International Journal of Computational Intelligence Systems, Vol. 9, No. 1 (2016) 184-201

Hyperrectangles Selection for Monotonic Classification by Using Evolutionary Algorithms

Javier García [1] , Adnan M. AlBar,[2] Naif R. Aljohani,[2] José-Ramón Cano [1] , Salvador García [3]

[1] Department of Computer Science, University of Jaén,
EPS of Linares, Calle Alfonso X el Sabio S/N,
Linares, 23700, Spain

E-mails: jgf00002@red.ujaen.es, jrcano@ujaen.es

[2] Information Systems Department, Faculty of Computing and Information Technology,
King Abdulaziz University, Kingdom of Saudi Arabia

E-mails: ambar@kau.edu.sa, nraljohani@kau.edu.sa

[3] Department of Computer Science and Artificial Intelligence, University of Granada,
Calle Periodista Daniel Saucedo Aranda S/N,
Granada, 18071, Spain

E-mail: salvagl@decsai.ugr.es

Received 4 July 2015

Accepted 3 January 2016

Abstract

In supervised learning, some real problems require the response attribute to represent ordinal values that should increase with some of the explaining attributes. They are called classification problems with monotonicity constraints. Hyperrectangles can be viewed as storing objects in $\mathbb{R}^n$ which can be used to learn concepts combining instance-based classification with the axis-parallel rectangle mainly used in rule induction systems. This hybrid paradigm is known as nested generalized exemplar learning. In this paper, we propose the selection of the most effective hyperrectangles by means of evolutionary algorithms to tackle monotonic classification. The model proposed is compared through an exhaustive experimental analysis involving a large number of data sets coming from real classification and regression problems. The results reported show that our evolutionary proposal outperforms other instance-based and rule learning models, such as OLM, OSDL, *k*-NN and MID; in accuracy and mean absolute error, requiring a fewer number of hyperrectangles.

Keywords: Monotonic Classification, Nested Generalized Examples, Evolutionary Algorithms, Rule Induction, Instance-based Learning.

1. Introduction

The classification with monotonicity constraints, also known as monotonic classification [1], is an ordinal classification problem where a monotonic restriction is present: a higher value of an attribute in an example, fixing other values, should not decrease its class assignment. The monotonicity of relations between the dependent and explanatory variables is very usual as a prior knowledge form in data classification [2]. To illustrate, while considering a credit card application [3], a $1000 to $2000 income may be

considered a medium value of income in a data set. If a customer A has a medium income, a customer B has a low income (i.e. less than $1000) and the rest of input attributes remain the same, there is a relationship of partial order between A and B: $B < A$. Considering that the application estimates lending quantities as output class, it is quite obvious that the loan that the system should give to customer B cannot be greater than the given to customer A. If so, a monotonicity constraint is violated in the decision.

The estimation of the knowledge about monotonicity in learning models is of a great interest for two main arguments [4]. Firstly, monotonicity imposes constraints on the prediction function. This decreases the size of the hypothesis space and also the complexity of the model. Secondly, the domain experts decide the acceptance or rejection of the models yielded if they are consistent with the domain knowledge, regardless of their accuracy [5].

Many data learning algorithms have been adapted to be able to handle monotonicity constraints in several styles. There are two steps to treat with monotonic classification problems. The first one is to preprocess the data [6] in order to "monotonize" the data set [7], rejecting the examples that violate the monotonic restrictions or selecting features to improve classification performance and avoid overfitting [8,9]; and the second one is to force learning only monotone classification functions. Proposals of this type are: classification trees and rule induction [10,11,12,13], neural networks [14] and instance-based learning [15,16,17].

Instance-based learning was baptized as learning family in [18] and considers a set of methods widely used in machine learning [19]. An analogous scheme for instance-based learning is the Nested Generalized Exemplar (NGE) theory. It was announced in [20] and perform various major adjustments to the instance-based learning model. The most relevant is that it allows two type of examples, standard examples represented as single points and generalized examples which fill a hyper-volume in $\mathbb{R}^n$. They are closely connected to the nearest neighbor classifier (NN) [21], in such a way that they are intended to enhance it. NGE learning algorithms are being more popular in recent years due to the simplicity and effectiveness of the outcome they provide.

In this manner, the hyperrectangles are generalization of examples in $\mathbb{R}^n$, according to NGE theory. Single and generalized examples coexist and hyperrectangles may be nested and inner hyperrectangles serve as exceptions to surrounding hyperrectangles. They constitute axis-parallel rectangle representations as in many of the rule learning systems [22]. Using this model, a new example can be classified by estimating the Euclidean distance between it and every of the hyperrectangles stored. The class is predicted according to the label associated with the nearest hyperrectangle. In the case that two or more hyperrectangles cover the example, it is necessary to resolve a possible conflict derived from different labels among the hyperrectangles [20].

The profits of combining hyperrectangles with instances to build classification models are pointed out in the literature [23,24,25]. Looking at rule induction [22], the modeling of decision surfaces derived from combinations between parallel axis separators and Voronoi diagrams (typical decision surfaces of NN classifiers) may adapt the prediction to examples drawn along curves, improving the performance in complex domains. Regarding instance-based classification [18], the hyperrectangles adds interpretability to the model and reduces storage requirements.

The generation of an optimal minimal number of hyperrectangles for classifying a set of points is NP-hard. Heuristic algorithms produce a large but finite subset of hyperrectangles from the training data. Nevertheless, it may be easy that almost all hyperrectangles modeled could be unnecessary. Thus, there is a need for selecting the most influential ones, and it can be done by using data reduction schemes [6]. Evolutionary Algorithms (EAs) [26] have been used for data reduction with promising results [27]. They have been favorably used in NGE learning in the past [28,29].

In this paper, we propose the utilization of EAs for hyperrectangles' selection in monotonic classification tasks. Our goal is to increase the performance in this type of problem by means of selecting the best suitable set of hyperrectangles which optimizes the nearest hyperrectangle classification with monotonicty constraints. We compare our algorithm

with other monotonic learning models belonging to both instance-based and rule learning paradigms. They are OLM [15], MID [10], OSDL [16] and monotonic k-NN [17]. The experimental design incorporates non-parametrical statistical testing [30,31]. The results show a significant improvement in accuracy whereas the number of examples stored in the final subset is further reduced. Besided, the main key point of our proposal is that the model built is composed of generalized examples which all fulfill the monotonicity constraints.

The paper is organized as follows. Section 2 provides an background in monotonic classification and the NGE learning model. In Section 3, all topics concerning the approach proposed are described. In Section 4 the experimentation framework is given and in Section 5 the results and analysis are presented. In Section 6, the conclusions are highlighted.

2. Background

We will do a brief review of the monotonic classification including the description of the most used techniques for monotonic classification in Subsections 2.1 and 2.2. The NGE classification will be described in Subsection 2.3.

2.1. Monotonic Classification

Ordinal classification problems are those in which the class is neither numeric nor nominal. Instead, the class values are ordered. For instance, a worker can be described as "excellent", "good" or "bad", and a bond can be evaluated as "AAA", "AA", "A", "A-", etc. Similar to a numeric scale, an ordinal scale has an order, but it does not possess a precise concept of distance. Ordinal classification problems are important, since they are fairly common in our daily life. Employee selection and promotion, determining credit rating, bond rating, economic performance of countries, industries and firms, and insurance underwriting, are examples of ordinal problem-solving in business. Rating manuscripts, evaluating lecturers, student admissions, and scholarships decisions for students, are examples of ordinal decision-making in academic life. Ordinal problems have been investigated in scientific disciplines such as information retrieval, psychology, and statistics for many decades [4] .

A monotonic classifier is one that will not violate monotonicity constraints. Informally, the monotonic classification implies that the assigned class values are monotonically non-decreasing (in ordinal order) with the attribute values. More formally, let $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ denote a set of examples with attribute vector $\mathbf{x}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,m})$ and a class, $\text{class}(\mathbf{x}_i)$, being n the number of instances and m the number of attributes. Let $\mathbf{x}_i \succeq \mathbf{x}_h$ iff $\forall_{j=1,\ldots,m}, \mathbf{x}_{i,j} \geqslant \mathbf{x}_{h,j}$.

A data set $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ is monotonic if and only if all the pairs of examples i, h are monotonic with respect to each other [10] (see equation 1).

$$\mathbf{x}_i \succeq \mathbf{x}_h \implies \text{class}(\mathbf{x}_i) \geqslant \text{class}(\mathbf{x}_h), \forall_{i,h} \quad (1)$$

Some monotonic ordinal classifiers require monotonic data sets to successfully learn, although there are others that are capable of learning from non-monotonic data sets as well.

2.2. Monotonic Classification Methods

Four monotonic learning methods are pioneers and well-known in the field of monotonic classification. In the following, we will discuss each one in detail.

- The Ordinal Learning Model (OLM) [15] is a very simple algorithm that learns ordinal concepts by eliminating non-monotonic pairwise inconsistencies. The generated concepts can be viewed as rules. During the learning phase, each example is checked against every rule in a rule-base, which is initially empty. If an example is inconsistent with a rule in the rule-base, one of them is selected at random while the other is discarded, but if the example is selected, it must be checked for consistency against all the other monotonicity rules. If it passes this consistency test, it is added as a rule. Consequently, the rule-base is kept monotonic at all times. Classification is done conservatively. All the rules are checked in decreasing order of class values against an attribute vector, and the vector is classified as the class of the first

rule that covers it. If such a rule does not exist, the attribute vector is assigned the lowest possible class.

- As its name implies, Ordinal Stochastic Dominance Learner (OSDL) [16], is based on the concept of Ordinal Stochastic Dominance (OSD). The rationale behind OSD can be given through an example: In life insurance, one may expect a stochastically greater risk to the insurer from older and sicker applicants than from younger and healthier ones. Higher premiums should reflect greater risks and vice versa. There are several definitions of stochastic ordering. The stochastic order computes when a random variable is bigger than another. Considering this order, stochastic dominance can be established as a form of stochastic order. In this case, a probability distribution over possible predictions can be ranked. The ranking depends of the nature of the data set. Stochastic dominance refers to a set of relations that may hold between a pair of distributions.
  The most commonly used, as stochastic ordering, is first OSD, which was used by [36] in the OSDL. For each vector $\mathbf{x}_i$, the OSDL computes two mapping functions: one that is based on the examples that are stochastically dominated by $\mathbf{x}_i$ with the maximum label (of that subset), and the second is based on the examples that cover (i.e., dominate) $\mathbf{x}_i$, with the smallest label. Later, an interpolation between the two class values (based on their position) is returned as a class.
- The monotonic *k*NN (M*k*-NN) was proposed in [17]. This method consists of two steps. In the first step, the training data is made monotone by relabeling as few cases as possible. This relabeled data set may be considered as the monotone classifier with the smallest error index in the training data. In the second step, we use a modified nearest neighbour rule to predict the class labels of new data so that violations of the restrictions of monotonicity will not occur. Considering the monotonic nearest neighbor rule, the class label assigned to a new data point $\mathbf{x}_0$ must lie in the interval $[\text{class}_{min}, \text{class}_{max}]$, where

$$\text{class}_{min} = \max\{\text{class}(\mathbf{x}_i) | \{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\} \wedge \mathbf{x}_i \leqslant \mathbf{x}_0\} \tag{2}$$

and

$$\text{class}_{max} = \min\{\text{class}(\mathbf{x}_i) | \{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\} \wedge \mathbf{x}_0 \leqslant \mathbf{x}_i\} \tag{3}$$

where $\{\mathbf{x}_i, \text{class}(\mathbf{x}_i)\}$ is the monotone data set. To conserve the monotonicity the choice of the class value for $\mathbf{x}_0$ must be in this interval.

- A monotone extension of ID3 (MID) was proposed by Ben-David [10] using an additional impurity measure for splitting, the total ambiguity score. However, the resulting tree may not be monotone anymore even when starting from a monotone data set. MID defines the *total-ambiguity-score* as the sum of the entropy score of ID3 and the order-ambiguity-score. This last score is defined in terms of the non-monotonicity index of the tree, which computes the number of pair branches that are non-monotonic regarding the total possible non-monotonic pairs there may be.

2.3. NGE Learning

NGE is a learning paradigm based on class exemplars, where an induced hypothesis has the graphical shape of a set of hyperrectangles in $\mathbb{R}^n$. Exemplars of classes are either hyperrectangles or single instances [20]. The input of an NGE system is a set of training examples, each described as a vector of pairs $numeric\_attribute/value$ and an associated class. Attributes can either be numerical or categorical. Numerical attributes are usually normalized in the $[0,1]$ interval.

In NGE, an initial set of hyperrectangles in $\mathbb{R}^n$ formed by single points directly taken from the data is generalized into a smaller set of hyperrectangles regarding the elements that it contains. Choosing which hyperrectangle is generalized from a subset of points or other hyperrectangles and how it is generalized depends on the concrete NGE algorithm employed.

The matching process is one of the central features in NGE learning and it allows some customiza-

tion, if desired. Generally speaking, this process computes the distance between a new example and an exemplar memory object (a generalized example). Let the example to be classified be termed as $\mathbf{x}_0$ and the generalized example as G, regardless of whether G is formed by a single point or if it has some volume.

The model computes a match score between $\mathbf{x}_0$ and G by measuring the Euclidean distance between two objects. The Euclidean distance is well-known when G is a single point. Otherwise, the distance is computed as follows (considering numerical attributes):

$$D_{\mathbf{x}_0 G} = \sqrt{\sum_{j=1}^{m} \left(\frac{\text{rdif}_j}{\max_j - \min_j} \right)^2} \tag{4}$$

$$\text{rdif}_j = \begin{cases} \mathbf{x}_{0,j} - \max(G_j) & \text{when } \mathbf{x}_{0,j} > \max(G_j), \\ \min(G_j) - \mathbf{x}_{0,j} & \text{when } \mathbf{x}_{0,j} < \min(G_j), \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where m is the number of attributes of the data, $\mathbf{x}_{0,j}$ is the value of the jth attribute of the example $\mathbf{x}_0$, $\max(G_j)$ and $\min(G_j)$ are the maximum and minimum values of G for the jth attribute and $\max_j$ and $\min_j$ are the maximum and minimum values for jth attribute in training data, respectively.

The distance measure represents the length of a line dropped perpendicularly from the point $\mathbf{x}_0$ to the nearest surface, edge or corner of G. Note that internal points in a hyperrectangle have a distance of 0 to that rectangle. In the case of overlapping rectangles, several strategies could be implemented, but it is usually accepted that a point falling in the overlapping area belongs to the smaller rectangle (the size of a hyperrectangle is defined in terms of volume). The volume is computed following the indications given in [23]. In nominal attributes, the distance is 0 when two attributes have the same categorical label, and 1 on the contrary.

3. Evolutionary Selection of Hyperrectangles for Monotonic Clasification

The approach proposed in this paper, named Evolutionary Hyperrectangle Selection for Monotonic clasification by CHC (*EHSMC-CHC*), is fully explained in this section. First, we approximate the NGE theory to monotonic classification in Subsection 3.1 by providing some definitions. Then, we describe the process for generating the initial set of hyperrectangles in Subsection 3.2. After this, we introduce the CHC model used as an EA to perform hyperrectangle selection in Subsection 3.3. Finally, the specific issues regarding representation and fitness function are specified in Subsection 3.4.

3.1. Definitions

We represent each instance with $\mathbf{x}_i = (\mathbf{x}_{i,1}; \mathbf{x}_{i,2}; .....; \mathbf{x}_{i,m}; \text{class}(\mathbf{x}_i))$, where $\mathbf{x}_{i,j}$ is the input value of the instance i in the attribute j, m is the total number of input attributes and $\text{class}(\mathbf{x}_i)$ is the class or output value of the instance i. Let the hyperrectangles be denoted by $H : (\prod_j A_j; C)$, where A_j is a condition belonging to the antecedent and C is the response. A formal definition is given next:

$$\begin{aligned} &H : A_1 \times A_2 \times ..... \times A_m \Rightarrow C, \\ &\text{with} A_j = \begin{cases} [j_{\min}, j_{\max}] & \text{if numerical,} \\ \{\alpha, \beta, ...\} & \text{if nominal,} \end{cases} \\ &C = \text{a value of the class,} \end{aligned} \tag{6}$$

where $j_{\min}$ and $j_{\max}$ are the minimum and maximum boundaries for the condition j respectively, and $\alpha, \beta, \ldots$ are the possible categorical values belonging to the domain of the attribute j, if it is nominal.

The hyperrectangles should not break monotonicity in the sense we saw in Subsection 2.1. Hence, we define a partial order relation in the set of disjoint hyperrectangles, deciding when a precedent is higher than, lower than or equal to another:

$$\begin{array}{ll} A_j \preceq A'_j & \text{if attribute } j \text{ is numeric and } j_{\max} \leqslant j'_{\min} \\ & \text{or if attribute } j \text{ is nominal and } A_j \subseteq A'_j, \\ A_j \succeq A'_j & \text{if attribute } j \text{ is numeric and } j_{\max} \geqslant j'_{\min} \\ & \text{or if attribute } j \text{ is nominal and } A'_j \subseteq A_j, \\ A_j = A'_j & \text{if attribute } j \text{ is numeric and} \\ & j_{\min} = j'_{\min} \text{ and } j_{\max} = j'_{\max} \\ & \text{or if attribute } j \text{ is nominal and } A_j \equiv A'_j. \end{array} \tag{7}$$

Then let two hyperrectangles H and H' be defined as:

$$\begin{array}{lll} H \preceq H' & \text{if } A_j \preceq A'_j \forall j & \text{and } \exists l, 1 \leqslant l \leqslant m, A_l < A'_l, \\ H = H' & \text{if } A_j = A'_j \forall j, & \\ H \succeq H' & \text{if } A_j \succeq A'_j \forall j & \text{and } \exists l, 1 \leqslant l \leqslant m, A_l > A'_l. \end{array} \tag{8}$$

Two hyperrectangles are comparable if $H \preceq H'$ or $H \succeq H'$, and non-comparable in contrary case, since we cannot establish an order between them. Hence, an hyperrectangle $H : (\prod_j A_j, C)$ will be non-monotonic with respect to another $H' : (\prod_j A'_j, C')$ if:

$$\begin{array}{l} H \preceq H' \text{ and } C > C' \text{ or} \\ H \succeq H' \text{ and } C < C' \text{ or} \\ H = H' \text{ and } C \neq C'. \end{array} \tag{9}$$

The distance between an instance $\mathbf{x}_i$ and an hyperrectangle H is defined as follows:

$$D_{\mathbf{x}_i H} = \sqrt{\sum_{j=1}^{m} \left(\frac{\text{dis}_j}{\text{Range}} \right)^2}, \tag{10}$$

where dis_j and Range are defined differently depending on the type of the attribute. If the attribute is numeric:

$$\begin{array}{l} \text{Range} = \max_j - \min_j, \\ \text{dis}_j = \begin{cases} \mathbf{x}_{i,j} - j_{\max} \text{ if } \mathbf{x}_{i,j} > j_{\max}, \\ j_{\min} - \mathbf{x}_{i,j} \text{ if } \mathbf{x}_{i,j} < j_{\min}, \\ 0 \text{ otherwise.} \end{cases} \end{array} \tag{11}$$

If the attribute is nominal:

$$\begin{array}{l} \text{dis}_j = \begin{cases} 0 & \mathbf{x}_{i,j} \in A_j, \\ 1 & \text{otherwise,} \end{cases} \\ \text{Range} = \text{Num. of possible values of the attribute.} \end{array} \tag{12}$$

where $\mathbf{x}_{i,j}$ is the value of the j-th attribute of the instance i, $j_{\max}$ and $j_{\min}$ are the maximum and minimum values of the hyperrectangle H in the attribute j and $\max_i$ and $\min_i$ are the maximum and minimum values of the attribute j in the data set.

The distance between two hyperrectangles H and H' is defined as:

$$D_{HH'} = \sqrt{\sum_{j=1}^{m} \left(\frac{\text{dis}_j}{\text{Range}} \right)^2}, \tag{13}$$

where dis_j and Range are defined as:

$$\begin{array}{l} \text{dis}_j = \left| \frac{j_{\max} + j_{\min}}{2} - \frac{j'_{\max} + j'_{\min}}{2} \right|, \\ \text{Range} = \max_j - \min_j, \end{array} \tag{14}$$

if the attribute j is numeric and

$$\text{dis}_j = 1 - \frac{\#(A_j \bigcap A'_j)}{\#(A_j \bigcup A'_j)}, \tag{15}$$

if the attribute j is nominal.

3.2. Getting the Initial Set of Hyperrectangles

We start from a training set TR with n instances which consists of pairs $(\mathbf{x}_i, \text{class}(\mathbf{x}_i)), i = 1, ..., n$. Each one of the n instances has m input attributes.

In this first phase will get a hyperrectangle set HS with N hyperrectangles whose rule representation consists of pairs $(H_i, \text{class}(H_i)), i = 1, ..., N$, where H_i defines a set of conditions $(A_1, A_2, ..., A_m)$ and $\text{class}(H_i)$ defines the associated class label of the hyperrectangle. Each one of the N hyperrectangles has m conditions which can be numerical conditions, expressed in terms of minimum and maximum values in intervals $[0, 1]$; or they can be categorical conditions, by using a set of possible values $A_i = \{v_{1i}, v_{2i}, ..., v_{vi}\}$, assuming that it has vi different values. Note that we make no distinction between a hyperrectangle with volume and minimal hyperrectangles formed by isolated points.

In this first stage of our method, we have used a simple heuristic which is fast and yields acceptable results. The heuristic yields a hyperrectangle from each example in the training set. For each one, it finds the $k-1$ nearest neighbors being the $k-$th neighbor an example of a different class. Then, each hyperrectangle is expanded considering these $k-1$ neighbors by using, in the case of numerical attributes, the minimal and maximal values as the limits of the interval defined, or getting all the different categorical values, in the case of nominal attributes, to form a subset of possible values from them.

Once all the hyperrectangles are obtained, the duplicated ones are removed, keeping one representative in each case. Hence $|HS| \leqslant |TR|$. Note that point hyperrectangles are possible to be obtained using this heuristic when the nearest neighbor of an instance belongs to a different class.

3.3. CHC Model

As an evolutionary computation method, we have used the CHC model [32]. CHC is a classical evolutionary model that introduces important aspects to obtain a trade-off between exploration and exploitation; such as incest prevention, restoration of the search process when it becomes locked and the fight among parents and offspring into the replacement process.

During each generation the CHC realizes the following stages:

- *NC* children born after mating of *NC* individuals of the father population.
- Then, among the 2*NC* individuals formed by parents and children takes place a struggle for survival in which only remain *NC* individuals for the new generation.

CHC also implements a form of heterogeneous recombination using HUX, a special recombination operator [32]. HUX exchanges half of the bits that differ between parents, where the bit position to be exchanged is randomly determined. CHC also employs a method of incest prevention. Before applying HUX to the two parents, the Hamming distance between them is measured. Only those parents who differ from each other by some number of bits (mating threshold) are mated. The initial threshold is set at $L/4$, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold is reduced by one.

CHC also performs a form of heterogeneous HUX recombination using a special operator recombination [32]. HUX exchanged half the bits that differ between parents, where the bit position to exchange is determined randomly. CHC also employs a method of preventing incest. Before applying HUX to the both parents, the Hamming distance between them is measured. Only parents who are distinguished by some number of bits (mating threshold) are coupled. The initial threshold is stable at L / 4, where L is the length of the chromosomes. If no offspring are inserted into the new population then the threshold it decrements by one.

No mutation is applied during the recombination phase. Instead, when the population converges or the search stops making progress (i.e., the difference threshold has dropped to zero and no new offspring are being generated which are better than any member of the parent population) the population is reinitialized to introduce new diversity to the search. The chromosome representing the best solution found over the course of the search is used as a template to reseed the population. Reseeding of the population is accomplished by randomly changing 35% of the bits in the template chromosome to form each of the other $NC-1$ new chromosomes in the population. The search is then resumed.

3.4. Representation and Fitness Function

Let $S \subseteq HS$ be the subset of selected hyperrectangles that result from the run of a hyperrectangle selection algorithm. Hyperrectangle selection can be considered as a search problem to which EAs can be applied. We take into account two important issues: the specification of the representation of the solutions and the definition of the fitness function.

- *Representation:* We use a binary representation, a chromosome consists of N genes (one from each hyperrectangle in HS) with two possible values: 0 and 1. If the gene is 1, the associated hyperrectan-

gle is included in the subset represented by S. If this is 0, this is not included.

- *Fitness Function:* Let S be a subset of hyperrectangles of HS and be coded by a chromosome. We define a fitness function based on the accuracy (classification rate) and the Non-Monotonic Index (NMI) evaluated over TR through the formula.

$$Fitness(S) = (1-\lambda)\cdot(\beta\cdot(\alpha\cdot clas\_rat + (1-\alpha)\cdot perc\_red) + (1-\beta)\cdot cover) + \lambda\cdot NMI\_red. \quad (16)$$

$clas\_rat$ means the percentage of correctly classified objects from TR using S. $perc\_red$ is defined as

$$perc\_red = 100\cdot\frac{|HS|-|S|}{|HS|}. \quad (17)$$

$cover$ refers the total coverage of examples in TR in the subset of selected hyperrectangles or, in other words, the number of examples of TR whose distance value has been equal to 0 (examples covered by hyperrectangles in S).
NMI_red is defined as

$$NMI\_red = 50 * \frac{NMI\_initial - NMI\_current}{NMI\_initial} \quad (18)$$

where $NMI\_initial$ is the number of monotonic violations computed by the predictions made for all the rules derived from the training set and divided by the total number of pairs of examples. $NMI\_current$ is similarly computed to $NMI\_initial$, but instead using the rules selected by the chromosome. NMI_red represents the relative reduction of NMI achieved by each chromosome and it is weighted by a constant factor of 50 due to the fact that this measure represents low values with respect to classification and reduction rates.

The objective of the EAs is to maximize the fitness function defined, i.e., maximize the classification rate and coverage while minimizing the number of hyperrectangles selected and the anti-monotonic index. Although the fitness function defined is focused on discarding single trivial hyperrectangles (points), exceptions could be present in special cases where points are necessary to achieve high rates of accuracy. Thus, the necessity of using single hyperrectangles or not will be determined by the tradeoff accuracy-coverage and will be conditioned by the problem tackled.

Regarding the parameters, we preserved the value of $\alpha = 0.5$ as the best choice, due to the fact that it was analyzed in previous works related to instance selection [33,34,35]. We have conducted several experimental evaluations to estimate the best values of these parameters. For the sake of shortness, we determined that a suitable value for β should fall near 0.66 and the value for λ should be close to 0.25.

It is worth mentioning that our objective is to identify the best values of the parameters that configure the evolutionary approach in a general set up. It is true that for a specific classification problem, these values could be tuned in order to optimize the results achieved, but this may affect to other aspects, like efficiency or simplicity. General rules can be given about this topic:

- The number of evaluations and population size are the main factors for yielding good results in accuracy and simplicity. The raise of these values has a negative effect on efficiency. In larger problems, it may be necessary to increase both values, but we will show in the experimental study that values of 10,000 and 50 work appropriately, respectively.
- Parameters α and β allow us to obtain a desired trade-off between the accuracy and the number of hyperrectangles created. In the case of obtaining poor accuracy rates in a specific problem we have to increase α or decrease β. In contrary case, when the rules yielded are numerous and we are interested in producing simpler models, we have to increase β or decrease α. Besides, a large value of λ penalizes the maximization of the accuracy and minimization of the number of hyperrectangles in favor of the loss of anti-monotonic index.

Regarding to the specification of the classification conflicts, we employ the same mechanisms as shown in [20]. In short, they are:

- If no hyperrectangle covers the example, the class

of the nearest hyperrectangle defines the prediction.

- If various hyperrectangles cover the example, the one with lowest volume is the chosen to predict the class, allowing exceptions within generalizations.
  Our approach computes the volume of a hyperrectangle in the following way:

$$V_H = \prod_{j}^{m} L_j, \tag{19}$$

where L_j is computed for each condition as

$$L_j = \begin{cases} H_{\text{upper}} - H_{\text{lower}} \text{ if numeric and } H_{\text{upper}} \neq H_{\text{lower}}, \\ 1 \text{ if numeric and } H_{\text{upper}} = H_{\text{lower}}, \\ \frac{\text{num. values selected}}{vi} \text{ if nominal.} \end{cases} \tag{20}$$

4. Experimental Framework

In this section, we present the experimental framework developed to analyze and compare our proposal EHSMC-CHC with other well-known algorithms presented in this domain. The study will incorporate two types of data set. On the one hand, we will involve real life data sets coming from standard classification problems whose classes are not initially ordered. For our purposes, we instigate the sorting of the class values, assigning each category a number. The order among classes is made according to the arrangement that achieves the fewest number of monotonicity violations reported in each data set. On the other hand, we will use regression data sets whose output attribute has a numeric domain which will be ordered into categorical values. In this second subset, the order of classes is natural and we tackle real ordinal prediction problems.

In the next subsection, the experimental methodology will be specified, including data sets, performance measures, parameters of the algorithms and statistical validation.

4.1. Experimental methodology

The elements included in the framework are the following:

- *Data sets*: The study includes a total of 30 data sets, as a result of the joint selection of standard classification data sets (CLAS) whose class attribute is transformed into an ordinal attribute, assigning each category a number in such a way that the number of pairs of non-monotonic examples is minimized, and regression data sets (REGR) whose class attribute is discretized into 4 or 10 categorical values, keeping the class distribution balanced. Also, four classical ordinal classification data sets (ORD) are included in this study (*era*, *esl*, *lev* and *swd*) [1]. Their main characteristics are described in Table 1. They are classical data sets used in the classification scope and extracted from the UCI and KEEL repositories [37,38]. The run of the algorithms has been done following a 10-fold cross validation procedure (10-fcv).
  Table 1 shows the name of the data sets and their number of instances, variables and classes. In case of containing missing values, the total number of instances of the original data set appears in parenthesis. In this paper, these instances have been ignored.
- *Evaluation metrics*:
  - Accuracy (Acc), defined as the number of successful hits relative to the total number of classifications. It has been by far the most commonly used metric for assessing the performance of classifiers for years [39].
  - Mean Absolute Error (MAE), calculated as the sum of the absolute values of the errors and then dividing it by the number of classifications. The errors between the real label and the predicted label are estimated by the difference between the ordinal class values. Various studies conclude that MAE is one of the best performance metrics in ordered classification [40].
  - Monotonic Accuracy (MAcc), computed as standard Acc, but only considering those examples that completely fulfill the monotonicity constraints. In other words, non-monotonic

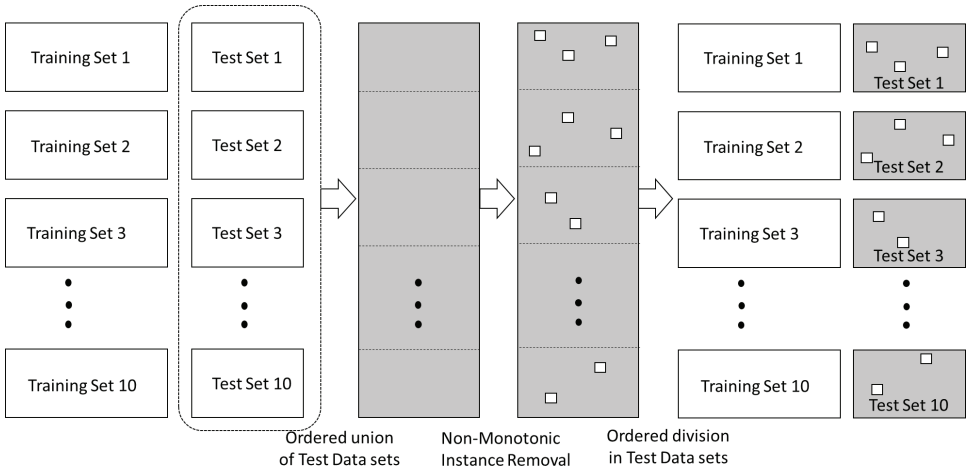

Figure 1: Process of transformation the 10-fcv to the new one by removing the non-monotonic instances from the test data sets.

comparable examples do not take part in the calculation of MAcc. The sense behind this is to simulate that future examples to be classified will check the monotonicity assumption. This metric serves as a manner of monotonicity level measurement of the predictions carried out. To address this, the process followed is presented in Figure 1 where the partitions used in 10-fcv are modified conserving the training data sets but removing the non-monotonic instances in the test data sets. We must point out that in the new validation data sets the test instances belong to the original data set, they are real instances, not artificial. In the same way for this new data sets obtained, we must highlight that the test data sets have been generated using different strategies than the noise removal filter or relabeling, searching to fair comparisons. The process presented in Figure 1 does not present any kind of randomness due to it is based in a deterministic greedy algorithm (see Algorithm 1).

- Monotonic Mean Absolute Error (MMAE), used to pursue the same goal of MAcc, but using MAE instead of Acc. Obviously, the data sets used in this case are the same than in the MAcc analysis.
- Non-Monotonicity Index (NMI) [41], defined as the number of clash-pairs divided by the total number of pairs of examples in the predictions made by an algorithm:

$$NMI = \frac{1}{n(n-1)} \sum_{x \in D} NClash(x), \qquad (21)$$

where x is an example from the data set D. $NClash(x)$ is the number of examples from D that do not meet the monotonicity restrictions (or clash) with x and n is the number of instances in D.

- Number of Rules (NRules), number of rules/hyperrectangles that compose the models produced by the algorithms.

- *Parameters configuration*: See Table 2.
- *Statistical procedures*: Several hypothesis testing procedures are considered to determine the most relevant differences found among the methods [30]. The use of non-parametric tests will be preferred to parametric ones, since the initial conditions that guarantee the reliability of the latter may not be satisfied. This could cause the statistical analysis to lose credibility. Friedman ranks test [31] is used to contrast the behavior of each algorithm. It highlights the significant differences between methods if they appear.

Table 1: Description of the 30 data sets used in the study.

Data set	Ins.	At.	Cl.	Type	Data set	Ins.	At.	Cl.	Type
appendicitis	106	7	2	CLAS	**hepatitis**	80 (155)	19	2	CLAS
bands	365 (539)	19	2	CLAS	**ionosphere**	351	33	2	CLAS
baseball10cl	337	16	10	REG	**iris**	150	4	3	CLAS
baseball4cl	337	16	4	REG	**machinecpu10cl**	209	6	10	REG
breast	277 (286)	9	2	CLAS	**machinecpu4cl**	209	6	4	REG
cleveland	297 (303)	13	5	CLAS	**movement_libras**	360	90	15	CLAS
dee10cl	365	6	10	REG	**newthyroid**	215	5	3	CLAS
dee4cl	365	6	4	REG	**pima**	768	5	2	CLAS
dermatology	358 (366)	34	6	CLAS	**sonar**	208	60	2	CLAS
ecoli	336	7	8	CLAS	**spectfheart**	267	4	2	CLAS
era	1000	4	9	ORD	**swd**	1000	10	4	ORD
esl	488	4	9	ORD	**vowel**	990	13	11	CLAS
german	1000	20	2	CLAS	**wdbc**	569	30	2	CLAS
glass	214	9	7	CLAS	**wine**	178	13	2	CLAS
haberman	306	3	2	CLAS	**wisconsin**	683 (699)	9	2	CLAS

Table 2: Parameters considered for the algorithms compared.

Algorithm	Parameters
Mk-NN	$k = \{1, 3\}$, distance = euclidean
OLM	modeResolution = conservative modeClassification = conservative
OSDL	classificationType = media, balanced = No weighted = No, tuneInterpolationParameter = No, lowerBound = 0, upperBound = 1 interpolationParameter = 0.5, interpolationStepSize = 10
MID	confidence = 0.25, 2 items per leaf, R = 1
EHSMC-CHC	Popul. Size = 50, Num. Evaluations = 10000 $\alpha = 0.5$, $\beta = 0.66$, $\lambda = 0.25$

Table 3: Results for Acc and MAE

	Acc						MAE					
	EHSMC-CHC	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**	**EHSMC-CHC**	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**
appendicitis	**0.8673**	0.8236	0.8018	0.7736	0.1855	0.1591	**0.1327**	0.1764	0.1982	0.2264	0.8145	0.8409
bands	0.6658	0.6194	0.3702	0.6219	**0.6859**	0.6250	0.3342	0.3806	0.6298	0.3781	**0.3141**	0.3750
baseball10cl	**0.2796**	0.2493	0.1988	0.2643	0.2494	0.2314	**1.3814**	1.4658	2.0048	1.5855	1.7870	1.8635
baseball4cl	**0.6647**	0.5850	0.5289	0.4983	0.4748	0.4961	**0.3678**	0.4802	0.5869	0.6028	0.6822	0.6611
breast	**0.7146**	0.6749	0.6817	0.5447	0.6272	0.6310	**0.2854**	0.3251	0.3183	0.4553	0.3728	0.3690
cleveland	0.5389	0.5517	**0.5657**	0.5655	0.5254	0.5253	0.9424	0.6931	0.8477	**0.6471**	0.7436	0.7774
dee10cl	**0.2875**	0.2633	0.1781	0.0985	0.2384	0.2766	1.3971	**1.2493**	2.6059	4.5396	1.8289	1.7619
dee4cl	**0.6493**	0.6053	0.4682	0.2522	0.5454	0.5809	**0.4085**	0.4221	0.7838	1.4927	0.5887	0.5259
dermatology	**0.9354**	0.9239	0.3800	0.1259	0.7898	0.8094	**0.1435**	0.1691	1.5331	1.7599	0.4625	0.4259
ecoli	**0.8126**	0.8007	0.5715	0.0324	0.5923	0.5893	**0.7016**	0.7738	1.5379	2.3482	1.4875	1.5829
era	0.1620	**0.2740**	0.1690	0.2320	0.1430	0.1410	2.1480	1.3630	2.1500	**1.2850**	2.3360	2.3400
esl	0.5353	0.6784	0.5963	**0.6785**	0.3956	0.3915	0.7750	0.3525	0.4324	**0.3521**	0.7418	0.7459
german	0.6970	0.6920	**0.7110**	0.3740	0.6370	0.6330	0.3030	0.3080	**0.2890**	0.6260	0.3630	0.3670
glass	0.6975	0.6200	0.3244	0.3379	**0.7172**	0.6783	0.5602	0.7595	1.7816	1.7729	**0.5408**	0.6460
haberman	**0.7514**	0.7120	0.3496	0.7158	0.4803	0.4771	**0.2486**	0.2880	0.6504	0.2842	0.5197	0.5229
hepatitis	**0.8343**	0.7735	0.2497	0.8118	0.8222	0.8030	**0.1657**	0.2265	0.7503	0.1882	0.1778	0.1970
ionosphere	**0.9174**	0.9002	0.6269	0.7407	0.7071	0.6702	**0.0826**	0.0998	0.3731	0.2593	0.2929	0.3298
iris	0.9467	**0.9533**	0.9333	0.3667	**0.9533**	**0.9533**	0.0533	**0.0467**	0.0667	0.9067	**0.0467**	**0.0467**
machinecpu10cl	0.2864	0.3298	0.3205	**0.3493**	0.3105	0.2962	1.6021	**1.2550**	1.4960	1.2631	1.4176	1.4938
machinecpu4cl	0.6457	0.6267	0.6219	0.5736	0.6267	**0.6505**	0.4352	**0.3924**	0.4452	0.4843	0.4117	0.3929
movement_libras	**0.7222**	0.6667	0.3000	0.0611	0.6139	0.5778	**1.2444**	1.5667	4.8306	7.0333	1.7528	2.0944
newthyroid	**0.9491**	0.9264	0.5636	0.1439	0.7957	0.7723	**0.0833**	0.1104	0.7662	0.8654	0.3165	0.3578
pima	0.6787	0.7265	0.7110	0.6563	**0.7358**	0.7320	0.3213	0.2735	0.2890	0.3437	**0.2642**	0.2680
sonar	0.7312	0.7460	0.4662	0.5419	**0.8555**	0.8307	0.2688	0.2540	0.5338	0.4581	**0.1445**	0.1693
spectfheart	**0.7942**	0.7608	0.1983	0.7830	0.7236	0.7050	**0.2058**	0.2392	0.8017	0.2170	0.2764	0.2950
swd	0.3930	0.5660	0.4040	**0.5820**	0.3360	0.3360	0.6630	0.4670	0.7510	**0.4350**	0.8810	0.8810
vowel	0.7838	0.7990	0.0909	0.0859	**0.9949**	0.9788	0.4313	0.5586	5.0000	4.8273	**0.0101**	0.0444
wdbc	**0.9364**	0.9350	0.3392	0.3568	0.3465	0.3341	**0.0636**	0.0650	0.6608	0.6432	0.6535	0.6659
wine	**0.9431**	0.9160	0.3042	0.3261	0.7542	0.8147	**0.0569**	0.0840	0.9536	0.9487	0.3180	0.2408
wisconsin	0.8230	0.9460	0.8872	0.9593	**0.9622**	**0.9622**	0.1770	0.0540	0.1128	0.0407	**0.0378**	**0.0378**
Average	0.6881	**0.6882**	0.4637	0.4485	0.5942	0.5887	0.5328	**0.4966**	1.1393	1.2423	0.6861	0.7107

Table 4: Results for MAcc and MMAE

	MAcc						MMAE					
	EHSMC-CHC	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**	**EHSMC-CHC**	**MID**	**OLM**	**OSDL**	**M1-NN**	**M-3NN**
appendicitis	**0.9431**	0.9111	0.8018	0.7736	0.1500	0.1167	**0.0569**	0.0889	0.1982	0.2264	0.2111	0.8500
bands	0.6663	0.6175	0.3724	0.6253	**0.6896**	0.6281	**0.3337**	0.3825	0.6276	0.3747	0.3635	0.3104
baseball10cl	0.2920	0.2700	**0.5783**	0.5350	0.2841	0.2647	**1.3342**	1.3383	0.5048	0.5411	1.8795	1.5814
baseball4cl	**0.7119**	0.6155	0.2237	0.3072	0.5209	0.5464	**0.3123**	0.4414	1.9201	1.3864	0.5367	0.6134
breast	**0.7902**	0.7534	0.7769	0.5710	0.6958	0.6996	**0.2098**	0.2466	0.2231	0.4290	0.2309	0.3042
cleveland	**0.5682**	0.5517	0.5575	0.5673	0.5536	0.5533	0.8920	0.6931	0.8448	**0.6583**	0.6963	0.6811
dee10cl	0.3204	0.2960	**0.5069**	0.2439	0.2776	0.3237	1.3093	1.1540	**0.7144**	1.5358	1.3429	1.4843
dee4cl	**0.6807**	0.6350	0.2085	0.0991	0.5898	0.6270	**0.3728**	0.3829	2.5892	4.7203	0.3818	0.5036
dermatology	**0.9370**	0.9313	0.4037	0.1334	0.8357	0.8489	**0.1360**	0.1583	1.5298	1.7417	0.3153	0.3405
ecoli	**0.8521**	0.8366	0.5653	0.0235	0.6696	0.6566	**0.4925**	0.6243	1.5889	2.3213	1.0180	1.0977
era	0.1626	**0.7239**	0.2837	0.5134	0.1811	0.1836	1.9900	**0.4726**	1.7797	0.5651	2.5840	2.0704
esl	0.7429	**0.8340**	0.6758	0.8119	0.4476	0.4399	0.2752	**0.1861**	0.3318	0.2035	1.6437	0.6796
german	0.7035	0.6973	**0.7260**	0.3815	0.6496	0.6465	0.2965	0.3027	**0.2740**	0.6185	0.3556	0.3504
glass	0.6983	0.6200	0.3227	0.3411	**0.7221**	0.6832	0.5580	0.7595	1.7892	1.7667	0.6372	**0.5307**
haberman	**0.9460**	0.8009	0.2777	0.7157	0.5857	0.5814	**0.0540**	0.1991	0.7223	0.2843	0.8754	0.4143
hepatitis	**0.8343**	0.7735	0.2497	0.8118	0.8222	0.8030	**0.1657**	0.2265	0.7503	0.1882	0.2000	0.1778
ionosphere	**0.9290**	0.9111	0.6576	0.7703	0.7416	0.7027	**0.0710**	0.0889	0.3424	0.2297	0.2212	0.2584
iris	0.9595	0.9590	0.9267	0.3667	**0.9662**	**0.9662**	0.0405	0.0410	0.0733	0.9133	**0.0271**	0.0338
machinecpu10cl	0.3302	0.3907	**0.6707**	0.6252	0.3831	0.3619	1.4441	1.0995	**0.3969**	0.4158	1.2460	1.2553
machinecpu4cl	0.6660	0.6640	0.3719	0.4290	0.6815	**0.7026**	0.4119	0.3570	1.3555	1.0573	**0.3382**	0.3496
movement_libras	**0.7226**	0.6715	0.3145	0.0643	0.6434	0.6049	**1.2474**	1.5329	4.7228	6.9883	1.5169	1.4765
newthyroid	**0.9561**	0.9366	0.5721	0.1504	0.8310	0.8065	**0.0687**	0.0924	0.7486	0.8591	0.3393	0.2562
pima	0.6787	0.7587	0.7005	0.6563	**0.7894**	0.7863	0.3213	0.2413	0.2995	0.3437	**0.2062**	0.2106
sonar	0.7457	0.7460	0.4662	0.5419	**0.8555**	0.8307	0.2543	0.2540	0.5338	0.4581	**0.1445**	0.1693
spectfheart	**0.7917**	0.7575	0.2008	**0.7917**	0.7315	0.7110	**0.2083**	0.2425	0.7992	**0.2083**	0.2880	0.2685
swd	0.4101	0.8485	0.4682	**0.8878**	0.3876	0.3876	0.6151	0.1561	0.6393	**0.1122**	1.0399	0.7730
vowel	0.7838	0.7990	0.0909	0.0859	**0.9949**	0.9788	0.4313	0.5586	5.0000	4.8273	0.0444	**0.0101**
wdbc	**0.9364**	0.9315	0.3073	0.2970	0.4626	0.4495	**0.0636**	0.0685	0.6927	0.7030	0.6184	0.5374
wine	**0.9462**	0.9229	0.3258	0.3258	0.7947	0.8595	**0.0538**	0.0771	0.9324	0.9490	0.2169	0.2581
wisconsin	0.8230	0.9529	0.8641	0.9571	**0.9708**	**0.9708**	0.1770	0.0471	0.1359	0.0429	**0.0280**	0.0292
Average	0.7176	**0.7373**	0.4823	0.4801	0.6303	0.6240	0.4732	**0.4171**	1.1020	1.1890	0.6516	0.5959

Table 5: Results for NMI and MRules

	NMI(%)						NRules		
	EHSMC-CHC	**MID**	**OLM**	**OSDL**	**M1-NN**	**M3-NN**	**EHSMC-CHC**	**MID**	**OLM**
appendicitis	5.44	**0.00**	0.22	**0.00**	14.34	15.37	9.8	**8.7**	23.3
bands	**0.00**	**0.00**	**0.00**	0.01	0.01	0.01	**49**	60	102.9
baseball10cl	**1.13**	10.68	12.58	12.45	14.09	13.98	**33.7**	93.2	215.7
baseball4cl	**0.00**	11.14	13.48	13.36	12.13	12.26	**40.2**	61.2	123.4
breast	**0.16**	0.64	3.48	2.35	3.78	3.78	25.5	142.9	**18.4**
cleveland	**0.00**	0.40	0.89	0.98	1.01	0.89	**13.3**	60.1	35.7
dee10cl	0.84	2.82	**0.02**	4.40	4.99	4.95	**27.5**	92.5	94.5
dee4cl	0.56	2.41	**0.02**	4.86	4.26	4.13	35.7	55.1	**27.5**
dermatology	0.02	**0.00**	0.05	0.19	0.16	0.16	13.3	**13.2**	36.9
ecoli	0.17	11.31	**0.02**	12.72	13.40	13.17	**33.4**	37.7	75.8
era	**8.79**	17.03	13.49	12.59	12.25	12.27	**9.4**	26.9	38.2
esl	**0.44**	61.70	61.92	61.56	63.28	63.30	**24.4**	55.4	39.6
german	**0.00**	0.02	0.06	0.05	0.05	0.06	**38.6**	288.8	288.8
glass	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**28.6**	33.4	191
haberman	3.46	**0.29**	0.56	3.80	19.72	19.74	25.4	**8.9**	25.2
hepatitis	**0.00**	**0.00**	0.10	0.10	0.40	**0.00**	**5.3**	9.8	17.3
ionosphere	0.09	**0.00**	0.29	0.23	0.45	0.40	**24.1**	24.5	116.8
iris	**0.00**	23.00	3.89	20.97	23.05	23.05	6.1	**4.4**	5.5
machinecpu10cl	**0.87**	5.18	4.88	4.92	6.78	6.93	**17.9**	53.3	58.9
machinecpu4cl	**0.20**	3.95	6.24	6.19	5.05	4.89	21.3	35.6	**20.4**
movement_libras	0.06	0.15	**0.02**	0.57	0.51	0.51	**44.1**	55.2	51.2
newthyroid	0.20	**0.05**	0.09	3.32	1.76	2.16	**8**	11.5	33.3
pima	**0.01**	3.47	0.12	3.67	5.50	5.52	**37.9**	52.7	39.3
sonar	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	28.8	**22.3**	187.2
spectfheart	**0.00**	**0.00**	**0.00**	**0.00**	0.03	0.03	**22.7**	27.6	239.5
swd	**1.50**	10.73	10.11	9.04	9.29	9.29	**4.1**	57.8	41.6
vowel	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	142.6	**104.1**	891
wdbc	0.01	**0.00**	**0.00**	1.37	3.35	3.34	17	**16.7**	64.3
wine	**0.00**	**0.00**	0.01	0.01	0.33	0.26	9.4	**7.9**	17.9
wisconsin	**0.00**	36.83	37.66	32.46	37.55	37.85	**6.2**	22.8	9.7
Average	**0.80**	6.73	5.67	7.07	8.58	8.61	**26.78**	51.47	104.36

Algorithm 1 Greedy Non-Monotonic Instances Removal Algorithm for test partitions.

```
function GREEDYREMOVE(T - dataset)
    while NumberOfTotalCollisions(T)>0 do
        maxColis=0, instancSelec=0;
        for each instance i in T do
            Colis=NumberOfCollisionsProduced(i,T);
            if Colis>maxColis then
                maxColis=Colis, instancSelec=i;
            end if
        end for
        T = T - instancSelec;
    end while
    return T
end function
```

5. Results and Analysis

This section shows the results obtained in the experimental study as well as the analysis based on them. Tables 3 and 4 report the results measured by accuracy, MAE and their monotonic versions explained before (MAcc and MMAE) in test data. Furthermore, Table 5 shows the NMI measured in the prediction for each algorithm and the number of rules reported for those algorithms which produce rules: MID and OLM. The best case in each data set is stressed in bold. The last row in each table shows the average measured by considering all data sets.

At a glance, observing Tables from 3 to 5, we can make the following analyses:

- The EHSMC-CHC proposal yields the second best average result in accuracy over test data, the first best average results is offered by MID. It is slightly more accurate than Mk-NN and much more accurate than OLM and OSDL.
- The MID proposal gets the best average result in MAE over test data. EHSMC-CHC obtains a very close result to MID, but it is clearly better than OLM and OSDL.
- By considering the monotonic measures, MAcc and MMAE, EHSMC-CHC is again slightly outperformed by MID, obtaining the second best results. Also, both are clearly superior to the rest of algorithms.
- The EHSMC-CHC proposal obtains the best average result in NMI over test data. The differences in NMI are very clear, thus our approach obtains very good monotonic predictions.
- Finally, the number of rules required by EHSMC-CHC is the lowest regarding the three algorithms compared. Hence, the models produces are simpler than the ones produced by MID and OLM.

In summary, our proposal EHSMC-CHC achieves competitive results in terms of accuracy and MAE by considering both classical and monotonic measures. It requires few rules and yields more monotonic models. Clearly, EHSMC-CHC is as good as MID in precision performance, but it requires a much lower number of generalized instances for building the model. With respect to the monotonicity in predictions, the NMI associated with EHSMC-CHC is also better than the reported by MID. Hence, EHSMC-CHC requires less rules and makes predictions with a higher monotonic index with similar behavior to MID.

Table 6 collects the results of applying the Wilcoxon test to EHSMC-CHC and the rest of the methods studied in this paper. In each one of the cells, three symbols can appear: +, = or –. They represent either that EHSMC-CHC outperforms (+), is similar to (=) or is worse (–) in performance than the rest of the methods. The value in brackets is the p-value obtained in the comparison. Other statistical

Table 6: Wilcoxon p-values reported comparing with EHSMC-CHC

Algorithm	Acc	MAE	MAcc	MMAE	NMI	NRules
MID	=(0.2580)	=(0.7577)	=(0.1642)	=(0.6071)	+(0.0034)	+(0.0009)
OLM	+(0.0001)	+(0.0002)	+(0.0001)	+(0.0020)	+(0.0258)	+(0.0000)
OSDL	+(0.0005)	+(0.0040)	+(0.0017)	+(0.0086)	+(0.0000)	—
M1-NN	+(0.0052)	+(0.0080)	+(0.0248)	+(0.0081)	+(0.0000)	—
M3-NN	+(0.0015)	+(0.0035)	+(0.0081)	+(0.0137)	+(0.0000)	—

Table 7: Rankings reported by the test of Friedman

Algorithm	Acc	MAE	MAcc	MMAE	NMI	NRules
EHSMC-CHC	**2.3**	**2.4333**	**2.3833**	**2.45**	**1.8833**	**1.4333**
MID	2.6667	2.5	2.9333	2.8667	2.6	2.05
OLM	4.5	4.7667	4.35	4.4333	3.2333	2.5167
OSDL	4.1333	4.1	4.3333	4.35	3.7833	—
M1-NN	3.4333	3.2833	3.3	3.4667	4.8167	—
M3-NN	3.9667	3.9167	3.7	3.4333	4.6833	—

studies can be performed by using non-parametric multiple comparison tests. These types of procedures study the set of results obtained by all the algorithms to compute a ranking that takes into account the multiple comparison. The smaller the ranking, the better the algorithm is. Rankings of the Friedman test are depicted in Table 7.

The results of the statistical tests allow us to highlight the following observations:

- EHSMC-CHC and MID are the best approaches compared with the other techniques regarding accuracy and MAE. No statistical differences are registered between both, thus they behave equally in performance.
- Our algorithm significantly outperforms OLM, OSDL, M1-NN and M3-NN in Acc, MAE, MAcc and MMAE.
- Considering NMI and NRules, our approach is the best according to the statistical report.

6. Conclusions

The purpose of this paper was to present a proposal of hyperrectangle selection for monotonic classification through evolutionary algorithms. The novel algorithm named *EHSMC-CHC* constitutes a novel approach for nested generalized exemplar learning in classification with monotonicity constrains. It builds a beginning set of hyperrectangles by using a heuristic on training data and then it carries out a selection process focused on maximizing the performance of several objectives: accuracy, coverage of examples and reduction of the monotonicity violations registered in the model learned with the lowest possible number of hyperrectangles.

The results showed that *EHSMC-CHC* will allow us to produce very accurate models with a low number of hyperrectangles with very few monotonicity ruptures. We have compared it with the most important monotonic learning approaches, including both rule and instance based learners, and the performance of the models obtained by *EHSMC-CHC* in several data sets overcomes the offered by them.

Acknowledgments

This work is supported by the National Research Project TIN2014-57251-P.

References

1. A. Ben-David, L. Serling and Y. Pao, "Learning and classification of monotonic ordinal concepts," *Computational Intelligence*, **5**, 45–49 (1989).
2. W. Kotłowski and R. Słowiński, "On nonparametric ordinal classification with monotonicity constraints," *IEEE Transactions on Knowledge and Data Engineering*, **25:11**, 2576–2589 (2013).
3. C. -C. Chen and S. -T. Li, "Credit rating with a monotonicity-constrained support vector machine model," *Expert Systems with Applications*, **41:16**, 7235–7247 (2014).
4. A. Ben-David, L. Sterling and T. Tran, "Adding monotonicity to learning algorithms may impair their accuracy," *Expert Systems with Applications*, **36:3**, 6627-6634 (2009).
5. A. Gegov, N. Gobalakrishnan and D. Sanders, "Filtration of Non-Monotonic Rules for Fuzzy Rule Base Compression," *International Journal of Computational Intelligence Systems*, **7:2**, 382-400 (2014).
6. S. García, J. Luengo and F. Herrera, "Data Preprocessing in Data Mining," Springer, 2015.
7. R. Potharst, A. Ben-David and M. C. van Wezel, "Two algorithms for generating structured and unstructured monotone ordinal datasets," *Engineering Applications of Artificial Intelligence*, **22:4-5**, 491-496 (2009).
8. Q. Hu, W. Pan, L. Zhang, D. Zhang, Y. Song, M. Guo and D. Yu, "Feature Selection for Monotonic Classification," *IEEE Transactions on Fuzzy Systems*, **20:1**, 69-81 (2012).
9. W. Pan, Q. Hu, Y. Song and D. Yu, "Feature selection for monotonic classification via maximizing monotonic dependency," *International Journal of Computational Intelligence Systems*, **7:3**, 543-555 (2014).
10. A. Ben-David, "Monotonicity maintenance in information-theoretic machine learning algorithms," *Machine Learning*, **19:1**, 29-43 (1995).
11. R. Potharst and A. J. Feelders, "Classification trees for problems with monotonicity constraints," *SIGKDD Explorations*, **4:1**, 1-10 (2002).
12. K. Dembczyński, W. Kotłowski, and R. Słowiński, "Learning rule ensembles for ordinal classification with monotonicity constraints," *Fundamenta Informaticae*, **94:2**, 163-178 (2009).
13. Q. Hu, X. Che, Z. Lei, D. Zhang, M. Guo and D. Yu, "Rank entropy-based decision trees for monotonic classification," *IEEE Transactions on Knowledge Data Engineering*, **24:11**, 2052-2064 (2012).
14. H. Daniels and M. Velikova, "Monotone and partially monotone neural networks," *IEEE Transactions on Neural Networks*, **21:6**, 906-917 (2010).
15. A. Ben-David, "Automatic generation of symbolic multiattribute ordinal knowledge-based DSSs: methodology and applications," *Decision Sciences*, **23**, 1357-1372 (1992).
16. S. Lievens, B. De Baets and K. Cao-Van, "A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting," *Annals of Operational Research*, **163:1**, 115-142 (2008).
17. W. Duivesteijn and A. Feelders, "Nearest neighbour classification with monotonicity constraints," In *ECML/PKDD* , **1**, 301-316 (2008).
18. D. W. Aha, D. Kibler and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, **6:1**, 37-66 (1991).
19. P. Flach, "Machine Learning: The Art and Science of Algorithms that Make Sense of Data," Cambridge University Press, 2012.
20. S. Salzberg, "A nearest hyperrectangle method," *Machine Learning*, **6:1**, 151-276 (1991).
21. T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, **13:1**, 21-27 (1967).
22. J. Fürnkranz, D. Gamberger and N. Lavrač, "Foundations of Rule Learning," Springer, 2012.
23. D. Wettschereck and T. G. Dietterich, "An experimental comparison of the nearest neighbor and nearest-hyperrectangle algorithms," *Machine Learning*, **19**, 5-27 (1995).
24. P. Domingos, "Unifying instance-based and rule-based induction," *Machine Learning*, **24**, 141-168 (1996).
25. O. Luaces and A. Bahamonde, "Inflating examples to obtain rules," *International Journal of Intelligent Systems*, **18:11**, 1113-1142 (2003).
26. D. Simon, "Evolutionary Optimization Algorithms," Wiley, 2013.
27. A. A. Freitas, "Data Mining and Knowledge Discovery with Evolutionary Algorithms," Springer, 2002.
28. S. García, J. Derrac, J. Luengo, C. J. Carmona and F. Herrera, "Evolutionary Selection of Hyperrectangles in Nested Generalized Exemplar Learning," *Applied Soft Computing*, **11:3**, 3032-3045 (2011).
29. S. García, J. Derrac, I. Triguero, C. J. Carmona and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowledge-Based Systems*, **25**, 3-12 (2012).
30. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, **7**, 1-30 (2006).
31. S. García, A. Fernández, J. Luengo and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, **180**, 2044-2064 (2010).
32. L. J. Eshelman, "The CHC adaptive search algorithm: How to safe search when enganging in nontraditional genetic recombination, " in G. Rawlings (Ed.), "Foundations of Genetic Algorithms". Morgan Kaufmann,

1991.
33. J. -R. Cano, F. Herrera and M. Lozano, "Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study," *IEEE Transactions on Evolutionary Computation*, **7:6**, 561-575 (2003).
34. J. Derrac, S. García and F. Herrera, "A survey on evolutionary instance selection and generation," *International Journal of Applied Metaheuristic Computing*, **1:1**, 60-92 (2010).
35. J. Derrac, C. Cornelis, S. García and F. Herrera, "Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection," *Information Sciences*, **186:1**, 73-92 (2012).
36. K. Cao-Van, "Supervised Ranking, from semantics to algorithms," *Ph.D. dissertation*, Ghent University (2003).
37. M. Lichman, "UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]". Irvine, CA: University of California, School of Information and Computer Science. (2013).
38. J. Alcalá-Fdez, A. Fernández, 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," *Journal of Multiple-Valued Logic and Soft Computing*, **17:2-3**, 255–287 (2011).
39. R. C. Prati, G. Batista, M. Monard, "A survey on graphical methods for classification predictive performance evaluation", *IEEE Transactions on Knowledge and Data Engineering*, **23:11**, 1601–1608 (2011).
40. N. Japkowicz, M. Shah, "Evaluating Learning Algorithms. A classification Perspective", Cambridge University Press, 2014.
41. H. Daniels, M. Velikova, "Derivation of monotone decision models from noisy data", *IEEE Transactions on Systems, Man and Cybernetics - Part C*, **36**, 705–710 (2006).