn classifier has demonstrated a better overall capacity for generalization than any other classifier on the datasets considered here.

Table 5 sums up the rankings of each classifier for each benchmark classification dataset regarding both training (Table 2) and testing (Table 3). In addition, each Table 5 cell displays (within parentheses) the overall ranking of each classifier. In the aforementioned sense, Table 5 indicates that the best two classifiers in training are the knn classifier (ranking 1) and the GRNN classifier (ranking 2). Nevertheless, both aforementioned classifiers performed poorly in testing with rankings 7 and 6, respectively. The GSA classifier achieved a moderate overall ranking of 4 in training and an improved overall ranking of 2 in testing. The INknn has demonstrated the worst overall ranking (8) in training and a moderate overall ranking (4) in testing. Finally, the proposed gsaINknn classifier has demonstrated a moderate overall ranking (5) in training and the best overall ranking (1) in testing.

Next, we examined the impact of different parameter initializations regarding the proposed gsaINknn classifier. Table 6 indicates the impact of different parameter initializations on the classification results as well as on the CPU processing times of the gsaINknn classifier regarding three different benchmark datasets. More specifically, different initial values of the parameters N_a (number of agents), IN weight and G_0 (gravitational constant), shown in columns two, three and four of Table 6, were employed resulting in the train and test classification accuracies (%) shown in the next two columns of Table 6, respectively. The last column of Table 6 displays the corresponding CPU processing times (in msec). Our experience with different parameter initializations for the gsaINknn classifier suggested that the best training/testing classification accuracies were typically obtained for smaller "IN weight" values (hence for larger number N_a of agents), whereas the initial value G_0 of the gravitational constant does not effect significantly the classification accuracies and not at all the CPU processing times as shown in Table 6.

Table 7 displays the CPU processing times (in msec) of all the classifiers on the benchmark datasets used in this work. We remark that a processing time shown in Table 7 includes both the time for training and the time for testing. It turns out that, even though the knn and GRNN classifiers require no training, fuzzy-ART is the fastest classifier here because fuzzy-ART mostly computes minima. The SOM and SVM classifiers are fairly fast because they carry out specific vector data processing. Likewise, the INknn classifier is fairly fast. Nevertheless, both the GSA and gsaINknn classifiers require substantially more time because they carry out stochastic search. More specifically, the number of agents employed by either classifier GSA or gsaINknn increases in proportion to the number of instances in a benchmark dataset thus leading to longer training times. Typically, the GSA is faster than the gsaINknn because the GSA does not involve INs. However, the gsaINknn may reduce the number of its INs, hence it may achieve smaller processing times. For example, for the Credit Approval dataset in Table 7, note that the GSA and gsaINknn classifiers have employed 240 and 10 agents resulting in processing times of 1,066,730 [msec] and 447,616 [msec], respectively.

Another point of interest regards the effectiveness of the GSA method. Note that alternative stochastic search and optimization methods, namely genetic algorithms, have been employed with an INknn classifier, and similar classification improvements have been reported. More specifically, genetic algorithms have improved the performance of INknn classifiers by estimating optimally the parameters of both positive valuation functions and dual isomorphic functions [27, 61] in classification experiments regarding the Iris and the Wine benchmark datasets. We point out that any differences between the classification results reported in [27, 61] and the corresponding results reported here (regarding the Iris and the Wine benchmark datasets) are not statistically significant. Nevertheless, a detailed comparison of alternative optimization methods for the INknn classifier is a topic for future research.

In conclusion, INs have demonstrated here comparatively a significant potential in classification applications, which (potential) is attributed to the fact that an IN can represent all-order data statistics [31]. In addition, the computational experiments in this work have demonstrated that INs without optimization result in a rather poor classification accuracy as shown in both Tables 4 and 5 for the INknn classifier; whereas, an optimization (pursued here by the GSA) of the INknn classifier has significantly improved the classification accuracy.

7. Conclusion

This work has presented an optimized, granular knn classifier (with k=1) in the metric lattice of Intervals' Numbers (INs). Optimization was justified on the grounds of improving the classification results. Since there are no analytic optimization methods currently available in the metric lattice of INs, we resorted to stochastic optimization techniques. In particular, this paper introduced a synergy, namely gsaINknn, of the gravitational search algorithm (gsa) for stochastic optimization with the INknn classifier. An application on twelve benchmark classification datasets from the literature has demonstrated that the gsaINknn classifier achieved better results than either the GSA or the INknn classifier alone. Moreover, our experimental results have demonstrated that the proposed gsaINknn classifier compares favorably with alternative classifiers from the literature.

Potential future work includes the consideration of more than one (N-tuple) IN per class as well as the employment of alternative tunable non-linear functions $v(x)$ and $\theta(x)$ rather than the functions $v(x)=x$ and $\theta(x)=1-x$ employed in this work. Alternative optimization methods to GSA will also be considered comparatively in a future work. The IN inputs to the gsaINknn classifier in this work have been trivial. However, Proposition 1 has paved the way for accommodating non-trivial IN inputs toward dealing with input uncertainty and/or ambiguity. Further future work extensions include the employment of Type-2 INs toward *Computing With Words* [41].

Table 2 Average classification accuracy % in 10 experiments (with ranking within parentheses) by different classifiers regarding training.

Benchmark dataset \ Classifier	**SOM**	**GRNN**	**fuzzy-ART**	**knn**	**GSA**	**SVM**	**INknn**	**gsaINknn**
Dermatology	96.05 (4)	**100.00 (1)**	95.50 (5)	**100.00 (1)**	99.08 (2)	**100.00 (1)**	96.17 (3)	99.08 (2)
Ecoli	87.55 (6)	98.52 (2)	87.81 (5)	**100.00 (1)**	89.90 (4)	90.98 (3)	84.30 (7)	71.12 (8)
Haberman's Survival	75.79 (5)	96.53 (2)	78.43 (4)	**98.52 (1)**	70.90 (7)	79.03 (3)	58.40 (8)	73.00 (6)
Iris	96.95 (6)	**100.00 (1)**	97.71 (5)	**100.00 (1)**	98.95 (2)	97.62 (4)	90.68 (7)	98.67 (3)
Pima Indians Diabetes	81.77 (3)	**100.00 (1)**	80.67 (5)	**100.00 (1)**	78.12 (6)	91.75 (2)	73.34 (7)	80.00 (4)
Ripley's	86.76 (6)	94.40 (2)	89.44 (3)	**100.00 (1)**	87.60 (5)	77.60 (8)	84.80 (7)	89.16 (4)
Wine	97.85 (5)	**100.00 (1)**	98.23 (4)	**100.00 (1)**	99.13 (3)	**100.00 (1)**	93.41 (6)	99.44 (2)
Credit Approval	78.96 (7)	**100.00 (1)**	85.18 (6)	**100.00 (1)**	88.35 (3)	98.84 (2)	85.44 (5)	88.15 (4)
Statlog (Heart)	83.33 (6)	**100.00 (1)**	87.72 (4)	**100.00 (1)**	89.11 (3)	99.68 (2)	82.78 (7)	87.56 (5)
Thyroid	95.42 (7)	99.94 (2)	96.64 (6)	**100.00 (1)**	99.56 (4)	97.66 (5)	75.89 (8)	99.89 (3)
Yeast	54.94 (7)	89.10 (2)	56.78 (6)	**100.00 (1)**	61.16 (4)	66.29 (3)	52.67 (8)	59.78 (5)
Breast Cancer Wisconsin (Diagnostic)	95.71 (5)	**100.00 (1)**	96.08 (4)	**100.00 (1)**	98.35 (3)	99.98 (2)	92.60 (6)	98.35 (3)

Table 3 Average classification accuracy % in 10 experiments (with ranking within parentheses) by different classifiers regarding testing.

Benchmark dataset \ Classifier	**SOM**	**GRNN**	**fuzzy-ART**	**knn**	**GSA**	**SVM**	**INknn**	**gsaINknn**
Dermatology	96.39 (4)	96.02 (5)	95.09 (6)	95.83 (3)	**96.99 (1)**	30.00 (8)	92.85 (7)	96.79 (2)
Ecoli	86.44 (2)	81.22 (8)	84.78 (4)	82.56 (7)	85.29 (3)	83.78 (5)	83.61 (6)	**97.67 (1)**
Haberman's Survival	74.33 (3)	69.56 (6)	74.67 (2)	65.33 (7)	69.71 (5)	**75.44 (1)**	53.64 (8)	71.99 (4)
Iris	96.22 (4)	96.44 (3)	96.44 (3)	96.22 (4)	97.78 (2)	96.22 (2)	90.86 (5)	**98.22 (1)**
Pima Indians Diabetes	**80.59 (1)**	71.07 (6)	68.44 (8)	69.56 (7)	71.53 (5)	73.60 (3)	76.64 (2)	72.02 (4)
Ripley's	85.38 (2)	82.10 (5)	82.66 (4)	80.40 (6)	84.09 (4)	75.50 (7)	**86.20 (1)**	84.12 (3)
Wine	95.83 (3)	95.62 (4)	95.42 (5)	94.38 (6)	97.88 (2)	74.58 (8)	91.54 (7)	**98.65 (1)**
Credit Approval	78.26 (6)	79.71 (5)	80.72 (4)	79.71 (5)	86.19 (2)	68.07 (7)	85.00 (3)	**86.67 (1)**
Statlog (Heart)	77.28 (5)	75.80 (6)	77.78 (4)	74.07 (7)	**84.11 (1)**	69.26 (8)	79.67 (3)	82.67 (2)
Thyroid	93.33 (8)	96.17 (5)	93.50 (7)	97.50 (3)	**98.16 (1)**	93.67 (6)	97.92 (2)	96.96 (4)
Yeast	53.86 (2)	46.37 (8)	50.14 (5)	50.95 (3)	46.60 (7)	**54.86 (1)**	49.25 (6)	50.35 (4)
Breast Cancer Wisconsin (Diagnostic)	94.40 (7)	97.50 (3)	95.66 (6)	95.96 (5)	**98.34 (1)**	83.04 (8)	96.39 (4)	98.11 (2)

Table 4 Overall average classification accuracies % (with ranking within parentheses) of the classifiers used in this work.

Overall Accuracy \ Classifier	**SOM**	**GRNN**	**fuzzy-ART**	**knn**	**GSA**	**SVM**	**INknn**	**gsaINknn**
Training datasets	85.92 (6)	98.21 (2)	87.52 (4)	**99.88 (1)**	88.35 (3)	80.87 (7)	80.40 (8)	87.02 (5)
Testing datasets	84.36 (3)	82.30 (5)	82.94 (4)	81.87 (6)	84.72 (2)	73.17 (8)	81.48 (7)	**86.18 (1)**

Table 5 Sums of rankings (with the resulting overall ranking within parentheses) of the classifiers used in this work.

Sum of Rankings \ Classifier	**SOM**	**GRNN**	**fuzzy-ART**	**knn**	**GSA**	**SVM**	**INknn**	**gsaINknn**
Training datasets	58 (7)	17 (2)	49 (6)	**12 (1)**	39 (4)	35 (3)	67 (8)	41 (5)
Testing datasets	44 (3)	60 (6)	54 (5)	59 (7)	31 (2)	64 (8)	51 (4)	**25 (1)**

Table 6 Impact of choosing different initialization parameters "N_a", "IN weight" and "G_0" on the classification accuracies % as well as on the CPU processing times (in msec) of the gsaINknn classifier regarding three of the benchmark datasets.

Benchmark Dataset	**N_a**	**IN weight**	**G_0**	**trainacc (%)**	**testacc (%)**	**proc time (msec)**
Iris	5	7	0.5	97.14	97.78	9,735
	5	7	1.0	98.10	97.78	9,735
	5	7	1.5	98.10	95.56	9,735
	5	7	2.0	99.05	95.46	9,735
	7	5	0.5	99.05	97.78	12,015
	7	5	1.0	99.05	97.78	12,015
	7	5	1.5	99.05	97.78	12,015
	7	5	2.0	97.14	95.46	12,015
Ripley's	5	25	0.5	86.40	89.30	10,030
	5	25	1.0	87.60	87.60	10,030
	5	25	1.5	88.40	82.30	10,030
	5	25	2.0	87.60	82.50	10,030
	25	5	0.5	88.40	84.10	33,797
	25	5	1.0	88.40	83.70	33,797
	25	5	1.5	88.40	84.00	33,797
	25	5	2.0	88.00	84.70	33,797
Breast Cancer	5	40	0.5	93.50	95.86	640,244
	5	40	1.0	93.50	95.86	640,244
	5	40	1.5	93.50	95.86	640,244
	5	40	2.0	93.50	96.45	640,244
	20	10	0.5	95.75	95.86	1,539,240
	20	10	1.0	97.25	98.22	1,539,240
	20	10	1.5	97.25	98.22	1,539,240
	20	10	2.0	97.50	97.04	1,539,240

Table 7 CPU processing times (in msec) of all the classifiers on all the benchmark datasets used in this work.

Benchmark dataset \ Classifier	**SOM**	**GRNN**	**fuzzy-ART**	**knn**	**GSA**	**SVM**	**INknn**	**gsaINknn**
Dermatology	688	249	107	174	25,926	521	535	172,317
Ecoli	176	82	27	39	1,788	201	127	23,269
Haberman's Survival	116	52	15	41	1,031	119	69	8,144
Iris	52	30	9	13	1,340	81	73	12,015
Pima Indians Diabetes	465	294	70	239	103,387	361	518	243,864
Ripley’s	101	31	14	27	4,822	70	60	33,926
Wine	179	77	29	61	13,021	133	180	47,943
Credit Approval	799	327	127	432	1,066,730	380	756	447,616
Statlog (Heart)	242	87	40	62	41,825	113	204	92,656
Thyroid	113	43	15	30	1,284	81	40	20,229
Yeast	812	801	124	650	6,810	2,587	440	87,174
Breast Cancer Wisconsin (Diagnostic)	759	621	193	493	1,543,530	326	796	1,539,240

References

[1] Athanasiadis, I.N., Kaburlasos, V.G., 2006. Air quality assessment using fuzzy lattice reasoning (FLR). In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI 2006), pp. 29–34.

[2] Bahrololoum, A., Nezamabadi-pour, H., Bahrololoum, H., Saeed, M., 2012. A prototype classifier based on gravitational search algorithm. Applied Soft Computing 12 (2), 819–825.

[3] Birkhoff, G., 1967. Lattice Theory, Colloquium Publications 25. American Math Society, Providence, RI.

[4] Carpenter, G.A., Grossberg, S., Rosen, D.B., 1991. Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks 4 (6), 759–771.

[5] Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., Rosen, D.B., 1992. Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. IEEE Transactions on Neural Networks 3 (5), 698–713.

[6] Chang, S.K., Mohammed, O.A., Hahn, S.-Y., 1994. Detection of magnetic body using artificial neural network with modified simulated annealing. IEEE Transactions on Magnetics 30 (5), 3644–3647.

[7] Chi, S.-C., Yang, C.C., 2006. Integration of ant colony SOM and k-means for clustering analysis. In: Gabrys, B., Howlett, R.J, Jain, L.C. (Eds.), Proceedings of the 10th Intl. Conf. on Knowledge-Based Intelligent Information and Engineering Systems (KES 2006), Part I. Springer, LNCS 4251, pp. 1–8.

[8] Cunningham, P., Delany, S.J., 2007. k-nearest neighbour classifiers. Technical Report UCD-CSI-2007-4.

[9] De Falco, I., Della Cioppa, A., Tarantino, E., 2007. Facing classification problems with Particle Swarm Optimization. Applied Soft Computing 7 (3), 652–658.

[10] Derrac, J., García, S., Herrera, F., 2014. Fuzzy nearest neighbor algorithms: taxonomy, experimental analysis and prospects. Information Sciences 260, 98-119.

[11] Frank, A., Asuncion, A., 2010. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science, Irvine, CA.

[12] Graña, M., 2009 (Ed.). Special issue on: Lattice computing and natural computing. Neurocomputing 72 (10-12), 2065–2066.

[13] Graña, M., 2012. Lattice computing in hybrid intelligent systems. In: Proceedings of the 12th International IEEE Conference on Hybrid Intelligent Systems (HIS 2012), Pune, India, 4-7 Dec. 2012, pp. 1-5.

[14] Graña, M., Chyzhyk, D., García-Sebastián, M., Hernández, C., 2011. Lattice independent component analysis for functional magnetic resonance imaging. Information Sciences 181 (10), 1910–1928.

[15] Graña, M., Savio, A.M., García-Sebastián, M., Fernández, E., 2010. A lattice computing approach for on-line fMRI analysis. Image and Vision Computing 28 (7), 1155–1161.

[16] Graña, M., Villaverde, I., Maldonado, J.O., Hernández, C., 2009. Two lattice computing approaches for the unsupervised segmentation of hyperspectral images. Neurocomputing 72 (10-12), 2111–2120.

[17] Huang, C.-L., Dun, J.-F., 2008. A distributed PSO-SVM hybrid system with feature selection and parameter optimization. Applied Soft Computing 8 (4), 1381–1391.

[18] Jamshidi Khezeli, Y., Nezamabadi-pour, H., 2012. Fuzzy lattice reasoning for pattern classification using a new positive valuation function. Advances in Fuzzy Systems 2012, Article ID 206121, 8 pages, doi: 10.1155/2012/206121.

[19] Jamshidi, Y., Nezamabadi-pour, H., 2013. A lattice computing algorithm for granular reasoning based on trapezoidal fuzzy numbers. Int. J. Granular Computing, Rough Sets and Intelligent Systems 3 (2), 160–177.

[20] Kaburlasos, V.G., 2004. FINs: lattice theoretic tools for improving prediction of sugar production from populations of measurements. IEEE Transactions on Systems, Man and Cybernetics - B 34 (2), 1017–1030.

[21] Kaburlasos, V.G., 2006. Towards a Unified Modeling and Knowledge-Representation Based on Lattice Theory, Studies in Computational Intelligence 27. Springer, Heidelberg, Germany.

[22] Kaburlasos, V.G., 2010. Granular fuzzy inference system (FIS) design by lattice computing. In: Corchado, E., Graña, M., Savio, A.M. (Eds.), Proceedings of the 5th Intl. Conf. on Hybrid Artificial Intelligence Systems (HAIS 2010), Part II. Springer, LNAI 6077, pp. 410–417.

[23] Kaburlasos, V.G., 2011 (Ed.). Special issue on: Information engineering applications based on lattices. Information Sciences 181 (10), 1771–1773.

[24] Kaburlasos, V.G., Kehagias, A., in press. Fuzzy inference system (FIS) extensions based on lattice theory. IEEE Transactions on Fuzzy Systems, DOI: 10.1109/TFUZZ.2013.2263807

[25] Kaburlasos, V.G., Moussiades, L., 2014. Induction of formal concepts by lattice computing techniques for tunable classification. Journal of Engineering Science and Technology Review 7 (1), 1-8.

[26] Kaburlasos, V.G., Pachidis, T., 2014. A Lattice-Computing ensemble for reasoning based on formal fusion of disparate data types, and an industrial dispensing application. Information Fusion 16, 68-83.

[27] Kaburlasos, V.G., Papadakis, S.E., 2006. Granular self-organizing map (grSOM) for structure identification. Neural Networks 19 (5), 623-643.

[28] Kaburlasos, V.G., Ritter G.X., 2007 (Eds.). Computational intelligence based on lattice theory, Studies in Computational Intelligence 67. Springer, Heidelberg, Germany.

[29] Kaburlasos, V.G., Athanasiadis, I.N., Mitkas, P.A., 2007. Fuzzy lattice reasoning (FLR) classifier and its application for ambient ozone estimation. International Journal of Approximate Reasoning 45 (1), 152–188.

[30] Kaburlasos, V.G., Papadakis, S.E., Amanatiadis, A., 2012. Binary image 2D shape learning and recognition based on lattice computing (LC) techniques. Journal of Mathematical Imaging and Vision 42 (2-3), 118–133.

[31] Kaburlasos, V.G., Papadakis, S.E., Papakostas, G.A., 2013. Lattice computing extension of the FAM neural classifier for human facial expression recognition. IEEE Transactions on Neural Networks and Learning Systems 24 (10), 1526–1538.

[32] Kaburlasos, V.G., Tsoukalas, V., Moussiades, L., 2014. FCknn: a granular knn classifier based on formal concepts. In: Proceedings of the World Congress on Computational Intelligence (WCCI) 2014, FUZZ-IEEE Program, Beijing, China, 6-11 July 2014.

[33] Kaburlasos, V.G., Papakostas, G.A., Pachidis, T., Athinellis, A., 2013. Intervals' numbers (INs) interpolation /extrapolation. In: Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2013), Hyderabad, India, 7-10 July 2013.

[34] Kohonen, T., Somervuo, P., 2002. How to make large self-organizing maps for nonvectorial data. Neural Networks, 15 (8-9), 945–952.

[35] Lang, K.J., Witbrock, M.J., 1988. Learning to tell two spirals apart. In: Touretzky, D., Hinton, G., Sejnowski, T. (Eds.), Proc of the Connectionist Models Summer School. Morgan Kaufmann, Mountain View, CA, pp. 52–59.

[36] Li, C., Zhou, J., 2011. Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. Energy Conversion and Management 52 (1), 374–381.

[37] Liu, Y., Sun, F., 2011. A fast differential evolution algorithm using k-nearest neighbour predictor. Expert Systems with Applications 38 (4), 4254–4258.

[38] Liu, H., Xiong, S., Fang, Z., 2011. FL-GrCCA: A granular computing classification algorithm based on fuzzy lattices. Computers & Mathematics with Applications 61 (1), 138–147.

[39] Liu, H., Xiong, S., Wu, C.-a., 2013. Hyperspherical granular computing classification algorithm based on fuzzy lattices. Mathematical and Computer Modelling 57 (3-4), 661–670.

[40] Lotfi Shahreza, M., Moazzami, D., Moshiri, B., Delavar, M.R., 2011. Anomaly detection using a self-organizing map and particle swarm optimization. Scientia Iranica 18 (6), 1460–1468.

[41] Mendel, J.M., 2007. Type-2 fuzzy sets and systems: an overview. IEEE Comput. Intel. Magazine 2 (1), 20–29.

[42] Nanda, S., 1989. Fuzzy lattices. Bulletin Calcutta Mathematical Society 81, 1–2.

[43] Pachidis, T., Kaburlasos, V.G., 2012. Person identification based on lattice computing k-nearest-neighbor fingerprint classification. In: Graña, M., Toro, C., Posada, J., Howlett, R.J., Jain, L.C. (Eds.), Proceeding of the 16th Intl. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2012). IOS Press, 2012, pp. 1720–1729.

[44] Papadakis, S.E., Kaburlasos, V.G., 2010. Piecewise-linear approximation of non-linear models based on probabilistically/possibilistically interpreted intervals' numbers (INs). Information Sciences 180 (24), 5060–5076.

[45] Papadakis, S.E., Kaburlasos, V.G., Papakostas, G.A., 2014. Two fuzzy lattice reasoning (FLR) classifiers and their application for human facial expression recognition. Journal of Multiple-Valued Logic and Soft Computing 22, 561-579.

[46] Polat, Ö, Yildirim, T., 2008. Genetic optimization of GRNN for pattern recognition without feature extraction. Expert Systems with Applications 34 (4), 2444–2448.

[47] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. GSA: a gravitational search algorithm. Information Sciences 179 (13), 2232–2248.

[48] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2011, Filter modeling using gravitational search algorithm. Engineering Applications of Artificial Intelligence 24 (1), 117–122.

[49] Rebollo-Ruiz, I., Graña, M., 2012. An empirical comparison of some approximate methods for Graph Coloring. In: E. Corchado et al. (Eds.), Proceedings of the 7th Intl. Conf. on Hybrid Artificial Intelligence Systems (HAIS 2012), Part I. Springer-Verlag, LNCS 7208, pp. 600–609.

[50] Rebollo, I., Graña, M., Cases, B., 2012. Effect of spatial percolation on the convergence of a graph coloring boid swarm. International Journal on Artificial Intelligence Tools 21 (6), 1250015, 19 pages, doi: 10.1142/S0218213012500157

[51] Rebollo-Ruiz, I., Graña, M., 2013. Swarm graph coloring for the identification of user groups on ERP logs. Cybernetics and Systems 44 (6-7), 489–504.

[52] Rutherford, D.E., 1965. Introduction to Lattice Theory. Oliver & Boyd Ltd., Edinburgh, Great Britain.

[53] Sarafrazi, S., Nezamabadi-pour, H., 2013. Facing the classification of binary problems with a GSA-SVM hybrid system. Mathematical and Computer Modelling 57 (1-2), 270–278.

[54] Si, T., Hazra, S., Jana, N.D., 2012. Artificial neural network training using differential evolutionary algorithm for classification. In: Satapathy, S.C., Avadhani, P.S., Abraham, A. (Eds.), Proc of the International Conference on Information Systems Design and Intelligent Applications (INDIA 2012). Springer, AISC 132, pp. 769–778.

[55] Specht, D.F., 1991. A general regression neural network. IEEE Trans on Neural Networks 2 (6), 568–576.

[56] Sussner, P., Esmi, E.L., 2009. An introduction to morphological perceptrons with competitive learning. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN 2009), pp. 3024–3031.

[57] Sussner, P., Esmi, E.L., 2011. Morphological perceptrons with competitive learning: lattice-theoretical framework and constructive learning algorithm. Information Sciences 181 (10), 1929–1950.

[58] Sussner, P., Valle, M.E., 2006. Gray-scale morphological associative memories. IEEE Transactions on Neural Networks 17 (3), 559–570.

[59] Sussner, P., Valle, M.E., 2006. Implicative fuzzy associative memories. IEEE Transactions on Fuzzy Systems 14 (6), 793–807.

[60] Taghipour, M., Moradi, A.R., Yazdani-Asrami, M., 2010. Identification of magnetizing inrush current in power transformers using GSA trained ANN for educational purposes. In: Proceedings of IEEE Conference on Open Systems (ICOS 2010), pp. 23–27.

[61] Tsoukalas, V.T., Kaburlasos, V.G., Skourlas, C., 2013. A granular, parametric KNN classifier. In: Proceedings of the 17th Panhellenic Conference on Informatics (PCI 2013), Thessaloniki, Greece, 19-21 September 2013, pp. 319-326.

[62] Valle, M.E., Grande Vicente, D.M., 2012. Sparsely connected autoassociative lattice memories with an application for the reconstruction of color images. J. Math. Imaging and Vision 44 (3), 195–222.

[63] Valle, M.E., Sussner, P., 2008. A general framework for fuzzy morphological associative memories. Fuzzy Sets and Systems 159 (7), 747–768.

[64] Valle, M.E., Sussner, P., 2011. Storage and recall capabilities of fuzzy morphological associative memories with adjunction-based learning. Neural Networks 24 (1), 75–90.

[65] Vapnik, V.N., 1999. The Nature of Statistical Learning Theory, 2nd ed. Information Science and Statistics. Springer, Heidelberg, Germany.

[66] Yin, M., Hu, Y., Yang, F., Li, X., Gu, W., 2011. A novel hybrid K-harmonic means and gravitational search algorithm approach for clustering. Expert Systems with Applications 38 (8), 9319–9324.