UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

Masterthesis

Fast k-Fuzzy Rough Cognitive Networks

Wouter Goossens
Quinten Moesen

Scriptie ingediend tot het behalen van de graad van master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

PROMOTOR :
dr. Gonzalo NAPOLES RUIZ

2018
2019

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Faculteit Bedrijfseconomische Wetenschappen

master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

Masterthesis

Fast k-Fuzzy Rough Cognitive Networks

Wouter Goossens
Quinten Moesen

Scriptie ingediend tot het behalen van de graad van master in de toegepaste economische wetenschappen: handelsingenieur in de beleidsinformatica

PROMOTOR :

dr. Gonzalo NAPOLES RUIZ

Fast k-Fuzzy-Rough Cognitive Networks

Wouter Goossens, Quinten Moesen, Carlos Mosquera, Gonzalo Nápoles

***Abstract*—Fuzzy-Rough Cognitive Networks (FRCNs) are neural networks that utilize rough information granules with soft boundaries to perform the classification process. Unlike other neuronal systems, FRCNs are lazy learners in the sense that we can build the whole model while classifying a new instance. This is possible because the weight matrix connecting the neurons is prescriptively programmed. Similar to other lazy learners, the processing time of FRCN notably increases with the number of instances in the training set, while their performance deteriorates in noisy environments. Aiming at coping with these issues, this paper presents a new FRCN-based algorithm termed Fast k-Fuzzy-Rough Cognitive Network. This variant implements a parallel approach of building the information granules as computed by k-fuzzy-rough sets. Numerical simulations on 35 classification datasets show a notable reduction in the processing time of the algorithm, while delivering competitive results when compared to other lazy learners in noisy environments.**

***Index Terms*—parallel granulation, noise, fuzzy-rough sets, granular cognitive mapping, lazy learners.**

I. Introduction

The continuously growing amount of raw data in our world has been challenging researchers to come up with better ways of building inference models able to extract useful insights from historical data. Pattern classification is one of the most researched topics in the field of machine learning. The classification problem consists of identifying the correct label (decision class) for an unseen object based on the available data [1]. A wide variety of models has been proposed to solve problems of these kinds. While often managing to deliver excellent results, many models lack the ability to explain how an object is exactly classified. That is why traditional neural networks are said to behave like *black boxes*. On the other hand, granular networks try to overcome this shortcoming by processing information granules instead of the traditional numeric representation of a classification problem [2]. The models discussed in this paper construct these information granules using either fuzzy sets [3], rough sets [4] or fuzzy-rough sets [5]. This granulation procedure allows representing the problem domain in a more concise manner. Yet granular classifiers are able to obtain competitive results when compared to traditional neural networks [6].

As an example of these granular systems, Nápoles et al. [7] introduced the *Rough Cognitive Networks* as a hybridization between rough sets [4] and fuzzy cognitive maps [8]. This lazy learner constructs a cognitive network by following a predefined set of rules that use the positive, boundary and negative rough regions for each decision class. One may notice that only the number of decision classes can enlarge the topology of the model, the amount of attributes does not have an influence on the size of the network. Additionally, the use of rough sets allows the model to cope better with uncertainty arising from inconsistencies. Despite all the advantages of this granular classifier, it remains to be sensitive to the value of the similarity threshold parameter. Therefore, Nápoles et al. [9] proposed an RCN-based multi-classifier system that does not require an explicit value for the similarity threshold parameter but rather induces different granularity degrees by using a variety of parameter values. However, the algorithm still constructs hard boundaries for its regions. The introduction of the *Fuzzy-Rough Cognitive Networks* (FRCNs) completely removed the need for an explicit parameter value [10]. This classifier constructs information granules with soft boundaries using fuzzy-rough sets.

In real-world problems, most datasets usually contain noisy instances. This causes the prediction rates of the FRCNs to degenerate as they utilize the nearest neighbors when computing the lower and upper approximations. If the nearest neighbor is a mislabeled sample, the values of fuzzy lower and upper approximations may be completely contaminated [11]. On the other hand, the information granulation of the FRCN model may become quite extensive when the number of instances increases. This happens due to the fact that for a dataset comprised of N instances, the algorithm has to execute the distance function $N \times (N-1)/2$ times. Consequently, the processing time drastically increases when more instances are added to the dataset.

The two main contributions of this paper attempt to overcome the previously described problems. Firstly, we attempt to reduce the noise effects on the performance of the algorithm by using a k-distance function instead of using sensitive operators as the *infimum* and *supremum*. Secondly, a parallel granulation approach is introduced to reduce the processing time of deriving the fuzzy-rough regions. In that regard, our parallel solution will focus on the computation of the similarity matrix and the fuzzy-rough regions attached to each decision class. Numerical simulations show the superiority of the new variant with respect to the the original FRCN algorithm and other lazy learners reported in the literature.

The structure of this paper is as follows. In Section II we introduce the reader to the fundamentals of fuzzy-rough set theory. The FRCN model will be outlined in Section III, whereas Section IV will elaborate on the two main contributions of this paper to the FRCN model. These contributions will be assessed by extensive experiments in Section V. The final remarks will be provided in Section VI.

G. Nápoles, W. Goossens and Q. Moesen are with the Faculty of Business Economics, Universiteit Hasselt, Belgium.
E-mail: gonzalo.napoles@uhasselt.be

C. Mosquera is with the Artificial Intelligence Lab, Vrije Universiteit Brussel, Belgium.

II. Preliminaries on Fuzzy-Rough Sets

Fuzzy-rough sets can be seen as a blend of fuzzy set and rough set theory that incorporates strengths of both theories. In rough set theory, objects are categorized using hard boundaries and traditional membership functions. This means that an object either belongs to an approximation, or it does not. The incorporation of fuzzy set theory allows us to create granules with soft boundaries. In fuzzy-rough set theory, an object can have a membership degree to each fuzzy-rough region. This allows us to replace the abrupt transitions between classes with more gradual ones. The FRCN classifier introduced by Nápoles et al. [12] uses the fuzzy-rough approach proposed by Inuiguchi et al. [5].

Let U denote the universe of discourse which incorporates all objects in the training dataset. We can define X_c as a partition of U which contains all elements classified with decision class D_c. The membership degree of $x \in U$ to a subset X_c is given by the following equation,

$$\mu_{X_c}(x) = \begin{cases} 1 & , x \in X_c \\ 0 & , x \notin X_c \end{cases}. \tag{1}$$

Additionally, we need to formulate a membership function representing the similarity between two instances x and y. This function, denoted by $\mu_P(y,x)$, can be constructed by combining the previously described membership degree $\mu_{X_c}(x)$ with a similarity degree $\varphi(x,y)$. This similarity degree denotes the complement of the normalized distance between two instances x and y. The FRCN classifier uses either the Heterogeneous Euclidean-Overlap Metric (HEOM) or the Heterogeneous Manhattan-Overlap Metric (HMOM) to compute the similarity degree [9]. The membership function can then be formalised as follows:

$$\mu_P(y,x) = \mu_{X_c}(x)\varphi(x,y) = \mu_{X_c}(x)(1-\delta(x,y)). \tag{2}$$

Using Equation 1 and 2, we can construct the membership function describing the lower approximations associated with the fuzzy set X_c. We use an implication function $\mathcal{I}$ so that $\mathcal{I}(0,0) = \mathcal{I}(0,1) = \mathcal{I}(1,1) = 0$ and $\mathcal{I}(1,0) = 1$,

$$\mu_{P_*(X_c)}(x) = \min\left\{\mu_{X_c}(x), \inf_{y \in \mathcal{U}} \mathcal{I}(\mu_P(y,x), \mu_{X_c}(y))\right\}. \tag{3}$$

Similarly, we can define the membership function for the upper approximation. To do that, we use a conjunction function $\mathcal{T}$ such that $\mathcal{T}(0,0) = \mathcal{T}(0,1) = \mathcal{T}(1,0) = 0$ and $\mathcal{T}(1,1) = 1$. Equation 4 shows this membership function for the upper approximation,

$$\mu_{P^*(X_c)}(x) = \max\left\{\mu_{X_c}(x), \sup_{y \in \mathcal{U}} \mathcal{T}_1(\mu_P(x,y), \mu_{X_c}(y))\right\}. \tag{4}$$

The lower and upper fuzzy-rough approximations are the main building-blocks of the FRCN classifier. In the next section, we will explain how to derive the network structure in a prescriptive way, without the need for an explicit learning process to compute the weight set.

III. Fuzzy-rough Cognitive Networks

When constructing an FRCN classification model, we first need to granulate the information space, so that a fuzzy-rough attribute space is created. We do this by categorizing objects into granules with soft boundaries. This translates into computing the positive and negative fuzzy-rough regions by using Equations 3 and 4, respectively,

$$\mu_{POS(X_c)}(x) = \mu_{P_*(X_c)}(x) \tag{5}$$

$$\mu_{NEG(X_c)}(x) = 1 - \mu_{P_*(X_c)}(x). \tag{6}$$

Algorithm 1 depicts the granulation process, where $D(x)$ returns the decision class attached to x.

Algorithm 1 Fuzzy-rough information granulation

```
for each $x \in U$ do
    if $D(x) = D_c$ then
        $X_c \leftarrow X_c \cup \{x\}$
    end if
    Compute $\mu_{X_c}(x)$ according to Equation 1
end for
for each $x \in U$ do
    for each subset $X_c$ do
        Compute $\mu_{POS(X_c)}(x)$ according to Equation 5
        Compute $\mu_{NEG(X_c)}(x)$ according to Equation 6
    end for
end for
```

After computing all fuzzy-rough regions, we build a neural network comprised of $2|D|$ input neurons, $|D|$ output neurons and $|D|(4+|D|)$ weights. Here, D represents the set of all decision classes. This construction procedure is summarized in Algorithm 2. Figure 1 displays the FRCN model for a classification problem with only two decision classes.

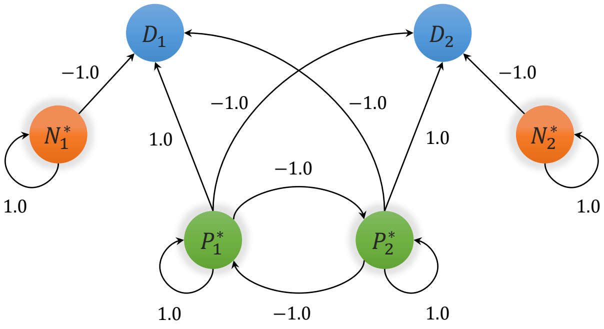

Fig. 1. Fuzzy-Rough Cognitive Network for binary classification.

In the FRCN algorithm, neural concept C_i denotes either a positive region (P_c^*), a negative region (N_c^*) or an output neuron (D_c). The weights connecting the neurons are assigned based on the following construction rules:

- R_1: If $C_i = P_c^*$ AND $C_j = D_c$ then $w_{ij} = 1.0$
- R_2: If $C_i = N_c^*$ AND $C_j = D_c$ then $w_{ij} = -1.0$
- R_3: If $C_i = P_c^*$ AND $C_j = D_{v \neq c}$ then $w_{ij} = -1.0$
- R_4: If $C_i = P_c^*$ AND $C_j = P_{v \neq c}$ then $w_{ij} = -1.0$

Algorithm 2 Network construction procedure

```
for each subset $X_c$ do
    Add a neuron $P_c^*$ as the $c$th positive region
    Add a neuron $N_c^*$ as the $c$th positive region
end for
for each decision class $D_c$ do
    Add a neuron $D_c$ as the $c$th decision class
end for
for each neuron $C_i$ do
    for each neuron $C_j$ do
        Configure $w_{ij}$ according to rules $R_1$-$R_4$
    end for
end for
```

After the network has been constructed from the information granules, the model can classify new objects via its reasoning mechanism. The initial activation value of an input neuron is computed based on the similarity degree between the new object y and $x \in U$, and the membership degree of x to the fuzzy-rough granular region. If $C_i = P_c^*$, then the activation rule can be formalized as:

$$A_i^{(0)} = \frac{\sum_{x \in \mathcal{U}} \mathcal{T}_2(\varphi(x,y), \mu_{POS(X_c)}(x))}{\sum_{x \in \mathcal{U}} \mu_{POS(X_c)}(x)} \tag{7}$$

where $\mathcal{T}_2$ denotes a conjunction function that may be different than the one used in Equation 4. If $C_i = N_c^*$, then the neuron is activated using the following rule:

$$A_i^{(0)} = \frac{\sum_{x \in \mathcal{U}} \mathcal{T}_2(\varphi(x,y), \mu_{NEG(X_c)}(x))}{\sum_{x \in \mathcal{U}} \mu_{NEG(X_c)}(x)}. \tag{8}$$

These activation values are computed for every decision class D_c as stated in Algorithm 3.

The network will now perform its neural reasoning process. The activation value of a neuron is computed using Equation 9 where M is the vector of incoming edges to the ith neuron. This equation corresponds to Kosko's activation function used in fuzzy cognitive maps [13],

$$A_i^{(t+1)} = f(\sum_{j=1}^{M} w_{ji} A_j^{(t)}) \tag{9}$$

where $A_i^{(t)}$ denotes the activation value of the ith neural processing entity on iteration t, whereas w_{ij} represents the causal weight connecting C_i and C_j.

Algorithm 3 Network activation procedure

```
for each decision class $D_c$ do
    Compute $A_x^{(0)}(P_c^*)$ according to Equation (7)
    Compute $A_x^{(0)}(N_c^*)$ according to Equation (8)
end for
```

The reasoning process continues until either a fixed point attractor, or a fixed number of iterations is reached. A fixed point attractor is reached when $A_i^{(t+1)} = A_i^{(t)}$ applies for each neuron i in the network. This reasoning mechanism is outlined in Algorithm 4.

Algorithm 4 Network reasoning procedure

```
for $t = 0$ to $T$ do
    converged ← True
    for each neuron $C_i$ do
        Compute $A_i^{(t+1)}$ according to Equation 9
        if $A_i^{(t)} \neq A_i^{(t+1)}$ then
            converged ← False
        end if
    end for
    if converged then
        return argmax$_c\{\mathrm{A}_x^{(t+1)}(D_c)\}$
    end if
end for
if not converged then
    return argmax$_c\{\mathrm{A}_x^{(T)}(D_c)\}$
end if
```

The class represented by the output neuron with the highest activation value is assigned to the new object y.

IV. Fast k-Fuzzy-Rough Cognitive Networks

In this section, we introduce the main contributions of this paper which aim at improving the performance of the FRCN model with respect to both prediction power and computational efficiency. Firstly, we present the k-fuzzy-rough set model to cope better with noisy datasets. As a second contribution, we propose a parallel granulation approach when computing the distances between all instances in the training set. This method will allow the k-FRCN algorithm to build its model using multiple threads, thus speeding up its performance.

A. k-Fuzzy-Rough Sets

For a given implication function $\mathcal{I}$ and $\mathcal{T}$-norm, we notice that the membership of an object to the fuzzy lower approximation $\mu_{P_*(X_c)}$ matches with the distance from x to its nearest neighbor from different classes, while the membership to the fuzzy upper approximation $\mu_{P^*(X_c)}$ is the similarity between x and the nearest neighbor in that set X_c.

Equation 10 uses the Kleene-Dienes implicator to compute the lower membership function,

$$\begin{aligned} \mu_{P_*(X_c)}(x) &= \inf_{y \in U} \max\big(1 - \mu_P(y,x), \mu_{X_c}(y)\big) \\ &= \inf_{y \in X_c} \max\big(1 - \mu_P(y,x), 1\big) \wedge \\ &\quad \inf_{y \notin X_c} \max\big(1 - \mu_P(y,x), 0\big) \\ &= 1 \wedge \inf_{y \notin X_c} \big(1 - \mu_P(y,x)\big) \\ &= \inf_{y \notin X_c} \delta(x,y). \end{aligned} \tag{10}$$

Equation 11 adopts the standard $\mathcal{T}$-norm to compute the upper membership function,

$$\begin{aligned}\mu_{P^*(X_c)}(x) &= \sup_{y\in U} \min\big(\mu_P(x,y), \mu_{X_c}(y)\big) \\ &= \sup_{y\in X_c} \min\big(\mu_P(x,y), 1\big) \vee \\ &\quad \sup_{y\notin X_c} \min\big(\mu_P(x,y), 0\big) \\ &= \sup_{y\in X_c} \mu_P(x,y) \vee 0 \\ &= \sup_{y\in X_c} \varphi(x,y). \end{aligned} \tag{11}$$

In k-fuzzy-rough sets, we attempt to reduce the noise effects by using a k-distance function instead of the sensitive inf and sup. Given an object x and a set of objects $Y = \{y_1, y_2, \ldots, y_n\}$, we define a distance function δ_k between x and Y as the distance from x to its k-th nearest neighbor in Y. Therefore, Equations 10 and 11 can be rewritten as follows:

$$\mu_{P_*(X_c)}(x) = \inf_{y\notin X_c} \delta(x,y) = \delta_1(x, U - X_c), \tag{12}$$

$$\mu_{P^*(X_c)}(x) = \sup_{y\in X_c} \varphi(x,y) = 1 - \delta_1(x, X_c). \tag{13}$$

Notice that we get the original fuzzy-rough set formulation if parameter k is set equal to one. On the other hand, we get more flexible definitions for the upper and lower approximations when $k > 1$. We refer to this as k-fuzzy-rough sets. The approximations can then be computed using:

$$\mu_{P_*(X_c)}(x) = \min\left\{\mu_{X_c}(x), \delta_k(x, U - X_c)\right\}, \tag{14}$$

$$\mu_{P^*(X_c)}(x) = \max\left\{\mu_{X_c}(x), 1 - \delta_k(x, X_c)\right\}. \tag{15}$$

It is worth mentioning that Inuiguchi's model [5] does not assume that $\mu_P(x,x) = 1, \forall x \in U$. Instead, we compute the minimum and the maximum when computing the $\mu_{P_*(X_c)}(x)$ and $\mu_{P^*(X_c)}(x)$, respectively. This feature allows preserving the inclusiveness of $P_*(X_c)$ in the fuzzy set X_c and the inclusiveness of X_c in $P^*(X_c)$.

B. Parallel granulation process

The Fast k-FRCN algorithm is a multi-threaded variant of the FRCN model that improves the efficiency of the model in large datasets. The algorithm uses k-fuzzy-rough sets to granulate all information. The features that are implemented in a multi-threaded way are: the computation of the distance matrix, the construction of the fuzzy-rough regions, and the computation of the similarity class for a given object.

The equal distribution of the distance computations induces balanced workloads among all threads. Only the distances below the main diagonal of the distance matrix are computed, as the distance function used by the FRCN model is symmetric. Figure 2 illustrates the computation of the distance matrix by using three processing threads.

The regions associated with each decision class and the similarity classes are computed in the same way. Firstly, the dataset is split into several chunks so that each thread processes a piece of data. Meanwhile, the main process waits until all tasks are completed. Afterwards, all parts are merged.

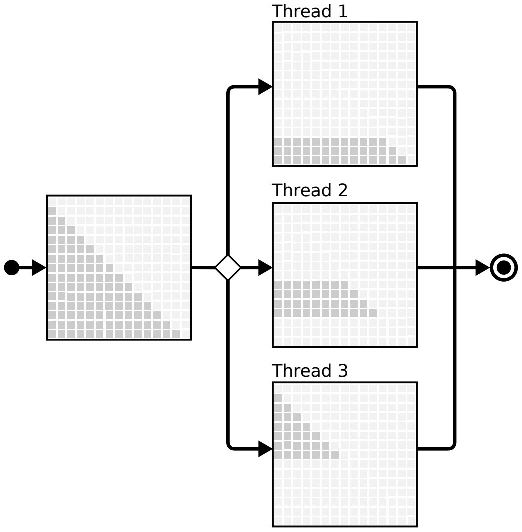

Fig. 2. Process to compute the distance matrix.

V. Numerical simulations

To validate the two contributions of this paper, we set up two experiments. In the first experiment, we set up a standard classification problem context where we compare the Fast-k-FRCN to the k-FRCN implementation. For each of the 35 datasets, 4 levels of noise have been generated (0%, 5%, 10% and 20%). Both algorithms have been trained for each dataset and noise level, resulting in 140 results per algorithm. The results from this first experiment are used to check if the Fast-k-FRCN does benefit from the parallelization of the distance calculation step in terms of processing time.

In our second experiment, the k-FRCN algorithm is compared with 6 state-of-the-art granular classifiers. The first goal of this experiment is to check whether our proposal can deliver competitive results when compared with other lazy learners. Additionally, we validate if the k-fuzzy-rough sets allow the algorithm to cope better with noisy datasets.

A. Proof of concept

Figure 3 displays the membership values for the lower and upper fuzzy-rough approximations associated with the *Setosa* decision class of the well-known *Iris* dataset. Because these approximations are constructed using fuzzy-rough sets, the unambiguous transitions of crisp sets are replaced with a continuous gradient.

However, this continuous gradient may be disrupted when noise is added to the dataset. Figure 4 shows the fuzzy-rough lower and upper approximations for the same decision class, but with 10% artificially added noise. It is noticeable that the transition becomes less smooth. The k-distance function of the k-fuzzy-rough sets tries to overcome this defect. The k-fuzzy-rough approximations, with k equal to 4, of the same decision class are again presented in Figure 5. Here, we can see that the approximations are able to recover from this noise and can construct a softer transition between the membership functions.

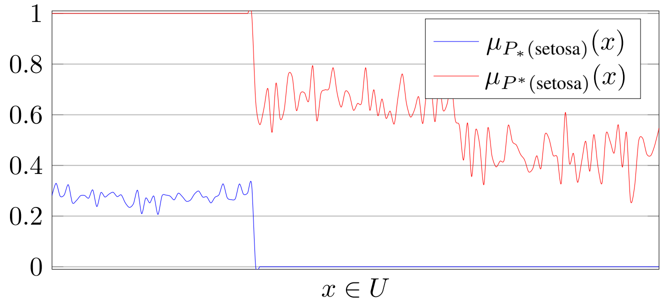

Fig. 3. Iris-setosa lower and upper fuzzy approximations.

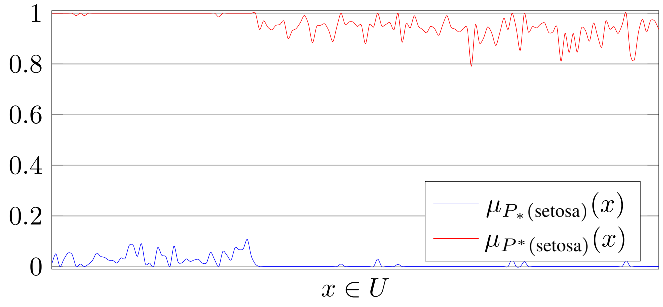

Fig. 4. Iris-setosa lower and upper approximations with 10% noise.

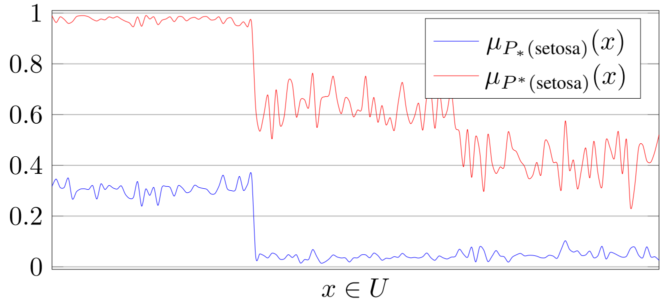

Fig. 5. Iris-setosa lower and upper approximations using 4-FRST with 10% noise.

B. Description of benchmark problems

To conduct our experiment, we selected 35 datasets suitable for pattern classification. The number of instances in these datasets ranges from 1599 to 12960. Using fairly large datasets, we can get a more clear insight in the reduction in time necessary to build the model. The number of attributes ranges from 2 to 240 and the number of classes from 2 to 100. A more extensive description of the datasets can be found in Table I.

All numerical attributes have been normalized and none of the included datasets are considered noisy or have any missing values. The noise mentioned in this paper is artificially added using a noise filter in Weka. This filter replaces the original decision class with a randomly selected decision class. This method is applied to the specified percentage of instances.

TABLE I
DESCRIPTION OF THE DATASETS

Dataset	Instances	Attributes	Classes	Imbalance
abalone	4174	8	28	689:1
banana	5300	2	2	1:1
bank	4521	16	2	8:1
car	1728	6	4	19:1
cardiotocography-10	2126	35	10	11:1
cardiotocography-3	2126	35	3	9:1
chess	3196	36	2	1:1
crowdsourced-mapping	10545	28	6	140:1
csj	2796	34	6	2:1
frogs-mfccs	7195	22	10	51:1
hypothyroid	3772	29	4	1740:1
mfeat-factors	2000	216	10	1:1
mfeat-fourier	2000	76	10	1:1
mfeat-karhunen	2000	64	10	1:1
mfeat-morphological	2000	6	10	1:1
mfeat-pixel	2000	240	10	1:1
mfeat-zernike	2000	47	10	1:1
mushroom	8124	22	2	1:1
musk2	6598	167	2	5:1
nursery	12960	8	5	2160:1
optdigits	5620	64	10	1:1
ozone	2536	72	2	34:1
page-blocks	5473	10	5	175:1
pendigits	10992	16	10	1:1
phoneme	5404	5	2	2:1
plant-margin	1600	64	100	1:1
plant-shape	1600	64	100	1:1
plant-texture	1599	64	100	1:1
segment	2310	19	7	1:1
spambase	4601	57	2	2:1
splice	3190	60	3	2:1
wall-following	5456	24	4	7:1
waveform	5000	40	3	1:1
wine-quality-white	4898	11	7	440:1
winequality-red	1599	11	11	68:1

C. Lazy learners used for comparison

1) IBk [14]: IBk is an implementation of the KNN algorithm that calculates the similarity between instances using a distance function. This algorithm never generates any abstractions of the dataset, but rather compares unseen instances with already classified ones. Therefore, it is called an instance-based algorithm. The similarity between two instances x and y can be computed as:

$$\varphi(x, y) = -\sqrt{\sum_{i=1}^{A} f(x_i, y_i)} \tag{16}$$

where $f(x_i, y_i) = (x_i - y_i)^2$ and i is a numerical attribute and $f(x_i, y_i) = (x_i \neq y_i)$ for nominal attributes. We define A as the set of attributes used to describe the two instances. These attributes are normalized so that all attributes contribute equally to the computed distance. The algorithm classifies an unseen instance based on the majority class of its k most similar instances. To handle noisy datasets, the algorithm keeps a record of the performance of each instance. Only instances that are known to correctly classify other instances are utilized in the classification process. This decision making can be enhanced by assigning a larger weight to neighbors that are more similar to the new instance.

2) KStar (K^*) [15] [16]: this algorithm uses an entropic distance measure instead of a specific distance function such as IBk. This entropic distance can be expressed as the complexity of trying to transform one instance x to another instance y. The K^* distance is then defined as the sum of the probabilities of all possible transformations between two instances. One advantage of this distance measure is that it can take the probability, that two values of a nominal attribute are quite similar, into account.

3) FuzzyRoughNN (FRNN) [17]: to elaborate on FRNN, we first introduce the fuzzy k-nearest neighbor (FNN) algorithm as in [18]. A major drawback of KNN is that every selected nearest neighbor is treated equally important in the classification process. A new way of calculating the membership of an instance to a class was introduced. This was done by defining membership as the function of the distance of the vector from its k-nearest neighbors and the memberships in the possible classes of the neighbor. Let NN be the set of k-nearest neighbors of a certain instance x. For an unlabeled instance the membership to the class can be calculated in the following manner:

$$\mu_C(x) = \sum_{y \in NN} F_{FNN}(y,x)\ \mu_C(y), \tag{17}$$

with

$$F_{FNN}(y,x) = \frac{||x-y||^{-2/(m-1)}}{\sum\limits_{X_i \in NN} ||x - X_i||^{2/(m-1)}}. \tag{18}$$

The m-parameter controls the overall weighting of the similarity. The intended effect of this parameter is that closer neighbors are weighted more heavily than farther ones. This happens when m is one. When m increases, this effect vanishes and neighbors are weighted more evenly. In [18], it is suggested that a parameter value of 2 is used. For every x, the algorithm will calculate the membership $\mu_C(y)$ to a certain decision class D_C. Two methods are described to do this in [18]: a crisp and a fuzzified membership. A crisp membership means that the membership will be 1 if the instance belongs to the class. Otherwise, the membership will be 0. The fuzzified membership proposed in [17] is given by:

$$\mu_C(y) = \begin{cases} 0.51 + 0.49\frac{|NN_C|}{K} & if\ x\ is\ in\ class\ C \\ 0.49\frac{|NN_C|}{K} & otherwise \end{cases} \tag{19}$$

where $|NN_C|$ is the length of all nearest neighbors that belong to decision class D_c and k is the predefined parameter that defines the number of nearest neighbors that are considered. Afterwards the decision class with the highest membership, calculated by Equation 18, is used to classify the unlabeled instance x.

One of the variants of FNN is FRNN. It integrates fuzzy-rough set theory and the FNN algorithm. The FRNN algorithm was proposed in [19]. It uses the k-nearest neighbors of a particular instance x to calculate the upper and lower approximations. Therefore, a $[0,1]$ valued fuzzy tolerance relation is used. This can be seen as another way to describe similarity. The mathematical formula for this relation is as follows:

$$F_{FRNN}(y,x) = \min_{a \in A} F_{a_{FRNN}}(y,x) \tag{20}$$

where A denotes the collection of all attributes of the dataset, while F_{aFRNN} can be calculated as follows:

$$F_{aFRNN}(y,x) = 1 - \frac{|a(y) - a(x)|}{|a_{\max} - a_{\min}|}. \tag{21}$$

To calculate the membership of x to a decision class C we first need approximations for the upper and lower boundaries $F_*C(x)$ and $F^*C(x)$. There are two variants of FRNN: FRNN-FRS and FRNN-VQRS, in this paper referenced as VQNN. The latter will be explained later in this paper. FRNN-FRS uses the traditional $\mathcal{T}$-norm and implicator approximations:

$$F_*C(x) = \min_{y \in NN} \mathcal{I}(F_{FRNN}(y,x), \mu_C(y)), \tag{22}$$

$$F^*C(x) = \max_{y \in NN} \mathcal{T}(F_{FRNN}(y,x), \mu_C(y)). \tag{23}$$

Depending on the variant, these approximations may be different. To get a definitive class for instance x, the algorithm combines the upper and lower approximation by calculating the mean of the two values. When this is done for every class, the class with the highest value is selected for x.

4) VQNN [19]: In fuzzy-rough set theory, *inf* and *sup* (Formula 3 and 4) operators are used to calculate the lower and upper approximations of a given set. These approximations will be the traditional rough set approximations when the set is crisp. This makes the model more susceptible to noise, as $\forall$ and $\exists$ do with a traditional set. This is why vaguely quantified rough sets were brought into existence. Instead of the traditional crisp quantifiers like "all" and "at least one", VQRS makes use of "most" and "some" to decide the lower and upper approximations which are defined by:

$$\mu_{P_*Qs(y)}(x) = Qs\left(\frac{\sum\limits_{y \in NN} \min(F_{FNN}(y,x), \mu_C(y))}{\sum\limits_{y \in NN} F_{FNN}(y,x)}\right),$$

$$\mu_{P^*Qm(y)}(x) = Qm\left(\frac{\sum\limits_{y \in NN} \max(F_{FNN}(y,x), \mu_C(y))}{\sum\limits_{y \in NN} F_{FNN}(y,x)}\right)$$

such that Qs and Qm are fuzzy quantifiers that model the linguistic quantifiers "some" and "most" respectively. A general definition can be formulated, which can be used to generate different fuzzy quantifiers, as follows:

$$Q_{(\alpha,\beta)} = \begin{cases} 0 & , x \leq \alpha \\ \frac{2(x-\alpha)^2}{(\beta-\alpha)^2} & , \alpha \leq x \leq \frac{\alpha+\beta}{2} \\ 1 - \frac{2(x-\alpha)^2}{(\beta-\alpha)^2} & , \frac{\alpha+\beta}{2} \leq x \leq \beta \\ 1 & , \beta \leq x \end{cases} \tag{24}$$

where α and β are parameters that can be used to satisfy a personal definition of "some" and "most".

5) FuzzyOwnershipNN (FONN) [17]: Another possibility to combine FNN and fuzzy-rough sets is FONN. It can handle uncertainty caused by overlapping classes and inadequate knowledge. To do this, the algorithm uses the confidence that an object x can be classified to a class C as membership, defined by:

$$\tau_C(x) = \frac{\sum\limits_{y \in U} F_{FONN}(y,x)\mu_C(y)}{|U|} \tag{25}$$

such that $F_{FONN}(y,x)$ is the fuzzy relation between y and x. This fuzzy relation can be written down as:

$$F_{FONN}(y,x) = \exp\big(-\sum_{a \in A} b_a(a(x) - a(y))^{2/(m-1)}\big) \tag{26}$$

where m again controls the overall weighting, while b_a defines the bandwidth of the membership,

$$b_a = \frac{|U|}{2\sum_{y \in U} ||a(x) - a(y)||^{2/(m-1)}}. \tag{27}$$

Firstly, the bandwidth is calculated for each attribute. Hereafter $\tau_C(x)$ is calculated for every class. The output of the algorithm will be the class with the highest membership value. There is no need for a parameter k in this algorithm, because distant neighbors will not influence the membership that much. Yet, every instance is considered.

6) OWANN [20], [21]: Vaguely quantified rough sets, as proposed in [22], and traditional models still show some crisp behavior or require parameter tuning. Therefore, ordered weighted average (OWA) fuzzy-rough sets were introduced in [23], using OWA operators as proposed in [24]. An OWA operator of dimension n can be defined as a mapping: $R^n \to R$ with an associated weight vector $w = \langle w_1, w_2, ..., w_n \rangle$. This vector has to fulfill two constraints:

$$\sum_{k=1}^{n} w_k = 1, \tag{28}$$

$$\forall k \in 1, 2, \ldots, n : w_k \in [0, 1]. \tag{29}$$

A very suitable way to replace the strict minimum and maximum operators of fuzzy-rough lower and upper approximations is the OWA aggregator. The purpose here is to calculate an OWA aggregation of a decreasingly ordered vector V of m scalar values, which can be formulated as:

$$OWA_w(V) = \sum_{i=1}^{k} w_i c_i \tag{30}$$

In this manner we can formulate the new upper and lower approximations of a fuzzy rough set as follows:

$$F_*C(x) = OWA_{w_*}\mathcal{I}\big(F_{FRNN}(y,x),\ \mu_C(y)\big),$$
$$F^*C(x) = OWA_{w^*}\mathcal{T}\big(F_{FRNN}(y,x),\ \mu_C(y)\big).$$

Similar to FRNN, the mean of the upper and lower approximations values determines the decision class that is selected for an unlabeled instance x.

D. Settings and Hyperparameter Optimization

All classification models built in this paper have been subjected to hyperparameter tuning to optimize the performance of each model for each particular dataset. A grid search approach has been used to find the optimal hyperparameter settings for each model build. This method is based on the approach of Nápoles et al. [25], but replaces the lowest average error with the highest average accuracy.

The procedure starts by splitting the dataset into 5 folds. It then builds a model for each hyperparameter configuration to detect the best hyperparameter settings. This model building procedure builds 5 different models each time keeping aside one fold. That fold is further split into the validation and test set. These are later used to calculate the validation and test accuracy for each model build. After building all 5 models for every hyperparameter configuration, the average validation and test accuracy is computed. These results are stored until all hyperparameter configurations have been built. The model resulting from the hyperparameter configuration with the highest validation accuracy will be used as the best performing model. The pseudo-code for this procedure can be found in Algorithm 5.

All previously described algorithms, that use a k parameter, have been optimized with k values ranging from 1 to 10. The lower α of the *OWANN* and *VQNN* classifiers has been trained with values 0.1 and 0.2, the upper α with 0.2 and 0.3. Their β parameters also have undergone hyperparameter tuning. We have used 0.6 and 0.8 for the lower β, and 0.8 and 1 for the upper β. The values for the fuzzifier parameter of FONN ranged from 1 to 10. *KStar* was optimized by using values from 5 to 40 in steps of 5 for the global blending option.

Algorithm 5 Hyperparameter tuning procedure

```
Randomly split dataset into 5 folds
for each parameter configuration $cf$: do
    for each fold $pt$: do
        Train model on remaining folds (not $pt$)
        Split fold $f$ in validation and test set
        Calculate validation accuracy
        Calculate test accuracy
    end for
    Calculate the average validation accuracy
    Calculate the average test accuracy
end for
Determine which configuration $bs$ reported the highest
    average validation accuracy
return average test accuracy with configuration $bs$
```

E. Exploring Fast k-FRCN's Processing Time

To assess the added value of the parallel implementation on algorithm's performance, we compare the time required to build the model with thread levels ranging from 1 to 10. If the thread level is set equal to 1, the classifier behaves exactly the same in terms of processing time as the original FRCN classifier. The average time to build the model per thread level is computed based the results of the best performing hyperparameter configuration for each dataset. Figure 6 shows the average time to build the model for each thread level.

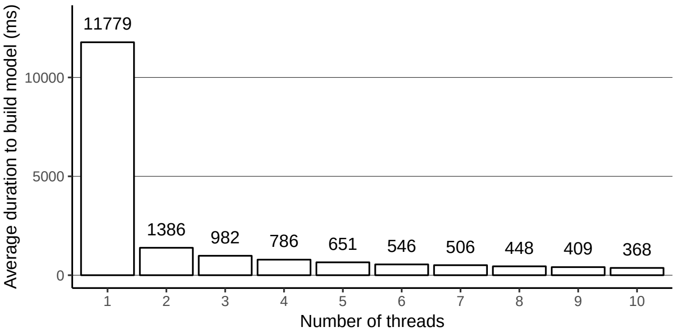

Fig. 6. Average duration in milliseconds to build model per thread level.

The time to build the classification model significantly decreases when the algorithm has access to at least two threads instead of one. When using two threads, the building time of the model is on average reduced by 88% compared to one thread. Adding more threads only marginally reduces the processing time. Moving from two to three threads, the time gain is 30%.

F. Exploring Fast k-FRCNs' Performance

In this section, we compare the performance of the k-FRCN model against 6 granular classifiers and the original FRCN algorithm. This performance comparison experiment has been conducted by using four different noise settings, that is, 0%, 5%, 10% and 20%.

Throughout this experiment, we use the kappa statistic as an accuracy measurement [26]. This statistic is a more robust measure because it takes the chance that an instance is correctly classified by chance into account.

Table II shows the average kappa per level of noise for each algorithm used in this experiment. The reader may notice that the performance of all algorithms degenerates as more noise is added to the problem. This is self-evident because it is not obvious for the algorithms how to distinguish an incorrectly labeled instance from a correctly labeled one. The performances of the algorithms over the 35 pattern classification datasets are visualized in Figures 7, 8, 9 and 10.

TABLE II
AVERAGE KAPPA PER LEVEL OF NOISE

Algorithm	0% noise	5% noise	10% noise	20% noise
k-FRCN	0.79	0.70	0.62	0.50
FRCN	0.77	0.68	0.59	0.45
FONN	0.78	0.71	0.65	0.56
FRNN	0.71	0.59	0.49	0.34
IBk	0.75	0.62	0.52	0.37
KStar	0.74	0.64	0.55	0.41
OWANN	0.75	0.65	0.58	0.46
VQNN	0.75	0.67	0.60	0.47

In order to determine whether the performances of all algorithms are statistically different, a Friedman test [27] has been conducted for each level of noise. This non-parametric test can be used to reject the null hypothesis if there is a significant difference among the algorithms [28]. Thus, the null hypothesis will be rejected when at least two classifiers are performing significantly different. The p-values for these tests are respectively $2.25E$-5, $2.14E$-18, $1.60E$-21 and $2.65E$-25. All p-values are smaller than the 5% significance level and therefore indicating a difference in performance between at least two algorithms in each noise environment. The Friedman ranks of all noise environments are shown in Figure 11, 12, 13 and 14. We can see that FONN is the best performing algorithm with k-FRCN following as second. The ranks of FONN and k-FRCN are the only ones that improve when more noise is added to the datasets. On the other hand, moving from 10% noise to 20% noise does not have an influence on the Friedman order of the algorithms.

Next, we perform a post-hoc analysis to test for significant differences between the proposed k-FRCN and the 7 other algorithms in terms of accuracy. This pairwise comparison is conducted using a Wilcoxon signed-rank test. This test will reject the null hypothesis when there is a significant difference between the two selected algorithms. Corrected values using Holm are also given for the p-values of the pairwise tests. This method will try to control for type I errors (false positive). Tables III, IV, V and VI show the results of the pairwise comparisons for each level of noise.

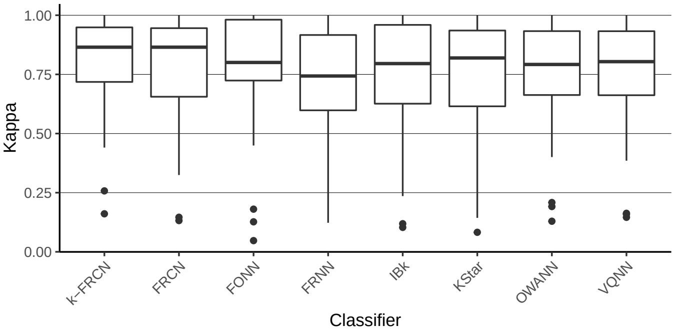

Fig. 7. Accuracy per classifier over 35 datasets without artificial noise.

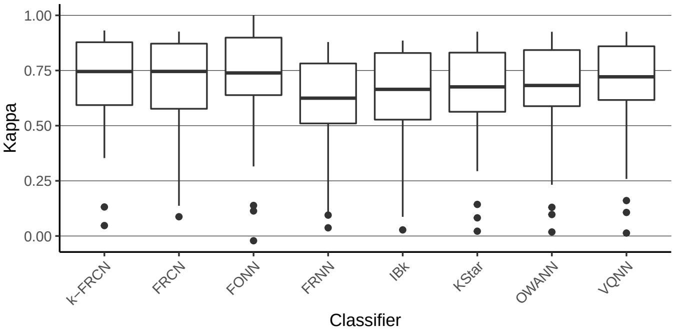

Fig. 8. Accuracy per classifier over 35 datasets with 5% artificial noise.

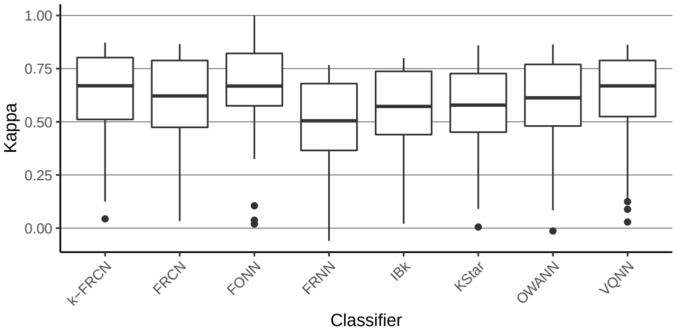

Fig. 9. Accuracy per classifier over 35 datasets with 10% artificial noise.

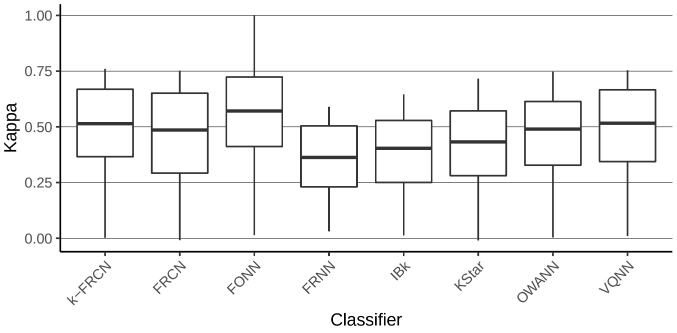

Fig. 10. Accuracy per classifier over 35 datasets with 20% artificial noise.

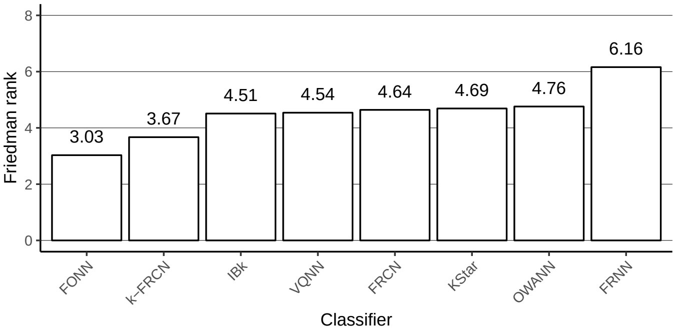

Fig. 11. Friedman rank for each classifier without artificial noise.

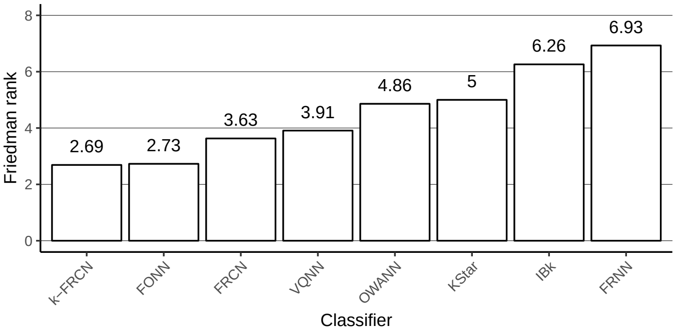

Fig. 12. Friedman rank for each classifier with 5% artificial noise.

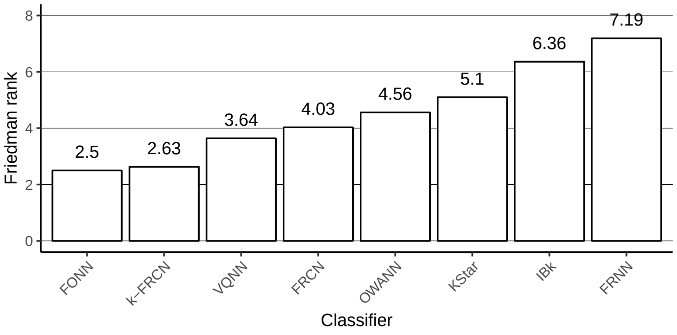

Fig. 13. Friedman rank for each classifier with 10% artificial noise.

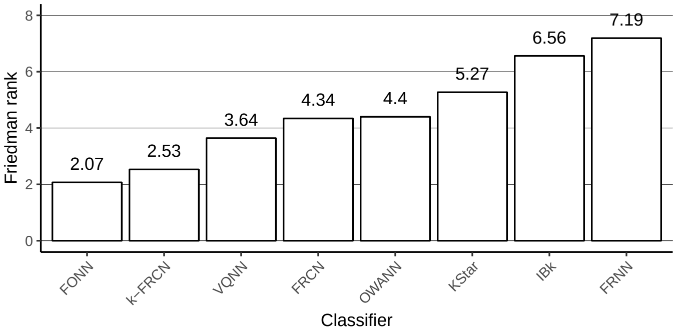

Fig. 14. Friedman rank for each classifier with 20% artificial noise.

TABLE III
WILCOXON SIGNED-RANK TEST (0% NOISE)

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	6	25	4	$2.57E$-4	$1.80E$-3
FONN	21	12	2	$8.58E$-1	$8.58E$-1
FRNN	9	25	1	$3.76E$-4	$2.26E$-3
IBk	14	19	2	$5.82E$-2	$1.16E$-1
KStar	12	22	1	$1.83E$-2	$7.32E$-2
OWANN	12	22	1	$1.32E$-2	$6.59E$-2
VQNN	13	21	1	$3.55E$-2	$1.06E$-1

TABLE IV
WILCOXON SIGNED-RANK TEST (5% NOISE)

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	7	27	1	$1.81E$-4	$9.05E$-4
FONN	15	20	0	$8.27E$-1	$8.27E$-1
FRNN	4	31	0	$2.90E$-7	$2.03E$-6
IBk	4	31	0	$1.60E$-6	$9.62E$-6
KStar	7	28	0	$4.44E$-4	$1.77E$-3
OWANN	8	27	0	$1.84E$-3	$5.53E$-3
VQNN	10	25	0	$1.18E$-2	$2.36E$-2

TABLE V
WILCOXON SIGNED-RANK TEST (10% NOISE)

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	4	30	1	5.42E-6	2.71E-5
FONN	18	17	0	4.41E-1	4.41E-1
FRNN	3	32	0	1.92E-9	1.34E-8
IBk	5	30	0	2.16E-8	1.30E-7
KStar	8	27	0	7.05E-6	2.82E-5
OWANN	5	30	0	1.51E-4	4.53E-4
VQNN	9	26	0	2.36E-3	4.72E-3

TABLE VI
WILCOXON SIGNED-RANK TEST (20% NOISE)

Algorithm	R^-	R^+	$R^=$	p-value	Holm
FRCN	7	28	0	2.58E-6	1.03E-5
FONN	21	14	0	9.68E-2	9.68E-2
FRNN	2	33	0	1.46E-9	1.02E-8
IBk	2	33	0	4.07E-9	2.44E-8
KStar	4	31	0	1.01E-7	5.06E-7
OWANN	8	27	0	8.43E-5	2.53E-4
VQNN	10	25	0	6.65E-3	1.33E-2

Overall, the simulation results over the 35 benchmark problems show that the k-FRCN classifier significantly outperforms all algorithms except for FONN in environments with 10% and 20% noise. This clearly indicates an improvement in the model with respect to the original FRCN classifier.

VI. CONCLUDING REMARKS

In this paper, we have proposed and validated a Fast k-FRCN variant of the existing FRCN classifier. The contribution of this model is two-folded. Firstly, the introduction of k-fuzzy-rough sets allows the model to perform better when the dataset is noisy. We have compared our proposed k-FRCN classifier to 7 state-of-the-art granular classifiers, including the original FRCN model. The k-FRCN model has proven to be capable of delivering very competitive results, especially in noisy environments. The model was able to outperform 6 of the other algorithms. Moreover, it achieved a higher accuracy than the original FRCN classifier on more than 70% of the datasets. Secondly, a parallel implementation of the information granulation procedure drastically reduces the time necessary to build the model. This can be accomplished by making more CPU threads available to the algorithm. The processing time decreased on average with 88% when at least two threads were at hand.

REFERENCES

[1] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, Wiley, New York, 2001.

[2] W. Pedrycz, G. Vukovich, Granular neural networks, Neurocomputing 36 (1) (2001) 205 – 224, rough-neuro computing.

[3] L. Zadeh, Fuzzy sets, Information and Control 8 (3) (1965) 338 – 353.

[4] Z. Pawlak, Rough Sets, Kluwer, 1991.

[5] M. Inuiguchi, W.-Z. Wu, C. Cornelis, N. Verbiest, Fuzzy-Rough Hybridization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 425–451.

[6] G. Benjamín, G. Nápoles, R. Falcon, R. Bello, K. Vanhoof, Performance analysis of granular versus traditional neural network classifiers: Preliminary results, 2018.

[7] G. Nápoles, I. Grau, E. Papageorgiou, R. Bello, K. Vanhoof, Rough cognitive networks, Knowledge-Based Systems 91 (2016) 46–61.

[8] B. Kosko, Fuzzy cognitive maps, International Journal of Man-Machine Studies 24 (1) (1986) 65 – 75.

[9] G. Nápoles, R. Falcon, E. Papageorgiou, R. Bello, K. Vanhoof, Rough cognitive ensembles, International Journal of Approximate Reasoning 85 (2017) 79 – 96.

[10] G. Nápoles, C. Mosquera, R. Falcon, I. Grau, R. Bello, K. Vanhoof, Fuzzy-rough cognitive networks, Neural Networks 97 (2018) 19 – 27.

[11] Q. Hu, S. An, X. Yu, D. Yu, Robust fuzzy rough classifiers, Fuzzy Sets and Systems 183 (1) (2011) 26–43.

[12] G. Nápoles, C. Mosquera, R. Falcon, I. Grau, R. Bello, K. Vanhoof, Fuzzy-rough cognitive networks, Neural Networks 97 (2018) 19–27.

[13] B. Kosko, Hidden patterns in combined and adaptive knowledge networks, International Journal of Approximate Reasoning 2 (4) (1988) 377 – 393.

[14] D. Aha, D. Kibler, Instance-based learning algorithms, Machine Learning 6 (1991) 37–66.

[15] J. G. Cleary, L. E. Trigg, K*: An instance-based learner using an entropic distance measure, in: 12th International Conference on Machine Learning, 1995, pp. 108–114.

[16] D. Tejera Hernández, An experimental study of k* algorithm, International Journal of Information Engineering and Electronic Business 7 (2015) 14–19.

[17] R. Jensen, C. Cornelis, Fuzzy-rough nearest neighbour classification 6499 (2011) 56–72.

[18] J. M. Keller, M. R. Gray, J. A. J. Givens., A fuzzy k-nearest neighbor algorithm, Transactions on Systems, Man, and Cybernetics 15 (15) (1985) 580–585.

[19] R. Jensen, C. Cornelis, Fuzzy-rough nearest neighbour classification and prediction, Theoretical Computer Science 412 (42) (2011) 5871 – 5884, rough Sets and Fuzzy Sets in Natural Computing.

[20] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, F. Herrera, Ifrowann: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification, IEEE Transactions on Fuzzy Systems 23 (5) (2015) 1622–1637.

[21] N. Verbiest, C. Cornelis, R. Jensen, Quality, frequency and similarity based fuzzy nearest neighbor classification, in: 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2013, pp. 1–8.

[22] C. Cornelis, M. De Cock, A. Radzikowska, Vaguely quantified rough sets, in: Lecture Notes in Artificial Intelligence, Vol. 4482, Springer, 2007, pp. 87–94.

[23] C. Cornelis, N. Verbiest, R. Jensen, Ordered weighted average based fuzzy rough sets, Rough Set and Knowledge Technology - 5th International Conference, RSKT 2010, Beijing, China, October 15-17, 2010. Proceedings 6401 78–85.

[24] R. R. Yager, On ordered weighted averaging aggregation operators in multicriteria decision making, ieee transactions on systems, man, and cybernetics, 8 (1): 183-190, Systems, Man and Cybernetics, IEEE Transactions on 18 (1988) 183 – 190.

[25] G. Nápoles, F. Vanhoenshoven, K. Vanhoof, Short-term cognitive networks, flexible reasoning and nonsynaptic learning, Neural Networks 115 (2019) 72 – 81.

[26] M. J. Warrens, Cohen's kappa is a weighted average, Statistical Methodology 8 (6) (2011) 473 – 484.

[27] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701.

[28] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks?, CoRR abs/1505.02288. arXiv:1505.02288.