International Journal of Approximate Reasoning 85 (2017) 79–96

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Rough cognitive ensembles

Gonzalo Nápoles [a,b,*], Rafael Falcon [c], Elpiniki Papageorgiou [d,a], Rafael Bello [b], Koen Vanhoof [a]

[a] Faculty of Business Economics, Hasselt Universiteit, Belgium
[b] Department of Computer Science, Universidad Central de Las Villas, Cuba
[c] School of Electrical Engineering and Computer Science, University of Ottawa, Canada
[d] Department of Computer Engineering, Technological Education Institute of Central Greece, Greece

ARTICLE INFO

Article history:
Received 18 April 2016
Received in revised form 14 March 2017
Accepted 21 March 2017
Available online 24 March 2017

Keywords:
Machine learning
Granular computing
Rough set theory
Fuzzy cognitive maps
Rough cognitive networks
Ensemble learning

ABSTRACT

Rough Cognitive Networks are granular classifiers stemming from the hybridization of Fuzzy Cognitive Maps and Rough Set Theory. Such cognitive neural networks attempt to quantify the impact of rough granular constructs (i.e., the positive, negative and boundary regions of a target concept) over each decision class for the problem at hand. In rough classifiers, determining the precise granularity level is crucial to compute high prediction rates. Regrettably, learning the similarity threshold parameter requires reconstructing the information granules, which may be time-consuming. In this paper, we put forth a new multiclassifier system classifier named *Rough Cognitive Ensembles*. The proposed ensemble employs a collection of Rough Cognitive Networks as base classifiers, each operating at a different granularity level. This allows suppressing the requirement of learning a similarity threshold. We evaluate the granular ensemble with 140 traditional classification datasets using different heterogeneous distance functions. After comparing the proposed model to 15 well-known classifiers, the experimental evidence confirms that our scheme yields very promising classification rates.

1. Introduction

Pattern classification [1] is one of the most ubiquitous real-world problems and certainly one at which humans really excel. It consists of identifying the right category (among those in a predefined set) to which an observed pattern belongs. These patterns are often described by a set of predictive attributes of numerical and/or nominal nature. Formally speaking, the pattern classification problem [1] is about building a mapping $f : \mathcal{U} \rightarrow \mathcal{D}$ that assigns to each instance $x \in \mathcal{U}$ described by the attribute set $\Psi = \{\psi_1 \ldots, \psi_M\}$ a decision class D_k from the K possible ones in $\mathcal{D} = \{D_1, \ldots, D_K\}$. The mapping is learned in a *supervised* fashion, i.e., by relying on an existing set of previously labeled examples that is used to train the classification model. The learning process is regularly driven by the minimization of a cost/error metric. More complex classification problems include learning in presence of class imbalance [2,3], class noise [4], partially labeled data [5] or multiple decision classes per object [6–8], among other scenarios.

The literature on classification models (henceforth simply called "*classifiers*") is vast and offers a myriad of techniques that approach the classification problem from multiple angles. Decision trees [9], neural networks [10], rule-based models

* Corresponding author.
E-mail address: gonzalo.napoles@uhasselt.be (G. Nápoles).

http://dx.doi.org/10.1016/j.ijar.2017.03.011
0888-613X/

[11], Bayesian networks [12], support vector machines [13] or *k*-nearest neighbors learners [14] stand among the most popular classifiers, each having its own inherent advantages and limitations.

A well-established trend within Machine Learning (ML) is that of combining the output of multiple base classifiers to predict the class label of a given instance. These models are called *multiclassifier systems* [15] and often perform better than any single classifier, especially if some diversity (either parametric or structural) is present among the set of base classifiers. This heterogeneity may come in the form of independent samples of the training data (bootstrapping), multiple parametric configurations, different types of base classifiers used as building blocks, etc. *Bayesian voting*, *bagging*, *boosting* and *stacking* are common ensemble creation methods [16]. AdaBoost [17] and random forests [18] are well-known ensemble learners that have shown great promise in solving pattern classification problems.

A recent survey [16] explores trends in this field, including: (a) using multiobjective optimization algorithms to derive several structural elements of the ensembles; (b) employing decomposition techniques; (c) resorting to negative correlation learning schemes; (d) incorporating elements from fuzzy logic and multiple-kernel learning and (e) endowing the models with deep learning features. Other reviews that focus on either a set of techniques or a particular domain can be found in the literature as well. For example, Zhao et al. [19] revised existing neural network ensembles whereas Yang et al. [20] related ensemble approaches used in bioinformatics.

Another ongoing ML avenue has led to the development of *granular classifiers* [21–26]. These are classification schemes that abide by the principles of *Granular Computing* [27–29], an emerging paradigm that relies on more symbolic constructs such as sets, intervals or similarity classes to replace numerical data as the underlying knowledge representation and reasoning vehicle. Granular classifiers offer important advantages over traditional ML techniques such as (1) a more human-centric manner of interacting with real-world information and its inherent uncertainty and (2) the ability to deal with the Big Data phenomenon by processing the deluge of raw data at higher levels of abstraction.

Recently, Nápoles and collaborators [30] introduced the notion of *rough cognitive mapping* in the pattern recognition context. Rough Cognitive Networks (RCNs) are granular classifiers featuring a synergy between Fuzzy Cognitive Maps (FCMs) [31] and Rough Set Theory (RST) [32]. Being more explicit, an RCN can be defined as a sigmoid FCM whose input concepts denote information granules (namely, the RST-derived positive, boundary and negative regions of the decision classes) whereas output concepts represent decision classes of the problem at hand. This granular model seems especially suitable to handle inconsistent data given the approximation space constructed over each output concept (decision class) by means of the three RST-based regions induced by a similarity relation. It is worth mentioning that two objects are considered inconsistent if their similarity degree exceeds a certain threshold, but they lead to different decisions.

RCN's classification performance has been found competitive with respect to state-of-the-art classifiers [33,34,30]. Despite these encouraging results, RCN-based models are still very sensitive to an input parameter denoting the similarity threshold upon which the rough information granules are built. Determining the exact granularity degree is not trivial, since higher values do not necessarily lead to optimal prediction rates. To overcome this drawback, Nápoles et al. [30] proposed a hyperparameter learning method to estimate the similarity threshold value from historical data. Nevertheless, evaluating a candidate solution implies reconstructing the granular regions from scratch, which is time-consuming in large datasets.

In this paper, we build upon the RCNs formalism by developing a *granular ensemble* model in order to deal with the RCN's parametric sensitivity. Rough Cognitive Ensembles (RCEs) use a collection of RCNs as base classifiers, each operating at a different granularity level. It is worth mentioning that the concept of *granular ensembles* seems to still be in its infancy though. A good starting point is the use of granular classifiers as base classifiers, given the momentum they are presently enjoying. The model proposed in this paper follows this idea, and it may well be one of the first granular ensemble techniques ever put forth. As a result, the hyperparameter learning method is no longer required, which notably increases the practical usability of this novel granular classifier in real-world applications.

The second contribution of this paper is oriented to the exhaustive evaluation of rough cognitive classifiers in presence of 140 ML datasets. As far as we know, RCNs have been applied to a few real-life problems, but there is no record of their performance in solving standard ML problems. This evaluation includes studying the algorithm's performance for several heterogeneous distance functions. After comparing the ensemble model to 15 state-of-the-art ML algorithms, the experimental evidence suggests that the proposed classifier produces highly competitive prediction rates.

The rest of this paper is structured as follows. Section 2 goes over important concepts that are related to this study such as *rough sets*, *three-way decision rules* and *fuzzy cognitive maps*. Section 3 emphasizes on the building blocks of the granular ensemble, Section 4 presents the foundations of the proposed ensemble classifier, while Section 5 describes three widely used heterogeneous distance functions. Moreover, Section 6 introduces the empirical simulations and its ensuing discussion. Section 7 formalizes the concluding remarks and future research directions to be explored.

2. Theoretical background

This section sheds light on several important concepts to this study, namely classical and extended rough sets, foundations of three-way decision rules, cognitive mapping and granular cognitive maps.

2.1. Rough set theory

RST is a methodology proposed in the early 1980s for handling uncertainty that is manifested in the form of inconsistent data [32,35,36]. Let $DS = (\mathcal{U}, \Psi \cup \{d\})$ denote a *decision system* where $\mathcal{U}$ is a non-empty finite set of objects called the universe of discourse, Ψ is a non-empty finite set of attributes describing any object in U and $d \notin \Psi$ represents the decision attribute. Any subset $X \in \mathcal{U}$ can be approximated by two crisp sets, which are referred to as its *lower* and *upper approximations* and denoted by $\underline{\Phi}X = \{x \in \mathcal{U} \mid [x]_\Phi \in X\}$ and $\overline{\Phi}X = \{x \in \mathcal{U} \mid [x]_\Phi \cap X \neq \emptyset\}$, respectively. In this classic formulation, the equivalence class $[x]_\Phi$ comprises the set of objects in $\mathcal{U}$ that are deemed inseparable from x according to the information contained in the attribute subset $\Phi \subseteq \Psi$.

The lower and upper approximations are the basis for computing the positive, negative and boundary regions of any set X. The *positive region* $POS(X) = \underline{\Phi}X$ includes those objects that are certainly contained in X; the *negative region* $NEG(X) = \mathcal{U} - \overline{\Phi}X$ denotes those objects that are certainly not related to X, while the *boundary region* $BND(X) = \overline{\Phi}X - \underline{\Phi}X$ captures the objects whose membership to the set X is uncertain, i.e., they might be members of X. These three rough regions comprise *information granules* upon which the RCN leans to map its set of input-type neurons and produce conclusions, as explained later in Section 3.

2.1.1. Using similarity relations in RST

In the classical RST formulation, two objects are considered inseparable w.r.t. $\Phi \subseteq \Psi$ if they have identical values for all attributes. This definition is adequate for nominal attributes but is too rigid when dealing with numerical ones, given that marginal differences between two numerical values could toss two nearly identical objects into different inseparability classes. To counter this stringent definition, the equivalence requirement on the inseparability relation R is relaxed by resorting to *similarity relations*.

Equation (1) shows an indiscernibility relation, where $0 \leq S(x, y) \leq 1$ is a similarity function. This weaker binary relation states that objects x and y are deemed inseparable as long as their similarity degree $S(x, y)$ exceeds a similarity threshold $0 \leq \xi \leq 1$. This user-specified parameter determines the granularity degree; therefore, its precise estimation may become essential for designing high-performing rough classifiers.

$$R : xRy \Longleftrightarrow S(x, y) \geq \xi \tag{1}$$

The similarity function could be formulated in a variety of ways. In this study, we assume that $S(x, y) = 1 - \delta(x, y)$, where $\delta(x, y)$ denotes the distance between objects x and y. As a further contribution, this paper investigates the effect of using different distance functions over the overall prediction rates computed by the proposed rough cognitive classifiers.

It should be noticed that the similarity relation R does not induce a *partition* of $\mathcal{U}$ into a set of equivalence classes but rather a *covering* of $\mathcal{U}$ into multiple similarity classes $\bar{R}(x)$. This suggests that an object x could simultaneously belong to more than one similarity class at the same time. Next, we revise the theoretical foundations of three-way decision rules, which comprise the core of rough cognitive mapping.

2.1.2. The three-way-decisions model

Yao [37] introduced the *three-way decision rules* based on three disjoint regions produced by the lower and upper approximations, namely, the positive region $POS(X)$, the boundary region $BND(X)$, and the negative region $NEG(X)$. By exploiting the positive and negative regions we can extract confident decisions while including the boundary regions in the reasoning process allows concluding an alternative decision that is neither positive nor negative. These classification rules are depicted as follows:

- $Des([x]_\Phi) \rightarrow_P Des(X)$, for $[x]_\Phi \subseteq POS(X)$
- $Des([x]_\Phi) \rightarrow_B Des(X)$, for $[x]_\Phi \subseteq BND(X)$
- $Des([x]_\Phi) \rightarrow_N Des(X)$, for $[x]_\Phi \subseteq NEG(X)$

In these rules $Des([x]_\Phi)$ represents the logic formula defining the equivalence class $[x]_\Phi$ and $Des(X)$ is the name of the concept. A *positive rule* is used for accepting, a *negative rule* for rejecting and a *boundary rule* for abstaining. More precisely, if $[x]_\Phi \subseteq POS(X)$, we accept x to be a member of the concept X. If $[x]_\Phi \subseteq NEG(X)$, we reject x to be a member of X. If $[x]_\Phi \subseteq BND(X)$, we neither accept nor reject x to be a member of X; instead, we make a decision of deferment, abstaining or non-commitment [38]. Such rules model our inability to make a definite acceptance or rejection decision in scenarios with insufficient information.

Within the classic rough set model, the above classification rules are unnecessarily restrictive, but one can generalize these rules to avoid such limitations. The studies in [39] and [38] introduced a general probabilistic approach, called *Decision-theoretic Rough Set model*, which uses two states and three actions to characterize the decision process.

As mentioned before, the abstract semantic of three-way decision rules is used for automatically constructing the RCN topology. This allows hybridizing RST elements with the neural reasoning mechanism of fuzzy cognitive mapping in an attempt to design an effective granular classifier.

2.2. Fuzzy cognitive maps

Without lack of generality, FCMs can be defined as interpretable recurrent neural networks widely used in modeling and simulation purposes [31]. They consist of a set of concepts (i.e., objects, variables or entities for a particular problem) and their causal relations. The activation value of such concepts (also called neurons) regularly takes values in the $[0, 1]$ interval. The higher the activation value of a neuron, the stronger its influence over the system under investigation. On the other hand, the strength of the causal relation between two concepts C_i and C_j is quantified by a weight $w_{ij} \in [-1, 1]$ and denoted via a directed edge from C_i to C_j. There are three possible types of causal relationships among neural processing entities that express the type of influence from one neuron to the others:

- If $w_{ij} > 0$ then an increment (decrement) in the cause C_i produces an increment (decrement) of the effect C_j with intensity $|w_{ij}|$.
- If $w_{ij} < 0$ then an increment (decrement) in the cause C_i produces a decrement (increment) of the neuron C_j with intensity $|w_{ij}|$.
- If $w_{ij} = 0$ then there is no causal relation between C_i and C_j.

The activation update rule in an FCM is displayed in Equation (2), with $A^{(0)}$ being the initial activation vector. This rule is iteratively repeated until a stop condition is met. A new activation vector is calculated at each discrete time step t. After a fixed number of iterations, the map will arrive at one of the following states: (i) equilibrium point, (ii) limited cycle or (iii) chaotic behavior [40]. The map is said to have converged if it reaches a fixed-point attractor. Otherwise, the process terminates after a maximum number of iterations T is reached. Whichever the case, the FCM output corresponds to the response vector at the last discrete time step.

$$A_i^{(t+1)} = f\left(\sum_{j=1}^{M} w_{ji} A_j^{(t)} + A_i^{(t)}\right), i \neq j \tag{2}$$

The function $f(\cdot)$ in Equation (2) denotes a monotonically non-decreasing nonlinear function used to clamp the activation value of each neuron to the allowed interval. Examples of such functions are the bivalent function, the trivalent function and the sigmoid variants [41]. In this study, we confine ourselves to using a sigmoid function $f(A_i) = 1/(1 + \exp(-\lambda A_i))$ since it has exhibited superior prediction capabilities [41]. In the sigmoid function, the parameter $\lambda > 0$ denotes the steepness factor and influences the sensitivity of the neuron. Nápoles et al. [30] empirically concluded that, in the framework of rough cognitive mapping, $2 \leq \lambda \leq 5$ are suitable values.

FCMs have been augmented with different types of information granules, thus giving rise to high-level constructs known as *Granular Cognitive Maps* (GCMs). Pedrycz [42] mentions the allocation of information granularity as a pivotal driving force behind the development of these types of granular structures and describes five protocols as its realization mechanisms. For example, Pedrycz [43] and his collaborators [44] put forth a granular representation of time series in which FCM nodes are generated after the cluster prototypes induced by fuzzy c-means [45] over the space of amplitude and change of amplitude. More recently, Nápoles et al. [8] extended this model to solve graded multilabel classification problems.

Homenda et al. [46] adopted numeric intervals as the granulation vehicle for their GCM weight matrix. As a result, each FCM weight is no longer a number but an interval. The authors elaborate on three methodologies for building a GCM from scratch by maintaining an adequate balance between specificity and generality in the design of the interval-based FCM weights and the ensuing map operations. They found the resulting GCM had a good degree of coverage without a loss in precision.

Another development in the realm of GCMs is that of RCNs, which is the underlying building block for the proposed model. Section 3 elaborates on RCNs and stresses their advantages and limitations.

3. Rough cognitive networks

Nápoles et al. [33,34,30] recently introduced the concept of *rough cognitive mapping*, which hybridizes RST and FCMs. In this granular model, the abstract semantics of three-way decision rules is used to define the FCM topology. This allows overcoming the limitations in the expression and architecture of traditional cognitive mapping as the network topology (i.e., map concepts and causal relations) are directly computed from historical data, removing the expert intervention requirements. Definition 1 formalizes the semantics behind this granular cognitive classifier.

Definition 1. A Rough Cognitive Network $\mathcal{R}_{(\xi_i)}$ is a sigmoid FCM whose input neurons represent rough approximation regions and the output neurons denote the set of class labels for a pattern classification problem.

The first step when constructing an RCN is related to the *granulation of the input data* using RST. The positive, boundary and negative regions of each decision class according to a predefined attribute subset $\Phi \subseteq \Psi$ are computed using the training dataset and a predefined similarity relation R to define the similarity classes (see Section 2.1). The similarity

relations may employ either the whole attribute set Ψ or a subset $\Phi \subseteq \Psi$, in case a feature selection technique has identified one as such.

It is important to notice that when solving ML problems with numerical attributes, the similarity class of a new instance $\bar{R}(x)$ might activate multiple decision rules. If $\bar{R}(x)$ only comprises instances related to a single decision class, then the similarity class will be completely contained within a specific decision region. If the similarity class activates the positive region of multiple decisions, the RCN will calculate confidence levels for each active rule and then compute the preference degree associated with each decision. In more complex scenarios where only negative and boundary regions are activated, RCNs are capable of using the knowledge about positive, negative and boundary regions for computing the decision class.

The second step when constructing an RCN is the *automated topology design*, where a sigmoid FCM is automatically created from the aforementioned RST-based information granules. In this scheme, each rough region is mapped to an input-type neural entity whereas each decision class gives rise to an output-type concept. Thus, the RCN graph will comprise at most $3|\mathcal{D}| + |\mathcal{D}| = 4|\mathcal{D}|$ neurons, with $\mathcal{D} = \{D_1, \ldots, D_K\}$ being the set of decision classes. It should be stated that output neurons do not influence other neurons since they are receiver concepts and are only used to compute the response vector. Rules R_1–R_4 formalize the direction and intensity of the causal weights in the proposed topology; these weights are estimated by using the abstract semantics of three-way decision rules.

- (R_1) IF C_i is P_k AND C_j is D_k THEN $w_{ij} = 1.0$
- (R_2) IF C_i is P_k AND C_j is $D_{v \neq k}$ THEN $w_{ij} = -1.0$
- (R_3) IF C_i is P_k AND C_j is $P_{v \neq k}$ THEN $w_{ij} = -1.0$
- (R_4) IF C_i is N_k AND C_j is D_k THEN $w_{ij} = -1.0$

In such rules, C_i and C_j represent two map neurons, P_k and N_k are the positive and negative regions related to the kth decision respectively while $-1 \leq w_{ij} \leq 1$ is the causal weight between the cause C_i and the effect C_j. The reader may observe that neurons can be grouped into four categories (positive, negative, boundary and decision neurons) which allows designing the network using the principles behind three-way decision rules.

Although the boundary regions are concerned with an abstaining decision, an instance $x \in BND(X_k)$ could be positively related to the kth decision alternative. Therefore, an additional rule considering the knowledge about boundary regions is introduced:

- (R_5) IF C_i is B_k AND C_j is D_v AND $BND(X_k) \cap BND(X_v) \neq \emptyset$ THEN $w_{ij} = 0.5$

The last step is related to the *network exploitation* on which the response vector $\mathcal{A}_x(\mathcal{D}) = [A_x(D_1), \ldots, A_x(D_k), \ldots, A_x(D_K)]$ is computed. The input pattern x is presented to the granular classifier as an input vector $A^{(0)}$ that allows activating the causal network. Rules R_6–R_8 formalize the method used to activate the input neurons, which is based on the inclusion degree of the pattern to each rough granular region.

- (R_6) IF C_i is P_k THEN $A_i^{(0)} = \frac{|\bar{R}(x) \cap POS(X_k)|}{|POS(X_k)|}$
- (R_7) IF C_i is N_k THEN $A_i^{(0)} = \frac{|\bar{R}(x) \cap NEG(X_k)|}{|NEG(X_k)|}$
- (R_8) IF C_i is B_k THEN $A_i^{(0)} = \frac{|\bar{R}(x) \cap BND(X_k)|}{|BND(X_k)|}$

Once the excitation vector $A^{(0)}$ has been computed, the reasoning rule depicted in Equation (2) is performed until either the network converges to a fixed-point or a maximal number of iterations is reached. Next, the class with the highest activation value is assigned to the pattern.

Fig. 1 displays an RCN for solving two-class pattern classification problems. Notice that we added a self-reinforcement positive causal connection to each input neuron with the goal of preserving its initial excitation level when performing the neural updating rule.

Rough cognitive mapping brings up the following advantages:

1. It allows handling pattern classification problems with both numerical and nominal attributes, since the rough regions are constructed on the basis of weak binary (similarity) relations.
2. It yields a numeric vector of preference degrees associated to decision classes instead of binary decisions for each alternative, which is often a desirable feature in decision-making scenarios.
3. The RCN topology is learned from historical data, thus freeing the user from an entangled and likely subjective modeling phase to derive the causal relationships among the different concepts.
4. The granulation step allows reasoning at a higher level of abstraction, thus reducing the dimensionality of the feature space. Consequently, the number of neural processing entities in the causal network does not scale up with the number of features describing the objects, which facilitates the processing of Big Dimensionality problems.

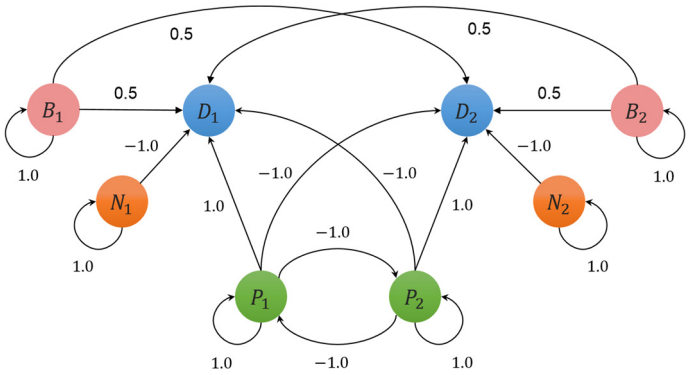

Fig. 1. RCN for pattern recognition problems with two decision classes.

4. Rough cognitive ensembles

A key issue when constructing an RCN is related to the proper selection of the similarity threshold. This parameter defines whether two objects are similar or not, which then influences the creation of the similarity classes upon which the rough approximations (RCN input nodes) are built. Regrettably, the RCN performance is highly sensitive to this user-specified parameter, thus small variations on the granularity degree may lead to quite different outcomes. As an alternative, Nápoles et al. [30] proposed a hyperparameter learning method to estimate the similarity threshold value from historical data. The main drawback of this procedure relies on its computational complexity due to the fact that a single objective function evaluation requires recalculating the lower and upper approximations of all decision classes, which could be time-consuming for large-scale datasets.

Let us assume that $\mathcal{U}_1 \subset \mathcal{U}$ is the training set and $\mathcal{U}_2 \subset \mathcal{U}$ is the held-out test (validation) set such that $\mathcal{U}_1 \cap \mathcal{U}_2 = \emptyset$. The computational complexity of building the upper and lower approximations is $O(|\Phi||\mathcal{U}_1|^2)$, with Φ being the attribute set, whereas the complexity of building the network topology is $O(|\mathcal{D}|^2)$, with $\mathcal{D}$ being the set of decision classes. Besides, the complexity of exploiting the granular network for $|\mathcal{U}_2|$ instances is $O(|\mathcal{U}_2||\Phi||\mathcal{U}_1|^2)$. This implies that the temporal complexity of evaluating a single parameter value is $O(\max\{|\Phi||\mathcal{U}_1|^2, |\mathcal{D}|^2, |\mathcal{U}_2||\Phi||\mathcal{U}_1|^2\})$. Due to the fact that $|\mathcal{U}_1| \geq |\mathcal{U}_2|$ in most machine learning scenarios, we can conclude that the overall complexity of this learning method is $O(T|\Phi||\mathcal{U}_1|^3)$, where T is the number of cycles. Unfortunately, this may negatively affect the practical usability of RCNs in solving real-world pattern classification problems.

On the other hand, estimating a suitable similarity threshold does not necessarily ensure that the granular network will produce optimal predictions. This happens because the validation and test sets must be disjoint, and so the parameter learning method fits the model to the examples in the validation set. Moreover, it might occur that $\bar{R}(x) = \emptyset$ for some testing object as a result of estimating a very strict threshold for validation examples. If so, the model uses the k-nearest neighbors to activate the network.

Next, we introduce the notion of Rough Cognitive Ensembles (RCEs) in an attempt to get rid of the parameter learning scheme while increasing the performance of rough cognitive mapping.

4.1. The ensemble architecture

Informally speaking, an RCE is an ensemble of several RCNs, where each individual granular network operates at a different granularity degree. To induce different granularity degrees, we use a random similarity threshold when constructing the three approximation regions associated with each base classifier. This approach attempts to eliminate the parameter learning requirement, which in fact does not necessarily lead to optimal prediction rates when testing the classifier. Definition 2 summarizes the semantics of the granular cognitive ensembles presented in this research.

Definition 2. A Rough Cognitive Ensemble $\Gamma_{(\mathcal{R})}$ is a multiclassifier system composed of a set $\mathcal{R} = \{\mathcal{R}_{(\xi_1)}, \ldots, \mathcal{R}_{(\xi_i)}, \ldots, \mathcal{R}_{(\xi_N)}\}$ of N different RCNs as base classifiers, where the ith granular cognitive network $\mathcal{R}_{(\xi_i)}$ is constructed from a (randomly) selected granularity degree.

Another important aspect when designing a classifier ensemble is related to the aggregation of multiple outputs. Combining the decisions of different models means amalgamating the various outputs into a single prediction. The simplest way to do this in classification models is adopting a standard (or weighted) voting scheme. Voting gives a significant improvement in classification accuracy and stability [47]. The higher the number of base classifiers, the merrier: predictions made by voting become more reliable as more votes are taken into account. Therefore, the classification result for each test instance is obtained by voting on all N classifiers.

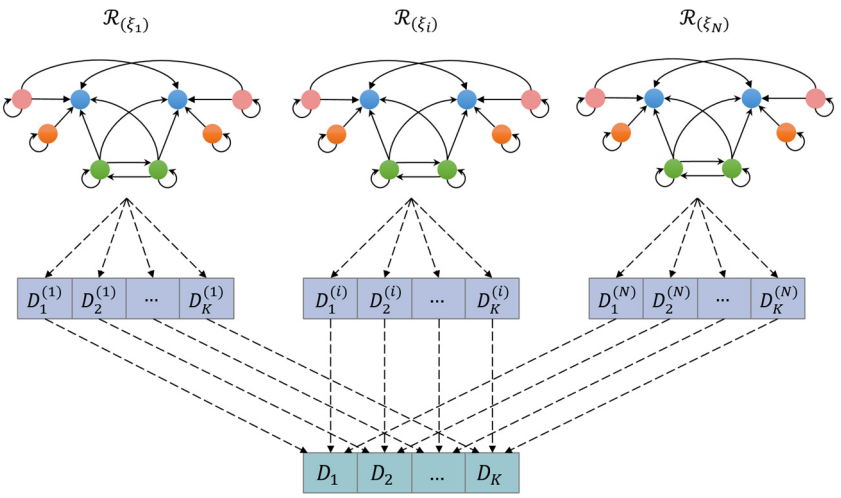

Fig. 2. Rough Cognitive Ensemble of N networks for problems with M classes.

Theoretical and empirical results [48,49,18] have shown that combining multiple base classifiers leads to optimal performance if these classifiers are not strongly correlated with one another. In the proposed granular ensemble, the diversity among base classifiers is promoted by using different similarity thresholds $[\xi_1, \ldots, \xi_i, \ldots, \xi_N]$ instead of using a single value. Fig. 2 displays an RCE comprised of N base classifiers with K decision classes, where $D_k^{(i)}$ denotes the kth decision class for the ith granular network $\mathcal{R}_{(\xi_i)}$ and D_k is the aggregated-type concept associated with the kth decision class. In this architecture, the function $A(.)$ is used to compute the activation value of all sigmoid neurons involved in the ensemble.

The reader may notice that if $\xi_i \leq \xi_j$ then $\bar{R}_{(\xi_j)}(x) \subseteq \bar{R}_{(\xi_i)}(x)$, which could produce correlated base classifiers. In order to increase diversity we can introduce randomization. Instance-based learners depend mostly on attributes used to compute the distance/similarity between objects. This suggests that we can promote diversity by using a random subset of attributes (i.e., random subspace method). However, in the context of rough classification, using random attributes may not be the best choice since rough approximations are often built upon a *reduct* of the attribute set.

Besides, unlike nearest-neighbor classifiers, RCN-based models are sensitive to perturbations on the training data due to the presence of the similarity threshold, even when using the same subset of attributes. Hence, we can perform *instance bagging* [50] in order to counter the correlation coming from the rule $\xi_i \leq \xi_j \Longrightarrow \bar{R}_{(\xi_j)}(x) \subseteq \bar{R}_{(\xi_i)}(x)$. Instance bagging attempts to neutralize the instability of the base classifier by modifying the original training set (i.e., deleting some instances and replicating others). During this process, instances are randomly sampled with replacement from the original dataset to create a new one with the same size [51]. This allows establishing a reasonable trade-off between ensemble diversity and accuracy.

4.2. The exploitation scheme

The exploitation phase of the proposed granular ensemble is focused on computing a vector $\mathcal{A}_x(\mathcal{D}^{(i)}) = [A_x(D_1^{(i)}), \ldots, A_x(D_k^{(i)}), \ldots, A_x(D_K^{(i)})]$ for each granular classifier. In this output vector, $A_x(D_k^{(i)})$ represents the activation degree of the kth output neuron according to the ith granular network, with x being the test instance. This allows computing the aggregated vector $\mathcal{A}_x(\mathcal{D}) = [A_x(D_1), \ldots, A_x(D_k), \ldots, A_x(D_K)]$ by combining the responses of the N base granular classifiers over K decision classes.

In order to activate the ensemble, we need to compute N excitation vectors $\{\mathcal{A}^{(0)}_{[x|\xi_i]}\}_{i=1}^{N}$ where $\mathcal{A}^{(0)}_{[x|\xi_i]}$ is used to perform the neural reasoning process in the ith cognitive network. The ith activation vector denotes the inclusion degree of the similarity class $\bar{R}_{(\xi_i)}(x)$ into each information granule, according to the corresponding similarity threshold ξ_i. Formally, the ith activation vector could be mathematically defined as follows:

$$\begin{aligned}\mathcal{A}^{(0)}_{[x|\xi_i]} = \big[&A^{(0)}_{[x|\xi_i]}(P_1), \ldots, A^{(0)}_{[x|\xi_i]}(P_k), \ldots, A^{(0)}_{[x|\xi_i]}(P_K),\\ &\ldots, A^{(0)}_{[x|\xi_i]}(N_1), \ldots, A^{(0)}_{[x|\xi_i]}(N_k), \ldots, A^{(0)}_{[x|\xi_i]}(N_K), \ldots,\\ &A^{(0)}_{[x|\xi_i]}(B_1), \ldots, A^{(0)}_{[x|\xi_i]}(B_k), \ldots, A^{(0)}_{[x|\xi_i]}(B_K)\big]\end{aligned}$$

where $A^{(0)}_{[x|\xi_i]}(P_k)$, $A^{(0)}_{[x|\xi_i]}(N_k)$ and $A^{(0)}_{[x|\xi_i]}(B_k)$ denote the activation degree of the kth positive, negative and boundary region, respectively. These values are computed according to the activation rules R_6–R_8.

When activating the granular ensemble classifier, output-type neurons remain inactive since their activation value is computed from the recurrent propagation of the evidence over the ith granular network. This implies that $A^{(0)}_{[x|\xi_i]}(D_1) =$

$\ldots = A^{(0)}_{[x|\xi_i]}(D_k) = \ldots = A^{(0)}_{[x|\xi_i]}(D_K) = 0$. Sometimes, there is additional knowledge about the probability of producing a specific decision class given an instance (e.g., resulting from an intermediate classification process). In these cases, we could use this knowledge to activate the output neurons in order to improve the performance.

Once the neural reasoning step is completed (i.e., either an equilibrium point is discovered or a maximal number of iterations is reached), the predicted class is derived from the aggregated output vector:

$$\mathcal{A}_x(D) = [A_x(D_1), \ldots, A_x(D_k), \ldots, A_x(D_K)].$$

This aggregated vector is obtained by voting on all N base classifiers over the K decision classes. Therefore, the activation degree $A_x(D_k)$ denotes the probability $P(D_k|x \in X)$ of producing the kth decision class given $x \in X$, and is computed according to Equation (2). The reader may notice that we can simulate the voting scheme in the cognitive network by connecting the kth output-type neuron $D^{(i)}_k$ with the corresponding decision neuron, and next performing a single-ahead reasoning process.

Sometimes we could identify scenarios on which multiple decision classes have the highest probability of being produced (i.e., two aggregated neurons bear the same maximal activation value). These situations are common in highly inconsistent problems where there is a lack of conclusive knowledge. To solve this issue, we select the decision class D_k associated with the closest neighbor of the test instance $x \in X$, such that D_k has the highest output probability. This heuristic provides a fair compromise between the decision class computed by the granular ensemble and the intrinsic relation among those instances that represent the same pattern.

5. Heterogeneous distance functions

As mentioned, the distance function plays a pivotal role when designing instance-based classifiers. As far as we know, there is no study on the impact of this component on the performance of RCN-based classifiers. In this section, we revise the mathematical formulation of three widely used distance functions taken from [52] that allow comparing heterogeneous instances, i.e., objects comprising both numerical and nominal attributes.

Let $\Phi = \{\phi_1, \ldots, \phi_M\}$ denote the attribute set, where ϕ_j can be either numerical or nominal, and it has a weight $0 \leq \omega_j \leq 1$ attached that quantifies its relevance. The dissimilarity function $\delta(x, y)$ between two instances x and y can be computed using one of the following distance functions:

- **The Heterogeneous Euclidean-Overlap Metric (HEOM)**. This heterogeneous distance function computes the normalized Euclidean distance between numerical attributes and an overlap metric for nominal attributes. Equations (3) and (4) define the HEOM distance function, where $x(j)$ and $y(j)$ are the normalized values of the jth attribute for heterogeneous instances x and y, respectively.

$$\delta_{HEOM}(x, y) = \sqrt{\frac{\sum_{j=1}^{|\Phi|} \omega_j \sigma_j(x, y)}{\sum_{j=1}^{|\Phi|} \omega_j}} \tag{3}$$

where

$$\sigma_j(x, y) = \begin{cases} 0 & \text{if } \phi_j \text{ is nominal } \wedge\, x(j) = y(j) \\ 1 & \text{if } \phi_j \text{ is nominal } \wedge\, x(j) \neq y(j) \\ (x(j) - y(j))^2 & \text{if } \phi_j \text{ is numerical} \end{cases} \tag{4}$$

- **The Heterogeneous Manhattan-Overlap Metric (HMOM)**. This heterogeneous variant is similar to the HEOM function since it replaces the Euclidean distance with the Manhattan distance when computing the dissimilarity between two numerical values. Equations (5) and (6) formalize the HMOM distance function, whose calculation requires less computational effort compared to HEOM's.

$$\delta_{HMOM}(x, y) = \frac{\sum_{j=1}^{|\Phi|} \omega_j \rho_j(x, y)}{\sum_{j=1}^{|\Phi|} \omega_j} \tag{5}$$

where

$$\rho_j(x, y) = \begin{cases} 0 & \text{if } \phi_j \text{ is nominal } \wedge\, x(j) = y(j) \\ 1 & \text{if } \phi_j \text{ is nominal } \wedge\, x(j) \neq y(j) \\ |x(j) - y(j)| & \text{if } \phi_j \text{ is numerical} \end{cases} \tag{6}$$

- **The Heterogeneous Value Difference Metric (HVDM)**. This function involves a stronger strategy for quantifying the dissimilarity between two discrete attribute values. Instead of computing the matching between attribute values, it measures the correlation between such attributes and decision classes. Equations (7) and (8) show the HVDM function variant adopted in this research.

$$\delta_{HVDM}(x, y) = \sqrt{\frac{\sum_{j=1}^{|\Phi|} \omega_j \tau_j(x, y)}{\sum_{j=1}^{|\Phi|} \omega_j}} \tag{7}$$

where

$$\tau_j(x, y) = \begin{cases} \frac{1}{K} \sum_{k=1}^{K} \left(\frac{\beta_{\phi_j, x(j), k}}{\beta_{\phi_j, x(j)}} - \frac{\beta_{\phi_j, y(j), k}}{\beta_{\phi_j, y(j)}} \right)^2 & \text{if } \phi_j \text{ is nominal} \\ (x(j) - y(j))^2 & \text{if } \phi_j \text{ is numerical} \end{cases} \tag{8}$$

whereas $\beta_{\phi_j, x(j)}$ is the number of instances in the training set that have value $x(j)$ for attribute ϕ_j, and $\beta_{\phi_j, x(j), k}$ denotes the number of instances that have value $x(j)$ for attribute ϕ_j and output class k.

In these distance functions, approximating the relevance of each attribute may result in improved prediction rates. To overcome this issue, we compute the *gain ratio* [53] associated with the jth attribute:

$$G(\phi_j, X) = \frac{I(X) - E(\phi_j, X)}{IC(\phi_j)} \tag{9}$$

where

$$I(X) = -\sum_{k=1}^{K} \frac{|X \cap D_k|}{|X|} log_2 \frac{|X \cap D_k|}{|X|} \tag{10}$$

measures the randomness of the distribution of available instances in X over K decision classes, whereas $E(\phi_j, X)$ is given by

$$E(\phi_j, X) = \sum_{l=1}^{L_j} \frac{|X_l|}{|X|} I(X_l) \tag{11}$$

where L_j is the number of possible values for the ϕ_j attribute and X_l denotes the instance set in X having value v_l for the ϕ_j attribute. In order to counter this metric's bias in favor of attributes with a larger number of values, the following normalization factor is introduced:

$$IC(\phi_j) = -\sum_{l=1}^{L_j} \frac{|X_l|}{|X|} log_2 \frac{|X_l|}{|X|} \tag{12}$$

In our approach, the gain ratio associated to the jth attribute replaces the weight ω_j when computing the dissimilarity degree between two heterogeneous instances. If $\sum \omega_j = 0$, then attribute weighting is not performed. Of course, we can adopt more sophisticated measures as the distance between partitions, but it will increase the computational cost.

6. Numerical simulations

In this section, we conduct several experiments in order to evaluate the prediction capabilities of the proposed ensemble classifier. With this goal in mind, we fist determine the best-performing variant among multiple ensemble models and demonstrate RCE's superiority with regards to the RCN-based classifiers. In these simulations we also evaluate the algorithm's performance for different distance functions. Afterwards, we compare the prediction capability of the best-performing ensemble method against 15 well-established classifiers across 140 pattern classification datasets.

6.1. Dataset characterization

We leaned upon 140 well-known pattern classification datasets taken from the KEEL [54] and UCI ML [55] repositories. These ML problems comprise different characteristics and allow evaluating the predictive capability of both state-of-the-art and granular algorithms under consideration.

Table 1 outlines the number of instances, attributes and decision classes for each dataset. The presence of noise and the imbalance ratio (calculated as the ratio of the size of the majority class to that of the minority class) are also given. In this paper, we say that a dataset is imbalanced if the number of instances belonging to the majority decision class is at least five times the number of instances belonging to the minority class.

In the adopted datasets, the number of attributes ranges from 2 to 262, the number of decision classes from 2 to 100, and the number of instances from 14 to 12,906. These ML problems involve 13 noisy and 47 imbalanced datasets, where the imbalance ratio ranges from 5:1 to 2160:1. In order to avoid the out-of-range issues in the adopted heterogeneous distance functions, the numerical attributes have been normalized. On the other hand, we replaced missing values by the mean or the mode depending if the attribute is numerical or nominal, respectively.

Table 1
Characterization of the ML datasets adopted for the simulations.

Dataset	Instances	Attributes	Classes	Noisy	Imbalance
abalone	4174	8	28	no	689:1
acute-inflammation	120	6	2	no	no
acute-nephritis	120	6	2	no	no
anneal	898	38	6	no	85:1
anneal.orig	898	38	6	no	85:1
appendicitis	106	7	2	no	no
arrhythmia	452	262	13	no	122:1
audiology	226	69	24	no	57:1
australian	690	14	2	no	no
autos	205	25	7	no	22:1
balance-noise	625	4	3	yes	5:1
balance-scale	625	4	3	no	5:1
ballons	16	4	2	no	no
banana	5300	2	2	no	no
bank	4521	16	2	no	7:1
blood	748	4	2	no	no
breast	277	9	2	no	no
bc-wisconsin-diag	569	31	2	no	no
bc-wisconsin-prog	198	34	2	no	no
bridges-version1	107	12	6	no	no
bridges-version2	107	12	6	no	no
car	1728	6	4	no	17:1
cardiotocography-10	2126	35	10	no	10:1
cardiotocography-3	2126	35	3	no	9:1
chess	3196	36	2	no	no
cleveland	297	13	5	no	12:1
colic	368	22	2	no	no
colic.orig	368	27	2	no	no
collins	500	23	15	no	13:1
contact-lenses	24	4	3	no	no
contraceptive	1473	9	3	no	no
credit-a	690	15	2	no	no
credit-g	1000	20	2	no	no
crx	653	15	2	no	no
csj	653	34	6	no	no
cylinder-bands	540	39	2	no	no
dermatology	358	34	6	no	5:1
echocardiogram	131	11	2	no	5:1
ecoli	336	7	8	no	71:1
ecoli0	220	7	2	no	no
ecoli-0vs1	220	7	2	no	no
ecoli1	336	7	2	no	no
ecoli2	336	7	2	no	5:1
ecoli3	336	7	2	no	8:1
ecoli-5an-nn	336	7	8	yes	71:1
energy-y1	768	8	38	no	no
energy-y2	768	8	38	no	no
eucalyptus	736	19	5	no	no
flags	194	28	8	no	15:1
glass	214	9	6	no	8:1
glass0	214	9	2	no	no
glass-0123vs456	214	9	2	no	no
glass1	214	9	2	no	no
glass-10an-nn	214	9	6	yes	8:1
glass2	214	9	2	no	no
glass-20an-nn	214	9	6	yes	8:1
glass3	214	9	2	no	6:1
glass-5an-nn	214	9	6	yes	8:1
glass6	214	9	2	no	6:1
haberman	306	3	2	no	no
hayes-roth	160	4	3	no	no
heart-5an-nn	270	13	2	yes	no
heart-statlog	270	13	2	no	no
hypothyroid	3772	29	4	no	1740:1
ionosphere	351	34	2	no	no
iris	150	4	3	no	no

Table 1 (*continued*)

Dataset	Instances	Attributes	Classes	Noisy	Imbalance
iris0	150	4	2	no	no
iris-10an-nn	150	4	3	yes	no
iris-20an-nn	150	4	3	yes	no
iris-5an-nn	150	4	3	yes	no
labor	57	16	2	no	no
led7digit	500	7	10	no	no
libras	360	90	15	no	no
liver-disorders	345	6	2	no	no
lung-cancer	32	56	3	no	no
lymph	148	18	4	no	40:1
mammographic	830	5	2	no	no
mfeat-factors	2000	216	10	no	no
mfeat-fourier	2000	76	10	no	no
mfeat-karhunen	2000	64	10	no	no
mfeat-morpho	2000	6	10	no	no
mfeat-pixel	2000	240	10	no	no
mfeat-zernike	2000	47	10	no	no
molecular-biology	106	57	2	no	no
monk-2	432	6	2	no	no
mushroom	5644	22	2	no	no
musk-1	476	167	2	no	no
musk-2	6598	167	2	no	5:1
new-thyroid	215	5	2	no	5:1
nursery	12960	8	5	no	2160:1
optdigits	5620	64	10	no	no
ozone	2536	72	2	no	33:1
page-blocks	5473	10	5	no	175:1
parkinsons	195	22	2	no	no
pendigits	10992	16	10	no	no
phoneme	5404	5	2	no	no
pima	768	8	2	no	no
pima-10an-nn	768	8	2	yes	no
pima-20an-nn	768	8	2	yes	no
pima-5an-nn	768	8	2	yes	no
planning	182	12	2	no	no
plant-margin	1600	64	100	no	no
plant-shape	1600	64	100	no	no
plant-texture	1599	64	100	no	no
postoperative	90	8	3	no	32:1
primary-tumor	339	17	22	no	84:1
saheart	462	9	2	no	no
segment	2310	19	7	no	no
solar-flare-1	323	5	6	no	11:1
solar-flare-2	1066	12	6	no	7:1
sonar	208	60	2	no	no
soybean	683	35	19	no	11:1
spambase	4601	57	2	no	no
spectfheart	267	44	2	no	no
spectrometer	531	101	48	no	29:1
splice	3190	60	3	no	no
sponge	76	44	3	no	23:1
tae	151	5	3	no	no
tic-tac-toe	958	9	2	no	no
vehicle	846	18	4	no	no
vehicle0	846	18	2	no	no
vehicle1	846	18	2	no	no
vehicle2	846	18	2	no	no
vehicle3	846	18	2	no	no
vertebral2	310	6	2	no	no
vertebral3	310	6	3	no	no
vote	435	16	2	no	no
vowel	990	13	11	no	no
wall-following	5456	24	4	no	6:1

(*continued on next page*)

Table 1 (*continued*)

Dataset	Instances	Attributes	Classes	Noisy	Imbalance
waveform	5000	40	3	no	no
weather	14	4	2	no	no
wine	178	13	3	no	no
wine-5an-nn	178	13	3	yes	no
winequality-red	1599	11	6	no	68:1
winequality-white	4898	11	7	no	439:1
wisconsin	683	9	2	no	no
yeast	1484	8	10	no	92:1
yeast1	1484	8	2	no	no
yeast3	1484	8	2	no	8:1
zoo	101	16	7	no	10:1

Each dataset has been partitioned using a 10-fold cross-validation procedure, i.e., the dataset has been split into ten folds, each one containing approximately 10% of the instances. For each fold, a ML algorithm is trained with the instances contained in the training partition (9 folds) and then tested with the current fold. It should be mentioned that test partitions are kept aside to evaluate the performance of the learned model.

6.2. State-of-the-art classifiers used for comparison

In this subsection, we outline the classifiers adopted for benchmarking purposes. The variants and default parameter settings implemented in Weka v3.6.11 [56] have been retained throughout the simulations. It should be highlighted that these default parameter values are common across all datasets, thus no algorithm performs hyperparameter tuning.

As Triguero et al. [57] stated, a good choice of parameter values increases the algorithms' performance over different data sources. Nevertheless, a robust model should allow good-enough results to be obtained even when its parameters might not have been optimized for a specific dataset. Observe that robustness is not a requirement for accuracy since we can optimize the model parameters and produce higher prediction rates, but it also increases the computational complexity of setting up the classifier.

In spite of the above facts, the key reason behind the decision of not performing hyperparameter learning relies on the aim of our research, namely: *to suppress the parametric requirements when constructing a Rough Cognitive Network*. Accordingly, the following simulations focus on examining the prediction capability of the ensemble model against well-established classifiers, even when no algorithm undergoes parameter tuning.

The classifiers used for comparison are: Decision Table (DT) [58], Naive Bayes (NB) [59], Naive Bayes Updateable (NBU) [59], Support Vector Machines (SMO) with sequential minimal optimization algorithm [60], Multilayer Perceptron (MLP) [61], Simple Logistic (SL) [62], Decision Tree (J48) [63], Fast Decision Trees (FDT) [64], Best-first Decision Trees (BFT) [65], Logistic Model Trees (LMT) [66], Random Trees (RT) [67], Random Forests (RF) [18], k-nearest neighbors learner (kNN) [68], K^* instance-based classifier (K*) [69] and Locally Weighted Learning (LWL) [70].

In the case of RCN variants, we set the similarity threshold to $\xi = 0.98$ for all datasets while the similarity threshold ξ_i associated with the ith base classifier is uniformly distributed in the $[0.96, 1)$ interval. Aiming at reducing the complexity of the granular ensemble, we employed $N = 10$ base classifiers in the subsequent numerical simulations.

6.3. Determining the best-performing granular classifier

The first simulation focused on determining the best-performing granular model. With this goal in mind, we evaluated the prediction capability of standard RCNs, the ensemble model without bagging (RCE) and the ensemble model performing instance bagging (RCB) across three heterogeneous distance functions. Fig. 3 displays the average Cohen's kappa coefficient achieved by each ML algorithm. The Cohen's kappa coefficient [71] measures the inter-rater agreement for categorical items. It is usually deemed a more robust measure than the standard accuracy since this coefficient takes into account the agreement occurring by chance.

In order to examine the existence of statistically significant differences in performance, we computed the Friedman two-way analysis of variances by ranks [72]. The Friedman test is a multiple-comparison nonparametric statistical method that detects whether at least two of the samples in a group represent populations with different median values or not. The test suggests rejecting the null hypothesis H_0 (p-value $= 7.989819\text{E}{-11} < 0.05$) using a confidence interval of 95%. This implies that there exist statistically significant performance differences between at least two algorithms across all the selected datasets. Fig. 4 shows the rank values computed by the Friedman test, where the RCB algorithm using the HVDM distance function emerged as the best-ranked granular classifier.

The next step is to determine whether the superiority of the RCB-HVDM classifier is statistically significant or not. By doing so, we resorted to the Wilcoxon signed rank test [73] and several post-hoc procedures to adjust the p-values, as recently suggested Benavoli et al. [74]. The post-hoc procedures are required since in pairwise analysis, if we try to draw a conclusion involving more than one pairwise comparison, we accumulate an error coming from their combination. Therefore,

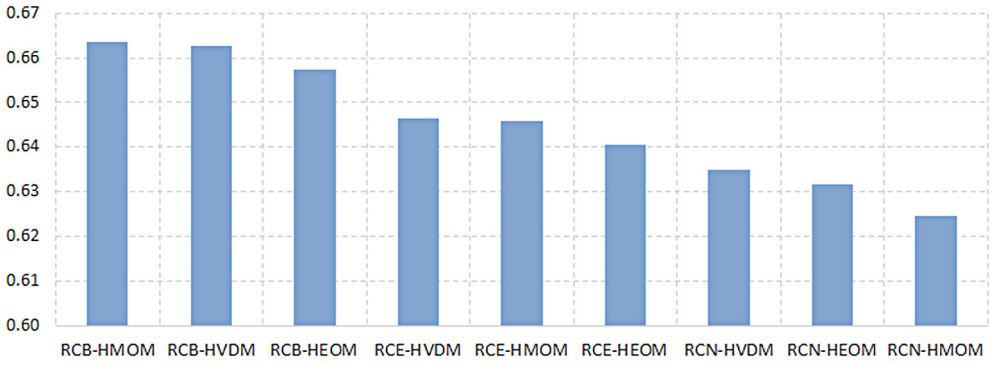

Fig. 3. Average Kappa measure computed by the rough classifiers.

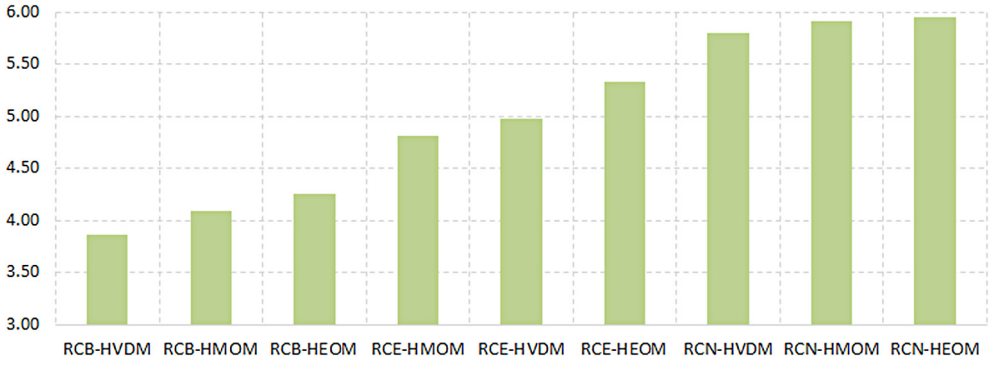

Fig. 4. Friedman's rank for the rough classifiers.

Table 2
Adjusted p-values according to different post-hoc procedures using the best-performing ensemble classifier (RCB-HVDM) as a control method.

Algorithm	p-value	Bonferroni	Holm	Holland	Hypothesis
RCN-HEOM	6.67E−10	5.33E−09	5.33E−09	5.33E−09	Rejected
RCN-HVDM	3.44E−08	2.75E−07	2.40E−07	2.40E−07	Rejected
RCN-HMOM	5.59E−08	4.47E−07	3.35E−07	3.35E−07	Rejected
RCE-HEOM	1.75E−06	1.40E−05	8.76E−06	8.76E−06	Rejected
RCE-HVDM	0.001107	0.008861	0.004430	0.004423	Rejected
RCE-HMOM	0.003157	0.025261	0.009473	0.009443	Rejected
RCB-HEOM	0.010440	0.083525	0.020881	0.020772	Rejected
RCB-HMOM	0.848561	1.000000	0.848561	0.848561	Accepted

we are losing control on the *Family-Wise Error Rate*, defined as the probability of making one or more false discoveries among all the hypotheses associated to multiple pairwise tests.

Table 2 reports the unadjusted p-value computed by the Wilcoxon signed rank test as well as the corrected p-values associated with each pairwise comparison using RCB-HVDM as the control method. We assume that a null hypothesis H_0 can be rejected if at least one of the adopted post-hoc procedures supports the rejection. The statistical analysis supports the superiority of the RCB-HVDM classifier as all the conservative hypotheses were rejected, save for the one concerning to the RCB-HMOM vs. RCB-HVDM pair. This suggests that RCB-HMOM and RCB-HVDM perform comparably although the RCB-HMOM algorithm is ranked first.

Fig. 5 displays the impact of using (a) different heterogeneous distance functions and (b) different ensemble strategies. From this figure we can formalize two key conclusions. First, performing instance bagging is convenient regardless of the underlying distance function. Second, it seems that selecting an ensemble scheme may reduce the negative effects of using a non-optimal distance function for a particular dataset.

The above simulations confirm the superiority of the RCE-based models over the base RCN classifier. Furthermore, the proposed ensemble classifier removed the need for an RCN parameter tuning stage, which became one of the main motivations of this study.

(a) Distance function.

(b) Ensemble scheme.

Fig. 5. Average Kappa measure according to different criteria.

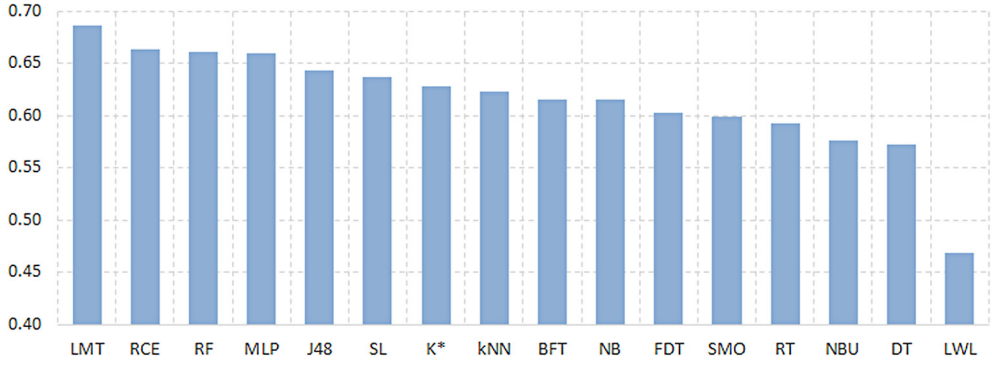

Fig. 6. Average Kappa measure computed by adopted classifiers.

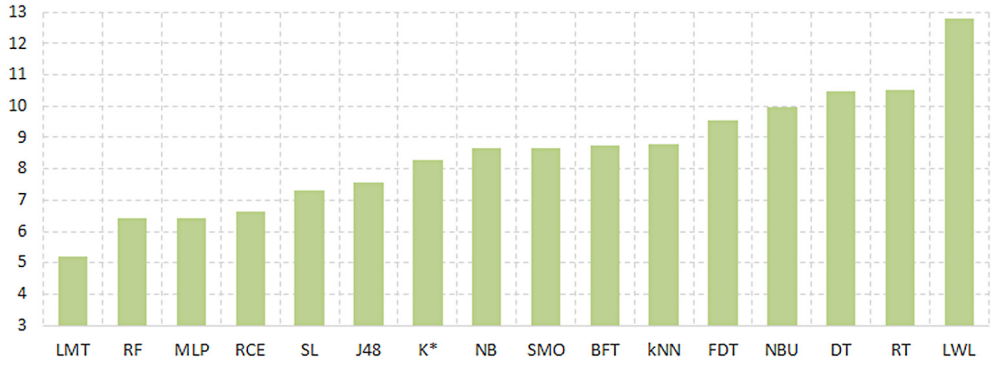

Fig. 7. Friedman's ranking for adopted classifiers.

6.4. Comparison against state-of-the-art classifiers

In this subsection, we compare the prediction ability of the best-performing granular ensemble (RCB-HVDM, hereinafter simply called RCE) against the traditional classifiers mentioned in Section 6.2. Analogously to the previous simulations, we utilized Cohen's kappa coefficient to quantify the algorithms' performance. Fig. 6 portrays the average Kappa measure attained by each algorithm across the 140 problems in our testbed.

For this experiment, the Friedman test suggests rejecting the null hypothesis (p-value $= 1.519139\text{E}{-10} < 0.05$) using a confidence level of 95%, thus there are significant differences between at least two algorithms across the selected datasets. Fig. 7 shows the rank values computed by the Friedman test. From these results, it is clear that LMT is the best-ranked algorithm, RCE is the fourth-best ranked, whereas LWL is the worst one. LMT refers to logistic model trees, i.e., classification trees with logistic regression functions at the leaves while LWL is an instance-based classifier that employs locally weighted linear regression to make predictions.

Table 3 shows the p-values reported by the Wilcoxon signed rank test and the corrected p-values according to the post-hoc procedures using the granular ensemble (RCE) as the control method. The results point to the fact that LML stands

Table 3
Adjusted p-values according to different post-hoc procedures using the proposed ensemble classifier (RCB-HVDM) as a control method.

Algorithm	p-value	Bonferroni	Holm	Holland	Hypothesis
LWL	3.22E−19	4.83E−18	4.83E−18	0.000000	Rejected
RT	3.11E−12	4.66E−11	4.35E−11	4.35E−11	Rejected
DT	6.67E−10	1.00E−08	8.67E−09	8.67E−09	Rejected
NBU	8.75E−08	1.31E−06	1.05E−06	1.05E−06	Rejected
kNN	4.47E−07	6.70E−06	4.91E−06	4.91E−06	Rejected
FDT	1.52E−06	2.28E−05	1.52E−05	1.52E−05	Rejected
NB	8.06E−05	0.001210	7.26E−04	7.25E−04	Rejected
SMO	0.000469	0.007038	0.003753	0.003747	Rejected
K*	0.000471	0.007072	0.003753	0.003747	Rejected
BFT	0.000510	0.007649	0.003753	0.003747	Rejected
LMT	0.000997	0.014962	0.004987	0.004977	Rejected
J48	0.059231	0.888460	0.236922	0.216691	Accepted
RF	0.399342	1.000000	1.000000	0.783288	Accepted
SL	0.486293	1.000000	1.000000	0.783288	Accepted
MLP	0.990836	1.000000	1.000000	0.990835	Accepted

as the best-performing classifier in our study, with no significant differences spotted between our proposal and RF, MLP, SL and J48, as the null hypothesis was accepted in each of these pairwise comparisons. Note however that SL and J48 report slightly lower Kappa measures. More importantly, the granular ensemble is capable of outperforming the remaining classifiers. This confirms the reliability of the granular ensemble in solving ML problems with a wide range of features.

The last experiment is oriented to characterizing those problems for which RCEs constitute the best algorithmic choice. Nevertheless, detecting these features manually from a table comprising 140 datasets and 15 algorithms is quite challenging and may lead to misleading conclusions.

Metalearning is a subfield of ML where automatic learning algorithms are applied on metadata concerning classification/prediction simulations. Although different researchers hold different views as to what the term exactly means, the main goal is to use such metadata (i.e., problems' features) to understand how learning models can become flexible in solving different kinds of classification problems [75]. Moreover, under the proper conditions, meta-learning also allows detecting the features on which a specific algorithm may outperform a set of baseline classification models.

In some scenarios, it is enough to predict whether the algorithm under analysis will be successful or not, while in others the purpose is to understand why the algorithm behaves the way it does.

In this paper, we use the well-known kNN classifier [68] as baseline algorithm since both belong to the same family (i.e., instance-based classifiers). The ML metadata include five features, namely: the number of instances, number of attributes, number of classes, whether the problem is noisy or not, and the imbalance ratio. Moreover, we consider a decision attribute with two classes (*P – positive*, *N – negative*). Observe that this dataset will comprise the algorithms' behaviors for adopted problems.

Therefore, a ML problem will be labeled as *positive* if the proposed algorithm was capable of outperforming the baseline classifier, otherwise it will be labeled as *negative*. More explicitly, we assume that an instance is *positive* if $\kappa(\text{RCE}) - \kappa(k\text{NN}) \geq 0.05$, with $\kappa(.)$ denoting the kappa coefficient for a particular problem. If $\kappa(\text{RCE}) - \kappa(k\text{NN}) \leq -0.05$ then the instance is labeled as *negative*, while those problems where $-0.05 < |\kappa(\text{RCE}) - \kappa(k\text{NN}) < 0.05$ were not included in the new dataset in order to increase the separability between instances used to build the metaclassifier.

For instance, the *audiology* problem includes 226 objects, 69 attributes, 24 decision classes and it does not involve noisy features, but a high imbalance degree (57:1). Since $\kappa(\text{RCE}) = 0.75$ and $\kappa(k\text{NN}) = 0.59$ we can infer the synthetic instance: "226, 69, 24, no, high, P". Following the same procedure, we obtain an artificial ML dataset comprising 63 instances (i.e., 47 labeled as positive and 16 labeled as negative). From this knowledge we can identify when our classification model is the best choice.

The next step focuses on training a decision tree classifier [63] to predict whether the ensemble will be adequate for solving a new ML problem or not. More importantly, decision tree classifiers allow deriving rules to interpret the prediction results. Fig. 8 displays the unpruned decision tree obtained after running the underlying learning scheme.

The decision tree classifier yielded an 86% accuracy, which can be interpreted as the confidence of the derived rule set. Such rules suggest that the granular ensemble outperforms the kNN algorithm in rather balanced problems, or alternatively in those problems that exhibit a certain imbalance degree but are defined by a reasonable number of attributes. Moreover, the derived rules have shown that our classifier seems to be less sensitive to noise, which could be a direct result of performing instance bagging over base classifiers with variable granularity levels.

7. Concluding remarks

In this paper, we introduced the notion of *Rough Cognitive Ensembles* as an alternative to deal with the hyperparameter learning requirements of the RCN algorithm. This granular classifier can be defined as a multiclassifier system composed of a set of RCNs, each operating at a different granularity degree. Tit should be noticed that inducing different granularity

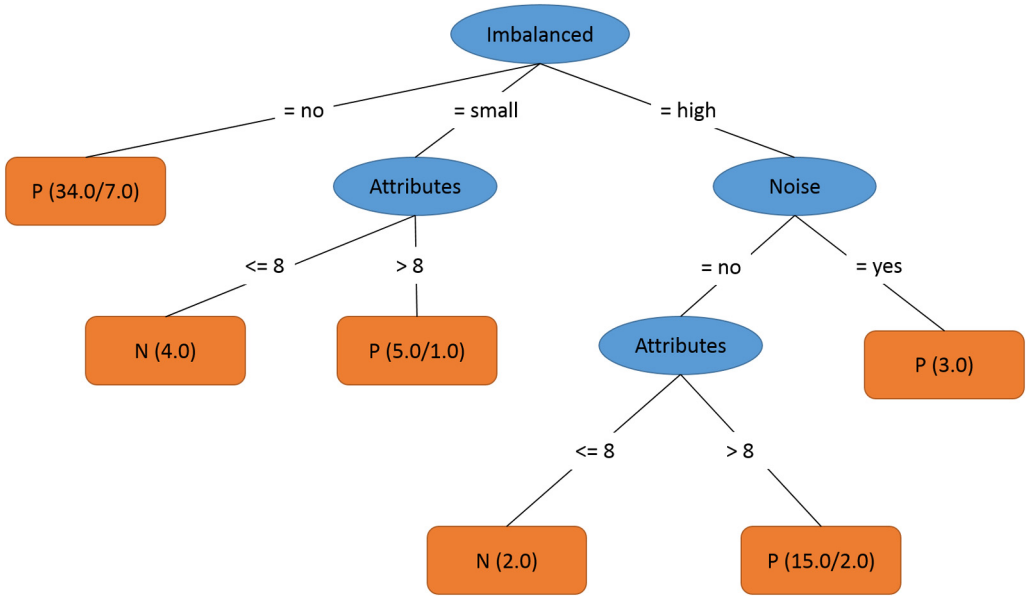

Fig. 8. Decision tree characterizing the ensemble performance.

degrees can be achieved by using a different similarity threshold for each base classifier. On the other hand, we perform instance bagging to reduce the correlation coming from the rule $\xi_i \leq \xi_j \Longrightarrow \bar{R}_{(\xi_j)}(x) \subseteq \bar{R}_{(\xi_i)}(x)$.

We conducted an extensive empirical analysis to assess the prediction capability of the proposed classifier. The results have corroborated that the ensemble model performing instance bagging leads to higher prediction rates regardless of the underlying dissimilarity functional. In spite of this fact, the HVDM function often reports improved prediction rates.

From the comparison between the best-performing ensemble model and 15 state-of-the-art classifiers, we can confirm the reliability of our proposal in solving pattern classification problems. The statistical analysis concluded that our granular classifier is capable of outperforming most ML algorithms while remaining competitive w.r.t. RF, MLP, SL and J48.

On the other hand, there is no doubt about the statistical superiority of RCEs when compared to other well-known instance-based learners. Aiming at explaining this behavior, we derived a rule set from a decision tree using the *k*NN learner as baseline classifier. The obtained rules have shown that the proposed algorithm performs well in presence of imbalanced classes as long as the problem comprises a reasonable number of attributes. In spite of these findings, the future research will be focused on exploring other appealing theoretical properties behind this novel rough classifier.

Acknowledgements

This work was supported by the Research Council of Hasselt University. The authors would like to thank the anonymous reviewers for their constructive remarks throughout the revision process.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd edition, John Wiley & Sons, 2012.
[2] Y. Sun, A.K. Wong, M.S. Kamel, Classification of imbalanced data: a review, Int. J. Pattern Recognit. Artif. Intell. 23 (04) (2009) 687–719.
[3] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141.
[4] B. Frénay, M. Verleysen, Classification in the presence of label noise: a survey, IEEE Trans. Neural Netw. Learn. Syst. 25 (5) (2014) 845–869.
[5] I. Cohen, F.G. Cozman, N. Sebe, M.C. Cirelo, T.S. Huang, Semisupervised learning of classifiers: theory, algorithms, and their application to human-computer interaction, IEEE Trans. Pattern Anal. Mach. Intell. 26 (12) (2004) 1553–1566.
[6] G. Tsoumakas, I. Katakis, Multi-label classification: an overview, Int. J. Data Warehous. Min. 3 (3) (2007) 1–13.
[7] W. Cheng, E. Hüllermeier, K.J. Dembczynski, Graded multilabel classification: the ordinal case, in: Proceedings of the 27th International Conference on Machine Learning, ICML-10, 2010, pp. 223–230.
[8] G. Nápoles, R. Falcon, E. Papageorgiou, R. Bello, K. Vanhoof, Partitive granular cognitive maps to graded multilabel classification, in: 2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE Computational Intelligence Society, 2016.
[9] S.B. Kotsiantis, Decision trees: a recent overview, Artif. Intell. Rev. 39 (4) (2013) 261–283.
[10] G.P. Zhang, Neural networks for classification: a survey, IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev. 30 (4) (2000) 451–462.
[11] H. Ishibuchi, T. Nakashima, T. Morisawa, Voting in fuzzy rule-based systems for pattern classification problems, Fuzzy Sets Syst. 103 (2) (1999) 223–238.
[12] N. Friedman, D. Geiger, M. Goldszmidt, Bayesian network classifiers, Mach. Learn. 29 (2–3) (1997) 131–163.
[13] M.A. Hearst, S.T. Dumais, E. Osman, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.
[14] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[15] T.G. Dietterich, Ensemble methods in machine learning, in: Multiple Classifier Systems, Springer, 2000, pp. 1–15.

[16] Y. Ren, L. Zhang, P. Suganthan, Ensemble classification and regression – recent developments, applications and future directions, IEEE Comput. Intell. Mag. 11 (1) (2016) 41–53.
[17] W. Fan, S.J. Stolfo, J. Zhang, The application of AdaBoost for distributed, scalable and on-line learning, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 1999, pp. 362–366.
[18] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[19] Y. Zhao, J. Gao, X. Yang, A survey of neural network ensembles, in: International Conference on Neural Networks and Brain, 2005, vol. 1, IEEE, 2005, pp. 438–442.
[20] P. Yang, Y. Hwa Yang, B.B. Zhou, A.Y. Zomaya, A review of ensemble methods in bioinformatics, Current Bioinformatics 5 (4) (2010) 296–308.
[21] S.K. Pal, L. Polkowski, Rough-Neural Computing: Techniques for Computing with Words, Springer-Verlag, Berlin, Heidelberg, 2004.
[22] L. Polkowski, P. Artiemjew, On granular rough computing: factoring classifiers through granulated decision systems, in: Rough Sets and Intelligent Systems Paradigms, Springer, 2007, pp. 280–289.
[23] W. Pedrycz, B.-J. Park, S.-K. Oh, The design of granular classifiers: a study in the synergy of interval calculus and fuzzy sets in pattern recognition, Pattern Recognit. 41 (12) (2008) 3720–3735.
[24] R. Al-Hmouz, W. Pedrycz, A. Balamash, A. Morfeq, From data to granular data and granular classifiers, in: 2014 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE, 2014, pp. 432–438.
[25] R. Al-Hmouz, W. Pedrycz, A. Balamash, A. Morfeq, Description and classification of granular time series, Soft Comput. 19 (4) (2015) 1003–1017.
[26] M. Szczuka, A. Jankowski, A. Skowron, D. Slezak, Building granular systems – from concepts to applications, in: Y. Yao, Q. Hu, H. Yu, J.W. Grzymala-Busse (Eds.), Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, in: Lecture Notes in Computer Science, vol. 9437, Springer International Publishing, 2015, pp. 245–255.
[27] A. Bargiela, W. Pedrycz, Granular Computing: An Introduction, The Springer International Series in Engineering and Computer Science, vol. 717, Springer Science & Business Media, 2012.
[28] L. Polkowski, P. Artiemjew, Granular Computing in Decision Approximation – An Application of Rough Mereology, Intelligent Systems Reference Library, vol. 77, Springer, 2015.
[29] W. Pedrycz, Granular Computing: Analysis and Design of Intelligent Systems, CRC Press, 2013.
[30] G. Nápoles, I. Grau, E. Papageorgiou, R. Bello, K. Vanhoof, Rough cognitive networks, Knowl.-Based Syst. 91 (2016) 46–61.
[31] B. Kosko, Fuzzy cognitive maps, Int. J. Man-Mach. Stud. 24 (1) (1986) 65–75.
[32] Z. Pawlak, Rough sets, Int. J. Comput. Inf. Sci. 11 (5) (1982) 341–356.
[33] G. Nápoles, I. Grau, K. Vanhoof, R. Bello, Hybrid model based on rough sets theory and fuzzy cognitive maps for decision-making, in: Rough Sets and Intelligent Systems Paradigms, Springer, 2014, pp. 169–178.
[34] G. Nápoles, I. Grau, R. Falcon, R. Bello, K. Vanhoof, A granular intrusion detection system using rough cognitive networks, in: R. Abielmona, R. Falcon, N. Zincir-Heywood, H. Abbass (Eds.), Recent Advances in Computational Intelligence in Defense and Security, Springer Verlag, 2015, Ch. 7.
[35] R. Bello, R. Falcon, W. Pedrycz, J. Kacprzyk, Granular Computing: at the Junction of Rough Sets and Fuzzy Sets, Springer Verlag, Berlin–Heidelberg, Germany, 2008.
[36] A. Abraham, R. Falcon, R. Bello, Rough Set Theory: a True Landmark in Data Analysis, Springer Verlag, Berlin–Heidelberg, Germany, 2009.
[37] Y. Yao, Three-way decision: an interpretation of rules in rough set theory, in: Rough Sets and Knowledge Technology, Springer, 2009, pp. 642–649.
[38] Y. Yao, The superiority of three-way decisions in probabilistic rough set models, Inf. Sci. 181 (6) (2011) 1080–1096.
[39] Y. Yao, Three-way decisions with probabilistic rough sets, Inf. Sci. 180 (3) (2010) 341–353.
[40] B. Kosko, Hidden patterns in combined and adaptive knowledge networks, Int. J. Approx. Reason. 2 (4) (1988) 377–393.
[41] S. Bueno, J.L. Salmeron, Benchmarking main activation functions in fuzzy cognitive maps, Expert Syst. Appl. 36 (3) (2009) 5221–5229.
[42] W. Pedrycz, W. Homenda, From fuzzy cognitive maps to granular cognitive maps, IEEE Trans. Fuzzy Syst. 22 (4) (2014) 859–869.
[43] W. Pedrycz, The design of cognitive maps: a study in synergy of granular computing and evolutionary optimization, Expert Syst. Appl. 37 (10) (2010) 7288–7294.
[44] W. Pedrycz, A. Jastrzebska, W. Homenda, Design of fuzzy cognitive maps for modeling time series, IEEE Trans. Fuzzy Syst. 24 (1) (2016) 120–130.
[45] J.C. Bezdek, R. Ehrlich, W. Full, FCM: the fuzzy c-means clustering algorithm, Comput. Geosci. 10 (2) (1984) 191–203.
[46] W. Homenda, A. Jastrzebska, W. Pedrycz, Granular cognitive maps reconstruction, in: 2014 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE, 2014, pp. 2572–2579.
[47] R. Bryll, R. Gutierrez-Osuna, F. Quek, Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets, Pattern Recognit. 36 (6) (2003) 1291–1302.
[48] K. Tumer, J. Ghosh, Classifier combining: analytical results and implications, in: Working Notes from the Workshop 'Integrating Multiple Learned Models', 13th National Conference on Artificial Intelligence, CiteSeer, 1996.
[49] K. Turner, N.C. Oza, Decimated input ensembles for improved generalization, in: International Joint Conference on Neural Networks, 1999, vol. 5, IJCNN'99, IEEE, 1999, pp. 3069–3074.
[50] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[51] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Elsevier, 2011.
[52] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, J. Artif. Intell. Res. 6 (1997) 1–34.
[53] J. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[54] J. Alcalá, 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, J. Mult.-Valued Log. Soft Comput. 17 (255–287) (2010) 11.
[55] M. Lichman, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2013.
[56] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[57] I. Triguero, S. García, F. Herrera, Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study, Knowl. Inf. Syst. 42 (2) (2015) 245–284.
[58] R. Kohavi, The power of decision tables, in: Machine Learning: ECML-95, Springer, 1995, pp. 174–189.
[59] G.H. John, P. Langley, Estimating continuous distributions in Bayesian classifiers, in: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.
[60] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Neural Comput. 13 (3) (2001) 637–649.
[61] R. Hecht-Nielsen, Theory of the backpropagation neural network, in: International Joint Conference on Neural Networks, 1989, IJCNN, IEEE, 1989, pp. 593–605.
[62] M. Sumner, E. Frank, M. Hall, Speeding up logistic model tree induction, in: Knowledge Discovery in Databases: PKDD 2005, Springer, 2005, pp. 675–683.
[63] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kauffman Publishers, 1993.
[64] J. Su, H. Zhang, A fast decision tree learning algorithm, in: Proceedings of the 21st National Conference on Artificial Intelligence, Vol. 1, AAAI'06, AAAI Press, 2006, pp. 500–505.

[65] H. Shi, Best-First Decision Tree Learning, Ph.D. thesis, CiteSeer, 2007.
[66] N. Landwehr, M. Hall, E. Frank, Logistic model trees, Mach. Learn. 59 (1–2) (2005) 161–205.
[67] Y. Amit, D. Geman, Shape quantization and recognition with randomized trees, Neural Comput. 9 (7) (1997) 1545–1588.
[68] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.
[69] J.G. Cleary, L.E. Trigg, et al., K*: an instance-based learner using an entropic distance measure, in: Proceedings of the 12th International Conference on Machine Learning, vol. 5, 1995, pp. 108–114.
[70] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning for control, in: Lazy Learning, Springer, 1997, pp. 75–113.
[71] N.C. Smeeton, Early history of the kappa statistic, Biometrics 41 (1985) 795.
[72] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701.
[73] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (1945) 80–93.
[74] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks?, J. Mach. Learn. Res. 17 (2016) 1–10.
[75] C. Lemke, M. Budka, B. Gabrys, Metalearning: a survey of trends and technologies, Artif. Intell. Rev. 44 (1) (2015) 117–130.