Quality, Frequency and Similarity Based Fuzzy Nearest Neighbor Classification

Verbiest, Nele; Cornelis, Chris; Jensen, Richard

Published in:
2013 IEEE International Conference on Fuzzy Systems (FUZZ)

DOI:
10.1109/FUZZ-IEEE.2013.6622340

Publication date:
2013

Citation for published version (APA):
Verbiest, N., Cornelis, C., & Jensen, R. (2013). Quality, Frequency and Similarity Based Fuzzy Nearest Neighbor Classification. In *2013 IEEE International Conference on Fuzzy Systems (FUZZ)* (pp. 1-8). IEEE Press. https://doi.org/10.1109/FUZZ-IEEE.2013.6622340

Quality, Frequency and Similarity Based Fuzzy Nearest Neighbor Classification

Nele Verbiest*, Chris Cornelis*†, Richard Jensen‡
*Department of Applied Mathematics and Computer Science
Ghent University, Belgium
Email: Nele.Verbiest@UGent.be
†Department of Computer Science and Artificial Intelligence
University of Granada, Spain
Email: chriscornelis@ugr.es
‡Department of Computer Science
Aberystwyth University, Ceredigion, Wales, UK
Email: rkj@aber.ac.uk

***Abstract*—This paper proposes an approach based on fuzzy rough set theory to improve nearest neighbor based classification. Six measures are introduced to evaluate the quality of the nearest neighbors. This quality is combined with the frequency at which classes occur among the nearest neighbors and the similarity w.r.t. the nearest neighbor, to decide which class to pick among the neighbor's classes. The importance of each aspect is weighted using optimized weights. An experimental study shows that our method, Quality, Frequency and Similarity based Fuzzy Nearest Neighbor (QFSNN), outperforms state-of-the-art nearest neighbor classifiers.**

Keywords*—*Fuzzy Rough Set Theory, Classification, Nearest Neighbors, Ordered Weighted Average

I. Introduction

Classification, one of the most studied fields in machine learning, is the problem of labeling an instance based on previously seen data. It has applications in diverse fields like spam detection, speech recognition, health care, gene classification and many more.

The K Nearest Neighbor (KNN, [1]) classifier is a widely used and well-known classifier. To classify an instance, it considers its K Nearest Neighbors and classifies it to the most frequently occurring class among these neighbors. Although this classifier has proven to be very useful in many real-world classification problems, it can be improved in many ways.

In [2], the Fuzzy Nearest Neighbor (FNN) classifier was presented. This classifier takes into account the similarity between the neighbors and the instance that has to be classified. Experimental studies, e.g. in [3] or at the end of this paper, show that FNN significantly improves KNN.

Although FNN performs well, many authors have tried to improve it further. The strategy that we focus on in this paper is to improve FNN using fuzzy rough set theory [4], [5].

Some previous efforts to do this have been reported in literature. In [6], it was noted that FNN cannot adequately handle imperfect knowledge. To address this problem, Sarkar [6] introduced a so-called fuzzy rough ownership function that, when plugged into the conventional FNN algorithm, produces class confidence values that do not necessarily sum up to 1. However, this method does not refer to the main ingredients of rough set theory, i.e., lower and upper approximation.

Another attempt to improve FNN using fuzzy rough set theory was made in [7], [8]. The authors proposed a nearest neighbor algorithm that measures the extent to which the nearest neighbors belong to the fuzzy lower and upper approximations of a certain class to predict the class of the target instance. In order to deal with noisy data, they took this fuzzy rough nearest neighbour (FRNN) approach one step further and used vaguely quantified rough sets (VQRSs, [9]). It was shown in [3] that both FRNN and its VQRS version have some serious shortcomings.

Finally, in [3], the Fuzzy Rough Positive Region based Nearest Neighbor (POSNN) classifier was presented. This method uses fuzzy rough set theory to measure the quality of the neighbors of a test instance and plugs that quality measure into the FNN method.

Although the main idea of POSNN, namely that the quality of the neighbors should be taken into account, is promising, POSNN in its current form cannot outperform FNN. Therefore, we will try to improve it in this paper.

First, we study in more detail how we can improve the quality measure used in POSNN.

POSNN uses the fuzzy rough positive region, which is based on the fuzzy rough lower approximation. The fuzzy rough lower approximation expresses to what extent instances that are indiscernible from an instance belong to the same class as that instance. In this work, we will additionally take into account the fuzzy rough upper approximation, as this measure expresses to what extent there exist instances that are indiscernible from an instance and have the same class. Summarizing, the lower approximation of an instance is high if instances from a different class are discernible from it, and the upper approximation is high if there are indiscernible instances from the same class. Taking both approximations into account will reveal more information about the quality of the instances.

Another problem with the quality measure used by POSNN is that it is susceptible to noise: one instance can alter the quality measure drastically. Therefore, we propose to use Ordered Weighted Average (OWA) fuzzy rough sets [10]. These OWA fuzzy rough sets use OWA operators [11] to soften the strict infimum and supremum operators used in classical fuzzy rough

sets.
The second improvement of QFSNN compared to previous nearest neighbor classifiers is that it uses a different strategy to combine the different aspects of the nearest neighbors. Concretely, after determining the nearest neighbors of a test instance t, three measures are calculated for each neighbor. First, the frequency of the class of each neighbor among all nearest neighbors is calculated. Next, we calculate the similarity between each nearest neighbor and the test instance t. Finally, we also calculate the quality of the neighbors using (OWA) fuzzy rough set theory. Now, instead of multiplying these values to obtain a final evaluation measure for the nearest neighbors, we combine them in a weighted sum. As such, each aspect is not taken equally into account: it can be that the frequency should be taken more into account than the quality, or the other way around. To find good weights, we calculate the training accuracy corresponding to different combinations of weights, and select the weights with the highest training accuracy.
The remainder of this paper is structured as follows: In Section II, we briefly discuss related work on nearest neighbor classification. In Section III, we first discuss how we can measure the quality of instances using (OWA) fuzzy rough set theory and then propose our method, QFSNN. In Section IV, we present the results obtained with QFSNN and compare QFSNN with other nearest neighbor methods. We conclude in Section V.

II. Related Work

In this section we briefly recall nearest neighbor classifiers on which our QFSNN technique is based, and against which we will compare QFSNN in the experimental evaluation.

A. K Nearest Neighbor (KNN)

The classical KNN classifier [1] classifies a test instance t as follows: it considers the K nearest training instances (called the nearest neighbors) of t with respect to a certain distance measure, generally the Euclidean distance, and then classifies t to the class that is most common among those neighbors. In case of ties, a class is picked at random.
An important drawback of KNN is that it considers all neighbors equally important, that is, it does not take into account the similarity of the neighbors with respect to the instance. As a result, KNN will be highly dependent on the parameter K.

B. Fuzzy Nearest Neighbor (FNN)

To overcome the drawbacks of KNN, the FNN classifier was developed by Keller [2]. This method also considers the set NN of K nearest neighbors of the test instance t, and then classifies t to the class C for which the following measure is maximal:

$$\frac{\sum_{c\in NN} R(x,t)C(x)}{\sum_{x\in NN} R(x,t)}. \tag{1}$$

In this formula, $R(x,t)$ is the similarity between the instances x and t, defined as follows:

$$\frac{1}{d_{eucl}(x,t)^2}, \tag{2}$$

where d_{eucl} refers to the Euclidean distance function.
The class membership $C(x)$ can be defined in two ways. The approach suggested in [2] is to define it as follows:

$$C(x) = \begin{cases} 0.51 + 0.49\frac{n_C}{K} & \text{if the class of } x \text{ is } C; \\ 0.49\frac{n_C}{K} & \text{else.} \end{cases} \tag{3}$$

Here, n_C is the number of nearest neighbors that are in class C. Another option is to use

$$C(x) = \begin{cases} 1 & \text{if the class of } x \text{ is } C; \\ 0 & \text{else.} \end{cases} \tag{4}$$

As we found in [3] that the latter definition results in better accuracy results, we will use this definition in the remainder of this paper.
The main idea of the FNN method is that not only the frequency at which a class occurs among the neighbors is important, but that the class memberships have to be weighted according to their similarity with respect to the test instance at hand.

C. Fuzzy Rough Nearest Neighbor (FRNN) and Vaguely Quantified Nearest Neighbor (VQNN) classification

In [7], [8], an attempt was made to improve FNN using fuzzy rough set theory. The main idea of FRNN is that a test instance t is assigned to the class C such that the sum of the memberships of t to the lower and upper approximation of this class C is maximal.
Although the idea in this paper is interesting, we showed in [3] that FRNN has some shortcomings. It turns out that FRNN classifies t to the class of the training instance x for which $R(x,t)$ is minimal. As a result, the classification of t only depends on one instance, and the concepts of fuzzy rough set theory are not fully exploited.
Another approach in [7], [8] to improve FNN is to use Vaguely Quantified Rough Sets (VQRS, [9]). The idea is the same as for FRNN: the test instance t is assigned to the class C such that the sum of memberships of t to the lower and upper approximation of this class C is maximal. The difference is that VQRSs are used instead of classical fuzzy rough sets.
In [3], we proved that VQNN and FNN return the same class for a test instance t, provided that the same similarity measure is used for both methods.

D. Positive Region based Nearest Neighbor (POSNN)

As FRNN and VQNN cannot improve FNN for classification tasks using fuzzy rough set theory, we proposed an alternative method POSNN in [3]. Instead of using the fuzzy rough approximations directly to classify an instance t, as in FRNN and VQNN, we used fuzzy rough set theory to measure the quality of the nearest neighbors, and included that quality as an extra weight for the class membership values. More specifically, we classified an instance t to the class for which

$$\frac{\sum_{c\in NN} R(x,t)C(x)POS(x)}{\sum_{x\in NN} R(x,t)} \tag{5}$$

is maximal, where $POS(x)$ is the fuzzy rough positive region membership of x, a fuzzy rough measure for the quality of x.

Although the idea of POSNN is more consistent and meaningful than the idea used in FRNN and VQNN, POSNN was found to only slightly improve FNN.

III. Quality, Similarity and Frequency based Fuzzy Nearest Neighbor (QFSNN) Classification

In this section we present our new nearest neighbor classification technique. We first discuss how we can measure the quality of instances using fuzzy rough set theory and then explain how we use it to improve FNN classification.

A. Using (OWA) fuzzy rough set theory to measure the quality of instances.

In this work we will use the fuzzy rough lower and upper approximation to measure the quality of instances. Before we introduce our ideas, we will fix the notation. We work in a decision system $(X, \mathcal{A} \cup \{d\})$ where $X = \{x_1, \ldots, x_n\}$ is a set of instances and $\mathcal{A} = \{a_1, \ldots, a_m\}$ is the set of features. The value of an instance $x \in X$ for a feature $a \in \mathcal{A}$ is denoted by $a(x)$ and is normalized such that $a(x) \in [0, 1]$. The feature $d \notin \mathcal{A}$ is a special feature: $d(x)$ denotes the class of the instance $x \in X$. In the classification context that we are working in, $d(x)$ is always nominal.
Fuzzy rough set theory revolves around the idea of indiscernibility between instances. The indiscernibility relation Ind that we will use is defined as follows:

$$\forall x, y \in X : Ind(x, y) = \min_{i=1,\ldots,m} Ind_{a_i}(x, y), \tag{6}$$

where the indiscernibility relation w.r.t. one feature $a \in \mathcal{A}$ is defined as:

$$\forall a \in \mathcal{A} : Ind_a(x, y) = 1 - |a(x) - a(y)|. \tag{7}$$

This indiscernibility relation can be used to approximate concepts, which are sets of instances in our context. Given a fuzzy set $S : X \to [0, 1]$, the lower approximation $Ind \downarrow S$ of S by means of Ind is a mapping $X \to [0, 1]$ defined as follows:

$$\forall x \in X : (Ind \downarrow S)(x) = \inf_{y \in X} \mathcal{I}(Ind(x, y), S(y)). \tag{8}$$

The mapping $\mathcal{I}$ is a fuzzy implicator, which is a mapping $[0, 1]^2 \to [0, 1]$ that is decreasing in the first, increasing in the second argument and for which the border conditions $\mathcal{I}(0, 0) = \mathcal{I}(1, 1) = \mathcal{I}(0, 1) = 1$ and $\mathcal{I}(1, 0) = 0$ hold. The implicator that we will use in this work is the Lukasiewicz implicator $\mathcal{I}_L$, given by:

$$\forall a, b \in [0, 1] : \mathcal{I}_L(a, b) = \min(1, 1 - a + b). \tag{9}$$

The lower approximation of an instance x expresses to what extent instances that are indiscernible from x belong to the concept S. We now consider the lower approximation of a particular concept, namely the instances $[x]_d$ that are in the same class as x:

$$\forall x \in X : [x]_d = \{y \in X | d(x) = d(y)\} \tag{10}$$

The value of the membership of x to the lower approximation of the class $[x]_d$ is given by:

$$\forall x \in X : (Ind \downarrow [x]_d)(x) = \inf_{y \in X} \mathcal{I}_L(Ind(x, y), [x]_d(y)). \tag{11}$$

This lower approximation can measure the quality of the instance x because it expresses to what extent instances that are indiscernible from x belong to the same class as x. Hence, it measures how typical x is for its class.
As $\mathcal{I}_L(a, 1) = 1$ for all $a \in [0, 1]$, we can rewrite Equation (11) as follows when using $\mathcal{I}_L$:

$$\forall x \in X : (Ind \downarrow [x]_d)(x) = \inf_{y \in X \setminus [x]_d} 1 - Ind(x, y) \tag{12}$$

This makes the intuition to use the lower approximation to measure the quality more clear: the membership to the lower approximation of the own class of an instance is low if instances from a different class of x are indiscernible from x. Note that this definition corresponds to the positive region of x, so this measure corresponds to the one that was used in POSNN [3]. We will refer to this first quality measure as Q_L.
Apart from the lower approximation, a fuzzy rough set also contains the upper approximation. Given a fuzzy set $S : X \to [0, 1]$, the fuzzy rough upper approximation of S by means of Ind is a mapping $X \to [0, 1]$ defined as follows:

$$\forall x \in X : (Ind \uparrow S)(x) = \sup_{y \in X} \mathcal{T}(Ind(x, y), S(y)). \tag{13}$$

The mapping $\mathcal{T}$ is a t-norm, a commutative, associative mapping $\mathcal{T} : [0, 1]^2 \to [0, 1]$ that is increasing in both arguments, and for which $\forall a \in [0, 1] : \mathcal{T}(a, 1) = a$ holds. In our work, we use the Lukasiewicz t-norm $\mathcal{T}_L$, defined as follows:

$$\forall a, b \in [0, 1] : \mathcal{T}_L(a, b) = \max(0, a + b - 1). \tag{14}$$

The upper approximation of an instance x expresses to what extent there exist instances that are indiscernable from x belonging to the concept S. As for the lower approximation, we study the membership to the upper approximation of an instance x with respect to its own class $[x]_d$:

$$\forall x \in X : (Ind \uparrow [x]_d)(x) = \sup_{y \in X} \mathcal{T}(Ind(x, y), [x]_d(y)). \tag{15}$$

At first sight, this measure expresses to what extent there exist instances that are indiscernible from x and that are in the same class as x. However, if we have a closer look at Equation (15), we see that this value is always 1: there always exists at least one instance that is indiscernable from x and that is in the same class as x, namely x itself. Therefore, we slightly adapt the definition of the upper approximation and use the following quality measure Q_U:

$$\forall x \in X : Q_U(x) = \sup_{y \in X \setminus \{x\}} \mathcal{T}(Ind(x, y), [x]_d(y)). \tag{16}$$

This measure can also be rewritten as follows:

$$\forall x \in X : Q_U(x) = \sup_{y \in [x]_d \setminus \{x\}} Ind(x, y). \tag{17}$$

That is, Q_U is high for x if there are instances (different from x) of the same class as x that are indiscernible from x.
Both Q_L and Q_U are interesting to measure the quality of instances. Therefore, we also consider Q_{LU}, defined as:

$$\forall x \in X : Q_{LU}(x) = Q_L(x) + Q_U(x). \tag{18}$$

This measure combines the two ideas captured in Q_L and Q_U: the instance x is of high quality if there are instances from the same class indiscernible from it, and if the instances that are from a different class are indiscernible from it.

Although Q_L, Q_U and Q_{LU} seem to be good quality measures, they are highly susceptible to noise. E.g., when there is one instance y that is falsely labeled $d(y) = d(x)$ and y is indiscernbile from x, then the quality $Q_U(x)$ will be high, based on this single mislabeled instance y alone.
To overcome this weakness, we will use OWA fuzzy rough sets [10] instead of classical fuzzy rough sets. The main reason why classical fuzzy rough sets are susceptible to noise is that they use the infimum and supremum operators. In OWA fuzzy rough set theory, these operators are replaced by OWA operators [11]. Recall that, given a weight vector $W = \langle w_1, \ldots, w_p \rangle$, the OWA_W operator aggregates p values $\{a_1, \ldots, a_p\}$ as follows:

$$OWA_W(a_1, \ldots, a_p) = \sum_{i=1}^{p} w_i b_i, \tag{19}$$

where $b_i = a_j$ if a_j is the i^{th} largest value in $\{a_1, \ldots, a_p\}$. The idea is that the weights W are associated to ordered positions: the higher values in $\{a_1, \ldots, a_p\}$ are assigned to the first weights in W, the lower values are associated with the last weights in W. Note that the OWA operator is a model for many standard aggregation operators. E.g., when $W = \langle 0, \ldots, 0, 1 \rangle$ is used, OWA_W is the minimum operator, and OWA_W with $W = \langle 1/n, \ldots, 1/n \rangle$ is the average operator.
The OWA operator can be used to soften the infimum and supremum operator. The supremum operator is the OWA_W operator with $W = \langle 1, 0, \ldots, 0 \rangle$. To make this operator less strict, we can use a weight vector $W_{\sup}$ with a decreasing range of weights. As a result, lower values will get lower weights than higher values, but are not totally ignored. Moreover, the final aggregation will take into account all values and does not depend on a single value. The weights that we will use in this paper are $W_{\sup} = \langle w_1, \ldots, w_p \rangle$ with

$$\forall i \in \{1, \ldots, p\} : w_i = \frac{2(p - i + 1)}{p(p + 1)}. \tag{20}$$

Completely analogously, we can define a soft infimum operator $OWA_{W_{\inf}}$ where $W_{\inf} = \langle w_1, \ldots, w_p \rangle$ with

$$\forall i \in \{1, \ldots, p\} : w_i = \frac{2i}{p(p + 1)}. \tag{21}$$

The resulting OWA fuzzy lower approximation is given by:

$$\forall x \in X : (Ind \downarrow^{OWA} S)(x) = \underbrace{OWA_{W_{inf}} \mathcal{I}(Ind(x, y), S(y))}_{y \in X}, \tag{22}$$

the OWA fuzzy upper approximation is defined by:

$$\forall x \in X : (Ind \uparrow^{OWA} S)(x) = \underbrace{OWA_{W_{sup}} \mathcal{T}(Ind(x, y), S(y))}_{y \in X}. \tag{23}$$

To measure the quality of instances using the OWA lower and upper approximation, we use the following definitions:

$$\forall x \in X : Q_L^{OWA}(x) = (Ind \downarrow^{OWA} [x]_d)(x) \tag{24}$$

and

$$\forall x \in X : Q_U^{OWA}(x) = (Ind \uparrow^{OWA} [x]_d)(x) \tag{25}$$

Again, we can combine both definitions to define Q_{LU}^{OWA}:

$$\forall x \in X : Q_{LU}^{OWA}(x) = Q_L^{OWA}(x) + Q_U^{OWA}(x) \tag{26}$$

Summarizing, we introduced the following six quality measures to measure the quality of instances $x \in X$:

- $Q_L(x) = \inf_{y \in X} \mathcal{I}(Ind(x, y), [x]_d(y))$
- $Q_U(x) = \inf_{y \in X} \mathcal{T}(Ind(x, y), [x]_d(y))$
- $Q_{LU}(x) = Q_L(x) + Q_U(x)$
- $Q_L^{OWA}(x) = \underbrace{OWA_{\inf} \mathcal{I}(Ind(x, y), [x]_d(y))}_{y \in X}$
- $Q_U^{OWA}(x) = \underbrace{OWA_{\sup} \mathcal{T}(Ind(x, y), [x]_d(y))}_{y \in X}$
- $Q_{LU}^{OWA}(x) = Q_L^{OWA}(x) + Q_U^{OWA}(x)$

B. Quality, Similarity and Frequency Based Fuzzy Nearest Neighbor (QFSNN) Classification

Using the quality measures discussed in the previous subsection, we now introduce a method to improve the fuzzy nearest neighbor classification.
To classify a test instance t, our method, QFSNN, first determines the K nearest neighbors NN of t among the training instances using the Euclidean distance measure, and then calculates three measures for each neighbor $x \in NN$:

- The quality $Q(x)$ of x, using one of the quality measures described in the previous subsection.
- The frequency $F(x)$ of the class of x among the instances in NN. That is, if the class of x is C and there are n instances in NN with class C, then the frequency $F(x)$ is given by $\frac{n}{K}$.
- The similarity $S(x) = R(x, t)$ between x and t, using the similarity measure proposed by Keller, as in Equation (2).

These three aspects of the nearest neighbors are all important for the classification. In [3], they were combined by multiplying the corresponding measures, but that approach did not significantly improve the accuracy. A reason for that might be that some aspects can be more important than others. We illustrate this in Figure 1, where a two-class classification problem is shown. Graphically, it is clear that the test instance t should be assigned to the grey dot class. However, if we apply state-of-the-art methods to this example with $K = 7$, they will all return the black dot class. KNN returns the black dot class because 4 out of the 7 nearest neighbors belong to the black dot class. FNN also takes into account the similarity between the test instance t and the nearest neighbors, but as some of the black dot class neighbors are almost equally close to t as the grey dot class neighbors, and there is one black dot class neighbor more, FNN will return the black dot class as well. POSNN also takes into account the quality of the nearest neighbors. Using the fuzzy rough quality measure, the black dot classes will be favored, as they are surrounded by many instances of their class. As a result, two out of three aspects that POSNN takes into account, namely frequency and quality, favor the black dot class, so it is very likely that POSNN will assign the test instance to the black dot class. It is clear that in this example, the similarity is a more important aspect than

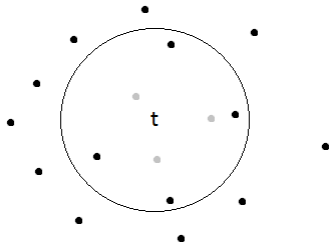

Fig. 1. Example of a two-class classification problem where KNN, FNN and POSNN assign the test instance t to the black dot class, and QFSNN assigns t to the grey dot class ($K = 7$).

the frequency or quality. For that reason, we associate a weight with each of the aspects and obtain the following evaluation measure $E(x)$ for an instance x:

$$E(x) = w_Q Q(x) + w_F F(x) + w_S S(x) \tag{27}$$

Our method QFSNN will return the class of the nearest neighbor $x \in NN$ for which $E(x)$ is maximal.
The question remains of course which weights to use for the evaluation function. To that goal, we consider a set of triplets of weights, and calculate the training accuracy corresponding to each triplet using a leave-one-out approach. The triplet that we use for the final classification will be the triplet corresponding to the highest training accuracy.
The triplets of weights that we use are constructed as follows. We consider a certain range $r \in \mathbb{N}$ and consider all possible triplets of weights where $w_Q, w_F, w_S \in \{1, \ldots, r\}$.
The procedure to calculate the training accuracy corresponding to a given triplet of weights is listed in Algorithm 1. For each training instance x, the nearest neighbors are calculated. Of course, the instance x is excluded from the possible nearest neighbors. Next, the nearest neighbors are evaluated using the function E with the given triplet of weights, and the algorithm keeps track of the nearest neighbor with the highest evaluation. If this nearest neighbor has the same class as the training instance, the training instance is classified correctly using these weights, and the accuracy is raised by one. The final accuracy returned is the number of correctly classified training instances divided by the total number of training instances.

IV. Experimental evaluation

In this section we evaluate the QFSNN classifier. We first discuss the set-up of the experiments and then present the results.

A. Experimental Set-Up

To evaluate QFSNN, we use 40 datasets from the KEEL dataset repository[1]. The datasets are listed in Table IV, together with the number of features, instances and classes they cover. We apply 10 fold cross-validation, that is, each dataset is divided into 10 folds. To classify the instances in a fold, the remaining 9 folds are used as training data. The final accuracy reported for each dataset is the average accuracy over all folds.

[1] www.keel.es

Algorithm 1 Algorithm to calculate the training accuracy corresponding to a triplet of weights using the leave-one-out procedure.

input: Training instances X, parameter K, triplet of weights (w_Q, w_F, w_S).
$Acc \leftarrow 0$
for $x \in X$ **do**
  $NN \leftarrow K$ nearest neighbors of x in $X \setminus \{x\}$
  $MaxE \leftarrow 0$
  $MaxN \leftarrow null$
  for $y \in NN$ **do**
    if $E(y) = w_Q Q(y) + w_F F(y) + w_S S(y) \geq MaxE$ **then**
      $MaxE \leftarrow E(y) = w_Q Q(y) + w_F F(y) + w_S S(y)$
      $MaxN \leftarrow y$
    end if
  end for
  if $Class(y) == Class(MaxN)$ **then**
    $Acc \leftarrow Acc + 1$
  end if
end for
Output $Acc/|X|$

We use the folds that can be found on the KEEL website.
We recall that we can measure the quality of the instances using the lower (Q_L), upper (Q_U) or sum of lower and upper (Q_{LU}) fuzzy rough approximation. Moreover, we can use the classical definition or the OWA extension (Q_L^{OWA}, Q_U^{OWA}, Q_{LU}^{OWA}). We use 5 different ranges r for training QFSNN: 5, 10, 20, 50 and 100.

B. Results

In this section we present the results obtained performing the experiments described in the previous subsection. The first aspect that we want to discuss is which method is most suited to measure the quality of the instances. First, we study if using the OWA extension is beneficial. In Figure 2, we show the average accuracy over all datasets of the QFSNN method, using range $r = 20$.
In the upper figure, we compare the accuracy results of QFSNN using Q_L and Q_L^{OWA} for measuring the quality of the instances. For lower values of K, the OWA extension improves the results clearly, when K increases, the difference is smaller.
In the next figure, we study the difference in accuracy between QFSNN using Q_U or Q_U^{OWA} for the quality measure. We can make the same conclusion as for Q_L and Q_L^{OWA}. The OWA extension improves QFSNN, and the differences are smaller for higher values of K. Note that the OWA extension influences the results more than for the lower approximation.
In the last figure, the accuracy results for Q_{LU} and Q_{LU}^{OWA} are shown. Again, the OWA extension improves the accuracy of QFSNN. For high values of K, the differences are smaller, for low values of K, the OWA extension improves the accuracy of QFSNN to a greater extent.
We conclude that the OWA extension improves the accuracy of the QFSNN classifier. We only showed the results for range $r = 20$, but the results for other ranges are similar. In the remainder of this work, we will only consider the OWA extension of the fuzzy rough approximation to measure the quality of the instances in the QFSNN classifier.

Next, we study which strategy is best to measure the quality of instances: using the lower, upper fuzzy rough approximation or the sum of both. In Table I, we show the average accuracy results over all datasets for different values of K. We now distinguish between different ranges for r because the trends are different for different values of r. For each range r and value K, the best result is highlighted.

Except for some ranges and values of K, Q_L^{OWA} has the worst results, although the differences are not large. In general, Q_U^{OWA} performs well for low values of K, and Q_{LU}^{OWA} performs better for high values of K.

In the remainder of the discussion we will use Q_{LU}^{OWA} to measure the quality of the instances, as the results of QFSNN are slightly more often better when using this quality measure.

Another aspect that we are interested in is which range is better. When the range r is higher, the running time is of course longer, so we prefer to use low values of r. In Table I We see that a higher value of r does not automatically lead to better accuracy values, on the contrary, for $r = 50$ or $r = 100$ the results are mostly worse than for lower ranges r. This is probably due to overfitting. In general, the results are best for $r = 10$, so we will use this range in the remainder of this paper.

Finally, we of course want to know if we succeeded in improving the FNN classifier. We use QFSNN where Q_{LU}^{OWA} is used to measure the quality of the instances and where range $r = 10$ is used to train the weights. We compare QFSNN against KNN and FNN for different values of K in Figure 3. The accuracy results are shown in Table IV for $K = 3$. Due to space constraints we cannot show the results for other values of K.

We see that QFSNN, POSNN and FNN clearly improve KNN. Moreover, QFSNN is better than FNN and POSNN for all values of K. The difference is bigger for lower values of K.

To test if the differences between FNN and QFSNN are significant, we perform the statistical Wilcoxon test [12]. This is a non-parametric pairwise test that aims to detect significant differences between two sample means; that is, the behavior of the two implicated algorithms in the comparison. For each comparison we compute R+, the sum of ranks of the Wilcoxons test in favor of QFSNN, R-, the sum of ranks in favor of FNN, and also the p-value obtained for the comparison. The observed values of the statistics are listed in Table II. We perform the test at the 10 percent significance level.

QFSNN is significantly better than FNN for $K = 3$ and $K = 4$. For the other values of K, we previously showed in Figure 3 that QFSNN is better on average then FNN, but this result is not significant.

C. Time complexity

It can be verified that, when n is the number of instances and m the number of attributes, the time complexity for classifying one test instance using KNN, FNN and POSNN is $\mathcal{O}(nm)$. When l is the number of test instances, the time complexity for the whole test dataset is $\mathcal{O}(nml)$.

Unfortunately, the increase in accuracy of QFSNN comes with an increase in time complexity. When r is the range, the number of considered triplets is r^3. For each triplet, each of the n training instances has to be classified. Classifying one training instance has time complexity $\mathcal{O}(nm)$, so the total time complexity to train the weights of QFSNN is $\mathcal{O}(r^3n^2m)$.

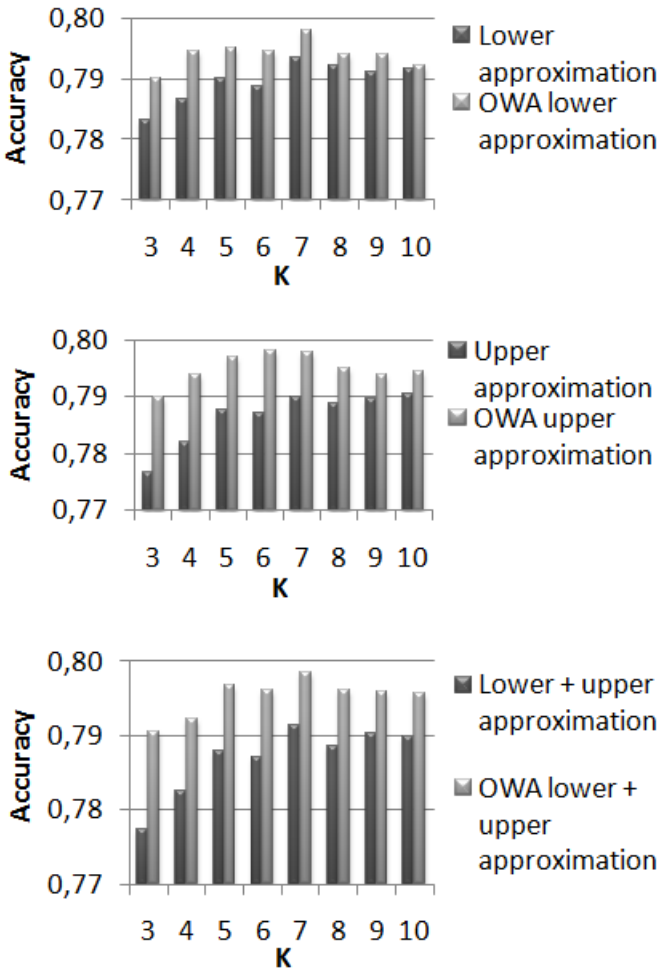

Fig. 2. Average accuracy of QFSNN with range $r = 20$ over all datasets for different values of K, comparing fuzzy rough approximations with their OWA extensions.

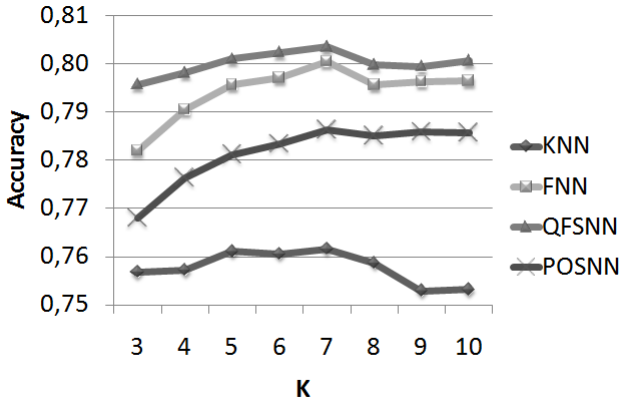

Fig. 3. Average accuracy of QFSNN with range $r = 10$ and using Q_{LU}^{OWA} for the quality measure, FNN and KNN over all datasets for different values of K.

TABLE I. AVERAGE ACCURACY OF QFSNN OVER ALL DATASETS FOR DIFFERENT VALUES OF K AND r, COMPARING THE QUALITY MEASURES Q_L^{OWA}, Q_U^{OWA} AND Q_{LU}^{OWA}.

	K	3	4	5	6	7	8	9	10
$r=5$	Q_L^{OWA}	0.7890	**0.7941**	0.7952	0.7943	0.7970	**0.7966**	0.7938	0.7938
	Q_U^{OWA}	0.7893	0.7923	**0.7967**	**0.7984**	0.7977	0.7939	0.7921	0.7942
	Q_{LU}^{OWA}	**0.7903**	0.7928	0.7960	0.7965	**0.7992**	0.7946	**0.7947**	**0.7947**
$r=10$	Q_L^{OWA}	0.7901	**0.7944**	0.7948	0.7960	**0.7986**	**0.7950**	0.7930	0.7925
	Q_U^{OWA}	**0.7906**	**0.7944**	**0.7968**	**0.7985**	0.7974	0.7940	0.7936	0.7951
	Q_{LU}^{OWA}	**0.7906**	0.7930	0.7960	0.7973	0.7984	0.7947	**0.7943**	**0.7955**
$r=20$	Q_L^{OWA}	**0.7899**	**0.7944**	0.7951	0.7945	0.7980	0.7941	0.7939	0.7920
	Q_U^{OWA}	**0.7899**	0.7937	**0.7968**	**0.7979**	0.7977	0.7950	0.7938	0.7943
	Q_{LU}^{OWA}	0.7905	0.7923	**0.7968**	0.7960	**0.7984**	**0.7962**	**0.7958**	**0.7956**
$r=50$	Q_L^{OWA}	0.7885	**0.7934**	0.7958	0.7952	0.7968	0.7935	0.7942	0.7937
	Q_U^{OWA}	**0.7910**	0.7929	0.7959	**0.7968**	0.7977	0.7944	0.7952	0.7942
	Q_{LU}^{OWA}	0.7904	0.7921	**0.7971**	0.7951	**0.7984**	**0.7949**	**0.7958**	**0.7948**
$r=100$	Q_L^{OWA}	0.7891	**0.7943**	0.7962	0.7941	0.7969	0.7944	0.7939	0.7935
	Q_U^{OWA}	**0.7900**	**0.7929**	**0.7965**	**0.7961**	0.7970	0.7946	0.7935	**0.7941**
	Q_{LU}^{OWA}	0.7893	0.7922	0.7964	0.7954	**0.7974**	**0.7953**	**0.7959**	0.7940
KNN		0.7518	0.7522	0.7561	0.7555	0.7566	0.7537	0.7479	0.7482
FNN		0.7769	0.7854	0.7906	0.7920	0.7954	0.7905	0.7912	0.7914
$POSNN$		0.7630	0.7713	0.7762	0.7783	0.7813	0.7800	0.7808	0.7807

TABLE II. OBSERVED VALUES OF THE NON-PARAMETRIC WILCOXON TEST FOR THE DIFFERENT VALUES OF K, COMPARING QFSNN TO FNN.

K	3	4	5	6	7	8	9	10
R +	537	535	476.5	499.5	419.5	446.5	441	448.5
R -	283	285	343.5	320.5	360.5	333.5	379	331.5
p-value	0.086	0.091	0.368	0.226	0.675	0.426	0.671	0.41

TABLE III. RUNNING TIMES FOR KNN, FNN, POSNN AND QFSNN OVER ALL DATASETS

Method	Running time (in s)
KNN	167
FNN	170
POSNN	303
QFSNN, Q_{LU}^{OWA}, $r=5$	374
QFSNN, Q_{LU}^{OWA}, $r=10$	386
QFSNN, Q_{LU}^{OWA}, $r=20$	445
QFSNN, Q_{LU}^{OWA}, $r=50$	923
QFSNN, Q_{LU}^{OWA}, $r=100$	4347

After the weights are determined, classifying all test instances has time complexity $\mathcal{O}(nml)$. The total time complexity of QFSNN is $\mathcal{O}(nm(l+r^3n))$.

In practice, however, the running time of QFSNN is acceptable for low values of r. In Table III, we show the running times needed to apply KNN, FNN, POSNN and QFSNN with different values for r to all 40 datasets. Recall that QFSNN with $r=5$ and $r=10$ lead to the best accuracy results. Their running times are only double of the running times of KNN and FNN. For higher values of r, the running time increases drastically.

V. CONCLUSION AND FURTHER WORK

In this work we introduced a new nearest neighbor classifier, QFSNN. When classifying a test instance, it considers three aspects to evaluate its nearest neighbors. As in KNN, it considers the frequency of the classes among the nearest neighbors, and as in FNN, it considers the similarity between the test instance and the neighbors. In addition, it also measures the quality of the nearest neighbors, using six measures based on (OWA) fuzzy rough set theory. These three aspects are combined by weighting them, the weights are optimized using a leave-one-out training procedure.

An experimental study shows that our method, QFSNN, improves the state-of-the-art nearest neighbor classifiers KNN and FNN. QFSNN is significantly better than FNN for low values of K.

In this work, we used a fixed weight vector for the OWA fuzzy rough quality measures, but in the future we want to investigate other weightings and automatic data-driven determination of weights. Another path we would like to explore in the future would be to test the performance of QFSNN for prediction tasks.

ACKNOWLEDGMENT

The paper has been partially supported by Spanish project TIN2011-28488.

REFERENCES

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] J. M. Keller, M. R. Gray, and J. R. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, pp. 580–585, 1985.

TABLE IV. ACCURACY RESULTS FOR KNN, FNN, POSNN AND QFSNN FOR $K = 3$

Name	# features	# instances	# classes	KNN	FNN	POSNN	QFSNN Q_{LU}^{OWA} $(r = 5)$	QFSNN Q_{LU}^{OWA} $(r = 10)$	QFSNN Q_{LU}^{OWA} $(r = 20)$	QFSNN, Q_{LU}^{OWA} $(r = 50)$	QFSNN Q_{LU}^{OWA} $(r = 100)$
appendicitis	7	106	2	0.8427	0.8327	0.8427	0.8691	**0.8791**	0.8509	0.8418	0.8600
australian	14	690	2	0.8478	0.8333	0.8333	0.8551	**0.8551**	0.8551	0.8522	0.8522
automobile	25	150	6	0.6334	0.7440	0.7423	0.7551	0.7492	**0.7564**	0.7421	0.7421
balance	4	625	3	0.8337	0.8337	0.8337	0.8705	0.8800	0.8800	**0.8832**	0.8816
bands	19	365	2	0.7146	0.7363	**0.7367**	0.7255	0.7285	0.7285	0.7309	0.7206
breast	9	277	2	0.6743	0.6595	0.6704	0.7367	0.7367	**0.7367**	0.7365	0.7367
bupa	6	345	2	0.6066	**0.6328**	0.6271	0.6238	0.6088	0.6209	0.6298	0.6298
car	6	1728	4	0.9231	0.9231	0.9231	0.9236	**0.9236**	0.9236	0.9236	0.9236
cleveland	13	297	5	**0.5514**	0.5391	0.5460	0.5495	0.5495	0.5495	0.5495	0.5495
contraceptive	9	1473	3	**0.4495**	0.4434	0.4393	0.4488	0.4488	0.4467	0.4495	0.4468
crx	15	653	2	0.8402	0.8296	0.8389	0.8612	**0.8612**	0.8567	0.8549	0.8520
dermatology	34	358	6	**0.9690**	**0.9690**	**0.9690**	**0.9690**	**0.9690**	**0.9690**	**0.9690**	**0.9690**
ecoli	7	336	8	0.8067	0.8127	**0.8337**	0.8097	0.8068	0.8037	0.8068	0.8097
flare	11	1066	6	0.6229	**0.7016**	0.5262	0.6960	0.6988	0.6988	0.6979	0.6979
german	20	1000	2	0.6960	0.7030	0.7060	0.7090	0.7130	**0.7170**	0.7160	0.7090
glass	9	214	7	0.7011	0.7270	**0.7494**	0.6991	0.6991	0.7041	0.6991	0.6991
haberman	3	306	2	0.7058	0.6733	0.6635	0.7220	**0.7220**	**0.7220**	**0.7220**	0.7156
hayes-roth	4	160	3	0.2500	0.6625	0.6813	0.7250	**0.7250**	**0.7250**	0.7188	0.7000
heart	13	270	2	0.7741	0.7704	0.7778	0.7815	0.7852	0.7889	**0.8000**	0.7852
hepatitis	19	80	2	**0.8251**	**0.8251**	**0.8251**	**0.8251**	**0.8251**	**0.8251**	**0.8251**	**0.8251**
housevotes	16	232	2	0.9181	**0.9250**	**0.9250**	0.9070	0.9070	0.9070	0.9070	0.9070
iris	4	150	3	**0.9400**	**0.9400**	**0.9400**	0.9333	0.9333	0.9333	0.9333	0.9333
led7digit	7	500	10	0.4520	**0.6580**	0.2220	0.6260	0.6260	0.6260	0.6260	0.6260
lymphography	18	148	4	0.7739	0.7803	0.7798	0.7798	0.7798	**0.7865**	0.7860	0.7860
mammographic	5	830	2	**0.8012**	0.7842	0.7324	0.7760	0.7786	0.7773	0.7785	0.7749
monk-2	6	432	2	**0.9629**	0.7739	0.7879	0.9629	**0.9629**	**0.9629**	**0.9629**	**0.9629**
movement-libras	90	360	15	0.7861	0.8694	0.8667	**0.8806**	0.8778	**0.8806**	0.8778	0.8778
newthyroid	5	215	3	0.9537	**0.9723**	**0.9723**	0.9675	0.9675	0.9675	0.9582	0.9628
pima	8	768	2	0.7293	0.7202	0.7280	0.7319	0.7358	**0.7397**	0.7384	0.7384
saheart	9	462	2	0.6818	0.6753	0.6774	0.6970	0.6970	0.6992	**0.7035**	0.6992
sonar	60	208	2	0.8307	0.8500	0.8452	**0.8507**	0.8460	0.8414	0.8364	0.8412
spectfheart	44	267	2	0.7120	0.7120	0.7120	0.7755	0.7755	0.7755	0.7755	**0.7792**
tae	5	151	3	0.4113	0.6438	0.6504	0.6504	0.6504	0.6500	**0.6633**	0.6500
tic-tac-toe	9	958	2	0.7756	0.7756	0.7756	**0.7767**	**0.7767**	**0.7767**	**0.7767**	**0.7767**
vehicle	18	846	4	0.7175	**0.9649**	0.7128	0.7104	0.7068	0.6974	0.7104	0.7010
vowel	13	990	11	0.9778	0.7187	0.9859	**0.9929**	**0.9929**	**0.9929**	**0.9929**	0.9929
wine	13	178	3	0.9549	**0.9879**	0.9549	0.9549	0.9660	0.9660	0.9663	0.9663
wisconsin	9	683	2	0.9638	0.9549	**0.9681**	0.9638	0.9638	0.9638	0.9638	0.9638
yeast	8	1484	10	0.5317	**0.9696**	0.5499	0.5492	0.5485	0.5485	0.5546	0.5593
zoo	16	101	7	0.9281	0.5492	**0.9683**	**0.9683**	**0.9683**	**0.9683**	**0.9683**	**0.9683**

[3] N. Verbiest, C. Cornelis, and R. Jensen, "Fuzzy rough positive region-based nearest neighbour classification," in *Proceedings of the 20th International Conference on Fuzzy Systems (FUZZ-IEEE2012)*, 2012, pp. 1961–1967.

[4] C. Cornelis, M. De Cock, and A. M. Radzikowska, "Fuzzy rough sets: From theory into practice," in *Handbook of Granular Computing*, 2008, pp. 533–552.

[5] D. Dubois and H. Prade, "Rough fuzzy sets and fuzzy rough sets," *International Journal of General Systems*, vol. 17, pp. 191–209, 1990.

[6] M. Sarkar, "Fuzzy-rough nearest neighbor algorithms in classification," *Fuzzy Sets and Systems*, vol. 158, no. 19, pp. 2134 – 2152, 2007.

[7] R. Jensen and C. Cornelis, "A new approach to fuzzy-rough nearest neighbour classification," in *Proceedings of the 6th International Conference on Rough Sets and Current Trends in Computing*, ser. RSCTC '08, 2008, pp. 310–319.

[8] ——, "Fuzzy-rough nearest neighbour classification and prediction," *Theoretical Computer Science*, vol. 412, pp. 5871–5884, 2011.

[9] C. Cornelis, M. De Cock, and A. M. Radzikowska, "Vaguely quantified rough sets," in *Proceedings of the 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, 2007, pp. 87–94.

[10] C. Cornelis, N. Verbiest, and R. Jensen, "Ordered weighted average based fuzzy rough sets," in *Proceedings of the 5th International Conference on Rough Sets and Knowledge Technology (RSKT 2010)*, 2010, pp. 78–85.

[11] R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183 –190, 1988.

[12] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, no. 6, pp. 80–83, 1945.