Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Sets and Systems 158 (2007) 2134–2152
FUZZY sets and systems
www.elsevier.com/locate/fss

Fuzzy-rough nearest neighbor algorithms in classification

Manish Sarkar*

Department of Computer Science and Engineering, 191 Auditorium Road, U-155, University of Connecticut, Storrs, CT 06269-3155, USA

Received 16 December 2005; received in revised form 6 April 2007; accepted 6 April 2007
Available online 22 May 2007

Abstract

In this paper, classification efficiency of the conventional *K*-nearest neighbor algorithm is enhanced by exploiting fuzzy-rough uncertainty. The simplicity and nonparametric characteristics of the conventional *K*-nearest neighbor algorithm remain intact in the proposed algorithm. Unlike the conventional one, the proposed algorithm does not need to know the optimal value of *K*. Moreover, the generated class confidence values, which are interpreted in terms of *fuzzy-rough ownership* values, do not necessarily sum up to one. Consequently, the proposed algorithm can distinguish between equal evidence and ignorance, and thus the semantics of the class confidence values becomes richer. It is shown that the proposed classifier generalizes the conventional and fuzzy KNN algorithms. The efficacy of the proposed approach is discussed on real data sets.

Keywords: *K*-nearest neighbor; Classifiers; Crisp; Rough; Fuzzy; Rough-fuzzy and fuzzy-rough

1. Introduction

1.1. Motivation

Most of the pattern classification techniques with numerical inputs can be classified into the following three groups: (i) *parametric*, (ii) *semiparametric* and (iii) *nonparametric* [5]. All these three techniques use a set of *training patterns* that already have class labels. The parametric and semiparametric classifiers need certain amount of *a priori* information about the structure of the data in the training set. In many cases, it is difficult to collect this kind of structural information. Then the nonparametric classification technique like the *K-nearest neighbor* (*KNN*) algorithm [11] becomes an attractive approach. It assigns the class label to the input pattern based on the class labels that majority of the *K*-closest (in some distance sense) neighbors from the training set posses. The advantages of the algorithm are (i) it is simple to implement, (ii) it works fast for small training sets, (iii) it does not need any *a priori* knowledge about the structure of the training set, and (iv) its performance asymptotically approaches the performance of the Bayes classifier.

* Tel.: +1 860 486 2584; fax: +1 860 486 4817.
E-mail address: manish_sarkar@hotmail.com.

0165-0114/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2007.04.023

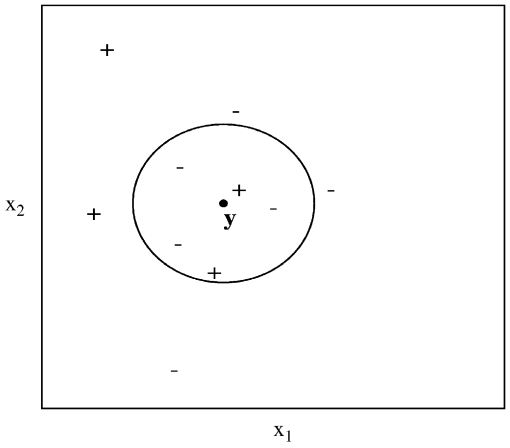

Fig. 1. The conventional 1-nearest (i.e., $K = 1$) neighbor algorithm will classify the test pattern $\boldsymbol{y}$ as positive, whereas the conventional 5-nearest neighbor algorithm will classify $\boldsymbol{y}$ as negative. Thus, the resultant classification depends on the choice of K.

The classification performance of the KNN algorithm usually varies significantly with different values of K (Fig. 1). The value of K implicitly indicates the space of the neighborhood around the test pattern. When K is one, the class label of the test pattern is determined just based on the class label of the closest neighbor. This scheme suffers if the class label of the closest neighbor is corrupted by noise. On the other hand, when K is equal to the total number of the training patterns, the information becomes too global, and the class label of the test pattern is determined based on the *a priori* probability of the classes. The large value of K may increase the classification efficiency as there are more bodies of evidence to classify the test pattern. But if the neighborhood is large, then the neighbors may belong to more than one class. It happens especially in the region where two classes overlap, or noise is present. Thus, it may increase the confusion in assigning the class label to the test pattern. Therefore, the optimal value of K can only be found by a trade-off, which is currently achieved using trial and error procedures.

Another drawback of the conventional KNN algorithm is that it provides the classification result for any pattern that can be fed to the algorithm. For instance, suppose each training pattern is a three-dimensional vector, and the number of classes is five. If an absurd, numerical and three-dimensional input pattern is fed to the KNN algorithm, then the algorithm classifies the input into any one of the five output classes. Strictly speaking, the algorithm should show all class confidence values zero to indicate that the input does not belong to any class. Surprisingly, the human brain possesses the ability to indicate what it can classify and what it cannot. For instance, if an observer knows only Caucasian and East Indian people, and if he sees some Chinese man, then he can readily point out that the Chinese man does not belong to any of the known classes. The conventional KNN algorithm lacks this characteristic, which is also known as the *possibilistic* classification ability [19].

The lack of possibilistic classification ability also affects the classification efficiency of the KNN algorithm. If a test pattern is far away from some of its neighbors, then those neighbors are not *similar* to the test pattern. It is more likely that the class label of such neighbors would be different from the class label of the test pattern, and hence those neighbors should not be allowed to decide the class label of the test pattern. However, in the conventional KNN algorithm, the use of the *relative closeness* of the neighbors results considerable influence from the furthest neighbor. Consequently, in the region where the noise is present or the classes are overlapping, the relative closeness or the *sharing* factor affects the classification result of the conventional KNN algorithm.

1.2. Problem definition

This paper attempts to generalize the conventional KNN algorithm so that (i) the classification efficiency becomes higher, (ii) the classification output acquires richer semantics, that is, it has the possibilistic interpretation, (iii) we do not need to find the optimal value of K by trial and error, and (vi) the simplicity, time complexity and model-free characteristics of the conventional KNN algorithm remain intact.

1.3. Related works

Several techniques have been reported in the literature to address some of the above issues [22,16,7]. Among them one popular algorithm, known as the *fuzzy KNN algorithm*, exploits the fuzzy uncertainty to enhance the classification result of the conventional KNN algorithm [16]. But, like the conventional one, the fuzzy KNN algorithm also collects the evidence from K-closest neighbors. Consequently, fuzzy KNN algorithms also need a difficult trade-off to choose the proper value of K.

In the conventional KNN algorithm, all neighbors receive equal importance. In the fuzzy KNN algorithm, the importance of a neighbor is determined based on the relative distance between the neighbor and the test pattern. Thus, a neighbor that is far away from the test pattern also receives considerable importance, although strictly speaking, the importance associated with this neighbor should be close to zero no matter what the other neighbors are. Hence, in the proposed algorithm, this relative distance is modified to the absolute distance. Consequently, the close neighbors have significant influence on the test pattern, and this influence decreases as the distance increases. Thus, the closeness property is captured as *fuzzy typicality*, rather than the *fuzzy sharing* as done in the fuzzy KNN algorithm. This sharing property does not allow the fuzzy KNN algorithm to have the possibilistic classification ability. In contrast, due to the fuzzy typicality, the proposed algorithm possesses the possibilistic classification capability. Instead of considering K-closest training patterns as the neighbors, the proposed algorithm considers all training patterns as the neighbors with different degrees. The degree depends on the fuzzy typicality of the training pattern. Thus, the proposed algorithm avoids the problem of choosing the optimal value of K.

There exist a few papers that have investigated the role of fuzzy-rough uncertainty in KNN algorithm. For example, in [3,4] a fuzzy-rough version of KNN algorithm was proposed. We would observe later that this approach suffers when the training pattern is noisy. In addition, to apply this technique, one has to have some prior ideas about the number of clusters the training set has.

1.4. Overview of the work

In this paper, we investigate the role of fuzzy and rough uncertainties in the principle of KNN, and subsequently we use these uncertainties to overcome some of the drawbacks of the conventional KNN algorithm. When the classes are overlapping, each neighbor may belong to more than one output class. In other words, the class confidence values of each neighbor can be fuzzy. Apart from that, when the neighbors are close, the input representations of all neighbors are similar. As a result, it is expected that the output class confidence values for all these neighbors would be similar. But, due to insufficient features, the neighbors may look similar, although they are not similar when the other features that are not accounted for are augmented. Hence, the class labels associated with two similar neighbors may be entirely different. Consequently, in the neighborhood region the relationship between the input features and the class labels may be a one-to-many mapping, and thus, the rough-uncertainty is generated. Obviously, the rough uncertainty can be completely avoided if we can successfully extract the essential features so that distinct feature vectors are used to represent the neighbors. But, it may not be possible to guarantee since our knowledge about the system generating the data is limited.

Therefore, the class labeling associates two kinds of uncertainties: (a) the fuzzy uncertainty due to overlapping classes and (b) the rough uncertainty due to insufficient features, which is actually due to the insufficient knowledge about the classification system. To model the situation, where both vagueness and approximation are present in the class labeling, and the fuzziness is present in the closeness of the neighbors, the concept of *fuzzy-rough* sets [9,10] can be employed. In this spirit, the output of the proposed algorithm is interpreted as a *fuzzy-rough ownership value*, which tries to capture both fuzzy and rough uncertainties, and also has the possibilistic classification capability. The proposed algorithm is referred to as fuzzy-rough nearest neighbor (FRNN) algorithm. Since we do not need to select the optimal value of K, the name of the algorithm does not contain any term K.

1.5. Organization

Section 2 describes the background ideas about fuzzy sets, rough sets, rough-fuzzy sets, fuzzy-rough sets and fuzzy KNN algorithm. Section 3 discusses the proposed method. The results are illustrated in Section 4. Finally the paper is concluded in Section 5.

2. Background

2.1. Fuzzy classification

In traditional two-state classifiers, where a class $\mathcal{C}_c$ is defined as a subset of the universal set $\boldsymbol{X}$, any input pattern $\boldsymbol{x} \in \boldsymbol{X}$ can either be a member or not be a member of the given class $\mathcal{C}_c$. This property of whether or not a pattern $\boldsymbol{x}$ of the universal set belongs to the class $\mathcal{C}_c$ can be defined by a *characteristic function* $\mu_{\mathcal{C}_c} : \boldsymbol{X} \to \{0, 1\}$ as follows:

$$\mu_{\mathcal{C}_c}(\boldsymbol{x}) = \begin{cases} 1 & \text{if and only if } \boldsymbol{x} \in \mathcal{C}_c, \\ 0 & \text{if and only if } \boldsymbol{x} \notin \mathcal{C}_c. \end{cases} \tag{2.1.1}$$

In real-life situations, boundaries between the classes may be overlapping. Hence, it is uncertain whether an input pattern belongs totally to the class $\mathcal{C}_c$. To consider such situations, in fuzzy sets [1] the concept of the characteristic function has been modified to the fuzzy *membership function* $\mu_{\mathcal{C}_c} : \boldsymbol{X} \to [0, 1]$.

The training of a C-class classifier for a set of input patterns $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is an assignment of the membership values $\mathcal{C}_c$ on each $\boldsymbol{x}_i \in \mathcal{X}$, $\forall c = 1, 2, \ldots, C$, $\forall i = 1, 2, \ldots, n$. If the membership values are crisp, then $\mathcal{X}$ is partitioned into C subgroups during the training process. In the fuzzy context, C subgroups of $\mathcal{X}$ are the set of values $\{\mu_{\mathcal{C}_c}(\boldsymbol{x})\}$ that can be conveniently arranged on a $C \times n$ matrix $U = [\mu_{\mathcal{C}_c}(\boldsymbol{x})]$. Based on the characteristic of U, the classification can be of the following three types [23]:

1. Crisp classification:

$$M_{\mathrm{hc}} = \left\{ U \in \Re^{Cn} | \mu_{\mathcal{C}_c}(\boldsymbol{x}) \in \{0, 1\}\ \forall c,\ \forall i; \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) = 1; 0 < \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) < n\ \forall c \right\}. \tag{2.1.2a}$$

2. Constrained fuzzy classification:

$$M_{\mathrm{fc}} = \left\{ U \in \Re^{Cn} | \mu_{\mathcal{C}_c}(\boldsymbol{x}) \in [0, 1]\ \forall c, \forall i; \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) = 1; 0 < \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) < n\ \forall c \right\}. \tag{2.1.2b}$$

3. Possibilistic classification:

$$M_{\mathrm{pc}} = \left\{ U \in \Re^{Cn} | \mu_{\mathcal{C}_c}(\boldsymbol{x}) \in [0, 1]\ \forall c, \forall i; 0 < \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) < n \forall c \right\}. \tag{2.1.2c}$$

It is obvious $M_{\mathrm{hc}} \subseteq M_{\mathrm{fc}} \subseteq M_{\mathrm{pc}}$.

Fig. 2(a) and (b) examine the role played by the crisp, constrained fuzzy and possibilistic classification approaches in a 2-class problem. In Fig. 2(a), the crisp membership values of pattern A in both classes are either 0 or 1. The constrained fuzzy membership values of the pattern A in both classes are about 0.5. On the other hand, the possibilistic membership values of the pattern A in both classes are 1 since it belongs to both classes completely. In Fig. 2(b), both patterns A and B are equidistant from the two classes. In the crisp classification, the membership value of A in one class will be 1, and in the other class it will be 0. It is true for the pattern B also. Obviously, this kind of membership assignment does not reflect the actual classification situation, since A and B belong to both classes. In the constrained fuzzy membership assignment, both patterns A and B will have membership values equal to 0.5. Although this membership assignment is better than the crisp counterpart, it fails to consider the pattern A as a more typical one than the pattern B. It is because here the membership assignment is a relative one, and it depends on the membership values to both classes. In the possibilistic membership assignments, the pattern A will receive equal membership values to both classes. It is true for B also. But, the membership of B to any class will be always less than that of pattern A. Therefore, the possibilistic assignment may not be summed up to one, and hence, it can distinguish between *equal evidence* and *ignorance* [32,19,29,8]. For instance, if the sum of the membership assignments for a pattern is close to zero, then possibly the pattern does not belong to any of the given classes. In other words, we are ignorant about the class label corresponding to the pattern. It is important in many fields, e.g., in medical diagnosis, where the patient may suffer from some unknown disease, or the symptoms are not sufficient enough for the diagnosis. Thus, the possibilistic assignment is more attractive compared to the crisp and constrained fuzzy membership assignments.

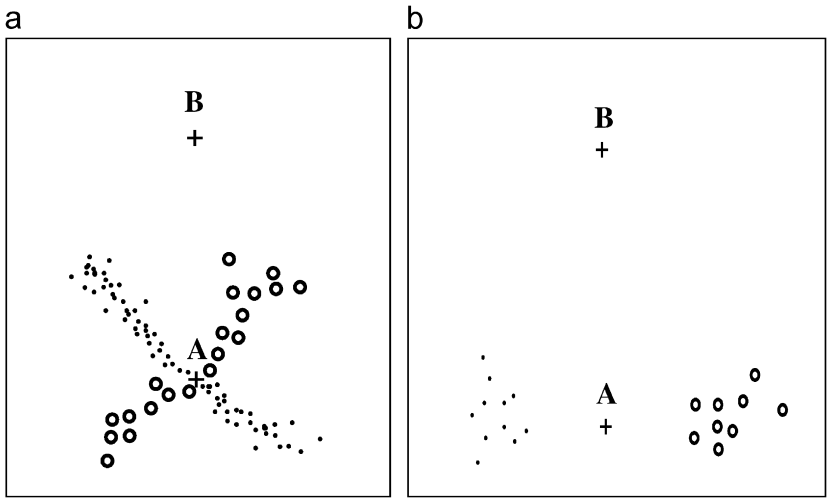

Fig. 2. In (a), possibilistic membership assignments would be able to indicate that (i) A belongs to both classes completely and equally, and (ii) B belongs to both classes partially and equally. In (b), possibilistic membership assignments would be able to show that (i) A belongs to both classes equally (ii) B belongs to both classes equally and (iii) from both classes, B is further away than A. Crisp and constrained fuzzy memberships would fail to convey this kind of information.

2.2. Rough sets

Let R be an equivalence relation on a universal set $\boldsymbol{X}$. Moreover, let $\boldsymbol{X}/\mathrm{R}$ denote the family of all equivalence classes induced on $\boldsymbol{X}$ by R [17,18,25]. One such equivalence class in $\boldsymbol{X}/\mathrm{R}$ that contains $\boldsymbol{x}$ is designated by $[\boldsymbol{x}]_R$. For any output class $\mathcal{C}_c \subseteq \boldsymbol{X}$, we can define the lower $\underline{R}(\mathcal{C}_c)$ and upper $\overline{R}(\mathcal{C}_c)$ approximations, which approach $\mathcal{C}_c$ as closely as possible from the inside and outside, respectively [25]. Here,

$$\underline{R}(\mathcal{C}_c) = \cup\{[\boldsymbol{x}]_R | [\boldsymbol{x}]_R \subseteq \mathcal{C}_c \text{ and } \boldsymbol{x} \in \boldsymbol{X}\} \tag{2.2.1a}$$

is the union of all equivalence classes in $\boldsymbol{X}/\mathrm{R}$ that are contained in $\boldsymbol{\mathcal{C}}_c$ and

$$\overline{R}(\mathcal{C}_c) = \cup\{[\boldsymbol{x}]_R | \cap \mathcal{C}_c \neq \phi \text{ and } \boldsymbol{x} \in \boldsymbol{X}\} \tag{2.2.1b}$$

is the union of all equivalence classes in $\boldsymbol{X}/\mathrm{R}$ that overlap with $\mathcal{C}_c$. The *rough set* $R(\mathcal{C}_c) = \langle \overline{R}(\mathcal{C}_c), \underline{R}(\mathcal{C}_c) \rangle$ is a representation of the given set $\boldsymbol{\mathcal{C}}_c$ by $\underline{R}(\mathcal{C}_c$ and $\overline{R}(\mathcal{C}_c)$. The set $\overline{R}(\mathcal{C}_c)$–$\underline{R}(\mathcal{C}_c)$ is a rough description of the boundary of $\mathcal{C}_c$ by the equivalence classes of $\boldsymbol{X}/\mathrm{R}$. The approximation is rough uncertainty free if $\underline{R}(\mathcal{C}_c) = \overline{R}(\mathcal{C}_c)$. Thus, when all patterns from an equivalence class do not carry the same output class label, the rough uncertainty is generated as a manifestation of the one-to-many relationship between that equivalence class and the output class labels [10].

2.3. Fuzzy-rough sets

A *rough-fuzzy set* [9,10] is a generalization of the rough set in the sense that here the output class is fuzzy [31]. Let $\boldsymbol{X}$ be a set, R be an equivalence relation defined on $\boldsymbol{X}$, and the output class $\mathcal{C}_c \subseteq \boldsymbol{X}$ be a fuzzy set. The rough-fuzzy set is a tuple $\langle \overline{R}(\mathcal{C}_c), \underline{R}(\mathcal{C}_c) \rangle$, where the lower approximation $\underline{R}(\mathcal{C}_c)$ and the upper approximation $\overline{R}(\mathcal{C}_c)$ are fuzzy sets of $\boldsymbol{X}/\mathrm{R}$, with membership functions defined by [9,10]

$$\mu_{\underline{R}(\mathcal{C}_c)}([\boldsymbol{x}]_R) = \inf\{\mu_{\mathcal{C}_c}(\boldsymbol{x}) | \boldsymbol{x} \in [\boldsymbol{x}]_R\} \quad \forall \boldsymbol{x} \in \boldsymbol{X}, \tag{2.3.1a}$$

and

$$\mu_{\overline{R}(\mathcal{C}_c)}([\boldsymbol{x}]_R) = \sup\{\mu_{\mathcal{C}_c}(\boldsymbol{x}) | \boldsymbol{x} \in [\boldsymbol{x}]_R\} \quad \forall \boldsymbol{x} \in \boldsymbol{X}. \tag{2.3.1b}$$

Here, $\mu_{\underline{R}(\mathcal{C}_c)}([\boldsymbol{x}]_R)$ and $\mu_{\overline{R}(\mathcal{C}_c)}([\boldsymbol{x}]_R)$ are the membership values of $[\boldsymbol{x}]_R$ in $\underline{R}(\mathcal{C}_c)$ and $\overline{R}(\mathcal{C}_c)$, respectively.

A *fuzzy-rough set* is a further generalization of the rough-fuzzy set. When the equivalence classes are not crisp, they are in the form of fuzzy clusters $F_1, F_2, \ldots, F_H$ generated by a *fuzzy weak partition* [9,10] of the input set $\boldsymbol{X}$. Here H is the number of clusters. The term fuzzy weak partition means that each F_j is a normal fuzzy set, i.e.,

$$\sup_{\boldsymbol{x}}\{\mu_{F_j}(\boldsymbol{x})\} = 1 \quad \text{and} \quad \inf_{\boldsymbol{x}}\left[\max_{j}\{\mu_{F_j}(\boldsymbol{x})\}\right] > 0, \tag{2.3.2}$$

while

$$\sup_{\boldsymbol{x}} \min_{F_i}\{\mu_{F_i}(\boldsymbol{x}), \mu_{F_j}(\boldsymbol{x})\} < 1 \quad \forall i,\ j \in \{1, 2, \ldots, H\},\ \ i \neq j. \tag{2.3.3}$$

Here $\mu_{F_j}(\boldsymbol{x})$ is the fuzzy membership function of the pattern $\boldsymbol{x}$ in the cluster F_j. In addition, the output class $\mathcal{C}_c$ may be fuzzy too. Given a weak fuzzy partition $F_1, F_2, \ldots, F_H$ on $\boldsymbol{X}$, the description of any fuzzy set $\mathcal{C}_c$ by means of the fuzzy partitions under the form of a lower and an upper approximation $\underline{\mathcal{C}_c}$ and $\overline{\mathcal{C}_c}$ is as follows:

$$\mu_{\underline{\mathcal{C}_c}}(F_j) = \inf_{\boldsymbol{x}}\{\max(1 - \mu_{F_i}(\boldsymbol{x}), \mu_{\mathcal{C}_c}(\boldsymbol{x}))\} \quad \forall \boldsymbol{x} \in \boldsymbol{X} \tag{2.3.4a}$$

and

$$\mu_{\overline{\mathcal{C}_c}}(F_j) = \sup_{\boldsymbol{x}}\{\min(\mu_{F_i}(\boldsymbol{x}), \mu_{\mathcal{C}_c}(\boldsymbol{x}))\} \quad \forall \boldsymbol{x} \in \boldsymbol{X}. \tag{2.3.4b}$$

The tuple $\langle\overline{\mathcal{C}_c}, \underline{\mathcal{C}_c}\rangle$ is called *fuzzy-rough* set. Here, $\mu_{\mathcal{C}_c}(\boldsymbol{x}) \in [0, 1]$ is the fuzzy membership of the input $\boldsymbol{x}$ to the class $\mathcal{C}_c$. The fuzzy-roughness appears when a fuzzy cluster contains patterns that belong to different classes.

2.4. K-nearest neighbor algorithms

Let $\boldsymbol{X} \subseteq \Re^N$ be the set of all possible input patterns, and $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \subseteq \boldsymbol{X}$ be a set of input training patterns for which the corresponding class labels are already known. In the conventional KNN algorithm, the Euclidean distances between the test pattern and all training patterns are calculated, and the test pattern is assigned the class label that most of the K-closest training patterns have [22]. Let the Euclidean distance between the test pattern $\boldsymbol{y} \in \boldsymbol{X} \subseteq \Re^N$ and the training pattern $\boldsymbol{x}_k \in \mathcal{X}$ be denoted by $\|\boldsymbol{y} - \boldsymbol{x}_k\| = \sqrt{\sum_{i=1}^{N}(y_i - x_{ki})^2}$. Since the number of training patterns is n, n such distances are calculated, and the closest K training patterns are identified as *neighbors*. The output of the conventional KNN algorithm attains a richer semantic when the output is interpreted as *a posteriori probability* [11]. Hence, instead of labeling the output class label equal to the class label that most of the neighbors have, the following class confidence values are assigned to the test pattern $\boldsymbol{y}$:

$$\begin{aligned} o_c(\boldsymbol{y}) &= \frac{1}{K}\ (\text{no. of neighbors with class label } c) \quad \forall c \\ &= \frac{1}{K}\sum_{k=1}^{K}\delta_c(\boldsymbol{x}_k) \quad \forall c, \end{aligned} \tag{2.4.1}$$

where $\delta_c(\boldsymbol{x}_k)$ is the characteristic function corresponding to the kth neighbor, i.e., $\delta_c(\boldsymbol{x}_k) = 1$ if $\boldsymbol{x}_k$ has the class label c, and $\delta_c(\boldsymbol{x}_k) = 0$ otherwise. Here, $o_c(\boldsymbol{y})$ is the *a posteriori* probability that $\boldsymbol{y}$ belongs to the class c. With this formulation, we can still derive the hard decision by assigning the class label j to the test pattern $\boldsymbol{y}$ where $o_j(\boldsymbol{y}) = \max_{1,2,\ldots,C}\{o_c(\boldsymbol{y})\}$ and C is the total number of classes.

In the conventional KNN algorithm, all K neighbors receive equal importance, and the class label of each training pattern is considered crisp. The fuzzy KNN algorithm refines the conventional KNN algorithm

1. by weighing the contribution of each of the K neighbors based on its distance to the test pattern. Evidently, the closest neighbor should receive the highest weight. Hence, the kth closest neighbor is weighted based on its relative distance with respect to all K-closest neighbors. Thus, the relative weight of the kth neighbor $k \in \{1, 2, \ldots, K\}$ is $(1/\|\boldsymbol{y} - \boldsymbol{x}_k\|^{2/(q-1)})/(\sum_{j=1}^{K} 1/\|\boldsymbol{y} - \boldsymbol{x}_j\|^{2/(q-1)})$, where $\|\boldsymbol{y} - \boldsymbol{x}_k\|$ is the Euclidean distance between $\boldsymbol{y}$ and $\boldsymbol{x}_k$, and q determines how strongly the distance is weighted while calculating each neighbor's contribution to the membership value. Here the denominator is used for normalization such that the sum of the weights is equal to 1.

2. by considering that each neighbor may belong to more than one class. It typically happens where the classes overlap. Hence, the crisp class membership of each training pattern, i.e., δ, is modified to the fuzzy membership function μ.

To consider the above two refinements, Eq. (2.4.1) is modified to the following [16]:

$$o_c(\boldsymbol{y}) = \sum_{k=1}^{K} \left(\frac{\dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_k\|^{2/(q-1)}}}{\sum_{j=1}^{K} \dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_j\|^{2/(q-1)}}} \right) \mu_{\mathcal{C}_c}(\boldsymbol{x}_k), \tag{2.4.2}$$

where $\mu_{\mathcal{C}_c}(\boldsymbol{x}_k)$ is the fuzzy membership of the kth neighbor in the cth class. In this case $o_c(\boldsymbol{y})$ is interpreted as the *fuzzy membership function* [18].

In the fuzzy KNN algorithm, the initial membership on each training pattern can be assigned in the following two ways [16]:

1. *Crisp membership*: Each training pattern can have complete membership in their known class and non-memberships in all other classes.
2. *Constrained fuzzy membership*: The K-nearest neighbors of each training pattern (say $\boldsymbol{x}_k$) are found, and the membership of $\boldsymbol{x}_k$ in each class is assigned as

$$\mu_{\mathcal{C}_c}(\boldsymbol{x}_k) = \begin{cases} 0.51 + 0.49 n_j/K & \text{if } j = c, \\ 0.49 n_j/K & \text{if } j \neq c. \end{cases} \tag{2.4.3}$$

The value n_j is the number of the neighbors found that belongs to the jth class. This initialization technique fuzzifies the memberships of the labeled samples that are in the region where classes are overlapping. Moreover, the patterns that are well away from the overlapping area are assigned with the complete membership in the known class. Consequently, a test pattern lying in the overlapping region will be influenced to a lesser extent by the labeled samples that are also in the overlapping area.

The steps of the fuzzy KNN algorithm are similar to that of the conventional KNN algorithm (Fig. 3). The time complexity of the algorithm for testing is $O(nr)$, where n and r are the sizes of the training and test sets, respectively.

The following result shows that the class membership assignments by the fuzzy KNN algorithm are constrained fuzzy. Consequently, the membership values assigned to each test pattern cannot distinguish between equal evidence and ignorance.

Axiom 1. In a C-class classification problem, the class membership assignments on the test patterns (say $\boldsymbol{y}$) by the fuzzy KNN algorithm are constrained fuzzy.

Proof. For the test pattern $\boldsymbol{y}$, if we consider sum of all outputs coming from all classes, we get

$$\begin{aligned} \sum_{c=1}^{C} o_c(\boldsymbol{y}) &= \sum_{c=1}^{C}\sum_{k=1}^{K} \left(\frac{\dfrac{\mu_{\mathcal{C}_c}(\boldsymbol{x}_k)}{\|\boldsymbol{y}-\boldsymbol{x}_k\|^{2/(q-1)}}}{\sum_{j=1}^{K} \dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_j\|^{2/(q-1)}}} \right) \\ &= \sum_{k=1}^{K} \left(\frac{\dfrac{\sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}_k)}{\|\boldsymbol{y}-\boldsymbol{x}_k\|^{2/(q-1)}}}{\sum_{j=1}^{K} \dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_j\|^{2/(q-1)}}} \right). \end{aligned}$$

Since the initialization is either crisp or constrained fuzzy, $\sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}_k) = 1$. Hence,

$$\sum_{c=1}^{C} o_c(\boldsymbol{y}) = \sum_{k=1}^{K} \left(\frac{\dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_k\|^{2/(q-1)}}}{\sum_{j=1}^{K} \dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_j\|^{2/(q-1)}}} \right) = \frac{\sum_{k=1}^{K} \dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_k\|^{2/(q-1)}}}{\sum_{j=1}^{K} \dfrac{1}{\|\boldsymbol{y}-\boldsymbol{x}_j\|^{2/(q-1)}}} = 1. \tag{2.4.5}$$

INPUT: (a) The training set $\mathcal{X}$ with already labeled patterns $\{\boldsymbol{x}_i \mid i = 1, 2, \ldots, n\}$
(b) The test pattern $\boldsymbol{y}$

ALGORITHM:
FOR $i = 1, 2, \ldots,$ to n
Determine the distance between $\boldsymbol{y}$ and $\boldsymbol{x}_i$.
IF $i \leq K$
Include $\boldsymbol{x}_i$ in the set of K-nearest neighbors.
ELSE IF ($\boldsymbol{x}_i$ is closer to $\boldsymbol{y}$ than any previous nearest neighbor)
Delete the farthest of the K-nearest neighbors.
Include $\boldsymbol{x}_i$ in the set of K-nearest neighbors.
END IF
END FOR
FOR $c = 1$ to C

$$o(c) = \frac{1}{K}(\text{no. of neighbors in class } \mathcal{C}_c) \tag{2.4.4}$$

END FOR
Crisp class label of $\boldsymbol{y}$ is j where $o(j) = max\{o(1), o(2), \ldots, o(C)\}$.

OUTPUT: (a) Class label of $\boldsymbol{y}$.
(b) Class confidence values $o(c)\ \forall c$.

Fig. 3. The conventional K-nearest neighbors algorithm. Here C is the total number of classes. If Eq. (2.4.4) is replaced by Eq. (2.4.2), then the resultant algorithm is the fuzzy K-nearest neighbors algorithm.

3. Fuzzy-rough nearest neighbor algorithms

3.1. Characterization of uncertainties

We can identify at least the following two uncertainties in the KNN principle:
Fuzzy uncertainty: It may arise due to the following two factors:

1. *How similar the test input and the neighbor is*: The similarity decreases as the distance between the test pattern and the neighbor increases. This similarity can be quantified in the form of fuzzy membership functions.
2. *How typical each neighbor is for a class*: The neighbor can partially belong to more than one class where the classes are overlapping. It is also possible that the neighbor does not belong to any class or belong to all classes with complete memberships. This kind of belongingness can be represented by possibilistic membership functions.

Rough uncertainty: It may appear due to the following two reasons:

1. *Neighbors and the test pattern are indistinguishable due to the lack of features*: When the spatial representations of all neighbors are similar, it is expected that the output class labels should also be similar. Due to the incomplete knowledge about the classification process, generally the input representation is not perfect. As a result, the test pattern and the neighbor, based on the available features appear similar, although they are not similar when the other features that are not accounted for are augmented. It makes the input–output relationship one-to-many, and thus the rough uncertainty appears. In some contexts these kinds of data sets are called *noisy* data sets.
2. *Neighborhood represents artificial similarity*: Using the term 'neighborhood', we usually consider a region or structure around the test pattern. This structure is not naturally owned by the data; rather it is artificial. In contrast, when we cluster a set of data, we obtain a natural structure. In a cluster, two members are close or similar to each other based on some criterion. In the KNN principle, using a forcibly assigned neighborhood, we assume that the neighbors and the test pattern are *similar*, and hence they should have similar class labels. In other words, in this case we forcibly call a training pattern neighbor since it is closer to the test pattern relative to some other training patterns, although from the spatial distance and distribution points of views they may not be close. When two such patterns

are artificially treated as similar, although their class assignments are quite different, a one-to-many relationship is created between the spatial similarity and the class labels. Consequently, the roughness emerges.

In the next subsection, we show how to exploit the fuzzy and rough uncertainties.

3.2. Exploitation of uncertainties

In the conventional KNN algorithm, the neighborhood is chosen in such a manner that it contains K-closest neighbors. In a dense region, this neighborhood region is small since the K-closest neighbors will be found within a close distance. Similarly, in a sparse region, this neighborhood will occupy a larger space. It implies that we are trying to construct a structure $W \subseteq \Re^N$ around the test pattern $\boldsymbol{y}$. If all neighbors are from a single class $\mathcal{C}_i$, then there is no uncertainty in the structure. Any test pattern that resides in the structure can be assigned to $\mathcal{C}_i$. However, if any neighbor belongs to another class $\mathcal{C}_j$, then the rough uncertainty arises in the structure. Although the neighbors are similar from the feature's perspective, they are not similar from the class label's perspective. It makes the relationship between the input representation and output class labels one-to-many. This uncertainty can be captured using the *rough ownership function* $r_{\mathcal{C}_c} : X \to [0, 1]$. The rough ownership function for the test pattern $\boldsymbol{y} \in \boldsymbol{X}$ in the output class $\mathcal{C}_c$ is defined by

$$r_{\mathcal{C}_c}(\boldsymbol{y}) = \frac{|W \cap \mathcal{C}_c|}{|W|}, \tag{3.2.1}$$

where W is the neighborhood region around $\boldsymbol{y}$ and $|\mathcal{C}_c|$ denotes the cardinality of the set $\mathcal{C}_c$. We do not call the definition (3.2.1) rough membership function [25] because it does not signify to what extent the test pattern is a member of a natural structure; rather it denotes to what extent the test pattern owns the artificial structure.

Now we make the whole situation slightly more complicated. Each neighbor can belong to more than one class, i.e., the class memberships are fuzzy. To accommodate both fuzzy and rough uncertainties, the rough ownership function is modified to the *rough-fuzzy ownership function*. The rough-fuzzy ownership function of a pattern $\boldsymbol{y} \in \boldsymbol{X}$ for the fuzzy output class $\mathcal{C}_c \subseteq \boldsymbol{X}$ is defined by

$$\imath_{\mathcal{C}_c}(\boldsymbol{y}) = \frac{|W \cap \mathcal{C}_c|}{|W|}, \tag{3.2.2}$$

where $|\mathcal{C}_c|$ means the cardinality of the fuzzy set $\mathcal{C}_c$. One way to determine the cardinality is to use [32]: $\mathcal{C}_c \stackrel{\text{def}}{=} \sum_{\boldsymbol{x}\in X} \mu_{\mathcal{C}_c}(\boldsymbol{x})$. In this case intuitively the fuzzy-rough ownership value is the volume occupied by the neighborhood structure with the training patterns from $\mathcal{C}_c$ divided by the complete volume of the structure. For the '$\cap$' (intersection) operation, we can use $\mu_{W\cap\mathcal{C}_c}(\boldsymbol{x}) = \mu_W(\boldsymbol{x})\mu_{\mathcal{C}_c}(\boldsymbol{x})\forall \boldsymbol{x} \in \mathcal{X}$. Here $\mu_W(\boldsymbol{x})$ is the fuzzy membership of $\boldsymbol{x}$ belonging to the structure W. Since W is crisp, $\mu_W(\boldsymbol{x}) \in \{0, 1\}$. Then,

$$\imath_{\mathcal{C}_c}(\boldsymbol{y}) = \frac{\sum_{\boldsymbol{x}\in X}\ \mu_W(\boldsymbol{x})\mu_{\mathcal{C}_c}(\boldsymbol{x})}{|W|} = \frac{1}{|W|}\sum_{\boldsymbol{x}\in W}\mu_{\mathcal{C}_c}(\boldsymbol{x}). \tag{3.2.3}$$

Next we make the situation even more complex, but closer to the reality. Till now we have assumed that each neighbor resides in the structure W equally and completely. Now each training pattern belongs to W with different degrees, i.e., the training pattern closer to the test pattern belongs to the neighborhood W to a high degree, and the training pattern far away from W supports the neighborhood structure by negligible amount. Therefore, W spans all training patterns, i.e., $\mathcal{X}$. The structure becomes very compact when all neighbors are very close to $\boldsymbol{y}$, and they all belong to the same class. Hence, we need to assign a neighbor a large weight if it forms a compact region. It can be satisfied using the concept of *fuzzy similarity* $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ between $\boldsymbol{y}$ and $\boldsymbol{x}$. The fuzzy similarity can be determined by finding the absolute distance between the test pattern and the neighbor. When we incorporate the concept of fuzzy similarity in the rough-fuzzy ownership function (Eq. (3.2.3)), it becomes the following *fuzzy-rough ownership function*:

$$\tau_{\mathcal{C}_c}(\boldsymbol{y}) = \tfrac{1}{|\mathcal{X}|}\sum_{\boldsymbol{x}\in\mathcal{X}}\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})\mu_{\mathcal{C}_c}(\boldsymbol{x}). \tag{3.2.4}$$

Although there are many possible t-norms that we could use in (3.2.3), we used an interactive t-norm operation. For example, if $\mu_W(\boldsymbol{x})$ or $\mu_{\mathcal{C}_c}(\boldsymbol{x})$ varies slightly in $\mu_{W\cap\mathcal{C}}(\boldsymbol{x}) = \mu_W(\boldsymbol{x})\mu_{\mathcal{C}_c}(\boldsymbol{x})\forall \boldsymbol{x} \in X$, then the value of $\mu_{W\cap C}(\boldsymbol{x})$ changes slightly. In contrast, if we choose a non-interactive t-norm operation like $\mu_{W\cap\mathcal{C}_c}(\boldsymbol{x}) = \min\{\mu_W(\boldsymbol{x}), \mu_{\mathcal{C}_c}(\boldsymbol{x})\}\forall \boldsymbol{x} \in \mathcal{X}$, then $\mu_{W\cap\mathcal{C}}(\boldsymbol{x})$ remains insensitive to any changes of $\mu_W(\boldsymbol{x})$ as long as $\mu_W(\boldsymbol{x}) \geqslant \mu_{\mathcal{C}_c}(\boldsymbol{x})$. Hence to keep $\iota_{\mathcal{C}_c}(\boldsymbol{y})$ smooth, we preferred interactive t-norms, like the one used in (3.2.3).

Before using the fuzzy-rough ownership function in the proposed algorithm, we show that the fuzzy-rough ownership function is able to quantify the fuzzy-roughness present in the data, and the quantified value lies in [0, 1].

Property 1. $0 \leqslant \tau_{\mathcal{C}_c}(\boldsymbol{y}) \leqslant 1$.

Proof. Since $0 \leqslant \mu_{\mathcal{C}_c}(\boldsymbol{x}) \leqslant 1$ and $0 \leqslant \tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) \leqslant 1$, the proof is trivial. □

Property 2. $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = 1$ or 0 if and only if no fuzzy-rough uncertainty is associated with the pattern $\boldsymbol{x}$.

Proof. *If part*: If no fuzzy-rough uncertainty is involved, then there does not exist any fuzziness. Hence, all neighbors must be very close to the test pattern $\boldsymbol{y}$ and each neighbor should belong to only one class. It implies that $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = 1\forall \boldsymbol{x}$, $\mu_{\mathcal{C}_c}(\boldsymbol{x}) = 1$ for some $j \in \{1, 2, \ldots, C\}$ and $\mu_{\mathcal{C}_c}(\boldsymbol{x}) = 0\forall c \neq j$. Since the absence of fuzzy-roughness also implies the absence of roughness, all neighbors must belong to the same class (say $\mathcal{C}_j$). Hence, $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = (1/|\mathcal{X}|)\sum_{j=1}^{|\mathcal{X}|} 1.1 = 1$ for $c = j$ and $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = 0$ for $c \neq j$.

Only if part: $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = 0$ in Eq. (3.2.4) implies that each term under the summation symbol, i.e., $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})\mu_{\mathcal{C}_c}(\boldsymbol{x})$ is separately zero. It means that either $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ or $\mu_{\mathcal{C}_c}(\boldsymbol{x})$ is zero, or both $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ and $\mu_{\mathcal{C}_c}(\boldsymbol{x})$ are zero for all $\boldsymbol{x}$. If $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = 0$, then $\boldsymbol{x}$ is very far away from $\boldsymbol{y}$. In that case there is no fuzzy-rough uncertainty. If $\mu_{\mathcal{C}_c}(\boldsymbol{x}) = 0$, then also no fuzzy-rough uncertainty exists as long as the class $\mathcal{C}_c$ is concerned. Thus, $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = 0$ indicates that the fuzzy-roughness is not present.

If $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = 1$, then both $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = 1$ and $\mu_{\mathcal{C}_c}(\boldsymbol{x}) = 1\forall \boldsymbol{x} \in \mathcal{X}$. It also indicates the absence of the fuzzy-roughness. □

The following three properties show that the rough-fuzzy ownership functions, rough ownership functions and fuzzy membership functions are the particular cases of the fuzzy-rough ownership functions.

Property 3. If $W \subseteq \mathcal{X}$ is crisp, then $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = \iota_{\mathcal{C}_c}(\boldsymbol{y})$.

Proof. Crisp W implies that $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ is crisp so that $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = 1$ when $\boldsymbol{x} \in W$ and $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = 0$ when $\boldsymbol{x} \notin W$. Hence, the proof comes directly from the definition (3.2.3) and (3.2.4). □

Property 4. If both $W \subseteq \mathcal{X}$ and $\mathcal{C}_c$ are crisp, then $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = r_{\mathcal{C}_c}(\boldsymbol{y})$.

Proof. Proof comes directly from the definition (3.2.1) and (3.2.4). □

Property 5. If $W \subseteq \mathcal{X}$ is crisp and $|W| = 1$, then $\tau_{\mathcal{C}_c}(\boldsymbol{y}) = \mu_{\mathcal{C}_c}(\boldsymbol{x})$, where $\boldsymbol{x}$ is the closest neighbor.

Proof. Proof comes directly from definition (3.2.4). □

Note that the output of the conventional KNN algorithm can be interpreted as the rough ownership function. Similarly, the output of the fuzzy KNN algorithm can be interpreted as a restricted form of $\tau_{\mathcal{C}_c}(\boldsymbol{y})$. The restriction is imposed by keeping (a) $K \leqslant |\mathcal{X}|$ and (b) $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ constrained fuzzy. Both conditions (a) and (b) are related because the sharing constraint in $\tilde{\mu}_{\boldsymbol{y}}(\mathbf{x})$ depends on K.

Next we prove that the fuzzy-rough ownership function is possibilistic, and hence we design the FRNN algorithm such that the outputs of the algorithm are the fuzzy-rough ownership values.

Axiom 2. For a C-class classification problem, the fuzzy-rough ownership function assignments are possibilistic even if the initial class memberships for the training patterns are crisp or constrained fuzzy.

Proof. For the test pattern $\boldsymbol{y}$,

$$\sum_{c=1}^{C} \tau_{\mathcal{C}_c}(\boldsymbol{y}) = \sum_{c=1}^{C} \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}\in\mathcal{X}} \tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) \mu_{\mathcal{C}_c}(\boldsymbol{x})$$
$$= \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}\in\mathcal{X}} \tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}).$$

If the initial class membership values for the training patterns are crisp or constrained fuzzy, then $\sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) = 1$. Also $0 \leqslant \tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) \leqslant 1$, and hence, $0 \leqslant \sum_{c=1}^{C} \tau_{\mathcal{C}_c}(\boldsymbol{y}) \leqslant 1$. Since $\sum_{c=1}^{C} \tau_{\mathcal{C}_c}(\boldsymbol{y})$ is not a constant, the resultant classification procedure is *possibilistic*.

If the initial class membership values for the training patterns are possibilistic, then $0 \leqslant \sum_{c=1}^{C} \mu_{\mathcal{C}_c}(\boldsymbol{x}) \leqslant C$. Moreover, $0 \leqslant \tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) \leqslant 1$. Hence, $0 \leqslant \sum_{c=1}^{C} \tau_{\mathcal{C}_c}(\boldsymbol{y}) \leqslant C$. Thus, the resultant classification procedure is *possibilistic* [18,23], and it can distinguish between equal evidence and ignorance. For example, for an isolated test pattern, we may observe $\tau_{\mathcal{C}_c}(\boldsymbol{y}) \approx 0$ for all c, and hence $\sum_c \tau_{\mathcal{C}_c}(\boldsymbol{y}) \approx 0$. In contrast, if $\boldsymbol{y}$ is from a dense region, then $\sum_c \tau_{\mathcal{C}_c}(\boldsymbol{y})$ may be more than 1. □

In the next subsection we design the FRNN algorithm such that its output is the fuzzy-rough ownership value.

3.3. Algorithm

The steps of the FRNN algorithm are very similar to that of the fuzzy KNN algorithm. The initial memberships of the training patterns can be assigned as crisp, constrained fuzzy or possibilistic. It depends on how much domain-specific knowledge the designer has. Unlike the fuzzy KNN algorithm, we do not need to fix K since we are using all training patterns as neighbors.

One way to determine $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ is by $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = 1/(1 + \kappa\|\boldsymbol{y} - \boldsymbol{x}\|^{2/(q-1)})$, where $\|\boldsymbol{y} - \boldsymbol{x}\|$ is the Euclidian distance between $\boldsymbol{y}$ and $\boldsymbol{x}$, and κ is a parameter that decides the bandwidth of the membership, i.e., the point at which $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ attains the value 0.5 [33]. Here $q \in (1, \infty)$ determines the shape of the membership function $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$. When q approaches one, the membership function tends to be a crisp membership function with a very sharp slope. When q approaches infinity, the slope is almost flat, and the membership function is maximally fuzzy. The role of q here is quite similar to the index of fuzziness in the concentration and dilation operators found in *fuzzy hedge* [17], and the index of fuzziness in *fuzzy C-means clustering algorithm* [1]. Another way to determine $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ is by using a Gaussian, i.e., $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x}) = \exp(-\kappa\|\boldsymbol{y} - \boldsymbol{x}\|^{2/(q-1)})$. Here also κ and q decide the shape of $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$.

There are at least two proposals to choose κ. One proposal is to keep κ fixed, and the other one is to make κ proportional to the inverse of the average distance between all neighbors and the test pattern $\boldsymbol{y}$, i.e., $\boldsymbol{\kappa} = 1/(2/|X|) \sum_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{y} - \boldsymbol{x}\|^{2/(q-1)}$. To make κ sensitive to the features, we can use the N-dimensional vector form of $\boldsymbol{\kappa}$, where

$$\boldsymbol{\kappa} = [\kappa_1, \kappa_2, \ldots, \kappa_N]' = \left[\frac{|\mathcal{X}|}{2\sum_{\boldsymbol{x}\in\mathcal{X}} \|y_1 - x_1\|^{2/(q-1)}}, \frac{|\mathcal{X}|}{2\sum_{\boldsymbol{x}\in\mathcal{X}} \|y_2 - x_2\|^{2/(q-1)}}, \ldots, \frac{|\mathcal{X}|}{2\sum_{\boldsymbol{x}\in\mathcal{X}} \|y_N - x_N\|^{2/(q-1)}} \right]'.$$

Then, Eq. (3.2.4) can be rewritten as

$$o_c(\boldsymbol{y}) = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}\in\mathcal{X}} \left[\frac{\mu_{\mathcal{C}_c}(\boldsymbol{x})}{1 + \sum_{i=1}^{N} \kappa_i (y_i - x_i)^{2/(q-1)}} \right] \tag{3.3.1}$$

or

$$o_c(\boldsymbol{y}) = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x}\in\mathcal{X}} \left[\mu_{\mathcal{C}_c}(\boldsymbol{x}) \exp\left(-\sum_{i=1}^{N} \kappa_i (y_i - x_i)^{2/(q-1)} \right) \right]. \tag{3.3.2}$$

The proposed algorithm is shown in Fig. 4.

INPUT: (a) Training data $\{\boldsymbol{x}_i \mid i = 1, 2, ..., n\}$ with fuzzy class labels.
(b) The test pattern $\boldsymbol{y}$.

ALGORITHM:

Compute $\boldsymbol{\kappa}$.
FOR $c = 1$ to C
Set $o(c)$ as zero
END FOR
FOR $i = 1$ to n
Determine squared weighted distance
$d = \sum_{j=1}^{N} \kappa_j (y_j - x_{ij})^2$ between $\boldsymbol{y}$ and $\boldsymbol{x}_i$.
FOR $c = 1$ to C
L: $o(c) = o(c) + \frac{1}{|\mathcal{X}|} \mu_{\mathcal{C}_c}(\boldsymbol{x}_i) \exp(-d^{1/(q-1)})$
END FOR
END FOR

Crisp class label of $\boldsymbol{y}$ is j where $o(j) = \max\{o(1), o(2), ..., o(C)\}$.

OUTPUT: (a) Class label of $\boldsymbol{y}$.
(b) Class confidence values $o(c)$ $\forall c$.

Fig. 4. The fuzzy-rough nearest neighbors algorithm. Here C is the total number of classes, and $q \in (1, \infty)$ is an index, which controls the fuzziness. Note that the line with the label L represents Eq. (3.3.2), and this line can be modified to represent Eq. (3.3.1).

The FRNN possesses a desirable approximation capability. Since $\exp(\cdot)$ is integrable over $\Re^N$, $\int_{\Re^N} \exp(\boldsymbol{y})\, dy \neq 0$, and $0 \leqslant \mu_{\mathcal{C}_c}(\boldsymbol{x})/|\mathcal{X}| \leqslant 1$, it can be easily shown that there always exists some FRNN model that can approximate (to an arbitrary accuracy) the given continuous classification function that maps any compact domain $X \subseteq \Re^N$ to $\Re^C$ (for details see [28]). This property is also known as the *universal approximation* capability [15].

Note that at first it may appear that since $K \ll n$, the conventional and fuzzy KNN algorithms are computationally cheaper than the FRNN algorithm. Actually all of them have O(nr) worst case time complexity, where n and r are the sizes of the training and test sets, respectively. In the crisp and fuzzy KNN algorithms, we need to compare the distance between all training patterns and the test pattern to find the K-closest neighbors (e.g., when $K = 1$, the distance is computed nr times for each class). Instead of this comparison, in the proposed algorithm we need to do some computations involving all training patterns and the test pattern.

3.3.1. Particular cases

The FRNN algorithm reduces to the following KNN algorithms when different conditions are imposed on it:

1. When the size of the artificial structure around the test pattern varies depending on the test pattern, i.e., K is an integer such that $0 \leqslant K \leqslant |\mathcal{X}|$, and $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ is a constrained fuzzy membership function, the FRNN algorithm is equivalent to the fuzzy KNN algorithm proposed by Keller et al. [16].
2. When K is an integer, $0 \leqslant K \leqslant |\mathcal{X}|$ and $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ is a crisp membership function, the output of the FRNN algorithm can be interpreted as the rough-fuzzy ownership value (see Property 3). The resultant classifier can be called the *rough-fuzzy KNN classifier*.
3. When K is an integer, $0 \leqslant K \leqslant |\mathcal{X}|$, and $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$ and $\mu_{\mathcal{C}_c}(\boldsymbol{x})$ are crisp membership functions, the output of the FRNN algorithm is equivalent to the rough ownership value (see Property 4). Alternatively the output can also be viewed as the *a posteriori* probability of the test pattern to belong to the output classes. Hence, the resultant classifier can be called a *rough KNN* or a conventional KNN classifier.
4. When $K = 1$, there is no rough uncertainty in the evidence supplied by the neighbors. If we ignore whether the nearest neighbor is close to or far away from the test pattern, then the output of the FRNN classifier can be interpreted as the fuzzy membership value of the test pattern to belong to the output class (see Property 5).

The conceptual difference between the FRNN classifier, and other parametric and semiparametric classifiers is as follows: In an FRNN classifier, every time a test pattern is encountered, an artificial structure is constructed around the test pattern. In parametric and semiparametric classifiers, we know the structure of the input space from some other source or from *a priori* knowledge. Hence, we are not interested to erect the structure around the test pattern; rather we intend to know in which part of the structure of the training set, the test pattern lies. For instance, in the radial basis function neural network [14], we find the structure of the input space through clustering the training data, and the classification is carried out based on in which structure the test pattern falls. In other words, in the radial basis function neural network, we find the classification function at all points of the inputs (this step is called *induction*), and then we find the value of the classification function at the test pattern (this step is called *deduction*) [30]. In the conventional and fuzzy KNN algorithms, we find the classification function only at the vicinity of the test pattern (this is called *transduction*). The advantage of the transduction over the induction–deduction scheme is that one does not need to find the values of the classification function at all possible points in X, rather the values need to be estimated only at the test pattern. But the transduction scheme suffers when the generalization using the local information is affected by the noise and outliers. The FRNN algorithm adopts a compromise by considering the local as well as the global contexts. It does not need to estimate the classification function at all possible input patterns, but it considers the influence of all training patterns while estimating the classification function only for the test patterns.

The advantages of the FRNN algorithm are as follows:

- No need to know the optimal value of K.
- It has the possibilistic classification ability.
- The conventional and fuzzy KNN algorithms are the particular cases of the proposed algorithm.
- The worst-case time complexity of the proposed algorithm is same as the conventional and fuzzy KNN algorithms.
- Unlike other parametric and semiparametric classifiers, the FRNN classifier does not need any *a priori* structural information about the training data. Note that in the classifiers based on the possibilistic clustering [19], we need to know the structural information (e.g., the number of the clusters).
- All other advantages of the conventional KNN algorithm are available here. For instance, new training patterns can be added at any time without any retraining. Also the FRNN algorithm is as simple as the conventional KNN algorithm.
- Instead of using Eq. (3.2.4), Bian and Shen [4] use Eqs. (2.3.4a) and (2.3.4b) to compute a lower and upper approximation of a fuzzy-rough membership value. The class label of the test pattern is decided based on the interval that spans from the lower approximation to the upper approximation. Therefore, in Bian et al.'s approach, the upper approximation of the test pattern depends on the maximum value of $\min(\mu_{F_i}(\boldsymbol{x}), \mu_{\mathcal{C}_c}(\boldsymbol{x}))$ among all training patterns. Hence, the upper approximation critically depends on a single training pattern among all possible training patterns. If this pattern is noisy, the upper approximation would be entirely different. Similar reasoning applies for the lower approximation. In contrast, the fuzzy-rough ownership value depends on all training patterns, and thus it produces a more robust classification result.

4. Results and discussion

The efficacy of the proposed algorithm is demonstrated through experiments conducted on real data sets. In most cases we have considered the class membership of the training patterns as crisp, and we have used Eq. (3.3.2) to compute the fuzzy-rough ownership function. In addition, we have assumed $q = 2$ in all studies.

4.1. Experiment 1

In the first experiment, we consider the vowels 'a', 'e', 'i', 'o' and 'u'. The data required for the training are collected from the vowel part of the utterances of the consonant–vowel pairs of three different speakers. The first three formants are used as features. The formants are extracted using linear prediction analysis [26]. The data set needed for this experiment is kept at http://www.geocities.com/fuzzy_rough/speechData.htm.

For the ease of visualization, we illustrate (Fig. 5) the characteristic of the data on a two-dimensional plane formed by the formants F_1 and F_2. The data set is noisy, and the classes are overlapping. We use the extracted features to constitute a training set of 230 examples. Here, our objective is to train various pattern classifiers as well as the FRNN on the same training set and compare their classification performances on a different test set consisting of 1508 patterns.

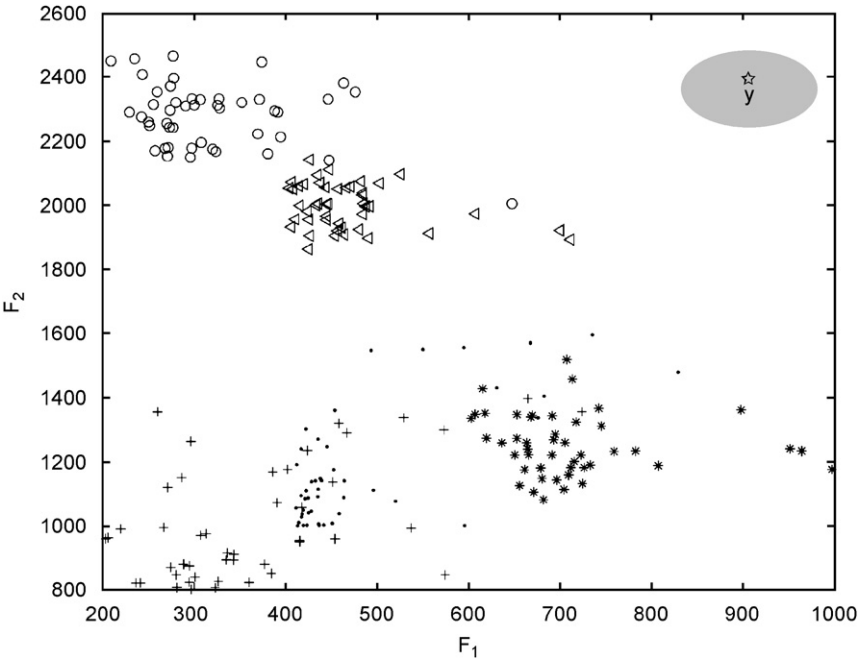

Fig. 5. Input speech data corresponding to the classes 'a', 'e', 'i', 'o' and 'u' are depicted in '*', '<', '0', '.' and '+', respectively. The data contain both rough and fuzzy uncertainties. Since the noisy pattern '**y**' is far away from all classes, the classification algorithm should not classify it to any class. The conventional and fuzzy KNN algorithms fail to achieve that, whereas the FRNN successfully does that.

Table 1
Results of vowel classification for different types of classification algorithms are shown in percentage of correct classification

Class	Bayes classifier (%)	Feedforward neural networks with backpropagation algorithm (%) No. of hidden nodes			Conventional KNN (%) ($K = 17$)	Fuzzy KNN (%) ($K = 21$)	FRNN (%)
		5	10	15			
'a'	84.89	80.77	69.49	70.23	96.08	89.71	99.51
'e'	82.86	86.93	83.28	71.32	95.59	94.12	97.06
'i'	87.33	82.67	80.30	81.34	91.67	91.67	79.41
'o'	40.29	72.89	76.56	63.94	87.25	82.35	78.43
'u'	89.86	81.94	79.92	72.99	57.35	61.27	79.41
Overall	77.05	81.04	76.75	71.97	85.58	83.82	86.76

The FRNN performs better than the other classifiers.

First, we use Bayes classifier for multivariate normal patterns with *a priori* probabilities $p_i = P_i/P$, where P_i denotes [24] the number of patterns in the ith class and P is the total number of training patterns. The covariance matrix for each class is estimated from the training patterns of that particular class. The classification performance of the Bayes classifier on the test set is shown in the second column of Table 1. The Bayes classifier gives optimal classification performance for the probabilistic classifiers [2], provided the parameters of the input distribution are estimated from the inputs collected over the whole input space. In practice, the distribution parameters are estimated based only on a finite number of training data. As a result, the performance of the Bayes classifier is no longer optimal, but its performance approaches the optimal one as the number of input data is made very large (theoretically, it is infinity). Nevertheless, in Fig. 6, the Bayes classifier, based on a finite number of training samples, is used to compare the performance of the proposed method.

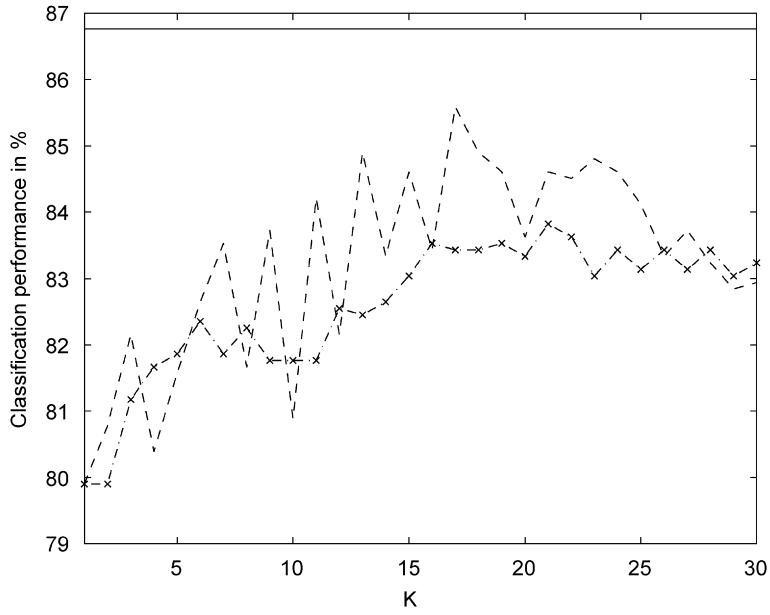

Fig. 6. This figure shows how the overall classification results of the conventional KNN (shown by '- -') and fuzzy KNN (shown by '- -') algorithms vary on the vowel data depending on K. The fluctuation in the classification performance makes it difficult to choose the optimal value of K. The classification performance of the FRNN, which does not depend on K, is shown by a solid, horizontal line at the top. The classification performance of the FRNN is always better than that of the conventional and fuzzy KNN algorithms.

Next, we use the backpropagation learning algorithm to train feedforward neural networks [14] with three input nodes, five output nodes and variable number of hidden nodes. The target patterns are five-dimensional vectors containing 1 in one location and 0 in all others. Here, we adopt the strategy of picking the output node with the highest activation value as the output class corresponding to an input. The learning-rate is adaptively changed in the following way: If the error decreases during training, then the learning-rate is increased by a predefined amount. In contrast, if the error increases, then the learning-rate is decreased, and the new weights and errors are discarded. As a result, the error always decreases or stays as it is. The momentum is kept constant throughout the process. We used the backpropagation learning algorithm on three different feedforward neural network architectures based on 5, 10 and 15 hidden nodes. We have repeated the training of each architecture for 5000 iterations with 25 different weight and bias initializations. The mean classification performance of the backpropagation algorithm over these three architectures is shown in the third, fourth and fifth columns of Table 1. Although for certain weight and bias initialization the overall classification performance is high, for some initializations the classification efficiency becomes quite low. Consequently, when the classification efficiency is averaged over 25 different initializations, the classification performance remains low.

Finally, we compare the conventional KNN, fuzzy KNN and FRNN algorithms. By trial and errors, we found that the classification results of the conventional and fuzzy KNN algorithms are maximum at $K = 17$ and 21, respectively. The sixth, seventh and eighth columns of Table 1 exhibit the performances of the conventional, fuzzy and FRNN algorithms. It can be observed that compared to all the other classifiers, the FRNN algorithm enhances the overall classification result significantly.

In Fig. 6 the variation of the overall classification efficiency is shown for $K = 1$ to 30. We can observe that the classification performance curves for both conventional and fuzzy KNN algorithms fluctuate for different values of K. When K is small, the classification results of both conventional and fuzzy KNN algorithms are low because the bodies of evidence collected from small number of neighbors are influenced by the presence of noise. On the other hand, when $K > 23$, again the performance tends to fall slowly. Here although we have more bodies of evidence, the bodies of evidence are conflicting, and it results in a confusion to assign the class label. It typically happens where the overlaps between the neighboring classes is high. The figure shows how difficult it is to choose the optimal value of K in real-life problems. The performance of the proposed method does not depend on K; hence, it traces a horizontal straight line in Fig. 6. The classification results of the conventional and fuzzy KNN are always lower than that of the FRNN.

Table 2
For the noisy pattern $\boldsymbol{y} = [900, 2400, 3500]'$ (see Fig. 5), which does not belong to any class, the classification outputs of the conventional KNN, fuzzy KNN and FRNN algorithms are shown

Class	Conventional KNN ($K = 1$)	Fuzzy KNN ($K = 4$)	FRNN
'a'	0	0	0.0739×10^{-10}
'e'	0.1250	0.1184	0.6089×10^{-10}
'i'	0.8750	0.8816	0.8075×10^{-10}
'o'	0	0	0.0018×10^{-10}
'u'	0	0	0.0001×10^{-10}
Sum of class confidence values	1.0	1.0	1.4922×10^{-10}

Both conventional and fuzzy KNN algorithms erroneously indicate that $\boldsymbol{y}$ belongs to the class 'i'. In contrast, the FRNN algorithm produces the class confidence values for all classes as almost zero. Moreover, the sum of the class confidence values for the FRNN algorithm is close to zero—which shows that the input pattern does not belong to any of the output classes.

Table 3
The comparative classification performance of the conventional KNN, fuzzy KNN and FRNN algorithms on the letter data set

Class	Conventional KNN (%) ($K = 1$)	Fuzzy KNN (%) ($K = 4$)	FRNN (%)
'D'	94.15	93.23	95.38
'G'	92.77	92.62	92.15
'O'	94.62	93.38	94.62
'Q'	87.69	90.31	89.85
Overall	92.31	92.38	93.00

The power of possibilistic classification is demonstrated in this paragraph. Fig. 5 shows a noisy test input $\boldsymbol{y} = [900, 2400, 3500]'$. Clearly this input does not belong to any class since it is far away from all classes. For this input, the class confidence values generated by the conventional KNN, fuzzy KNN and FRNN algorithms are shown in Table 2. Both conventional and fuzzy KNN algorithms indicate with more confidence that the test pattern belongs to the class 'i', and thus fail to indicate that the input pattern does not belong to any class. On the other hand, the proposed algorithm produces the class confidence values for all classes as almost zero. In addition, for the proposed algorithm, the sum of the class confidence values is almost zero, which implies total ignorance. In other words, the proposed algorithm indicates that the input pattern does not belong to any of the output classes.

4.2. Experiment 2

For this experiment, we consider the letter image recognition data set, which has been obtained from UCI machine learning repository [6,12]. The aim is to identify each of large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were collected from 20 different fonts. Each letter within these 20 fonts was randomly distorted to produce a file of 20 000 unique stimuli. Each stimulus was later converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 to 15. We have collected input data only for the letters D, G, O and Q, because the characters are similar, and it is difficult to discriminate them. The training set consists of 400 patterns, and the test set consists of 2600 patterns. The best classification results for the conventional KNN ($K = 1$), fuzzy KNN ($K = 4$) and FRNN algorithms are shown by Table 3. Although the performance differences among the FRNN, conventional KNN and fuzzy KNN classifiers are not large, the FRNN shows the best result, and to achieve that we do not need any trial and error procedure.

Fig. 7 shows how the performance curves of the conventional and fuzzy KNN algorithms vary when K varies between 1 and 30. Unlike the previous experiment, the classification performance of the conventional KNN decreases rapidly from the beginning. In contrast, the performance of the fuzzy KNN initially increases, and then decreases when K is increased. This figure shows that the performance of the FRNN always maintains a gap with that of the conventional and fuzzy KNN algorithms.

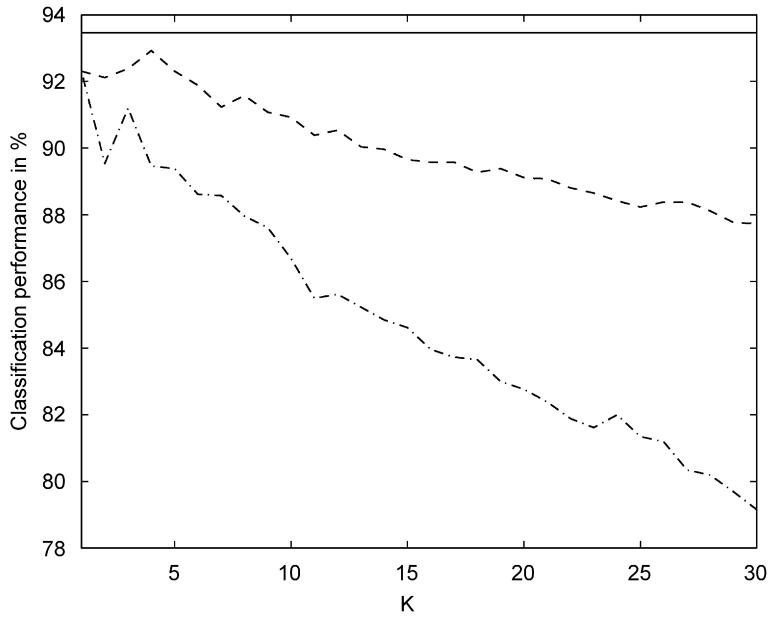

Fig. 7. This figure shows how the classification performance of the conventional KNN (shown by '—· — · — ·—') and fuzzy KNN (shown by '- -') algorithms vary on the letter data depending on K. The classification performance of the FRNN, which does not depend on K, is shown by a solid horizontal line. The classification performance of the FRNN is always better than that of the conventional and fuzzy KNN algorithm, and to design the FRNN we do not need to find suitable values for K.

4.3. Experiment 3

To find how the proposed classifier performs on a data set, which already has fuzzy class labels, we conducted another experiment. The objective of this experiment was to diagnose whether a subject was suffering from a serious head injury or not. We collected 100 training patterns and 250 test patterns. For each pattern following features were considered: original value of cerebral perfusion pressure (CPP), original value of intracranial pressure (ICP), original value of Glasgow coma scale (GCS), original conditions of the pupils (e.g., brisk, fixed, one brisk and both fixed), operability of lesion. Although the above features are crisp, the output class labels, serious head injury and non-serious head-injury, are fuzzy. Using domain-specific knowledge, three medical experts classified each training pattern into those two classes, and the class membership values assigned by them were possibilistic. For each training pattern, the average value of the class membership values assigned by all three experts was treated as the final class membership value. Using these membership values, the proposed classifier was trained, and the overall classification performance on the test data was 76.34%.

When the possibilistic class membership values of the training set were made crisp, and the resultant training data were fed to a crisp KNN, the classification result on the test data became 69.74%. When the possibilistic class membership values of the training data were converted to constrained fuzzy class labels (i.e., sum of membership values equal to 1), and a fuzzy KNN was trained using this training data, the classification result on the test data set improved to 72.1%. However, this classification result was still less than the classification result of the proposed algorithm.

5. Summary and conclusion

The conventional KNN algorithm assigns the class label of the input pattern based on the class labels that majority of the K-closest (in some distance sense) neighbors from the training set posses. In the conventional KNN algorithm, all K neighbors receive equal importance and the class label of each training pattern is considered crisp. The fuzzy KNN algorithm refines the conventional KNN algorithm (a) by weighing the contribution of each of the K neighbors based on its distance to the test pattern and (b) by making the class membership of each training pattern fuzzy. Although both conventional and fuzzy KNN algorithms are simple and nonparametric, the classification performance depends on the

choice of K, and the class confidence values generated by the algorithms are not possibilistic. To partially avoid the above two problems, and to enhance the classification result, this paper attempts to use fuzzy-rough uncertainty.

The proposed method explores the fuzzy uncertainty that is present in terms of (a) the absolute similarity between the test pattern and the neighbors, (b) the belongingness of the test pattern into the output classes. In the proposed method, each training pattern is considered neighbor to the test pattern with varying degree, and hence we do not need to determine the appropriate value of K. The proposed algorithm also exploits rough uncertainty, which is present because (a) the test pattern and the neighbors may be indistinguishable due to the lack of features, and (b) the neighborhood represents the artificial similarity. Both fuzzy and rough uncertainties are quantified using the fuzzy-rough ownership function, and it is shown that the fuzzy-rough ownership function possesses the possibilistic classification ability. Finally, the proposed algorithm is built such that its output can be interpreted as the fuzzy-rough ownership function. The computational complexity of the proposed algorithm is same as that of the conventional and fuzzy KNN algorithms. It is also shown that the conventional and fuzzy KNN algorithms are the particular cases of the proposed algorithm. The enhanced classification result of the proposed algorithm is shown on some real-life data sets.

The good performance of the FRNN algorithm does not imply that the FRNN algorithm will be good for all classification problems. In fact there is no known single algorithm that performs well on all diagnosis problems (if there were, we would have observed only one classification algorithm available for diagnosis). This work, however, highlights the potential usefulness of the fuzzy-rough uncertainty in the nearest neighbor principle.

The FRNN algorithm inherits certain drawbacks from the KNN algorithms. Some of the drawbacks are (a) we need to store the training data; hence for a large training set it may take large space, and (b) for every test pattern, the distance should be computed between the test pattern and all training data. Thus, long time may be needed for the testing. To speed-up the computation, we can use approaches similar to the fast versions of the KNN algorithm [20,21,13]. This paper does not attempt to improve the space and time complexity of the FRNN approach, but shows better classification results using a simple technique.

In the proposed classifier, we need to use two parameters: $q \in (1, \infty)$ and κ. The choice of q and κ decide the shape of $\tilde{\mu}_{\boldsymbol{y}}(\boldsymbol{x})$. When q approaches one, the membership function tends to be a crisp membership function with a very sharp slope. When q approaches infinity, the slope is almost flat, and the membership function is maximally fuzzy. Although we reported all results using $q = 2$, with higher values of q like $q = 4, 6$, the classification results did not change significantly.

We have discussed two proposals to choose κ: (a) keep κ fixed for all dimensions and (b) make κ proportional to the inverse of the average distance between all neighbors and the test pattern. It was observed that the second proposal usually yielded slightly better results. It is possibly because the second proposal considers the distribution of test data along all dimensions. One interesting future work would be how to improve classification results of the proposed algorithm further by selecting proper values of q and κ.

In Eq. (3.2.3), we have used a product-based t-norm operation. The impact of different t-norm operations on classification results would be an interesting future work.

The FRNN algorithm is useful to construct modular classifiers. In the modular classifiers, we divide a complex classification problem into several subclassification problems, and we use one classifier for each subclassification problem. In other words, each subclassifier is supposed to deal with only a part of the complete input space. The final classification result is generated by combining the outputs of all the modular classifiers. But combining the individual solutions provided by the modules becomes difficult as each module claims that the test pattern falls in its input space. It typically happens when each classifier produces certain output whenever input is fed to the classifiers. Ideally, the output of the classifier should be close to zero when the input does not belong to the input space corresponding to the classifier. In other words, the classifier should have the capability to indicate the *ignorance*. Since the FRNN algorithm has the possibilistic classification ability, it can be used more efficiently to build modular classifiers. Our future work would be directed along that direction.

Acknowledgments

Discussion with Dr. Leong Tze-Yun and other lab members of Medical Computing Lab has helped me to write the paper. The work of this paper has been supported by the Strategic Research Grant no. RP960351 from the National Science and Technology Board and the Ministry of Education, Singapore.

References

[1] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.
[2] J.C. Bezdek, A review of probabilistic, fuzzy, and neural models for pattern recognition, in: C.H. Chen (Ed.), Fuzzy Logic and Neural Network Handbook, McGraw-Hill, New York, 1996.
[3] H. Bian, L. Mazlack, Fuzzy-rough nearest-neighbor classification approach, in: 22nd Internat. Conf. of the North American Fuzzy Information Processing Society—NAFIPS, 2003.
[4] H. Bian, W. Shen, Fuzzy-rough nearest-neighbor classification method: an integrated framework, in: Proc. IASTED Internat. Conf. on Applied Informatics, Austria, 2002, pp. 160–164.
[5] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[6] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, ⟨http://www.ics.uci.edu/∼mlearn/mlrepository.html⟩, 1998.
[7] B.V. Dasarathy (Ed.), Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, IEEE Computer Society Press, Los Alamitos, CA, 1991.
[8] D. Dubois, H. Prade, Possibility Theory: An Approach to Computerized Processing of Uncertainty, Plenum Press, New York, Plenum, 1988.
[9] D. Dubois, H. Prade, Rough-fuzzy sets and fuzzy-rough sets, Internat. J. Gen. Systems 17 (2–3) (1990) 191–209.
[10] D. Dubois, H. Prade, Putting rough sets and fuzzy sets together, in: R. Slowinski (Ed.), Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, Kluwer Academic Publishers, Dordrecht, 1992.
[11] R. Duda, P. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[12] P.W. Frey, D.J. Slate, Letter recognition using Holland-style adaptive classifiers, Mach. Learning 6 (2) (1991).
[13] J. Friedman, J. Bentley, R. Finkel, An algorithm for finding best matches in logarithmic time, ACM Trans. Math. Software 3 (3) (1977) 209–226.
[14] S. Haykin, Neural Networks—A Comprehensive Foundation, Macmillan College Publishing Company, New York, 1994.
[15] J.S.R. Jang, C.T. Sun, E. Mijutani, Neuro-Fuzzy and Soft Computing, Prentice-Hall, Englewood Cliffs, NJ, 1997.
[16] J.M. Keller, M.R. Gray, J.A. Givens, A fuzzy K-nearest neighbor algorithm, IEEE Trans. Systems Man Cybernet. 15 (4) (1985) 580–585.
[17] G.S. Klir, T.A. Folger, Fuzzy Sets, Uncertainty and Information, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[18] G.S. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic Theory and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1995.
[19] R. Krishnapuram, J.M. Keller, A possibilistic approach to clustering, IEEE Trans. Fuzzy Systems 1 (2) (1993) 98–110.
[20] L. Mico, J. Oncina, Comparison of fast neighbor classifiers for handwritten character recognition, Pattern Recognition Lett. 19 (1998) 351–356.
[21] L. Mico, J. Oncina, R.C. Carrasco, A fast branch and bound nearest neighbour classifier in metric space, Pattern Recognition Lett. 17 (1996) 731–739.
[22] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
[23] N.R. Pal, J.C. Bezdek, On cluster validity for the fuzzy C-means model, IEEE Trans. Fuzzy Systems 3 (3) (1995) 330–379.
[24] S.K. Pal, D. Dutta Majumder, Fuzzy Mathematical Approach to Pattern Recognition, Wiley (Halsted Press), New York, 1986.
[25] Z. Pawlak, S.K.M. Wong, W. Ziarko, Rough sets: probabilistic versus deterministic approach, Internat. J. Man–Machine Stud. 29 (1988) 81–95.
[26] L.R. Rabiner, B.H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[27] M. Sarkar, Rough-fuzzy functions in classification, Fuzzy Sets and Systems 132 (3) (2002) 353–369.
[28] F. Scarselli, A.C. Tsoi, Universal approximation using feedforward neural networks: a survey of some existing methods, and some results, Neural Networks 11 (1) (1998) 15–37.
[29] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, Princeton, 1976.
[30] V.N. Vapnik, Statistical Learning Theory, Springer, New York, 1995.
[31] L.A. Zadeh, Fuzzy Sets, Inform. Control 8 (1965) 338–353.
[32] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets and Systems 1 (1978) 3–28.
[33] H.Z. Zimmerman, Quantifying vagueness in decision models, European J. Oper. Res. 22 (1985) 148–158.