Applied Soft Computing Journal 74 (2019) 567–582

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Interpretable fuzzy partitioning of classified data with variable granularity

Ciro Castiello [a,c,*], Anna Maria Fanelli [a], Marco Lucarelli [b], Corrado Mencar [a,c]

[a] Department of Informatics, University of Bari Aldo Moro, Bari, 70125, Italy
[b] SITAEL S.p.A., Mola di Bari, Bari, Italy
[c] Member of the INdAM Research Group GNCS

HIGHLIGHTS

- Fuzzy systems get knowledge from data, granulating input features in fuzzy partitions.
- DC*: our proposal to designs interpretable fuzzy partitions with optimal granularity.
- DC* performs a 2-stage clustering process and is specific for classification problems.
- The number of clusters is minimized through a mechanism relying on the A* algorithm.
- Resulting fuzzy systems show linguistic interpretability and classification accuracy.

ARTICLE INFO

Article history:
Received 17 April 2018
Received in revised form 16 September 2018
Accepted 20 October 2018
Available online 29 October 2018

Keywords:
Clustering
A*
Interpretability
Fuzzy partitions
Information granulation

ABSTRACT

Fuzzy rule-based systems are effective tools for acquiring knowledge from data and represent it in a linguistically interpretable form. To achieve interpretability, input features are granulated in fuzzy partitions. A critical design decision is the selection of the granularity level for each input feature. This paper presents an approach, called DC* (Double Clustering with A*), for automatically designing interpretable fuzzy partitions with optimal granularity. DC* is specific for classification problems and is mainly based on a two-stage process: the first stage identifies clusters of multidimensional samples in order to derive class-labeled prototypes; in the second stage the one-dimensional projections of such prototypes are further clustered along each dimension simultaneously, thus minimizing the number of clusters for each feature. Moreover, the resulting one-dimensional clusters provide information to define fuzzy partitions that satisfy a number of interpretability constraints and exhibit variable granularity levels. The fuzzy sets in each partition can be labeled by meaningful linguistic terms and used to represent knowledge in a natural language form. Experimental results on both synthetic and real data show that the derived fuzzy partitions can be exploited to define very compact fuzzy rule-based systems that exhibit high linguistic interpretability and good classification accuracy.

1. Introduction

Information granulation is the process of forming meaningful entities, called information granules, that exhibit functional and descriptive representations of observational data, adhering to some level of abstraction [1]. Information granules are generally defined as agglomerates of data, arranged together due to their similarity, functional adjacency, indistinguishability, coherence or alike. They are the building blocks for information abstraction, since information granules highlight high-level properties and relationships about an universe of discourse, whereas they hide useless low-level details pertinent to single data. Once formed, information granules help to understand hidden relationships among data.

Granular computing is a paradigm oriented towards representing and processing information granules; it embraces a number of modeling frameworks based on different forms of representation, depending on the nature of data as well as on the applicative domain [2–5]. Fuzzy set theory is a convenient modeling framework for granular computing, leading to the so-called Theory of Fuzzy Information Granulation (TFIG) [1]. The use of TFIG for granulating data produces information granules that are defined as compositions of fuzzy sets. Fuzzy sets capture the perceptive character of concepts as conceived by human beings; as a consequence,

* Corresponding author.
E-mail addresses: ciro.castiello@uniba.it (C. Castiello), annamaria.fanelli@uniba.it (A.M. Fanelli), marco.lucarelli@sitael.com (M. Lucarelli), corrado.mencar@uniba.it (C. Mencar).

https://doi.org/10.1016/j.asoc.2018.10.040
1568-4946/© 2018 Elsevier B.V. All rights reserved.

fuzzy information granules can describe hidden relationships in a way close to mental representation of concepts. This feature helps users to understand data through information granules, especially if these are described in natural language.

Deriving fuzzy information granules from data and describing them in natural language are not easy tasks. Fuzzy information granules are often defined in terms of one-dimensional fuzzy sets defined on each input feature. This involves the granulation of each feature into a fuzzy partition. The fuzzy sets composing each fuzzy partition should accurately represent data in an interpretable way, i.e. they should be shaped so as to be easily tagged with linguistic terms. Also, the number of derived fuzzy sets should be as small as possible since humans have limited ability to remember and to deal with descriptions [6]. Interpretability and accuracy are conflicting requirements (a trade-off is often demanded) because the need to preserve interpretability constrains the parameters of a model. This introduces a bias that in most cases prevents the model from reaching the accuracy level which often characterizes unconstrained, not interpretable alternatives.

To reduce the bias, techniques for interpretability preservation should not introduce constraints other than those strictly necessary for their own purpose. One way to tackle this problem is to enable variable granularity in fuzzy partitions, dealing with the shape and the number of the fuzzy sets in every partition. Many techniques impose a fixed granularity level (e.g., by using predefined fuzzy sets); on the other hand, interpretability can be also preserved whenever information granules with variable granularity are allowed. If the fuzzy sets conform to data, while preserving interpretability, then the resulting information granules are more representative of the underlying data thus enabling the realization of fuzzy models that are more accurate but still readable.

In this paper we present DC* (Double Clustering with A*), a method for generating interpretable fuzzy partitions with variable granularity. Each fuzzy set of the partition is defined so as to conform to available data and, at the same time, to satisfy a number of interpretability constraints. The number of fuzzy sets for each partition is also optimized according to available data, i.e. a minimal number of fuzzy sets is defined for each input feature, thus enhancing readability. As an interesting effect, some features could be granulated with a single fuzzy set: this enables their safe removal from the linguistic description of the derived information granules. DC* requires only one hyper-parameter to be imposed by the user, that is the upper bound for the number of information granules to be derived from data.

DC* is an instance of DC*f* (Double Clustering *f*ramework), a framework for generating interpretable information granules [7]. DC*f* is an abstract framework that can be implemented in a number of ways to achieve different techniques for interpretable information granulation. DC* is specifically designed for pre-classified data and performs a two-step clustering process to generate fuzzy partitions. First, DC* identifies cluster prototypes in the multidimensional data space via the LVQ1 algorithm, so as to exploit class information and to find class-aware clusters. Then, it clusters the projections of these prototypes along each dimension by a properly defined search procedure based on A* [8]. The objective is to derive the minimal number of fuzzy sets per feature, so that a compact and interpretable description of data can be provided. The optimization process involves all the features simultaneously, thus configuring a combinatorial search problem: A* is endowed with a heuristic function to tackle this problem while attenuating the computational burden. Also, differently from alternative techniques, the simultaneous generation of fuzzy partitions allows to preserve the multi-dimensional relations in the original data.

The fuzzy sets of each partition generated by DC* can be combined to form fuzzy information granules, which can be directly translated into human-comprehensible fuzzy rules to be used for classification tasks. Experimental results highlight how the resulting classifiers require very few rules to perform accurate classifications which compare to alternative methods. On the overall, DC* can be intended as a method for linguistic fuzzy modeling, where interpretability is a primary requirement and class information is exploited to generate fuzzy partitions with optimal granularity.

Some early works on DC* appear in literature [9–11], mainly presenting some technical advances. In this paper we give a comprehensive description of the method, including all the formal details and the most recent advancements. Additionally, a complete experimentation of the method is performed with a comparison with different approaches.

The paper is organized as follows. In Section 2 we present some related works concerning interpretable fuzzy information granulation. In Section 3 we introduce the Double Clustering framework and we describe DC* in detail. In Section 4 we portray some experimental results to illustrate the effectiveness of the proposed method. Conclusive remarks are drawn in Section 5.

2. Related work

The concept of interpretability in the context of knowledge-based systems can be roughly intended as the similarity between a somehow compiled knowledge base (usually resulting from an inductive learning process) and the description provided by a human while observing the same entities analyzed during the automatic learning process [12]. In essence, interpretability is a quality that puts humans at the center of intelligent processing. As human observation mainly relies on perceptions instead of measurements, fuzzy logic has been widely recognized as a suitable mathematical tool for human-centric information processing [13,14]. In this context, information granulation plays a crucial role as it corresponds to a basic cognitive task along with causation and organization of perceptions [1]. Nevertheless, interpretability of information granules is not granted by the mere use of fuzzy logic: information granulation processes must be endowed with the capability of preserving interpretability while building up the information granules. This requires two modeling decisions: (i) a computable definition of interpretability, and (ii) a suitable technique for interpretable information granulation.

Defining interpretability is not a trivial task because it involves the identification of properties pertaining to the human mind [15,16]. Nevertheless, several speculative works and psychological studies have been carried out, which enable to partially define interpretability along two main axes [17]. On the structural axis, interpretability is mainly defined in terms of complexity of the description of fuzzy information granules; on the semantic axis, interpretability is seen as a degree of co-intension between the explicit semantics of an information granule and the implicit semantics that takes form in the human mind when the linguistic description of the information granule is communicated to the user [18,19]. In order to put things in operation, several aspects of interpretability are formally defined in terms of constraints and criteria (either hard or soft) which can be included in learning processes to preserve interpretability in both axes [17,20–22].

Information granulation is usually a preliminary step for defining rule bases that describe some complex relationships among data. Many methods for interpretable information granulation are essentially based on the structural facet of interpretability: they are aimed at generating fuzzy information granules that accurately describe the underlying data and are characterized by a simple description. This requires a trade-off because the structural interpretability constraints introduce a bias in the granulation process that could affect its capability to derive an accurate description of data. For this reason, many information granulation methods are based on evolutionary computation with single or multiple objectives [23–25].

A simple yet common approach to generate interpretable fuzzy information granules is to fix the fuzzy partition for each input feature. In this way the problem of information granulation is simplified into the selection of the most suitable combination of fuzzy sets (one for each feature) that define each information granule [26]. The rigidness of this approach can be attenuated first by learning fuzzy information granules that are well adapted to data, then by modifying them in order to satisfy a number of interpretability constraints (usually merging similar fuzzy sets on the same feature) [27–30]. As an alternative approach, interpretable fuzzy partitions are generated first, then they are fine-tuned so as to better represent the underlying data [31–33]. This approach, however, often leads to fuzzy partitions that do not satisfy many interpretability constraints.

In most cases, the number of fuzzy sets, as well as their initial granularity, is fixed for each input feature. This is a simplifying design hypothesis which, however, may fail to capture the essence of data: features can be more or less relevant when inducing knowledge from data; this requires different levels of granulation which, in the extreme case, could reduce a feature to a single information granule (which implies the uselessness of that feature). In view of maximizing the interpretability of the final model, the coarsest possible granularity is desirable because this leads to a simpler representation of knowledge. It is important, therefore, to find a granularity level for each input feature that optimally balances simplicity with accuracy in describing the underlying data.

Different approaches have been applied to define an appropriate granulation level for each input feature. In some cases, fuzzy sets are defined by a preliminary discretization of continuous attributes followed by a fuzzification of the resulting partition [34]. An alternative is to cluster the values of a single feature into a number of clusters that is determined by some validity index [35]. Also, features can be granulated in sequence by trying to maximize accuracy [36].

All these approaches generate fuzzy partitions by taking into account one feature at a time. This makes impossible to exploit multi-dimensional relationships among data in order to generate a fuzzy partition. A common approach to capture multi-dimensional relationships is to cluster data in the whole input space [37–39]. However, the resulting fuzzy sets are hard to be described in linguistic terms because they usually violate several interpretability constraints. To preserve interpretability without managing features sequentially, genetic algorithms can be used to evolve fuzzy partitions simultaneously [40–43]. However, genetic algorithms require several hyper-parameters that must be set through several trial-and-error processes; they involve also randomness of the search process that could be hard to control, especially when the search space is very large.

As an alternative approach, a multi-objective, incremental, approach can be applied to generate a family of interpretable fuzzy partitions, which are offered to the user for a final choice that represents the best trade-off between interpretability and accuracy [44]. This approach, named Hierarchical Fuzzy Partitioning (HFP), has been implemented in tools that are currently used for designing interpretable fuzzy systems, like FisPro [45] or GUAJE [46, 47].

The method presented in this paper is different from these approaches because it is capable of granulating all features simultaneously and, at the same time, is based on an optimal procedure involving the A* search algorithm.

3. Double clustering with A* (DC*)

The proposed method for designing interpretable fuzzy partitions is an instance of a general framework, the Double Clustering framework (DC*f*) [7], which is defined by three main stages that operate sequentially on multi-dimensional numerical data:

- *Data compression.* Multi-dimensional data are compressed in a reduced number of prototypes whose dimensions are the same as those pertaining to the original data. The objective of this stage is to find hidden relationships among data that can be expressed in terms of data aggregations (see Section 3.1).
- *One-dimensional clustering.* The prototypes obtained in the first stage are projected onto each dimension. Then, the prototype projections are clustered along each dimension with the objective of defining the structure and the granularity of the fuzzy partitions (see Section 3.2).
- *Fuzzy granulation.* Information coming from the previous stage is exploited to define fuzzy partitions along each dimension. Eventually, the fuzzy sets defined on each partition are combined to define fuzzy information granules that represent data abstractions denoted in a linguistic form (see Section 3.3).

DC* is characterized by the ability of exploiting data class information for automatically deriving the number of one-dimensional clusters along each dimension. To this pursuit, data compression is performed by the LVQ1 algorithm [48], one-dimensional clustering is performed by a specific implementation of the A* algorithm [8] and fuzzy granulation is obtained by a procedure aimed at compiling trapezoidal fuzzy sets that define strong fuzzy partitions. In the following subsections the three stages are described in detail.

3.1. Data compression

Let

$$\mathbf{X} = [m_1, M_1] \times \cdots \times [m_n, M_n] \subseteq \mathbb{R}^n \tag{1}$$

be a n-dimensional Universe of Discourse,

$$C = \left\{c^1, c^2, \ldots, c^{n_C}\right\}$$

a finite set of classes, $\mathbf{D} \subset \mathbf{X}$ a finite set of n_D available data samples, and $\mathrm{cl} : \mathbf{D} \mapsto C$ a classification function that assigns a class label to each available data sample.[1]

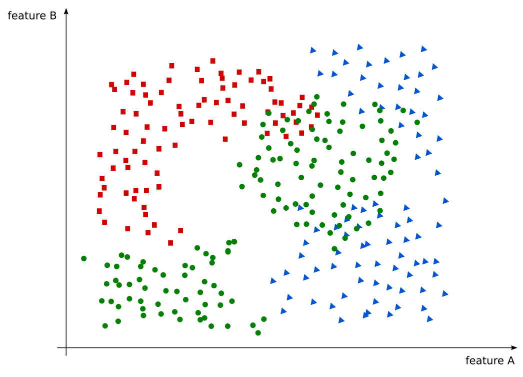
The data compression stage is aimed at defining a collection $\mathbf{P} \subset \mathbf{X}$ of $n_P \ll n_D$ multidimensional prototypes — representing aggregate information of the available samples — along with a classification function $\mathrm{cl} : \mathbf{P} \mapsto C$, so that the class of most data samples surrounding a prototype is the same as the prototype class. An illustration of such a compression of data is depicted in Fig. 1.

A straightforward approach to produce the collection of prototypes comes from the application of Learning Vector Quantization (LVQ) [48] because it does not require the storage of a partition matrix (as required by many clustering algorithms such as Fuzzy C-Means) and it exploits class information to derive classified prototypes to represent classified data in a compressed form.

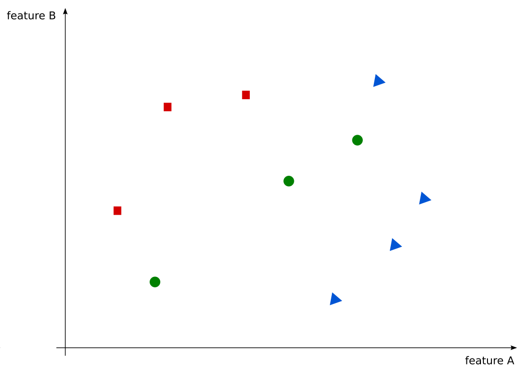
In this work we use the simplest version of LVQ (LVQ1), which is the core of the data compression stage, as described in Algorithm 1. (Euclidean distance has been used; however any distance function can be applied.)

LVQ1 requires the specification of the number n_P of prototypes: this is the single hyper-parameter requested from the user (it just suffices for the rest of the parameters in the algorithm to be set to default values). We use this hyper-parameter to regulate the compression ratio of the samples in order to set the granularity level of the fuzzy information granules defined in the third stage of DC*f*.

[1] In this work, crisp classification has been adopted. However, soft classification with confidence degrees [49] is possible in principle and matter of future developments.



(a) Data samples.



(b) Derived prototypes.

Fig. 1. The data compression stage applied on a dataset characterized by two features and three different classes. The final aggregation is expressed in terms of ten prototypes.

Algorithm 1 Data compression

```
Input: dataset $\mathbf{D}$
Input: classification function $\mathrm{cl} : \mathbf{D} \mapsto C$
Input: the number of prototypes $n_P$
Input: learning rate $\alpha$
Input: tolerance $\epsilon$
Input: max. iterations $\max_i$
Output: the collection of prototypes $\mathbf{P}$ with classification $\mathrm{cl} : \mathbf{P} \mapsto C$
  (* Step 1: prototype initialization. Prototypes are randomly initialized with data samples. The class distribution of prototypes reflects the class distribution in data. *)
  $\mathbf{P} \leftarrow \emptyset$
  for $i \leftarrow 1$ to $n_C$ do
    $\mathbf{D}^i \leftarrow \{\mathbf{x} \in \mathbf{D} | \mathrm{cl}(\mathbf{x}) = c^i\}$
    $n_P^i \leftarrow \mathrm{round}((|\mathbf{D}^i| / |\mathbf{D}|) * n_P)$ (* num. of prototypes of class i *)
    for $j \leftarrow 1$ to $n_P^i$ do
      $\mathbf{x} \leftarrow \mathrm{random\_selection}(\mathbf{D}^i)$
      $\mathbf{p} \leftarrow \mathbf{x}$
      $\mathbf{P} \leftarrow \mathbf{P} \cup \{\mathbf{p}\}$
      $\mathrm{cl}(\mathbf{p}) \leftarrow \mathrm{cl}(\mathbf{x})$
    end for
  end for
  (* Step 2: Vector quantization *)
  $i \leftarrow 1$
  do
    $E \leftarrow 0$
    for each $\mathbf{x} \in \mathbf{D}$ do
      $\mathbf{p} \leftarrow \arg\min_{\mathbf{p}' \in \mathbf{P}} ||\mathbf{p}' - \mathbf{x}||$ (* In case of ties a random choice is applied *)
      $\mathbf{p}_{old} \leftarrow \mathbf{p}$
      if $\mathrm{cl}(\mathbf{x}) = \mathrm{cl}(\mathbf{p})$ then
        $\mathbf{p} \leftarrow \mathbf{p} + \alpha(\mathbf{x} - \mathbf{p})$
      else
        $\mathbf{p} \leftarrow \mathbf{p} - \alpha(\mathbf{x} - \mathbf{p})$
      end if
      $E \leftarrow E + \|\mathbf{p} - \mathbf{p}_{old}\|$
    end for
    $i \leftarrow i + 1$
    $\alpha \leftarrow \alpha - \alpha/\max_i$
  while $E > \epsilon$ and $i < \max_i$
```

3.2. One-dimensional clustering

In this stage the prototype collection $\mathbf{P}$ is projected onto each dimension $d = 1, 2, \ldots, n$. Also, we require that the elements inside each projection are sorted in ascending order. To this aim, we need to define a projection operator to be applied on each dimension d:

$$\Pi_d(\mathbf{x}) = \Pi_d(x_1, x_2, \ldots, x_n) = x_d \qquad \forall \mathbf{x} \in \mathbf{X}$$

i.e., Π_d returns the dth coordinate of a vector, and a sorting function that, given a set $A \subset \mathbb{R}$ of elements, carries out a sorted sequence:

$$\mathrm{sort}\, A = \left(x^i | x^i \in A\right)$$

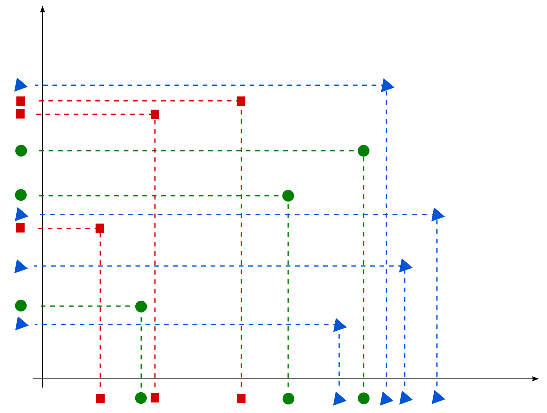


Fig. 2. Projections of the prototype collection depicted in Fig. 1b.

such that $x^i \leq x^{i+1}$.

Given the prototype collection $\mathbf{P}$, a projection of its elements over every dimension can be conceived. For each dimension, the projection of the different prototypes is sorted:

$$P_d = \mathrm{sort}\{\Pi_d(\mathbf{p}) | \mathbf{p} \in \mathbf{P}\} = \left(p_d^i\right)_{i=1,2,\ldots,n_P}$$

being $d = 1, 2, \ldots, n$. Each projection retains class information by inheriting, for each element inside the projection, the class label of the multidimensional prototypes. Formally speaking, for each $d = 1, 2, \ldots, n$ a function $\mathrm{cl}_d : P_d \mapsto C$ is derived by defining $\mathrm{cl}_d(p) = \mathrm{cl}(\mathbf{p})$ whenever $p = \Pi_d(\mathbf{p})$. In Fig. 2 the projections on a bi-dimensional space of a collection of prototypes are illustrated.

Given a projected prototype sequence P_d, clustering is performed by operating on the related sequence of *cuts*. Roughly speaking, a cut is a boundary of an information granule and is formally defined by the midpoint between two elements inside a projection labeled with different classes. More precisely, the sequence of all midpoints is defined as

$$Q_d = \left(\frac{p_d^i + p_d^{i+1}}{2}\right)_{i=1,2,\ldots,n_P-1}$$

The sequence of cuts is the sub-sequence $T_d \subseteq Q_d$ where $\mathrm{cl}_d\left(p_d^i\right) \neq \mathrm{cl}_d\left(p_d^{i+1}\right)$. The cardinality of T_d is denoted by n_{T_d}. Any sub-sequence $S_d \subseteq T_d$ of cardinality n_{S_d} determines a clustering of the projection P_d, where each cluster is defined by all the elements inside P_d that are not separated by any cut in S_d. Fig. 3 illustrates the sequences T_d and a subsequence S_d of cuts (with $d = 1,2$) related to the projections depicted in Fig. 2.

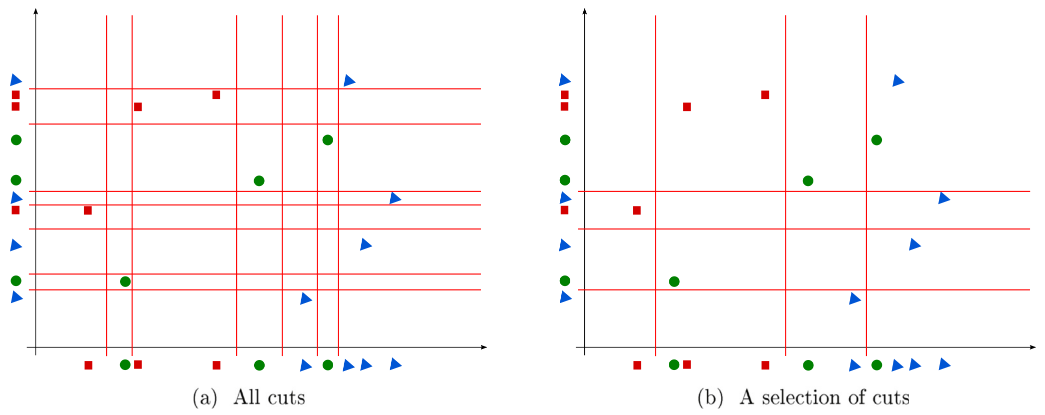


Fig. 3. The collection of all cuts and an example of selection identified for the dataset reported in Fig. 2.

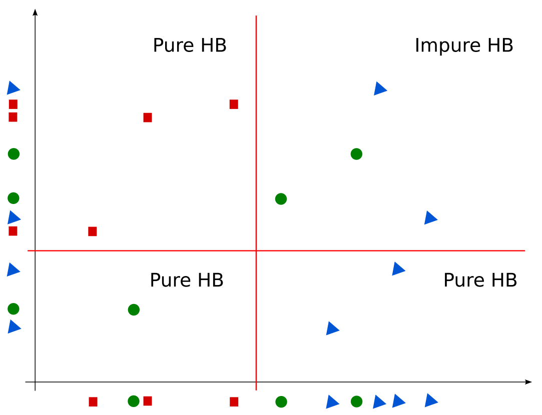


Fig. 4. A bi-dimensional Universe of Discourse partitioned into pure and impure hyper-boxes.

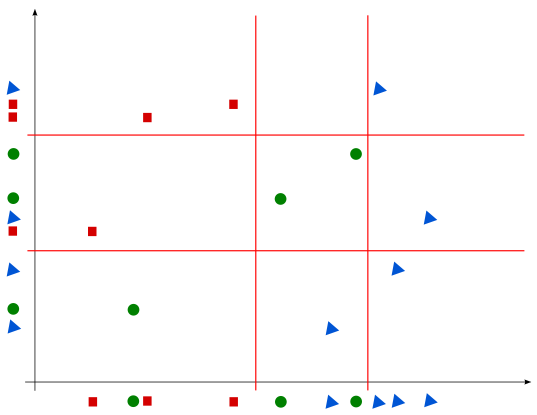


Fig. 5. An optimal configuration of cuts.

The objective of one-dimensional clustering is to provide for the coarsest granulation of the feature space, subject to separating prototypes of different classes into different granules. To obtain such a result, an *optimal* configuration of sub-sequences S_d, spanning the entire set of feature dimensions, has to be identified. In order to define optimality, the concept of *hyper-box* must be introduced first. To this pursuit, given a sub-sequence of cuts

$$S_d = (s_d^i)_{i=1,\dots,n_{S_d}} \tag{2}$$

we define an extended sequence that adds the boundary points (1) in dimension d to the set of cuts:

$$\bar{S}_d = \left(m_d, s_d^1, s_d^2, \dots, s_d^{n_{S_d}}, M_d\right)$$

A hyper-box is simply defined as the Cartesian product of intervals, one for each dimension $d = 1, \dots, n$, delimited by cuts in $\bar{S}_d$:

$$\mathbf{B}_{k_1,k_2,\dots,k_n} = \left[s_1^{k_1-1}, s_1^{k_1}\right] \times \cdots \times \left[s_n^{k_n-1}, s_n^{k_n}\right] \tag{3}$$

where $k_d = 1, \dots, n_{S_d} + 1$, with the convention of denoting $s_d^0 = m_d$, and $s_d^{n_{S_d}+1} = M_d$.

A hyper-box contains zero or more multi-dimensional prototypes in **P**. A hyper-box is said *pure* if it does not contain prototypes or all prototypes it contains belong to the same class; otherwise it is said *impure*.

Fig. 4 depicts the previously illustrated bi-dimensional space with a configuration of cuts producing one impure and three pure hyper-boxes (HBs).

A pure and non-empty hyper-box is a surrogate of an information granule: if prototypes contained in a hyper-box are surrounded by data samples, then most of those samples are also inside the hyper-box. In this sense, a configuration of cuts spanning the entire set of feature dimensions is optimal if it produces *only* pure hyper-boxes in such a way that the number of cuts is *minimized*. The objective of one-dimensional clustering is therefore to find such an optimal configuration of cuts (see an example in Fig. 5).

The clustering problem has exponential complexity since the number of candidate solutions is $\prod_d 2^{n_{T_d}} \sim \mathcal{O}(2^{n \cdot n_P})$. To tackle the problem, we adopt a strategy based on the A* algorithm. A* is a well-known general informed search algorithm that can be applied to any discrete problem provided that it is fully specified by a suitable structure. In DC*, we specialize A* so as to operate an informed search on the space defined by the set of all possible clustering configurations. Although the computational complexity of A* is still exponential in the worst case, a careful design of all its components can carry out a fast search of the optimal solution in most cases.

The adopted implementation of A* explores a *search space* $\mathbf{\Sigma}$, i.e. a set of configurations of cuts – called nodes – that represent candidate solutions. In our problem a candidate solution is a partition of the Universe of Discourse into hyper-boxes which are completely characterized by cuts. Therefore, a convenient way to represent a node is through a tuple:

$$\sigma = (S_1, S_2, \dots, S_n)$$

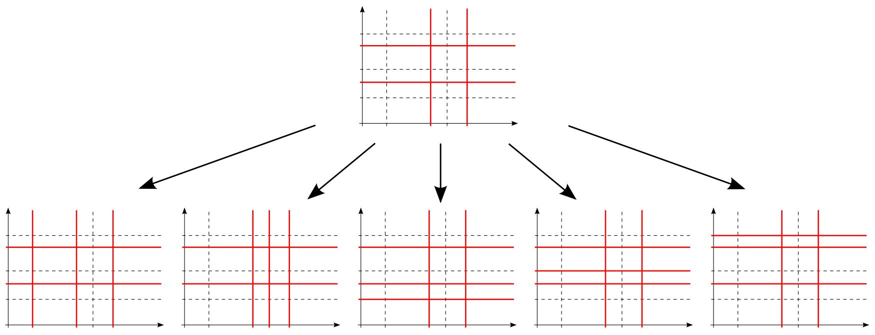


Fig. 6. An example of application of the successor operator.

being S_d a sub-sequence of cuts standing on dimension d as in Eq. (2). The initial node from which A* starts the search is:

$$\sigma_0 = \left(\underbrace{\emptyset, \emptyset, \ldots, \emptyset}_{n}\right)$$

which corresponds to a single hyper-box that coincides with the entire Universe of Discourse.

The structure of A* used in this work[2] is reported in Algorithm 2.

Algorithm 2 The A* algorithm used for one-dimensional clustering

```
Input: the initial node $\sigma_0$
Input: the goal test $\mathcal{G} : \mathbf{\Sigma} \mapsto \{\text{true}, \text{false}\}$
Input: the successor operator $O : \mathbf{\Sigma} \mapsto 2^{\mathbf{\Sigma}}$
Input: the path-cost function $g : \mathbf{\Sigma} \mapsto \mathbb{N}$
Input: the heuristic function $h : \mathbf{\Sigma} \mapsto \mathbb{N}$
Output: the optimal solution $\sigma^*$
  closed $\leftarrow \emptyset$ (* the set of all visited nodes *)
  open $\leftarrow \sigma_0$ (* the queue of nodes to be visited *)
  while open $\neq \emptyset$ do
    $\sigma \leftarrow$ dequeue(open) (* pick the most promising node *)
    closed $\leftarrow$ closed $\cup\{\sigma\}$ (* mark the node as visited *)
    if $\mathcal{G}(\sigma) =$ true then (* node is optimal: return it *)
      $\sigma^* \leftarrow \sigma$
      return $\sigma^*$
    else
      $\Sigma \leftarrow O(\sigma)$ (* node is not optimal: generate successors *)
      for each $\sigma' \in \Sigma$ do
        cost $\leftarrow g(\sigma') + h(\sigma')$ (* estimate cost of the node *)
        queue(open, $\sigma'$, cost) (* add node to priority queue *)
      end for
    end if
  end while
```

A* requires the specification of a goal test, the successor operator, the cost function and a heuristic function. The goal test simply verifies if the cuts defining a node provide for pure hyper-boxes only. In such a case, the search algorithm is allowed to terminate: because of the operating mode of A*, the first goal node met during the process is optimal for sure, being characterized by the minimum number of cuts.

3.2.1. *Successor operator*

The successor operator $O : \mathbf{\Sigma} \mapsto 2^{\mathbf{\Sigma}}$ generates a set of nodes given an input node. Each node is distinguished from the previous one by the presence of one additional cut in one dimension only. Formally, given two nodes $\sigma = (S_1, S_2, \ldots, S_n)$ and $\sigma' = (S'_1, S'_2, \ldots, S'_n)$, then $\sigma' \in O(\sigma)$ if and only if these two conditions are satisfied:

1. The nodes σ and σ' are distinguished by one dimension d only, i.e. $S_d \neq S'_d$ and $S_{d'} = S'_{d'}$ for each $d' \neq d$;

[2] We adopt a simplified version of A* because we do not need to track the path from the initial node to the goal node.

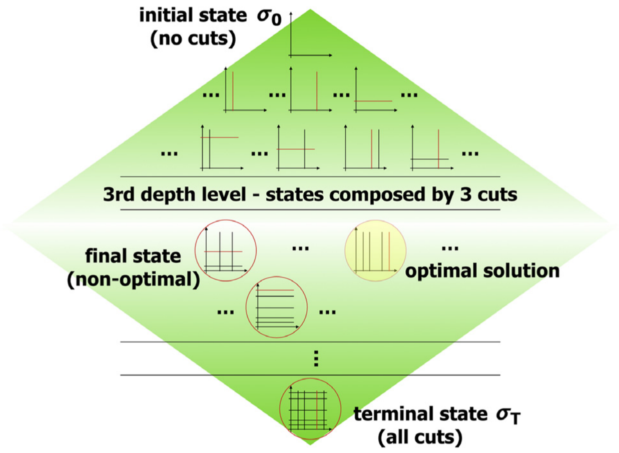


Fig. 7. A synopsis of the search space, structured according to the successor operator.

2. The sequences S_d and S'_d are distinguished by one cut t only, i.e. $S'_d \setminus S_d = \{t\}$.

Therefore, starting from a node σ with $n_\sigma = \sum_d n_{S_d}$ cuts, the successor operator produces $n_T - n_\sigma$ nodes with $n_\sigma + 1$ cuts, being $n_T = \sum_d n_{T_d}$. In Fig. 6 an example of successors generation is depicted.

The additive property of the successor operator enables the generation of the terminal node:

$$\sigma_T = (T_1, T_2, \ldots, T_n) \tag{4}$$

i.e. the node where all cuts are used. By construction, the cuts of the terminal node generate only pure hyper-boxes (although possibly not in minimal number): this guarantees that A* will eventually terminate on a goal node. The overall structure of the search space is schematized in Fig. 7.

3.2.2. *Cost function*

The cost function $f : \mathbf{\Sigma} \mapsto \mathbb{N}$ evaluates a node σ in the A* search phase: the lower the cost, the more promising the node in view of reaching a solution for the problem. The cost function is defined as the sum of the *path-cost* function and the *heuristic* function:

$$f(\sigma) = g(\sigma) + h(\sigma) \tag{5}$$

The path-cost function $g : \mathbf{\Sigma} \mapsto \mathbb{N}$ simply counts the number of cuts that are used in a node, i.e.

$$g(\sigma) = \sum_d n_{S_d} = n_\sigma$$

while the heuristic function is more complex because it must estimate the minimum number of cuts required to reach a goal node from the current node.

The heuristic function covers a fundamental role in the A* search process since it is useful to estimate the cost of an optimal solution starting from a given node. To ensure optimality of A*, the heuristic function must be *admissible*, i.e. it must never overestimate the true cost; however, an estimation closer to the true cost translates into a more efficient search process performed by A*.

The heuristic function of DC* exploits class information to compute an estimate of the cost function. In this paper we propose a heuristic function that exploits the number of classes to estimate the cost, thus resulting more informative as the number of class labels increases. To define the heuristic function, we consider an impure hyper-box[3] **B** generated by a node[4] σ. Since **B** is impure, it includes at least two prototypes belonging to different classes. We denote by $n_c^{\mathbf{B}}$ the number of distinct class labels attached to the prototypes in **B**. (Obviously, $n_c^{\mathbf{B}} \geq 2$.) To make **B** pure, it must be split into at least $n_c^{\mathbf{B}}$ hyper-boxes by a number of cuts. The objective, therefore, is to find the minimum number of cuts splitting **B** into $n_c^{\mathbf{B}}$ hyper-boxes.

Let $T_d^{\mathbf{B}} \subseteq T_d$ be the subset of cuts standing on dimension d that intersect **B** according to its definition (3), i.e.

$$T_d^{\mathbf{B}} = \left\{ t \in T_d | s_d^{k_d} < t < s_d^{k_d+1} \right\}$$

and let $S_d^{\mathbf{B}}$ be a selection of such cuts, i.e. $S_d^{\mathbf{B}} \subseteq T_d^{\mathbf{B}}$. For each $d = 1, 2, \ldots, n$, we can use the cuts in $S_d^{\mathbf{B}}$ to split **B** in a number n_B of hyper-boxes which can be evaluated as:

$$n_B = \prod_{d=1}^{n} \left(|S_d^{\mathbf{B}}| + 1 \right) = \prod_{d=1}^{n} (s_d + 1)$$

being s_d the cardinality of $S_d^{\mathbf{B}}$. Thus, a necessary (yet not sufficient) condition for selecting the cuts to split **B** into pure hyper-boxes is: $n_B \geq n_c^{\mathbf{B}}$.

In order to guarantee admissibility, the number of selected cuts in each dimension must be minimal. Furthermore, we are interested in the number of cuts for each dimension, but not in their effective positioning (because we do not consider the location of the prototypes in the hyper-box). Thus, the following minimization problem must be solved:

$$\begin{aligned} \text{minimize} : & \sum_d s_d \\ \text{subject to} : & \prod_{d=1}^{n} (s_d + 1) \geq n_c^{\mathbf{B}} \\ & 0 \leq s_d \leq \left| T_d^{\mathbf{B}} \right| \end{aligned} \tag{6}$$

An efficient way to solve (6) is through Algorithm 3. The result of the minimization problem is denoted by $h_{\mathbf{B}}$.

The value $h_{\mathbf{B}}$ is an optimistic estimate of the number of cuts required to split the hyper-box **B** into pure hyper-boxes. To define the heuristic function for a node σ (possibly including more than one impure hyper-box) we must take into account that a single cut may intersect more than one hyper-box. In order to estimate the number of cuts, impure hyper-boxes are grouped together if they share at least one interval. More formally, given two (impure) hyper-boxes $\mathbf{B}_{k_1,k_2,\ldots,k_n}$ and $\mathbf{B}'_{k'_1,k'_2,\ldots,k'_n}$, they are *connected* if there exists a dimension d such that $k_d = k'_d$. Such hyper-boxes share the same interval $\left[s_d^{k_d-1}, s_d^{k_d} \right]$ and both of them could be split by a single cut.[5] We will denote by conn **B** the set of all impure hyper-boxes connected to **B** (see Fig. 8 for a schematic example).

[3] We will use the symbol **B** instead of $\mathbf{B}_{k_1,k_2,\ldots,k_n}$ when there is no risk of confusion.

[4] By definition, a node that is not a solution induces at least one impure hyper-box.

Algorithm 3 Minimization of cuts for a single hyper-box

```
Input: the number of classes $n_c^{\mathbf{B}}$
Input: the max. number of cuts for each dimension $t_d = \left|T_d^{\mathbf{B}}\right|$, $d = 1, 2, \ldots, n$
Output: the minimal number of cuts $s_d$, $d = 1, 2, \ldots, n$
  $s_d \leftarrow 0$ for each $d = 1, 2, \ldots, n$
  $i \leftarrow 0$
  while $\prod_{d=1}^{n} (s_d + 1) < n_c^{\mathbf{B}}$ do
    $i \leftarrow (i \mod n) + 1$
    if $s_i < t_i$ then
      $s_i := (s_i + 1)$
    end if
  end while
```

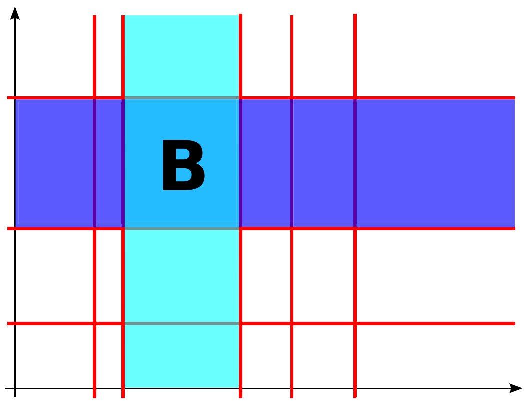


Fig. 8. An example of connected hyper-boxes.

Let $\mathcal{B}$ be the set of all impure hyper-boxes generated by a node σ. The procedure for computing the heuristic function for σ is reported in Algorithm 4. In essence, the procedure iterates over the impure hyper-boxes starting from one hyper-box with maximum number of different class labels. Once a hyper-box is selected, the heuristic value is summed to an accumulator and all connected hyper-boxes are removed. The procedure stops when the collection of hyper-boxes to scan is empty and the value of the accumulator is returned.

Algorithm 4 The heuristic function

```
Input: a node $\sigma$
Output: the heuristic value $h(\sigma)$
  $\mathcal{B} \leftarrow$ the set of impure hyper-boxes generated by $\sigma$
  $h(\sigma) \leftarrow 0$
  while $\mathcal{B} \neq \emptyset$ do
    $\mathbf{B}_{\max} \leftarrow \arg\max_{\mathbf{B} \in \mathcal{B}} n_c^{\mathbf{B}}$
    $h(\sigma) \leftarrow h(\sigma) + h_{\mathbf{B}_{\max}}$
    $\mathcal{B} \leftarrow \mathcal{B} \setminus \text{conn}\mathbf{B}_{\max}$
  end while
```

3.2.3. Priority queue

A* requires a priority queue to temporarily store the nodes that need to be evaluated. With reference to Algorithm 2, the queue is updated at each iteration by adding all the nodes yielded by the successor operator applied to the node under consideration. The nodes in the queue are sorted according to a priority level, so that any node extracted from the queue has the highest priority.

[5] It is easy to show that, if two impure hyper-boxes are connected, then there must be a cut t belonging to T_d that lays in the interval $\left[s_d^{k_d-1}, s_d^{k_d} \right]$.

The priority level is determined by its potentiality of being a goal node or the parent of an optimal node. The priority is inversely proportional to the cost function applied to a node: the smaller the cost, the higher the priority.

Several nodes may have the same priority in the queue. In fact, this is a common condition that is verified in the earliest stages of DC*, when there is not enough information to discriminate among different nodes in terms of their potentiality of being parents of an optimal solution. To avoid a random selection of such nodes, some additional information can be used to guide the selection of a node among the nodes with the highest priority. To this aim, the queue is endowed with a multiple-level priority. An array of priorities $\left[\pi_1(\sigma), \pi_2(\sigma), \ldots, \pi_{n_\pi}(\sigma)\right]$ is attached to each node in the queue: nodes are then sorted according to the first priority level; nodes with the same priority at the first level are sorted according to the priority at the second level, and so on. The priority at the first level is defined by the cost function in order to preserve the correctness of A*, i.e.

$$\pi_1(\sigma) = f(\sigma)$$

being f the cost function defined in Eq. (5), while the remaining priority levels could be used to promote nodes with some desirable qualities.

In DC*, the second priority level is defined to promote nodes that correspond to coarse-grained information granules. Coarse information granules embrace a wide fraction of the Universe of Discourse; therefore they could describe properties that apply to a great number of samples. In other words, promoting coarse-grained information granules could favor the generality of the granulation provided by a solution. To achieve this feature, the second priority level is defined as follows: given a node σ generated by a parent node by adding a cut t_k on dimension d (as described in Section 3.2.1), then

$$\pi_2(\sigma) = \min\{t_k - t_{k-1}, t_{k+1} - t_k\}$$

being t_{k-1} and t_{k+1} the cuts belonging to S'_d, respectively preceding and following t_k. If the cut t_k is very close to an existing cut, then the node σ generates hyper-boxes that are too thin in dimension d. This, in turn, translates into information granules that are too fine-grained. The node σ is therefore penalized, by assigning a second-level priority equal to the distance (so that the smaller the distance, the lower the priority).

A third priority level is also defined in DC*, in order to promote nodes that require few input features for representing information granules. This is achieved by simply counting the input features where at least one cut is defined:

$$\pi_3(\sigma) = -\sum_{d=1}^{n}\left(\chi_{\neq}(S_d, \emptyset)\right)$$

being $\chi_{\neq}(A, B) = 1$ if $A \neq B$, 0 otherwise. The less input features are required by the node, the higher is this priority value. It is worth observing that feature selection comes as a by-side effect of the optimal search procedure performed by A*; therefore this priority level is used just as a guide when more than one node have the highest priorities at both the first and second level. In practice, this third level is used quite rarely, therefore there is no need to define further priority levels in the queue: in case of nodes with the same highest priority at all levels, a random choice is performed.

3.3. *Fuzzy granulation*

Once an optimal solution has been found by A*, it is translated into a granulation of the Universe of Discourse. This requires the following steps:

1. Fuzzy partition of the selected input features;
2. Generation of the multi-dimensional information granules;
3. Linguistic representation of the information granules in terms of fuzzy classification rules.

The first step is accomplished by taking into account the sets of cuts selected for each feature. If no cuts are selected for a specific feature, that feature is discarded. Since DC* is a cut-based granulation method, we resort to trapezoidal fuzzy sets in order to define fuzzy partitions. This kind of sets, in fact, enable the definition of Strong Fuzzy Partitions (SFPs) — thus preserving a number of interpretability requirements — and are more flexible than triangular fuzzy sets [50]. Trapezoidal fuzzy sets are defined as follows:

$$T[a,b,c,d](x) = \begin{cases} \dfrac{x-a}{b-a}, & x \in [a,b[\\ 1, & x \in [b,c] \\ \dfrac{x-d}{c-d}, & x \in]c,d] \\ 0, & x < a \vee x > d \end{cases}$$

with the convention that $T[a,a,c,d](a) = 1$ and $T[a,b,c,c](c) = 1$.

Formally, let $\sigma^* = \left(S_1^*, S_2^*, \ldots, S_n^*\right)$ be the optimal solution found by A* and let d be such that

$$S_d^* = \left(s_d^{*i}\right)_{i=1}^{n_{S_d^*}} \neq \emptyset$$

The sequence S_d^* divides the dth input feature into $n_{S_d^*}+1$ intervals $\left[s_d^{*i-1}, s_d^{*i}\right]$ for $i = 1, 2, \ldots, n_{S_d}+1$, with the convention of denoting $s_d^{*0} = m_d$ and $s_d^{*n_{S_d}+1} = M_d$.

The final fuzzy partition is defined by the collection of trapezoidal fuzzy sets $T\left[a_d^i, b_d^i, c_d^i, d_d^i\right]$ for $i = 1, 2, \ldots, n_{S_d^*}+1$. In this way, the Universe of Discourse is partitioned like in Fig. 9. Among several alternatives to define the parameters of a Trapezoidal SFP (TSFP) we adopt the Variable Fuzziness technique, which gives more fuzziness to information granules with coarser granularity [50]. The Variable Fuzziness technique can be formalized as follows.

For each $i = 1, \ldots, n_{S_d^*}$, let $\Delta_i = s_d^{*i} - s_d^{*i-1}$. If $\Delta_i \leq \Delta_{i+1}$ then

$$c_d^i = \frac{s_d^{*i-1} + s_d^{*i}}{2}, \; d_d^i = 2s_d^{*i} - c_d^i.$$

This ensures that the descending part of the ith fuzzy set starts from the midpoint of the ith interval and intersects the ith cut at 0.5. The ascending part of the $(i+1)$-th fuzzy set is defined accordingly to ensure that the fuzzy partition is strong:

$$a_d^{i+1} = c_d^i, \; b_d^{i+1} = d_d^i.$$

If $\Delta_i > \Delta_{i+1}$ the ascending part of the $(i+1)$-th fuzzy set intersects the ith cut at 0.5 and ends at the midpoint of the $(i+1)$-th interval. Formally:

$$b_d^{i+1} = \frac{s_d^{*i} + s_d^{*i+1}}{2}, \; a_d^{i+1} = 2s_d^{*i} - b_d^{i+1}.$$

Consequently,

$$c_d^i = a_d^{i+1}, \; d_d^i = b_d^{i+1}.$$

Finally, the definition of the leftmost and rightmost fuzzy sets is completed by setting $a_d^1 = b_d^1 = m_d$ and $c_d^{n_{S_d^*}+1} = d_d^{n_{S_d^*}+1} = M_d$.

Multi-dimensional information granules are defined by combining the fuzzy sets belonging to the selected features. However, only those combinations that describe the underlying data are retained, while all the others are discarded. This is achieved by

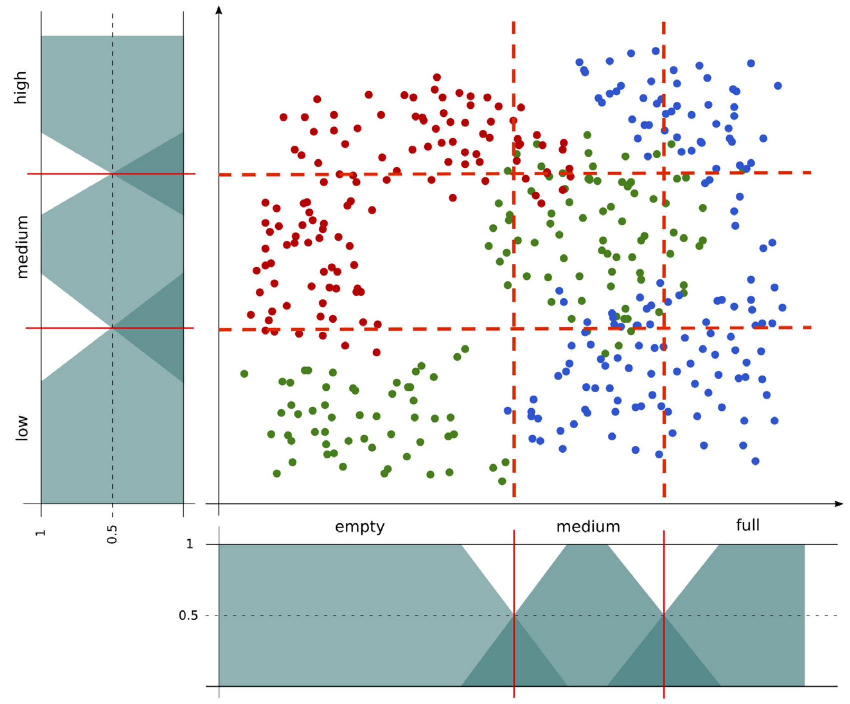


Fig. 9. The final fuzzy partition of the bi-dimensional space performed in terms of trapezoidal fuzzy sets.

defining fuzzy information granules from the non-empty hyper-boxes of the optimal solution as the Cartesian product of trapezoidal fuzzy sets[6]:

$$\widetilde{\mathbf{B}}_{k_1,k_2,\ldots,k_n}(\mathbf{x}) = \bigwedge_{d=1}^{n} T\left[a_d^{k_d}, b_d^{k_d}, c_d^{k_d}, d_d^{k_d}\right](x_d), \qquad (7)$$

being $\wedge$ a t-norm. By definition, all such hyper-boxes are pure, therefore they can be tagged with the class label of the multi-dimensional prototypes falling in them. As a consequence, each fuzzy information granule can be linguistically represented as a classification rule. To achieve this, a linguistic variable is defined for each selected feature. The linguistic variable assigns a linguistic term to each fuzzy set of the fuzzy partition. If L_d^i designates the linguistic term assigned to fuzzy set $T\left[a_d^i, b_d^i, c_d^i, d_d^i\right]$, then the classification rule of the fuzzy information granule (7) follows the schema

IF x_1 is $L_1^{k_1}$ AND ... AND x_n is $L_n^{k_n}$ THEN c,

being c the class of the corresponding hyper-box. In the schema, the discarded features are not included to improve readability. An optimal solution has a set of non-empty (pure) hyper-boxes, therefore a collection of fuzzy classification rules can be derived. This gives rise to a fuzzy rule-based classifier: given an input $\mathbf{x}$, the membership degrees of all information granules are computed according to Eq. (7); then the class of the information granule with the highest membership degree is assigned to the input. Inputs not belonging to any information granule are not classified.[7]

4. Experimental results

The objective of the experimentation is to evaluate DC* in terms of both accuracy and interpretability. To this end we arranged three experimental sessions. The first session is aimed at verifying the ability of the heuristic function, which plays a key role in the A* search procedure, in accelerating the search process towards a goal state. Then we test DC* on a number of benchmark datasets: in this case the focus is pointed on evaluating the interpretability and the accuracy exhibited by the derived predictive models. Comparative experimental sessions are reported to assess the capabilities of DC* with respect to a number of other classification techniques. Among them, a specific method known in literature, namely the Hierarchical Fuzzy Partitioning algorithm (HFP) is specially considered for a thorough comparative analysis.

4.1. Evaluation of the heuristic function

The goal of this preliminary experimental stage is to test the effectiveness of the heuristic function adopted to drive the A* search. To this aim, we referred to a specifically built synthetic problem named "k-chessboard" which allows to investigate the heuristic function behavior as the complexity of the problem at hand is growing.

As depicted in Fig. 10, the k-chessboard is a particular bi-dimensional dataset with 2 classes, shaped to form a regular grid, where points of the same class must not be adjacent (with respect to the orthogonal directions). The value of k refers to the number of samples over each grid side: higher values for k correspond to problems of increasing complexity, requiring more information granules to provide proper partitioning. Since we are only interested in the performances of the heuristic function, we skip the data compression stage, thereby the points of the chessboard coincide with the prototypes used in the second stage of DC*. The k-chessboard problem consists in separating points of different classes by cutting the plane horizontally or vertically. The problem admits one solution only, coinciding with the terminal state (4), where all the possible cuts are included. As such, this kind of solution stands as the worst scenario for the searching process carried out by DC*. In fact, it requires to penetrate through the whole search graph, heavily stressing the A* search procedure. This is the main reason why this particular problem has been chosen for the preliminary experimental stage. Moreover, given an intermediate configuration of cuts, it is possible to calculate

[6] If a feature d' is discarded, a dummy fuzzy set $T[a_{d'}, b_{d'}, c_{d'}, d_{d'}] = 1$ is used.
[7] This is counted as a classification error.

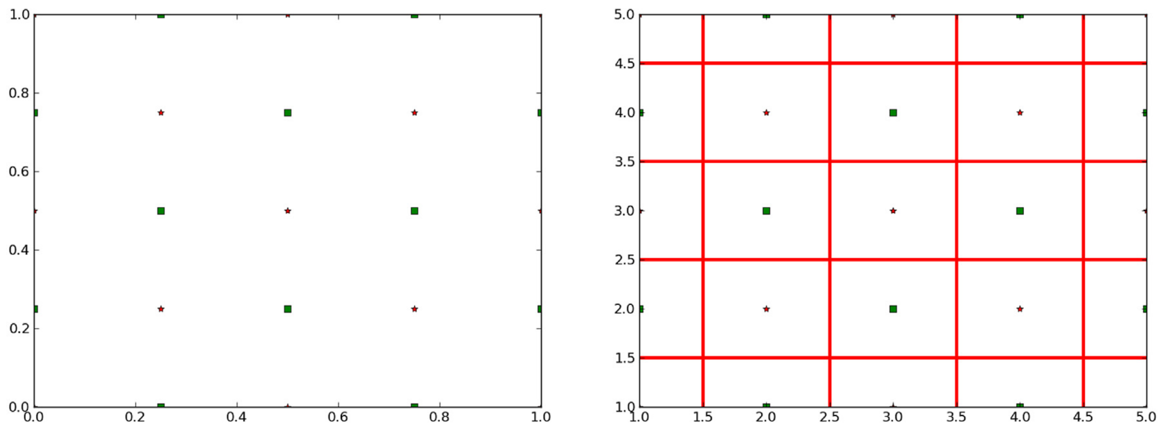


Fig. 10. The k-chessboard dataset with $k = 5$. On the left side the dataset distribution is depicted. On the right side, the resolutive configuration of cuts.

the exact number of cuts required to reach the final state. Thus, in order to evaluate the effectiveness of the heuristic function, two baselines have been identified:

- a constant-valued heuristic function (which is ultimately uninformative for the search process);
- the optimal heuristic function (which is predictable for the chessboard problem).

Both of them have been adopted in the experimentation illustrated in the following; such a parallel evaluation aims at comparing the behavior of the heuristic function implemented in DC* with respect to the utmost adverse and favorable conditions. Particularly, the optimal heuristic function directly drives the search process toward the exploratory path involving the minimum number of nodes, that is the depth-first search dictated by the consecutive addition of a new cut at each step (until all the possible cuts are considered). The constant heuristic, on the other hand, simply collapses the evaluation of each node to the constant value 1: this disables any discriminating power, thus failing to prevent the search process from stagnating in inefficient breadth-first directions.

Experiments have been conducted over nine progressive k-chessboard datasets, with k ranging from 2 to 10, recording the number of tested nodes while passing through the adoption of the three different heuristic functions. It can be noted that, for the optimal heuristic case, the number of tested nodes is linearly determined by the value of k (being the total number of all possible cuts equal to $2k - 2$).

Fig. 11 summarizes the experiment results highlighting the dependence between the complexity of the problem and the number of tested nodes. In particular, if the problem complexity increases, then the number of tested nodes: (a) linearly grows when the optimal heuristic function is applied (best scenario); (b) exponentially grows when the uninformative heuristic function is applied (worst scenario); (c) exponentially grows when the DC* heuristic function is applied, but an appreciable reduction of the number of tested nodes is registered. This demonstrates an increased supply of informative power of the DC* heuristic function when compared with the constant heuristic function.

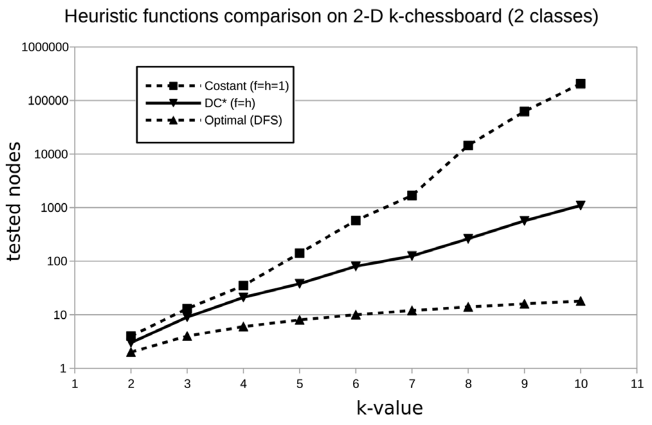


Fig. 11. Heuristic functions evaluation on the k-chessboard problem. The number of tested nodes is reported during the application of the constant, the optimal, and the DC* heuristic functions, as problem complexity increases.

Table 1
Illustration of the datasets involved in the experimental sessions.

Dataset	ID	#samples	#features	#classes
Appendicitis	Ap	106	7	21
Balance	Bl	625	4	3
Banana	Bn	5300	2	2
Beer styles	BS	400	3	8
Bupa	Bu	345	6	2
Cardiotocography	CTG	2126	21	3
Hayes-Roth	Hy	160	4	3
Ionosphere	Ion	351	33	2
Iris	Ir	150	4	3
Monk-2	Mo	432	6	2
Newthyroid	Nth	215	5	3
Page-Blocks	PB	5472	10	5
Phoneme	Ph	5404	5	2
Pima	Pi	768	8	2
Saheart	Sh	462	9	2
Sonar	So	208	60	2
Thyroid	Thy	7200	21	3
Vertebral 2	V2	310	6	2
Vertebral 3	V3	310	6	3
Wisconsin breast cancer	WBC	683	9	2
Wine	Wi	178	13	3

4.2. Evaluation of DC* on benchmark datasets

We tested DC* by referring to a number of benchmark datasets widely employed in literature. Particularly, we selected a collection of 21 datasets concerning classification problems, among those freely available online (we referred to the UCI [51], KEEL [52], and Mendeley[8] repositories). As illustrated in Table 1, the selected datasets pertain to classification problems showing relevant differences in terms of complexity, number of samples, features and classes involved. In this sense, they appear suitable to investigate the DC* results in terms of both interpretability and accuracy in a variety of scenarios.

[8] doi:10.17632/n4b6734rfn.1.

We established to launch four runs for each dataset, starting from an initial number of prototypes which is equal to the number of classes and then doubling such number at each successive run. Additionally, the test sessions have been conducted on the basis of 10-fold cross validation for each dataset.

Tables 2–3 summarize the obtained results, reported in terms of mean values evaluated at the end of the 10-fold cross validation sessions. A number of considerations spring from the analysis of the values reported in table. As concerning interpretability, DC* shows the capability to produce predictive models which describe data by a reduced number of rules. Such a property, which is relevant to increase the readability of the derived base of knowledge, is preserved even while increasing the number of prototypes. Furthermore, the capability of DC* to operate a proper feature selection is manifest when comparing the number of features involved in the final models with those originally describing the problems at hand. Actually, the classification models seldom resort to the entire set of features (less than half of the total features has been employed for almost every problem) and in some cases the reduction ratio appears to be quite drastic. This contributes to simplify the structure of the derived models. Finally, the reported values of standard deviation, evaluated both for the number of rules and features, indicate that a good description stability is also appreciable. On the other hand, the observed values of standard deviation on classification errors are motivated by the high sensibility of LVQ to initialization settings.

DC* is designed to provide predictive models exhibiting a fair trade-off between accuracy and interpretability. In this sense, even if the experimental session was not oriented to provide optimal results in absolute terms of classification error, the accuracy values reported in table appear to be quite satisfactory. Moreover, it should be emphasized that such results, differently from what happens when dealing with a number of different data mining methods, have been obtained by setting one hyper-parameter only, i.e. the number of prototypes to start the data compression stage (whose role in determining the granularity of the final model is self-evident). In this way, the application of DC* allows to automatically detect the suitable fuzzy information granules that are identified in terms of number, shape and amplitude.

For the majority of datasets, the best accuracy results (reported in bold in Tables 2–3) have been obtained while setting for the data compression stage a higher number of prototypes. This is consistent with the idea that a greater number of prototypes shifts the balance of the trade-off towards accuracy (at the expense of interpretability). However, in limited cases the performance of the derived models gets worse: this can be explained in terms of the over-fitting occurring during some experimentations, thus preventing the final model to correctly classify new problem samples.

In order to qualitatively appreciate the interpretability of the FRBCs derived by DC*, in Fig. 12 an example of FRBC induced from BS data is depicted. It can be observed that, for each feature, the number of fuzzy sets is automatically determined and all fuzzy sets can be labeled by intelligible linguistic terms. Rules have an immediate readability and linguistically explain each class of the dataset.

4.3. Comparative evaluation

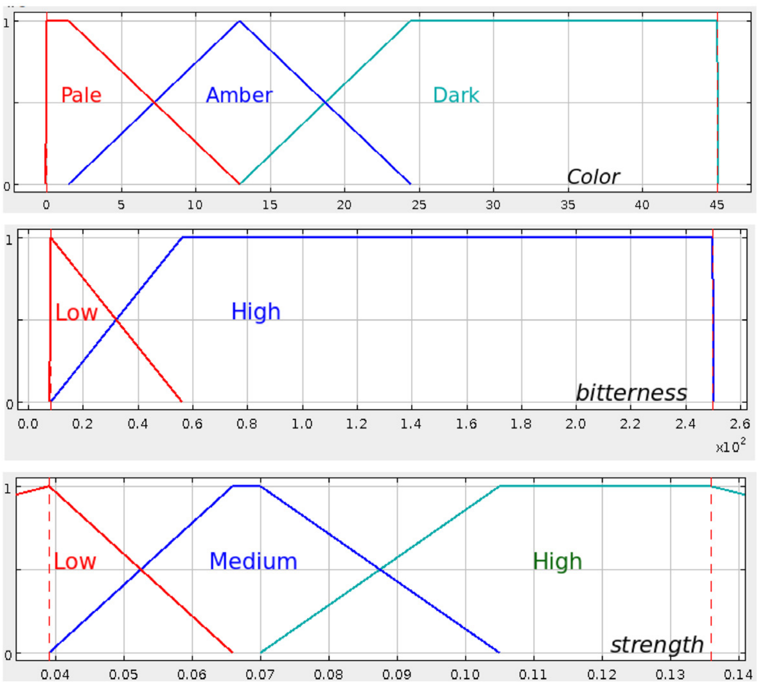
In order to complete the assessment of DC*, we set up a twofold experimental session devoted to a comparison evaluation. Our experiments involved both a pool of heterogeneous classifiers (standing as a general class of reference algorithms) and a different granulation method (which is consistent with the general principles underlying DC*).

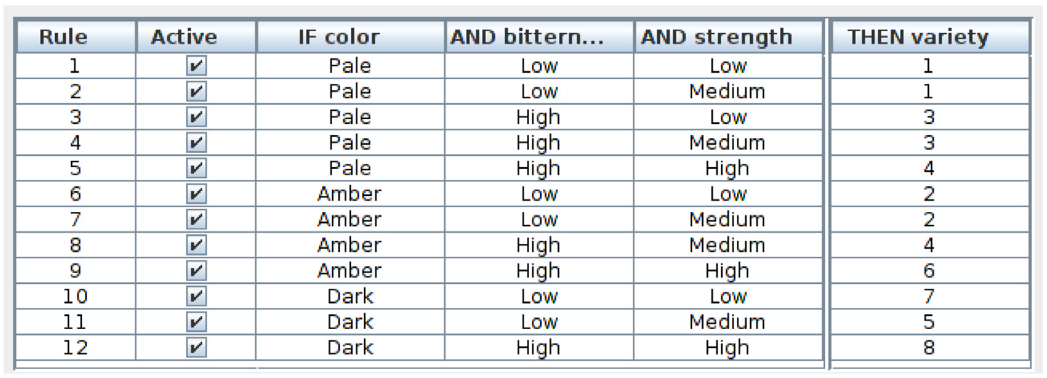
As a preliminary evaluation, we compared the behavior of DC* with the performance exhibited by a number of classification methods popularized in literature. Such a comparison has been performed on the basis of the previously described datasets which were used to train five classifiers: CN2 rule inducer (CN2), C4.5 classification tree (C4.5), Naive Bayes classifier (NB), Support Vector Machine for Classification (C-SVM), and k-Nearest Neighbor (kNN). We used Orange[9] to perform the experiments, with default parameters for each method; in particular we used $k = 5$ for k-NN and Gaussian kernel for C-SVM. We performed a 10-fold cross validation session for each dataset: the average performance results are reported in Table 4, together with the DC* classification error results coming from Tables 2–3. We are reporting here the performance results of all the parametrizations previously adopted to launch DC*. In fact, the aim of this preliminary evaluation is not the selection of the best DC* model to be tested against the other methods; instead, we want to contextualize DC* in the panorama of classical approaches which do not emphasize interpretability as a main concern. To do that, we simply refer to the previously attained performance results: in this way, each instance of DC* can be seen as an alternative method to be compared with others. In Table 4 the labels DC*×n (with $n = 1, 2, 4, 8$) refer to the choice of the number of prototypes per class settled to launch DC* (in the way it has been described while introducing Tables 2–3): starting from an

Table 2
Experimental results obtained by applying DC* on the datasets pertaining to 10 out of 21 benchmark problems.

Dataset	#Prot.	%Error(±stDev)	#Rules(±stDev)	#Features(±stDev)
Ap	2	18.18 ± 13.48	2 ± 0	1 ± 0
	4	**15.45 ± 12.89**	2 ± 0	1 ± 0
	8	22.73 ± 10.95	2.2 ± 0.4	1.2 ± 0.4
	16	20 ± 15.10	3.8 ± 1.8	1.9 ± 0.83
Bl	3	43.17 ± 9.57	3 ± 0	2 ± 0
	6	45.71 ± 6.76	3.9 ± 0.3	2 ± 0
	12	31.75 ± 3.95	9.2 ± 1.17	3 ± 0
	24	**27.46 ± 8.84**	12.6 ± 3.32	3.4 ± 0.49
Bn	2	49.64 ± 3.04	2 ± 0	1 ± 0
	4	47.98 ± 4.66	2.5 ± 0.67	1.3 ± 0.46
	8	39.74 ± 7.84	4.6 ± 1.2	1.9 ± 0.3
	16	**30.64 ± 5.86**	8 ± 1.26	2 ± 0
BS	8	29 ± 12.90	8 ± 0	2.5 ± 0.5
	16	21 ± 7	8.5 ± 0.67	2.7 ± 0.46
	32	12.5 ± 7.5	9.9 ± 1.14	2.8 ± 0.4
	64	**9.5 ± 6.3**	11.8 ± 1.25	3 ± 0
Bu	2	47.14 ± 6.16	2 ± 0	1 ± 0
	4	46.57 ± 7.88	2.2 ± 0.4	1.2 ± 0.4
	8	47.14 ± 9.41	3.8 ± 0.4	2 ± 0
	16	**40.86 ± 5.12**	7 ± 1.79	2.7 ± 0.64
CTG	3	45.45 ± 7.73	3 ± 0	2 ± 0
	6	**20.56 ± 3.60**	4 ± 0	2 ± 0
	12	21.97 ± 7.21	4 ± 0	2 ± 0
	24	29.58 ± 7.90	4.5 ± 1.28	2.3 ± 0.46
Hy	3	57.5 ± 10.75	3 ± 0	1.8 ± 0.4
	6	59.38 ± 12.26	3.4 ± 0.49	1.8 ± 0.4
	12	54.38 ± 15.32	6.8 ± 1.94	2.5 ± 0.67
	24	**38.75 ± 18.29**	14.9 ± 2.55	3.3 ± 0.46
Ion	2	41.67 ± 5.12	2 ± 0	1 ± 0
	4	38.89 ± 10.17	2 ± 0	1 ± 0
	8	**25 ± 9.46**	2.2 ± 0.6	1.1 ± 0.3
	16	28.33 ± 9.36	3.4 ± 0.92	1.7 ± 0.46
Ir	3	31.33 ± 15.51	3 ± 0	2 ± 0
	6	22.67 ± 10.41	3.3 ± 0.46	2 ± 0
	12	12 ± 12.22	3.1 ± 0.3	1.4 ± 0.49
	24	**8 ± 5.81**	3.2 ± 0.6	1.5 ± 0.5
Mo	2	45.68 ± 17.02	2 ± 0	1 ± 0
	4	45.90 ± 12.23	2.2 ± 0.6	1.1 ± 0.3
	8	25.90 ± 15.01	2.6 ± 0.92	1.3 ± 0.46
	16	**24.32 ± 14.87**	3.6 ± 1.2	1.7 ± 0.46

9 http://orange.biolab.si/.





Rule	Active	IF color	AND bittern...	AND strength	THEN variety
1	✓	Pale	Low	Low	1
2	✓	Pale	Low	Medium	1
3	✓	Pale	High	Low	3
4	✓	Pale	High	Medium	3
5	✓	Pale	High	High	4
6	✓	Amber	Low	Low	2
7	✓	Amber	Low	Medium	2
8	✓	Amber	High	Medium	4
9	✓	Amber	High	High	6
10	✓	Dark	Low	Low	7
11	✓	Dark	Low	Medium	5
12	✓	Dark	High	High	8

Fig. 12. An example of FRBC generated from the BS dataset.

initial number that is equal to the classes of the problem at hand, the prototype number has been doubled at each subsequent run.

DC* is a method aimed at balancing accuracy and interpretability. Since they are conflicting objectives, it comes with no surprise that DC* often shows accuracy performances which do not match those produced by alternative classifiers that have been specifically designed to maximize accuracy only. Nevertheless, the table shows that in the majority of cases (related to the Ap, Bl, Bn, Hy, Ion, Ir, Mo, PB, Ph, Pi, Sh, and Wi datasets) the performance of DC* lies within the range determined by the accuracy values registered when the alternative methods are applied.

In the remaining cases, a higher price is paid for generating interpretable models with DC*, which exhibits a rough loss of 5%–10% in terms of accuracy.

Finally, we are interested in evaluating the interpretability/accuracy trade-off of the fuzzy models returned by DC*. To this aim, we need to compare these models with those identified by some specific alternative algorithm sharing a common set of features with DC*. Due to the particular nature of the proposed approach, such an inquiry is not straightforward. Among the variety of methods proposed in literature, we selected the Hierarchical Fuzzy Partitioning algorithm (HFP) [44], which appears to be one of the most suitable for our purpose. In fact, HFP is designed to derive fuzzy partitions from data, based on a clustering process followed by an iterative sequence of fuzzy partition merging, taking into account both accuracy and interpretability issues. However, HFP differs from DC* for a number of key points, notably because HFP provides a family of models (ranging from the simplest one to those more complex and accurate), leaving to the user the commitment for the final choice. Furthermore, HFP requires the set-up of a number of hyper-parameters (which have been left to default values in our experiments) and it requires the specification of a fixed number of fuzzy sets per features as well as a predetermined partitioning method.

HFP has been applied to the collection of datasets illustrated in Table 1, performing a new set of 10-fold cross validation sessions.[10] In practice, we exploited the HFP implementation provided by the FisPro tool [45]. The obtained results are reported in Tables 5–6 (where the performance of DC* is recalled for the sake of comparison).

Since HFP provides a family of models, it makes sense to select among them some specific instances: the models providing respectively the best accuracy results and the simplest structure (in terms of lower number of rules) stand as appropriate candidates. However, the latter had to be discarded being characterized by a

[10] We used the same folds adopted for the experiments described in Section 4.2.

Table 3
Experimental results obtained by applying DC* on the datasets pertaining to the remaining 11 out of 21 benchmark problems.

Dataset	#Prot.	%Error(± stDev)	#Rules(± stDev)	#Features(± stDev)
Nth	3	26.36 ± 12.82	3 ± 0	2 ± 0
	6	23.18 ± 6.88	3.1 ± 0.3	1.3 ± 0.46
	12	16.36 ± 8.67	3 ± 0	1.2 ± 0.4
	24	**14.09 ± 10.26**	3.2 ± 0.6	1.3 ± 0.64
PB	5	27.3 ± 8.49	5 ± 0	2.9 ± 0.3
	10	13.18 ± 2.96	5.6 ± 0.66	2.7 ± 0.46
	20	15.97 ± 4.91	5.8 ± 0.87	2.7 ± 0.46
	40	**10.47 ± 2.54**	8 ± 1.26	3.2 ± 0.4
Ph	2	41.09 ± 22.2	2 ± 0	1 ± 0
	4	53.31 ± 22.93	2.5 ± 0.81	1.2 ± 0.4
	8	38.71 ± 22.22	3.3 ± 1.1	1.7 ± 0.64
	16	**31.79 ± 16.33**	6.4 ± 2.24	2.7 ± 0.64
Pi	2	36.49 ± 3.87	2 ± 0	1 ± 0
	4	37.53 ± 5.24	2 ± 0	1 ± 0
	8	**32.21 ± 5.74**	2.2 ± 0.6	1.1 ± 0.3
	16	35.97 ± 10.21	3.2 ± 0.98	1.6 ± 0.49
Sh	2	43.4 ± 14.99	2 ± 0	1 ± 0
	4	42.55 ± 12.41	2 ± 0	1 ± 0
	8	**34.68 ± 4.86**	2.7 ± 0	1.4 ± 0.49
	16	37.02 ± 7.56	5.2 ± 1.17	2.5 ± 0.5
So	2	49.05 ± 12.24	2 ± 0	1 ± 0
	4	49.52 ± 9.80	2 ± 0	1 ± 0
	8	**37.14 ± 11.43**	2 ± 0	1 ± 0
	16	41.43 ± 9.54	3.7 ± 45.83	2 ± 0
Thy	3	46.78 ± 8.02	3 ± 0	2 ± 0
	6	35.89 ± 15.38	3.1 ± 0.3	1.9 ± 0.3
	12	17.83 ± 10.68	3.3 ± 0.46	2 ± 0
	24	**13.39 ± 8.72**	3.3 ± 0.46	1.8 ± 0.4
V2	2	34.19 ± 8.19	2 ± 0	1 ± 0
	4	30.65 ± 8.69	2 ± 0	1 ± 0
	8	**27.42 ± 10.02**	2 ± 0	1 ± 0
	16	31.61 ± 9.66	3.6 ± 1.11	1.9 ± 0.54
V3	3	55.81 ± 5.41	3 ± 0	2 ± 0
	6	41.29 ± 8.75	3.4 ± 0.49	2 ± 0
	12	35.16 ± 12.53	4.7 ± 1.62	2.3 ± 0.46
	24	**32.26 ± 6.92**	7.5 ± 1.2	3 ± 0.45
WBC	2	13.91 ± 5.03	2 ± 0	1 ± 0
	4	**12.61 ± 5.06**	2 ± 0	1 ± 0
	8	13.19 ± 5.78	2 ± 0	1 ± 0
	16	13.62 ± 4.16	2 ± 0	1 ± 0
Wi	3	46.11 ± 16.30	3 ± 0	2 ± 0
	6	40 ± 11.86	3.2 ± 0.4	2 ± 0
	12	27.22 ± 5.80	3.6 ± 0.49	2 ± 0
	24	**23.33 ± 11.06**	4.1 ± 0.7	2.1 ± 0.3

Table 4
Performance results (expressed in terms of the average error obtained at the end of 10-fold cross validation sessions) of five standard classifiers compared with DC*. The four DC*-related columns correspond to the different settings of the initial parameters adopted to run the 10-fold cross validation session for each dataset (in the way they have been described in Tables. 2-3).

Dataset	CN2	C4.5	NB	C-SVM	kNN	DC* × 1	DC* × 2	DC* × 4	DC* × 8
Ap	16.0	17.9	17.0	13.2	13.2	18.2	15.5	22.7	20.0
Bl	20.8	21.1	8.3	14.7	29.3	43.2	45.7	31.8	27.5
Bn	16.4	12.2	33.9	38.1	11.3	49.6	48.0	39.7	30.6
BS	9.0	4.5	7.5	2.5	2.5	29.0	21.0	12.5	9.5
Bu	37.1	36.2	34.8	34.5	31.0	47.1	46.6	47.1	40.9
CTG	9.4	6.9	20.0	12.3	9.1	45.5	20.6	22.0	29.6
Hy	20.0	21.9	14.4	18.1	54.4	57.5	59.4	54.4	38.8
Ion	32.5	7.4	11.4	5.7	15.7	41.7	38.9	25.0	28.3
Ir	10.7	4.0	10.0	3.3	4.7	31.3	22.7	12.0	8.0
Mo	5.5	3.8	37.1	25.3	36.9	45.7	45.9	25.9	24.3
Nth	7.9	8.8	5.1	4.7	6.5	26.4	23.2	16.4	14.1
PB	4.3	4.8	15.6	4.9	4.2	27.3	13.2	16.0	10.5
Ph	16.0	13.2	25.2	61.5	12.2	41.1	53.3	38.7	31.8
Pi	33.2	28.3	26.0	27.5	25.8	36.5	37.5	32.2	36.0
Sh	35.1	43.1	32.3	33.0	33.4	43.4	42.6	34.7	37.0
So	27.5	20.3	22.7	15.0	19.8	49.1	49.5	37.1	41.4
Thy	1.4	N/A	7.3	5.6	6.0	46.8	35.9	17.8	13.4
V2	20.0	20.6	23.5	16.1	21.9	34.2	30.7	27.4	31.6
V3	17.1	23.2	21.0	16.8	24.2	55.8	41.3	35.2	32.3
WBC	6.3	5.6	2.5	2.8	3.8	13.9	12.6	13.2	13.6
Wi	12.9	3.9	1.1	1.7	31.5	46.1	40.0	27.2	23.3

Table 5
Comparative evaluation of DC* and HFP (with reference to datasets pertaining to 10 out of 21 benchmark problems). The lowest error values and the highest A/I indices are highlighted in bold with the indication of the exact Wilcoxon signed rank test results: ($^{=}$) no statistically significant difference; (†) statistically significant difference for $p = 10\%$; (††) statistically significant difference for $p = 5\%$; (†††) statistically significant difference for $p = 1\%$.

Dataset	Method	err(%)	#features	#rules	A/I
Ap	DC*($n_P = 4$)	**15.5 ± 12.9**$^{=}$	1 ± 0	2 ± 0	**42.3**$^{=}$
	HFP-FR	17.3 ± 14.4	1 ± 0	2 ± 0	41.4
	HFP-BA	12.7 ± 10.1	4.4 ± 1.1	17.8 ± 11.0	4.9
Bl	DC*($n_P = 24$)	27.5 ± 8.8	3.4 ± 0.5	12.6 ± 3.3	6.0
	HFP-FR	**23.7 ± 6.4**$^{=}$	3.2 ± 0.4	12.8 ± 1.6	**6.1**$^{=}$
	HFP-BA	13.5 ± 4.1	4 ± 0	140.4 ± 10.8	0.6
Bn	DC*($n_P = 16$)	**30.6 ± 5.8**†††	2 ± 0	8 ± 1.2	8.9
	HFP-FR	48.2 ± 1.8	2 ± 0	8 ± 0	6.5
	HFP-BA	42.5 ± 1.5	1 ± 0	2 ± 0	**28.7**†††
BS	DC*($n_P = 64$)	**9.5 ± 6.3**†††	3.0 ± 0	11.8 ± 1.2	**7.8**†††
	HFP-FR	48.5 ± 3.2	3.0 ± 0	12.2 ± 0.6	4.2
	HFP-BA	48.5 ± 3.2	3.0 ± 0	11.8 ± 0.6	4.4
Bu	DC*($n_P = 16$)	**40.8 ± 5.1**$^{=}$	2.7 ± 0.6	7.0 ± 1.7	**9.0**$^{=}$
	HFP-FR	42.8 ± 5.5	3.3 ± 0.4	7.0 ± 0.8	8.3
	HFP-BA	42.6 ± 5.7	3.9 ± 1.0	9.7 ± 3.6	6.8
CTG	DC*($n_P = 6$)	20.6 ± 3.6	2 ± 0	4 ± 0	19.9
	HFP-FR	**19.6 ± 2.2**$^{=}$	2 ± 0	4 ± 0	**20.1**$^{=}$
	HFP-BA	17.4 ± 2.5	7.9 ± 1.9	120.8 ± 95.2	0.7
Hy	DC*($n_P = 24$)	**38.7 ± 18.2**$^{=}$	3.3 ± 0.4	14.9 ± 2.5	**4.1**$^{=}$
	HFP-FR	46.2 ± 14.3	2.5 ± 0.5	14.1 ± 2.3	4.0
	HFP-BA	48.1 ± 14.2	2.5 ± 0.5	14.2 ± 3.0	3.9
Ion	DC*($n_P = 8$)	**25.0 ± 9.5**††	1.1 ± 0.3	2.2 ± 0.6	**35.4**$^{=}$
	HFP-FR	30.0 ± 11.2	1 ± 0	2 ± 0	35.0
	HFP-BA	18.3 ± 11.2	5.2 ± 2.9	282.8 ± 606.5	0.3
Ir	DC*($n_P = 24$)	**8.0 ± 5.8**$^{=}$	1.5 ± 0.5	3.2 ± 0.6	**29.5**†††
	HFP-FR	12.7 ± 6.3	2 ± 0	4 ± 0	21.8
	HFP-BA	5.3 ± 7.2	3 ± 0	8.5 ± 0.8	11.1
Mo	DC*($n_P = 16$)	**24.3 ± 14.8**$^{=}$	1.7 ± 0.4	3.6 ± 1.2	**23.2**††
	HFP-FR	28.1 ± 10.5	2.0 ± 0	4.0 ± 0	18.0
	HFP-BA	9.7 ± 10.6	4.1 ± 0.8	62.4 ± 38.9	2.6

very poor classification performance. Therefore, we referred to the number of rules exhibited by the best DC* model for each dataset and, correspondingly, we considered the HFP model with an equal (or next to) number of rules. In this way, our experiments helped to identify a couple of HFP models for each dataset: HFP with best accuracy (HFP-BA) and HFP with a predetermined (fixed) number of rules (HFP-FR).[11]

In order to provide a quantitative assessing of the interpretability/accuracy trade-off, we introduced a simple trade-off index, namely the “A/I-index” reported in Tables 5–6. Such index is defined as the ratio between the accuracy (i.e. the complement of the error) of the model and the number of its rules, multiplied by 100: higher values of the A/I index correspond to better trade-off balances. Moreover, for assessing the capability to operate a suitable feature selection, which is a common property both for DC* and HFP, we also reported the total number of features involved in each derived model.

We integrated our investigation by computing, for each dataset, the exact Wilcoxon signed-rank (eWs-r) test [53] over both the

[11] It should be observed that the definition of the HFP-BA and HFP-FR models is accomplished on the basis of the 10-fold sessions performed on training sets. When the models are successively evaluated on test sets, the “fixed-rule” model may happen to outperform the one tagged as “best-accuracy”. We register twice such an infrequent occurrence, while investigating the Hy and So datasets.

Table 6
Comparative evaluation of DC* and HFP (with reference to datasets pertaining to the remaining 11 out of 21 benchmark problems). The lowest error values and the highest A/I indices are highlighted in bold with the indication of the exact Wilcoxon signed rank test results: ($^{=}$) no statistically significant difference; (†) statistically significant difference for $p = 10\%$; (††) statistically significant difference for $p = 5\%$; (†††) statistically significant difference for $p = 1\%$.

Dataset	Method	err(%)	#features	#rules	A/I
Nth	DC*($n_P = 24$)	$\mathbf{14.1 \pm 10.3}^{\dagger\dagger}$	1.3 ± 0.6	3.2 ± 0.6	$\mathbf{27.9}^{=}$
	HFP-FR	22.7 ± 8.4	2 ± 0	3.2 ± 0.4	24.6
	HFP-BA	17.3 ± 5.3	4 ± 0	7.8 ± 1.6	10.6
PB	DC*($n_P = 40$)	10.5 ± 2.5	3.2 ± 0.4	8 ± 1.26	$\mathbf{11.5}^{=}$
	HFP-FR	$\mathbf{10.0 \pm 1.3}^{=}$	3.8 ± 0.9	9 ± 1.41	10.2
	HFP-BA	7.7 ± 0.7	7 ± 1	37.8 ± 6.81	2.4
Ph	DC*($n_P = 16$)	31.7 ± 16.3	2.7 ± 0.6	6.4 ± 2.2	$\mathbf{11.7}^{\dagger\dagger}$
	HFP-FR	$\mathbf{30.4 \pm 3.4}^{=}$	2.9 ± 0.3	7.8 ± 0.6	8.9
	HFP-BA	26.1 ± 8.4	4.0 ± 1.0	51.6 ± 40.6	2.8
Pi	DC*($n_P = 8$)	$\mathbf{32.2 \pm 5.7}^{=}$	1.1 ± 0.3	2.2 ± 0.6	$\mathbf{32.1}^{\dagger\dagger\dagger}$
	HFP-FR	34.2 ± 4.6	2 ± 0	4 ± 0	16.5
	HFP-BA	28.1 ± 4.5	5.8 ± 1.17	39 ± 16.6	1.8
Sh	DC*($n_P = 8$)	34.6 ± 4.8	1.4 ± 0.4	2.7 ± 0	$\mathbf{26.2}^{\dagger\dagger\dagger}$
	HFP-FR	$\mathbf{33.1 \pm 3.0}^{=}$	2.0 ± 0	3.9 ± 0.3	17.3
	HFP-BA	31.2 ± 4.7	6.7 ± 0.6	47.3 ± 12.0	1.6
So	DC*($n_P = 8$)	37.1 ± 1.1	1.0 ± 0	2.0 ± 0	31.4
	HFP-FR	$\mathbf{35.7 \pm 0}^{=}$	1.0 ± 0	2.0 ± 0	$\mathbf{32.1}^{=}$
	HFP-BA	40.4 ± 5.3	8.3 ± 1.4	148.7 ± 100	0.6
Thy	DC*($n_P = 24$)	13.4 ± 8.7	1.8 ± 0.4	3.3 ± 0.46	$\mathbf{26.8}^{\dagger}$
	HFP-FR	$\mathbf{7.4 \pm 1.0}^{\dagger\dagger}$	2 ± 0	4 ± 0	23.2
	HFP-BA	7.3 ± 1.0	4.9 ± 2.0	22.6 ± 17.3	4.1
V2	DC*($n_P = 8$)	$\mathbf{27.4 \pm 10.0}^{\dagger\dagger}$	1 ± 0	2 ± 0	$\mathbf{36.3}^{\dagger}$
	HFP-FR	34.2 ± 10.6	1 ± 0	2 ± 0	32.9
	HFP-BA	26.8 ± 7.9	2.3 ± 0.6	5.5 ± 3.61	13.3
V3	DC*($n_P = 12$)	$\mathbf{32.3 \pm 6.9}^{\dagger\dagger\dagger}$	3 ± 0.5	7.5 ± 1.2	9.4
	HFP-FR	44.2 ± 8.9	2 ± 0	4 ± 0	$\mathbf{14.0}^{\dagger\dagger\dagger}$
	HFP-BA	44.2 ± 8.9	2.1 ± 0.3	4.6 ± 1.28	12.1
WBC	DC*($n_P = 4$)	$\mathbf{12.6 \pm 5.1}^{\dagger\dagger}$	1 ± 0	2 ± 0	$\mathbf{43.7}^{\dagger\dagger}$
	HFP-FR	16.2 ± 6.6	1 ± 0	2 ± 0	41.9
	HFP-BA	4.8 ± 2.1	4.9 ± 0.8	34.7 ± 18.0	2.7
Wi	DC*($n_P = 24$)	23.3 ± 11.1	2.1 ± 0.3	4.1 ± 0.7	19.1
	HFP-FR	$\mathbf{20.0 \pm 8.3}^{=}$	2 ± 0	4 ± 0	$\mathbf{20.0}^{=}$
	HFP-BA	7.2 ± 6.6	5.4 ± 1.1	39.6 ± 20.3	2.3

error percentages and the A/I index values for each fold. This kind of test is useful to assess the significance of the difference emerging when DC* and HFP are compared. More precisely, the eWs-r test allows to verify if it is possible to reject (with a prefixed confidence value p) the hypothesis that the reported performance measures are statistically equivalent: in that case, we are able to assert the supremacy of a model with a reduced risk of mistaking. For each dataset the significance test compares the DC* model with the HFP version (-BA or -FR) exhibiting the highest average A/I index. Three p-values have been used to appreciate the statistical differences (see the caption of Tables 5–6 for further details).

HFP-BA models generally outperform DC* models in terms of accuracy, but their employment is often impractical since the number of their rules is prohibitively high in most cases. Anyway, the discrepancy between the classification error values sometimes is quite reduced and in the majority of cases the number of features involved in the DC* models is lower than the corresponding value reported for the HFP-BA models. When turning to consider HFP-FR models, we focus on the comparison of models that are similar in terms of their overall structure, and the reported numbers of involved features are more consistent too (even if in some cases the feature selection process performed by DC* is still preferable).

At a first glance, the results reported in Tables 5–6 indicate that the DC* models exhibit mean values of the A/I index that are greater than those produced by HFP for 15 out of 21 datasets. However, the statistical analysis reported in Tables 5–6 is worthy of the following remarks:

- For 4 out of the 5 cases where the HFP-FR models appear to show a (slight) supremacy in terms of the A/I index value, the eWs-r test established that no statistically significant difference can be asserted among the involved evaluated values. Moreover, in the single case where the null difference hypothesis can be rejected (i.e., the one related to the V3 dataset), the DC* model is still able to exhibit a better performance in terms of classification error.
- Among the 15 cases where the DC* models appear to show a supremacy in terms of the A/I index value, the eWs-r test established the lack of statistically significant difference only in 6 experimental sessions (i.e., those related to the Ap, Bu, Hy, Ion, Nth, and PB datasets). For the remaining cases the null difference hypothesis can be rejected with a substantial level of confidence: p is even less than 1% for the BS, Sh, Ir, and Pi datasets.
- In a single case (related to the Bn dataset) the HFP-BA model proved to be the best in terms of A/I index among all the models. However, DC* was able to produce a much better accuracy performance for this dataset (while relying on a restrained number of fuzzy rules).
- As concerning accuracy, the eWs-r test confirms the primacy of DC* over HFP in 7 cases. No significant differences are observed when considering all the other datasets, with the exception of the Thy case where HFP shows higher accuracy than DC*. In this case, however, DC* produces a better interpretability-accuracy tradeoff. (It should be recalled that the statistical analysis focusing on accuracy involved DC* and the HFP version exhibiting the highest average A/I index.)

The analysis of the extensive experiments we have conducted lets us conclude that DC* outperforms HFP in terms of interpretability-accuracy tradeoff (as measured by the A/I index). At the same time DC*, far from penalizing accuracy, appears to be significantly preferable even on this side in several cases. All in all, we postulate that such a performance can be ascribed to the peculiar process performed by DC*, which allows to build up fuzzy information granules with variable granularity that are well-adapted to data, yet preserving the required interpretability constraints.

5. Concluding remarks

DC* is a method to extract interpretable fuzzy information granules from pre-classified numerical data. As an instance of DC*f*, it is based on two clustering stages: the first one is aimed at discovering multi-dimensional relationships among data, which are represented as classified prototypes that are projected onto each axis and clustered simultaneously in the second clustering stage. To perform the second task, which is intrinsically hard in complexity, a procedure based on A* has been adopted. A relevant effort of this investigation has been devoted to the definition of a heuristic function for A*, as well as to the convenient organization of data in a multi-dimensional priority queue.

The simultaneous clustering process performed in the second stage enables the partition of each feature in a variable number of fuzzy sets, with different granularity. At the limit (which is often encountered in experiments), a feature is partitioned in a single fuzzy set, i.e. the feature can be safely removed during the subsequent description of the derived fuzzy information granules. This involves an automatic feature selection process, which is particularly appreciable in knowledge discovery tasks.

The results of the second clustering stage is a collection of cuts, i.e. points in each axis that can be used to define trapezoidal fuzzy sets satisfying a number of interpretability constraints. This makes the linguistic tagging of such fuzzy sets very easy. As a consequence, the resulting information granules can be expressed in a simplified natural language, thus enabling an immediate understanding of the granulation process by non-technical users. This paves the way for a successful application of DC* in some real-world contexts like medical data analysis, educational data mining, etc.

The entire granulation process is controlled by a single hyper-parameter, namely the number of multi-dimensional prototypes to be defined in the first clustering process. This parameter directly translates into the desired granulation level of the available data. A higher number of information granules usually corresponds to a greater accuracy, though balanced by possible over-fitting. In general, however, a certain stability on the number of information granules has been observed during experiments, thus fine-tuning of this hyper-parameter is not required.

On the overall, DC* proposes a new way for data granulation, which exhibits some concrete advantages over alternative methods available in literature, and paves the way for further refinements and extensions which are matter of future investigations.

References

[1] L.A. Zadeh, Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, Fuzzy Sets and Systems 90 (1997) 111–127, http://dx.doi.org/10.1016/S0165-0114(97)00077-8.

[2] A. Bargiela, W. Pedrycz, Granular Computing: An Introduction, Kluwer Academic Publishers, Boston, Dordrecht, London, 2003, http://dx.doi.org/10.1007/978-1-4615-1033-8.

[3] T.Y. Lin, Granular computing on binary relations: Part I and II, in: L. Polkowski (Ed.), Rough Sets in Knowledge Discovery, Vol. 1, Physica-Verlag Heidelberg, 1998, pp. 286–318.

[4] W. Pedrycz, A. Bargiela, Granular clustering: a granular signature of data., IEEE Trans. Syst. Man Cybern. Part B Cybern 32 (2) (2002) 212–224, http://dx.doi.org/10.1109/3477.990878.

[5] Y.Y. Yao, Granular computing using neighborhood systems, in: Advances in Soft Computing: Engineering Design and Manufacturing, Springer-Verlag, London, 1999, pp. 539–553, http://dx.doi.org/10.1007/978-1-4471-0819-1_40.

[6] T. Saaty, M.S. Ozdemir, Why the magic number seven plus or minus two, Math. Comput. Modelling 38 (3–4) (2003) 233–244, http://dx.doi.org/10.1016/S0895-7177(03)90083-5.

[7] G. Castellano, A.M. Fanelli, C. Mencar, DCf: a double clustering framework for fuzzy information granulation, in: 2005 IEEE International Conference on Granular Computing, Vol. 2, IEEE, 2005, pp. 397–400, http://dx.doi.org/10.1109/GRC.2005.1547320.

[8] S.J. Russell, P. Norvig, Artificial Intelligence, A Modern Approach, second ed., in: Prentice Hall series in artificial intelligence, Prentice Hall, 2003, p. 874.

[9] M. Lucarelli, C. Mencar, C. Castiello, A.M. Fanelli, A new heuristic function for DC*, in: F. Masulli, G. Pasi, R. Yager (Eds.), Fuzzy Logic and Applications: 10th International Workshop, WILF 2013, Genoa, Italy, November 19-22, 2013. Proceedings, Springer International Publishing, 2013, pp. 44–51, http://dx.doi.org/10.1007/978-3-319-03200-9_5.

[10] M. Lucarelli, C. Castiello, A.M. Fanelli, C. Mencar, Interpretable knowledge discovery from data with DC*, in: J.M. Alonso, H. Bustince, M. Reformat (Eds.), Proceedings of the 2015 Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology, in: Advances in Intelligent Systems Research, vol. 89, Atlantis Press, Gijon, Spain, 2015, pp. 815–822, http://dx.doi.org/10.2991/ifsa-eusflat-15.2015.

[11] C. Mencar, A. Consiglio, G. Castellano, A.M. Fanelli, Improving the classification ability of DC* algorithm, in: F. Masulli, S. Mitra, G. Pasi (Eds.), Applications of Fuzzy Sets Theory (7th International Workshop on Fuzzy Logic and Applications, WILF 2007, Proceedings), Vol. 4578, Springer Berlin / Heidelberg, 2007, pp. 145–151, http://dx.doi.org/10.1007/978-3-540-73400-0_18.

[12] R.S. Michalski, A theory and methodology of inductive learning, Artificial Intelligence 20 (1983) 111–161, http://dx.doi.org/10.1016/0004-3702(83)90016-4.

[13] A. Bargiela, W. Pedrycz (Eds.), Human-Centric Information Processing Through Granular Modelling, in: Studies in Computational Intelligence, vol. 182, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, http://dx.doi.org/10.1007/978-3-540-92916-1.

[14] L.A. Zadeh, Is there a need for fuzzy logic?, Inform. Sci. 178 (13) (2008) 2751–2779, http://dx.doi.org/10.1016/j.ins.2008.02.012.

[15] J.M. Alonso, C. Castiello, C. Mencar, Interpretability of fuzzy systems: Current research trends and prospects, in: J. Kacprzyk, W. Pedrycz (Eds.), Springer Handbook of Computational Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 219–237, http://dx.doi.org/10.1007/978-3-662-43505-2_14.

[16] C. Mencar, C. Castiello, A.M. Fanelli, Interpretability assessment of fuzzy rule-based classifiers, in: V. Di Gesù, S.K. Pal, A. Petrosino (Eds.), Fuzzy Logic and Applications: 8th International Workshop, WILF 2009 Palermo, Italy, June 9-12, 2009 Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 155–162, http://dx.doi.org/10.1007/978-3-642-02282-1_20.

[17] M.J. Gacto, R. Alcala, F. Herrera, Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures, Inform. Sci. 181 (20) (2011) 4340–4360, http://dx.doi.org/10.1016/j.ins.2011.02.021.

[18] C. Mencar, C. Castiello, R. Cannone, A.M. Fanelli, Design of fuzzy rule-based classifiers with semantic cointension, Inform. Sci. 181 (20) (2011) 4361–4377, http://dx.doi.org/10.1016/j.ins.2011.02.014.

[19] C. Mencar, C. Castiello, R. Cannone, A.M. Fanelli, Interpretability assessment of fuzzy knowledge bases: A cointension based approach, Internat. J. Approx. Reason. 52 (4) (2011) 501–518, http://dx.doi.org/10.1016/j.ijar.2010.11.007.

[20] C. Mencar, A.M. Fanelli, Interpretability constraints for fuzzy information granulation, Inform. Sci. 178 (24) (2008) 4585–4618, http://dx.doi.org/10.1016/j.ins.2008.08.015.

[21] S. Zhou, J. Gan, Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling, Fuzzy Sets and Systems 159 (23) (2008) 3091–3131, http://dx.doi.org/10.1016/j.fss.2008.05.016.

[22] E. Lughofer, On-line assurance of interpretability criteria in evolving fuzzy systems - Achievements, new concepts and open issues, Inform. Sci. 251 (2013) 22–46, http://dx.doi.org/10.1016/j.ins.2013.07.002.

[23] O. Cordon, A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems : Designing interpretable genetic fuzzy systems, Internat. J. Approx. Reason. 52 (2011) 894–913, http://dx.doi.org/10.1016/j.ijar.2011.03.004.

[24] K. Cpalka, K. Lapa, A. Przybyl, M. Zalasinski, A new method for designing neuro-fuzzy systems for nonlinear modelling with interpretability aspects, Neurocomputing 135 (2014) 203–217, http://dx.doi.org/10.1016/j.neucom.2013.12.031.

[25] P.K. Shukla, S.P. Tripathi, A review on the interpretability-accuracy trade-off in evolutionary multi-objective fuzzy systems (EMOFS), Information 3 (3) (2012) 256–277.

[26] A.G. Evsukoff, S. Galichet, B.S. de Lima, N.F. Ebecken, Design of interpretable fuzzy rule-based classifiers using spectral analysis with structure and parameters optimization, Fuzzy Sets and Systems 160 (7) (2009) 857–881, http://dx.doi.org/10.1016/j.fss.2008.08.010.

[27] R. Belohlavek, M. Krupka, Grouping fuzzy sets by similarity, Inform. Sci. 179 (15) (2009) 2656–2661, http://dx.doi.org/10.1016/j.ins.2009.03.020.

[28] R.P. Paiva, A. Dourado, Interpretability and learning in neuro-fuzzy systems, Fuzzy Sets and Systems 147 (1) (2004) 17–38, http://dx.doi.org/10.1016/j.fss.2003.11.012.

[29] G. Leng, X.-J. Zeng, J.A. Keane, An improved approach of self-organising fuzzy neural network based on similarity measures, Evol. Syst. 3 (1) (2012) 19–30, http://dx.doi.org/10.1007/s12530-012-9045-6.

[30] E. Lughofer, J.-L. Bouchot, A. Shaker, On-line elimination of local redundancies in evolving fuzzy systems, Evol. Syst. 2 (2011) 165–187, http://dx.doi.org/10.1007/s12530-011-9032-3.

[31] M. Antonelli, P. Ducange, B. Lazzerini, F. Marcelloni, Learning concurrently partition granularities and rule bases of Mamdani fuzzy systems in a multi-objective evolutionary framework, Internat. J. Approx. Reason. 50 (7) (2009) 1066–1080, http://dx.doi.org/10.1016/j.ijar.2009.04.004.

[32] E.R.R. Kato, O. Morandin, M. Sgavioli, B.D. Muniz, Genetic tuning for improving wang and Mendel's fuzzy database, in: Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on, 2009, pp. 1015–1020. http://dx.doi.org/10.1109/ICSMC.2009.5346036.

[33] L. Stepnickova, M. Stepnicka, A. Dvorak, New results on redundancies of fuzzy/linguistic IF-THEN rules, in: Proceedings of the 8th Conference of the European Society for Fuzzy Logic and Technology, Atlantis Press, Paris, France, 2013, pp. 400–407, http://dx.doi.org/10.2991/eusflat.2013.62.

[34] M. Zeinalkhani, M. Eftekhari, Fuzzy partitioning of continuous attributes through discretization methods to construct fuzzy decision tree classifiers, Inform. Sci. 278 (2014) 715–735, http://dx.doi.org/10.1016/j.ins.2014.03.087.

[35] A.M. Acilar, A. Arslan, A novel approach for designing adaptive fuzzy classifiers based on the combination of an artificial immune network and a memetic algorithm, Inform. Sci. 264 (2014) 158–181, http://dx.doi.org/10.1016/j.ins.2013.12.023.

[36] A. Bouchachia, E. Lughofer, D. Sanchez, D. Wang, X.J. Zeng, J.A. Keane, A simplified structure evolving method for Mamdani fuzzy system identification and its application to high-dimensional problems, Inform. Sci. 220 (2013) 110–123, http://dx.doi.org/10.1016/j.ins.2011.12.033.

[37] F. Afsari, M. Eftekhari, E. Eslami, P.Y. Woo, Interpretability-based fuzzy decision tree classifier a hybrid of the subtractive clustering and the multi-objective evolutionary algorithm, Soft Comput. 17 (2013) 1673–1686, http://dx.doi.org/10.1007/s00500-013-0981-2.

[38] J. Abonyi, Supervised Fuzzy Clustering Based Initialization of Fuzzy Partitions for Decision Tree Induction, in: Soft Computing in Industrial Applications, in: Advances in Soft Computing, Vol. 75, Springer, 2010, pp. 31–39, http://dx.doi.org/10.1007/978-3-642-11282-9_4.

[39] A. Quteishat, C.P. Lim, K.S. Tan, A modified fuzzy min-max neural network with a genetic-algorithm-based rule extractor for pattern classification, IEEE Trans. Syst. Man Cybern. - Part A: Syst. Hum. 40 (3) (2010) 641–650, http://dx.doi.org/10.1109/TSMCA.2010.2043948.

[40] J. Casillas, Embedded genetic learning of highly interpretable fuzzy partitions, in: Proceedings of Joint 2009 International Fuzzy Systems Association World Congress, IFSA 2009 and 2009 European Society of Fuzzy Logic and Technology Conference, EUSFLAT 2009, Lisbon, Portugal, 2009, pp. 1631–1636.

[41] A. Fernández, M.J. del Jesus, F. Herrera, Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets, Internat. J. Approx. Reason. 50 (3) (2009) 561–577, http://dx.doi.org/10.1016/j.ijar.2008.11.004.

[42] M.J. Gacto, R. Alcala, F. Herrera, Integration of an index to preserve the semantic interpretability in the multiobjective evolutionary rule selection and tuning of linguistic fuzzy systems, IEEE Trans. Fuzzy Syst. 18 (3) (2010) 515–531, http://dx.doi.org/10.1109/TFUZZ.2010.2041008.

[43] C.H. Nguyen, W. Pedrycz, T.L. Duong, T.S. Tran, A genetic design of linguistic terms for fuzzy rule based classifiers, Internat. J. Approx. Reason. 54 (1) (2013) 1–21, http://dx.doi.org/10.1016/j.ijar.2012.07.007.

[44] S. Guillaume, B. Charnomordic, Generating an interpretable family of fuzzy partitions from data, IEEE Trans. Fuzzy Syst. 12 (3) (2004) 324–335, http://dx.doi.org/10.1109/TFUZZ.2004.825979.

[45] S. Guillaume, B. Charnomordic, Learning interpretable fuzzy inference systems with FisPro, Inform. Sci. 181 (20) (2011) 4409–4427, http://dx.doi.org/10.1016/j.ins.2011.03.025.

[46] J.M. Alonso, L. Magdalena, Generating understandable and accurate fuzzy rule-based systems in a java environment, in: A.M. Fanelli, W. Pedrycz, A. Petrosino (Eds.), Fuzzy Logic and Applications (Proc. of WILF 2011), in: LNAI, vol. 6857, Springer-Verlag Berlin Heidelberg, Trani, Bari (Italy), (ISSN: 0302-9743) 2011, pp. 212–219, http://dx.doi.org/10.1007/978-3-642-23713-3_27.

[47] J.M. Alonso, C. Castiello, M. Lucarelli, C. Mencar, Modeling interpretable fuzzy rule-based classifiers for Medical Decision Support, in: R. Magdalena, E. Soria, J. Guerrero, J. Gómez-Sanchis, A. Serrano (Eds.), Medical Applications of Intelligent Data Analysis: Research advancements, IGI Global, 2012, pp. 255–272, http://dx.doi.org/10.4018/978-1-4666-1803-9.ch017.

[48] T. Kohonen, Self-Organizing Maps, in: Information Sciences, vol. 30, Springer Verlag, 2001, http://dx.doi.org/10.1007/978-3-642-56927-2.

[49] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, T. Radauer, Integrating new classes on the fly in evolving fuzzy classifier designs and their application in visual inspection, Appl. Soft Comput. 35 (2015) 558–582.

[50] C. Mencar, M. Lucarelli, C. Castiello, A.M. Fanelli, Design of strong fuzzy partitions from cuts, in: Proceedings of the 8th Conference of the European Society for Fuzzy Logic and Technology, in: Advances in Intelligent Systems Research, Atlantis Press, Paris, France, 2013, pp. 424–431, http://dx.doi.org/10.2991/eusflat.2013.65.

[51] M. Lichman, UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml.

[52] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera, Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework, J. Mult.-Valued Logic Soft Comput. 17 (2–3) (2011) 255–287.

[53] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83, http://dx.doi.org/10.1037/0003-066X.54.8.594.