Information Sciences 433–434 (2018) 96–114

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Multi-label classification using a fuzzy rough neighborhood consensus

Sarah Vluymans [a,b,c,*], Chris Cornelis [a], Francisco Herrera [c,d], Yvan Saeys [a,b]

[a] Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Belgium
[b] Data Mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium
[c] Department of Computer Science and Artificial Intelligence, University of Granada, Spain
[d] Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia

ARTICLE INFO

Article history:
Received 15 May 2017
Revised 16 November 2017
Accepted 23 December 2017
Available online 25 December 2017

Keywords:
Fuzzy rough set theory
Multi-label classification
Nearest neighbor classification

ABSTRACT

A multi-label dataset consists of observations associated with one or more outcomes. The traditional classification task generalizes to the prediction of several class labels simultaneously. In this paper, we propose a new nearest neighbor based multi-label method. The nearest neighbor approach remains an intuitive and effective way to solve classification problems and popular multi-label classifiers adhering to this paradigm include the MLKNN and IBLR methods. To classify an instance, our proposal derives a consensus among the labelsets of the nearest neighbors based on fuzzy rough set theory. This mathematical framework captures data uncertainty and offers a way to extract a labelset from the dataset that summarizes the information contained in the labelsets of the neighbors. In our experimental study, we compare the performance of our method with five other nearest neighbor based multi-label classifiers using five evaluation metrics commonly used in multi-label classification. Based on the results on both synthetic and real-world datasets, we are able to conclude that our method is a strong competitor to nearest neighbor based multi-label classifiers like MLKNN and IBLR.

1. Introduction

Research in machine learning concerns the ability to learn a confident prediction model based on a set of observations. For example, in a classification context, a learner trains a classification model on the given elements, of which the outcomes are known, and uses the derived model to predict the outcome of previously unseen instances. In the traditional single-label setting, each observation is associated with one outcome, its class label. Multi-label learning [16,18,60] represents a more general approach, where an observation can belong to several classes at the same time, that is, more than one class label can be associated with the same instance. The total number of classes is known, but the number of labels per instance can differ across the dataset. Multi-label learning can be more challenging than single-label learning or learning all possible classes independently, as correlations between some classes may be present. An example of such a situation is the existence of a label hierarchy, which needs to be taken into account in the prediction process [37]. Application domains of multi-label classification include image processing (e.g. [24,49]), text categorization (e.g. [29,31]) and bioinformatics (e.g. [5,44]).

* Corresponding author.
E-mail address: sarah.vluymans@ugent.be (S. Vluymans).

https://doi.org/10.1016/j.ins.2017.12.034
0020-0255/

In a multi-label dataset, every instance x is described by a number of input features and associated with a labelset. This labelset is represented as a binary vector $L_x = \langle l_1(x), l_2(x), \ldots, l_m(x) \rangle$, with m the total number of possible class labels in the dataset and $(\forall i)(l_i \in \{0, 1\})$. The value $l_i(x)$ indicates whether or not x belongs to class l_i. The task of a multi-label classifier is to predict the complete labelset of a target instance. This is inherently different from single-label classification, where only one outcome label needs to be predicted.

Several approaches to multi-label classification have been proposed in the literature. The recent overview book [18] discerns between two main families, the data transformation methods and method adaptation algorithms. The former group of methods applies a transformation to the multi-label dataset, such that it degenerates to one or more easier-to-handle single-label problems, on which a single-label classifier can be applied. Two well-known representatives of this family are the binary relevance (BR, [17]) and label powerset (LP, [3]) transformations. BR creates m binary single-label datasets, one for each class. Each dataset contains the same instances as the original multi-label dataset, but their labelsets are transformed to a single label. For the dataset associated with class l_i, an instance x receives the label 'positive' when $l_i(x) = 1$ and 'negative' otherwise. The LP transformation on the other hand creates only one single-label dataset. Each possible labelset receives an identifier, such that labelsets that entirely coincide are associated with the same identifier. This identifier is used as the single new class label. The second family of multi-label classification algorithms handle the multi-label dataset directly and are often based on modifications or generalizations of existing single-label classification schemes. An example is the MLKNN method proposed in [59].

In this paper, we focus on nearest neighbor methods for multi-instance classification, of which MLKNN is an example. The nearest neighbor approach [11] is an intuitive way to predict an outcome for a new observation based on a set of known instances. In its simplest form, it requires no training phase, since no classification model is built. Instead, all known instances are stored in memory as prototypes. In order to predict the outcome of a target instance, its nearest element (or set of nearest elements) is extracted from the prototype set and the prediction is derived from the outcomes of these neighbors. In particular, to classify an instance x, the k nearest neighbor classifier (kNN) locates the k nearest elements among the stored instances and aggregates their class labels to a prediction for x. In single-label classification, this is commonly achieved by a majority vote. The kNN classifier is a simple and understandable algorithm and remains popular in the machine learning community [46]. Several multi-label classifiers based on or extending kNN have been proposed in the literature. We propose a new member of this family in this contribution and use a novel way to aggregate the labelsets of the k nearest neighbors to a prediction based on fuzzy rough set theory.

Fuzzy rough set theory [12] is an alternative to traditional set theory and models uncertainty in data. It covers two complimentary aspects of uncertainty, namely vagueness (fuzziness) and indiscernibility (roughness). The former relates to unclear descriptions of concepts, to which elements can belong to a certain degree. As an example, the set of elements that are *similar* to a given element x is necessarily fuzzy, since some elements are intrinsically more similar to x than others and making a strict division between similar and non-similar is difficult. The membership degree of an element to a fuzzy set is represented by a real number between 0 and 1. Roughness in a dataset concerns the issue when observations that are indiscernible with respect to their descriptive features have distinct outcomes. In such a situation, it is challenging to sharply delineate the outcome concept based on the input features. Instead, a lower and an upper approximation are provided. Fuzzy rough set theory was developed as a hybridization of fuzzy set theory [58] and rough set theory [32] and has been used successfully in a variety of machine learning techniques [40]. It provides a framework to approximate a concept by two fuzzy sets, the fuzzy rough lower and upper approximation.

The fuzzy rough approximation operators are essentially based on a similarity relation that measures the degree to which elements are similar to each other. As such, they are related to nearest neighbor approaches. In this paper, we use these operators to derive a consensus prediction from the labelsets of the k nearest neighbors of a target instance. In particular, each of the neighbors of the target instance may have a different labelset and the challenge is to aggregate this information to one predicted labelset. Fuzzy rough set theory forms an ideal means to do so. Based on the similarity of the neighbors to the target, an appropriate consensus labelset is derived. We will experimentally show that our approach can outperform state-of-the-art nearest neighbor based multi-label classifiers. Following the recent study of Reyes et al. [34], we limit the comparison of our proposal to the state-of-the-art within the same classifier family, the nearest neighbor methods

The remainder of the paper is structured as follows. In Section 2, we review the existing nearest neighbor based multi-label classifiers. Section 3 recalls fuzzy rough set theory and describes our proposed classification method. Our method is carefully evaluated in an experimental study, of which the set-up is described in Section 4. Section 5 lists and analyzes the experimental results. Finally, Section 6 concludes the paper. We note that additional content, including the full experimental results, is made available at our web page http://www.cwi.ugent.be/sarah.php.

2. Related work: nearest neighbor based multi-label classifiers

To focus our discussion, we consider a subgroup of multi-label classifiers, namely those based on the nearest neighbor paradigm. Several nearest neighbor based multi-label classifiers have been proposed in the literature. We provide an overview in this section. In our experimental study, we select a number of these methods to compare among each other and against our proposed classifier. This selection is made based on the popularity and performance of these methods in other experimental studies. We also take into account how many parameters should be set by the user. The more parameters that need to be set manually, the less attractive a method is for practical use.

2.1. Basic unweighted approaches

Two basic approaches were proposed in [35], by combining the LP and BR transformations with the single-label kNN classifier. The LPKNN method classifies an instance by first locating its k nearest neighbors and then predicting the most prevalent labelset among these elements. The BRKNN method is equivalent to using the single-label kNN method within a binary relevance setup, but runs considerably faster, since it computes the nearest neighbors of an instance only once instead of m times. The classifier assigns x to a class l when this label is present in at least half of its nearest neighbors. Using this heuristic, it is possible that x is not assigned to any class at all. To address this issue, the authors of [35] developed two extensions, called BRKNN-a and BRKNN-b. For each class l, these methods set the label confidence score to the percentage of the nearest neighbors of x that belong to class l. When none of the classes is present in at least half of its neighbors, BRKNN-a assigns x to the class with the highest label confidence score, while BRKNN-b assigns x to the s classes with the highest label confidence score, where s is the average size of the labelsets of the neighbors of x. In the later work of Geng et al. [15], the authors proposed the BRKNN-new algorithm as an improvement of BRKNN by considering label correlations.

The class-balanced k nearest neighbor (BKNN, [43]) and multi-label categorical kNN method (ML-CKNN, [22]) compute class scores based on the nearest neighbors of an instance within a class and predict the classes for which the score exceeds a given threshold. The BRkSC and BRSC methods of Liu et al. [28] follow a similar idea, but are based on the shell-nearest neighbor algorithm [61]. The so-called class certainty factors are determined based on the approximate shell-nearest neighbors of the instance to classify.

2.2. Basic weighted approaches

It is a common idea to incorporate weights in a nearest neighbor method, in order to increase the importance of certain neighbors. In [8], the authors propose a ranking-based KNN approach that assigns weights to the nearest neighbors of an element to reflect how trustworthy their labels are. To classify an instance, its k nearest neighbors are ranked according to this trust measure and a weighted voting strategy is used to derive the prediction. Similarly, the study of Herrera et al. [48] proposed a distance-weighted multi-label k nearest neighbor algorithm called MLC-DWkNN, which assigns an instance to class l when the sum of the weights of the neighbors that belong to l exceeds the sum of the weights of the neighbors that do not. Another weighted nearest neighbor approach to multi-label classification is the Margin-Ratio kNN method (Mr.kNN, [27]). Relevance values of training instances, computed based on a modified fuzzy c-means clustering method, are used as weights in the voting procedure of a k nearest neighbor classifier.

2.3. MLKNN and related methods

The MLKNN method from [59] was introduced as one of the first lazy multi-label classifiers and remains popular today. It makes label predictions based on the maximum a posteriori principle and the k nearest neighbors of a target instance. Simply put, it counts the occurrences of all classes among the neighbors and evaluates how likely the presence of a class label is based on these counts.

It has been pointed out in the literature that a limitation of the MLKNN method is that it does not take label correlations into account. The DMLKNN method of Younes et al. [52,56] deals with this shortcoming by considering a global maximum a posteriori rule. The FSKNN method [23] uses a fuzzy similarity measure to first group the training elements into clusters. It reduces the computational cost of the neighbor search of MLKNN, since it only uses a subset of the clusters to locate them.

2.4. Other nearest neighbor based multi-label classifiers

The authors of [6] proposed the IBLR method that combines nearest neighbor based learning and logistic regression. The class labels of the neighbors of an element are considered as supplementary features. One classifier, a logistic regression model, is trained for each class, but dependencies between labels are taken into account. The IBLR+ generalization takes additional features of the target instance into account, aside from its neighborhood information. In IBLR, the bias term of the logistic regression model is constant, while it depends on the target instance in IBLR+.

The MLRS method was proposed in [57] and is a nearest neighbor method based on neighborhood rough sets [20,51]. At training time, this method computes, for each pair of labels, the conditional probability that a label l_i occurs when the presence of label l_j is already known. Next, to classify an instance x, its k nearest neighbors are located, based on which the marginal probabilities for the presence of the class labels is determined. Finally, for each class label, the probability that x belongs to this class is calculated. As additional input, this method requires a threshold on the class probability values, above which a class label is accepted.

Two fuzzy nearest neighbor approaches to multi-label classification are the FkNN [2] and Fuzzy Veristic kNN (FV-kNN, [55]) methods. The former is an adaptation of the fuzzy nearest neighbor classifier [26], while the latter uses a fuzzy kNN rule based on the theory of veristic variables, which can take on several values at once with different degrees.

Other proposals include a kernel-based kNN approach for multi-label propagation [25], the Mallows method [7] based on calibrated label ranking [14] and the Mallows model [30], the multi-label ranking method kNN-MLR* [4], the RW.kNN

method [47] based on random walks in a neighborhood graph and the evidential multi-label kNN method (EML-kNN, [53,54]) based of an extension of the Dempster-Shafer framework to the multi-label setting.

Finally, we can refer the reader to the recent work of Reyes et al. [34], where the authors reviewed several nearest neighbor multi-label classifier and proposed a new method called MLDGC based on the data gravitation based algorithm. Their proposal is compared to several representatives of the nearest neighbor multi-label classifier family. The authors of this study also outline guidelines for new proposals within this classifier family, which we follow in this paper.

3. Fuzzy rough multi-label classifier

In this section, we present our proposal, that uses fuzzy rough set theory to construct an appropriate consensus among the labelsets of the k nearest neighbors of a target instance. In Section 3.1, we first provide the necessary background on fuzzy rough set theory. Section 3.2 introduces our proposal.

3.1. Fuzzy rough set theory

Rough set theory was proposed in [32] and deals with indiscernibility in data. At its core lies the definition of an equivalence relation, that partitions the universe in equivalence classes. Two elements that belong to the same equivalence class are indiscernible based on the measured features. Commonly, this is the case when they take on the same value for all features. When a concept C can be described as an exact union of equivalence classes, there is no ambiguity in its definition. In the other case, when C contains some equivalence classes only partially, uncertainty is present. Rough set theory describes such a set C by two other sets, the lower approximation $\underline{C}$ and upper approximation $\overline{C}$, that approximate C in two ways. The former is defined as the union of equivalence classes fully included in C, while the latter corresponds to the union of the equivalence classes that have a non-empty intersection with C.

In order to make rough sets more flexible, e.g. allowing a graded rather than strict similarity between elements, the authors of [12] combined it with fuzzy set theory [58] and proposed fuzzy rough set theory. When C is a fuzzy set, any element x can partially belong to it, expressed by its membership degree $C(x) \in [0, 1]$. Likewise, a fuzzy relation $R(\cdot, \cdot)$ takes on values between 0 and 1 and expresses how strongly its two arguments are related. In fuzzy rough set theory, a concept C is approximated by two fuzzy sets, the fuzzy rough lower and upper approximation. In the general implicator/t-norm fuzzy rough set model [33], the membership degree of an element x to the lower approximation of C is computed as

$$\underline{C}(x) = \min_{y \in X} [\mathcal{I}(R(x, y), C(y))], \tag{1}$$

where the implicator $\mathcal{I} : [0, 1]^2 \to [0, 1]$ is an operator that is decreasing in its first argument, increasing in the second and satisfies the boundary conditions $\mathcal{I}(0, 0) = \mathcal{I}(0, 1) = \mathcal{I}(1, 1) = 1$ and $\mathcal{I}(1, 0) = 0$. The relation $R(\cdot, \cdot)$ computes the feature similarity between two elements and is assumed to satisfy at least reflexivity ($(\forall x)(R(x, x) = 1)$) and symmetry ($(\forall x, y)(R(x, y) = R(y, x))$). The upper approximation is defined as

$$\overline{C}(x) = \max_{y \in X} [\mathcal{T}(R(x, y), C(y))], \tag{2}$$

where the t-norm $\mathcal{T} : [0, 1]^2 \to [0, 1]$ is a commutative and associative operator that is increasing in both arguments and satisfies the boundary condition $(\forall a \in [0, 1])(\mathcal{T}(a, 1) = a)$.

It has been noted in the literature that the traditional fuzzy rough set model is highly sensitive to noise in the data and several robust alternatives have been proposed, see e.g. the review paper [21]. An example noise-tolerant version is the OWA-based fuzzy rough set model proposed in [10], which replaces the minimum and maximum operators by ordered weighted average (OWA, [50]) aggregations. The OWA aggregation with weight vector $W = \langle w_1, w_2, \ldots, w_p \rangle$ of a set of values $V = \{v_1, v_2, \ldots, v_p\}$ is defined as $\text{OWA}(V) = \sum_{i=1}^{p} w_i v'_i$, where v'_i is the ith largest value in V. The OWA-based lower and upper approximations are respectively given by

$$\underline{C}(x) = \text{OWA}_{W_L}(\{\mathcal{I}(R(x, y), C(y)) \mid y \in X\}) \tag{3}$$

and

$$\overline{C}(x) = \text{OWA}_{W_U}(\{\mathcal{T}(R(x, y), C(y)) \mid y \in X\}). \tag{4}$$

The weight vectors W_L and W_U should be chosen appropriately, such that they resemble a minimum and maximum operator respectively. The OWA-based model reduces to the traditional model (1)–(2), when the weight vectors are set to $W_L = \langle 0, 0, \ldots, 0, 1 \rangle$ and $W_U = \langle 1, 0, \ldots, 0, 0 \rangle$. Three other possible choices [38] are additive, exponential and inverse additive weights, for which the lower approximation weight vectors are defined as

$$W_L^{add} = \left\langle \frac{2}{p(p+1)}, \frac{4}{p(p+1)}, \ldots, \frac{2(p-1)}{p(p+1)}, \frac{2}{p+1} \right\rangle$$

$$W_L^{exp} = \left\langle \frac{1}{2^p - 1}, \frac{2}{2^p - 1}, \ldots, \frac{2^{p-2}}{2^p - 1}, \frac{2^{p-1}}{2^p - 1} \right\rangle$$

$$W_L^{invadd} = \left\langle \frac{1}{pD_p}, \frac{1}{(p-1)D_p}, \ldots, \frac{1}{2D_p}, \frac{1}{D_p} \right\rangle,$$

with $D_p = \sum_{i=1}^{p} \frac{1}{i}$, the pth harmonic number. For these weighting schemes, for a given length p, the upper approximation weight vector W_U is a reversal of W_L.

3.2. Proposed method: Fuzzy ROugh NEighborhood Consensus (FRONEC)

In this section, we develop our fuzzy rough multi-label classifier FRONEC, short for fuzzy rough neighborhood consensus. It is related to the BRKNN and LPKNN methods (Section 2.1), that first locate the k nearest neighbors of the instance x to classify. Afterward, BRKNN and LPKNN predict the labels of x by aggregating the labelsets of these neighbors. The former does so by retaining the labels that appear in at least 50% of the neighbors, while the latter predicts the labelset that most frequently occurs among the neighbors. We posit that fuzzy rough set theory can provide a more suitable way to construct a consensus labelset based on the k nearest neighbors of x.

We base our consensus derivation on the notion of the fuzzy rough positive region. In the traditional fuzzy rough set model, the membership degree of x to the positive region is defined as

$$POS(x) = \max_{y \in X}[\min_{z \in X}[\mathcal{I}(R(z,x), R_d(y,z))]], \tag{5}$$

where X is the training set and relation $R_d(\cdot,\cdot)$ expresses the label similarity of two elements. This expression can be summarized as $POS(x) = \max_{y \in X}[q_x(y)]$, where $q_x(y)$ measures the quality of y relevant to x. The $POS(\cdot)$ operator locates the element y for which the membership degree of x to the fuzzy concept $R_d(y,\cdot)$ is largest. In single-instance learning, the label similarity relation is crisp, that is, two elements x and y either have the same class label ($R_d(x,y) = 1$) or they do not ($R_d(x,y) = 0$). In this case, the definition of the positive region reduces to the membership degree of x to its own decision class [9]. This value has been used as an instance quality measure in e.g. [39].

Within FRONEC, when classifying x, we use a quality metric $Q_x(\cdot)$ that is based on the definition of the positive region. Our method proceeds as follows:

1. Define the set $N(x)$ of the k nearest neighbors of x in the training set.
2. Construct the set Y that consists of elements y for which $Q_x(y)$ is largest (Section 3.2.1). Multiple elements y may tie for the highest value.
3. Any class that appears in at least half of the elements of Y is predicted for x.

Our experiments show that the third step is often superfluous, as the labelsets of all instances in Y usually coincide. Note that the elements in Y are not required to be located in the vicinity of x (e.g. $Y \not\subseteq N(x)$). Only their labelsets, which represent an appropriate consensus of those of the neighbors, are of importance. The elements in Y have the largest encountered values of $Q_x(y)$, that is, the quality of their labelset with relation to x is highest. The selection of the elements with the largest Q_x values relates back to the presence of the maximum operator in (5).

3.2.1. Instance quality measure

The instance quality measure $Q_x(\cdot)$ is based on the definition of the fuzzy rough positive region (5). We make two notable modifications:

1. In the interest of computational cost, we reduce the set of elements over which the minimum is taken to the k nearest neighbors of x. This is exactly the set for which we wish to find a consensus labelset.
2. We replace the minimum operator by an OWA aggregation. In this way, our quality measure can benefit from the superior robustness properties of the OWA-based fuzzy rough set model.

We consider three different definitions for $Q_x(\cdot)$, given the instance x to classify. The quality measure in FRONEC-1 stays closest to (5) and is defined as

$$Q_x^{(1)}(y) = \mathrm{OWA}_{W_L}(\{\mathcal{I}(R(z,x), R_d(y,z)) \mid z \in N(x)\}). \tag{6}$$

This measure is related to the OWA-based fuzzy rough lower approximation (3). In version FRONEC-2, we consider the use of the fuzzy rough upper approximation and set

$$Q_x^{(2)}(y) = \mathrm{OWA}_{W_U}(\{\mathcal{T}(R(z,x), R_d(y,z)) \mid z \in N(x)\}). \tag{7}$$

Finally, the third version FRONEC-3 combines both and computes

$$Q_x^{(3)}(y) = \frac{Q_x^{(1)}(y) + Q_x^{(2)}(y)}{2}. \tag{8}$$

3.2.2. Label similarity relation

A crucial component of the quality calculation is the label similarity relation $R_d(\cdot, \cdot)$. In a multi-label dataset, $R_d(x, y)$ expresses how similar the labelsets of instances x and y are. The comparison between two labelsets L_x and L_y can be summarized in a 2x2 table:

	$L_y = 1$	$L_y = 0$
$L_x = 1$	a	b
$L_x = 0$	c	d

The value of a relates to classes that are present in both labelsets, d to classes that are absent in both and b and c to classes that occur in only one of the two sets. They can be aggregated to $R_d(x, y) = \frac{a+d}{a+b+c+d}$. We consider two ways to calculate the values a, b, c and d.

Firstly, the *Hamming based similarity relation* $R_d^{(1)}(\cdot, \cdot)$ is the complement of the normalized Hamming distance between the labelsets of two elements. It is defined as

$$R_d^{(1)}(x, y) = 1 - \frac{|L_x \,\Delta\, L_y|}{m}, \tag{9}$$

where the Δ operator constructs the symmetric difference between its two arguments and the $|\cdot|$ operation measures the cardinality of the resulting set. This relation can take on only a limited set of values, namely those in the set $\left\{\frac{i}{m} \mid i = 0, 1, \ldots, m\right\}$. Using this relation, the values a, b, c and d in the above table are set to the number of times their corresponding combination occurs (e.g. a is the number of classes present in both L_x and L_y).

In our second approach, a *label distribution based similarity relation* $R_d^{(2)}(\cdot, \cdot)$, a, b, c and d are based on label distribution information. Let $P = \langle p_1, p_2, \ldots, p_m \rangle$ be the prior class probability vector, where p_i represents the ratio of training elements that belong to the ith class. Relation $R_d^{(2)}$ takes into account whether a label is common or rare. It is based on the following idea. If a rare class label is present in both labelsets, this should be rewarded more than when they both contain the same common class label. Similarly, when they both do not contain a common class label l, this should be rewarded more than when a rare class is not present. This relation rewards unexpected behavior (presence of rare labels, absence of common labels) more than expected behavior. The calculation is presented in Algorithm 1. In line 5, an update of a is

Algorithm 1 $R_d^{(2)}(\cdot, \cdot)$ based on the label distribution.

Input: Training set X, labelsets L_1 and L_2, prior class probabilities $P = \langle p_1, p_2, \ldots, p_m \rangle$
Output: $R_d^{(2)}(L_1, L_2)$

```
1:  a ← 0, b ← 0, c ← 0, d ← 0
2:  for l = 1, 2, ..., m do
3:    if (L1)_i = (L2)_i then
4:      if (L1)_i = (L2)_i = 1 then
5:        a ← a + (1 − p_i)
6:      else
7:        d ← d + p_i
8:    else
9:      if (L1)_i = 1 then
10:       b ← b + 1/2
11:     else
12:       c ← c + 1/2
13: R_d(L1, L2) ← (a+d)/(a+b+c+d)
```

performed when both labelsets contain label l_i. We add $(1 - p_i)$ to the current value of a. This implies that a higher reward is given to less common labels. In line 7, label l_i is absent in both labelsets. Since this is more unexpected for a common label, a higher reward is given depending on how common l_i is. In lines 10 and 12, the values of b and c are updated when a label is present in only one of the two labelsets. We add $\frac{1}{2}$ to the current value, which is the average of p_i and $(1 - p_i)$.

It should be clear from their description that the characteristics of these two label similarity relations are different. As an example, we evaluate the similarity between selected labelsets of a dataset included in our experimental study, with $P = \langle 0.439, 0.234, 0.151, 0.650, 0.250, 0.202, 0.565, 0.209, 0.066, 0.049 \rangle$. We assess the label similarity of an instance x with $L_x = \langle 1, 1, 0, 1, 1, 1, 1, 1, 1, 0 \rangle$ with several other elements:

	$R_d^{(1)}$	$R_d^{(2)}$		$R_d^{(1)}$	$R_d^{(2)}$
$L_1 = \langle 1, 1, 0, 1, 1, 1, 1, 1, 0, 0\rangle$	0.9000	0.9029	$L_6 = \langle 1, 0, 0, 0, 0, 1, 0, 0, 0, 0\rangle$	0.4000	0.3419
$L_2 = \langle 1, 1, 0, 1, 1, 0, 1, 0, 0, 0\rangle$	0.7000	0.6712	$L_7 = \langle 0, 1, 0, 1, 0, 0, 0, 0, 0, 0\rangle$	0.4000	0.3049
$L_3 = \langle 0, 0, 0, 1, 0, 1, 1, 1, 0, 0\rangle$	0.6000	0.5627	$L_8 = \langle 0, 0, 0, 0, 0, 0, 0, 0, 1, 0\rangle$	0.3000	0.2447
$L_4 = \langle 1, 1, 1, 1, 0, 0, 1, 0, 0, 0\rangle$	0.5000	0.4637	$L_9 = \langle 0, 0, 0, 1, 0, 0, 0, 0, 0, 0\rangle$	0.3000	0.1357
$L_5 = \langle 1, 0, 0, 1, 0, 0, 1, 0, 0, 0\rangle$	0.5000	0.3821	$L_{10} = \langle 0, 0, 0, 0, 0, 0, 0, 0, 0, 1\rangle$	0.1000	0.0324

As stated above, the Hamming based relation $R_d^{(1)}$ can only take on a limited set of values, while relation $R_d^{(2)}$ has no such prior restriction. It is important to note that labelsets that have the same value for the first relation, do not necessarily coincide in the second relation. For example, consider L_8 and L_9, which both only contain one label and have a Hamming-based similarity of 0.3 with L_x. However, the label distribution based relation value of the former is almost twice as large as that of the latter. The reason is that both L_x and L_8 contain the rare class label l_9, while the shared label l_4 of L_x and L_9 is far more common in the dataset. We also observe large differences between some values of the two relations. As a result, we can expect different classification results from the FRONEC method when either $R_d^{(1)}$ or $R_d^{(2)}$ is used.

3.2.3. Computational complexity

Having described all components of our proposal, we are now in a position to evaluate its computational complexity. Despite its nearest neighbor approach, which generally implies a lazy learning nature and therefore a negligible training phase, FRONEC can precompute and store all pairwise label similarity values between the training instances. As should be clear from the quality measures definitions (6)–(8), label similarity comparisons are only computed between training elements. This allows these results to be precomputed, such that repeated calculations are avoided at classification time. The cost of this calculation is quadratic in the number of instances, but needs to be performed only once during training. The cost of one pairwise label similarity calculation is linear in the number of labels, independent of whether $R_d^{(1)}$ or $R_d^{(2)}$ is used.

To classify an instance x, the procedure described in the introduction of Section 3.2 is applied:

1. The first step is to construct the set $N(x)$ of the k nearest neighbors of x in the training set. This process has a computational cost that is linear in the number of instances and linear in the number of features. The latter is due to the cost of the feature similarity (or distance) calculation. These feature similarity values are also used in the second step. If these values can be stored, repeated calculations are avoided.
2. Next, the quality value $Q_x(y)$ is calculated for every training element y. Since the quality of y depends on the target instance x, it can not be precomputed during training. Since all $R(\cdot, x)$ and $R_d(\cdot, \cdot)$ similarity values have already been determined, the construction of the sets to aggregate in (6) and (7) is linear in the number of neighbors k. The OWA aggregation has a cost of $\mathcal{O}(k \log(k))$, due to the sorting step in its definition. Quality calculation (8) is slightly more costly, since it includes both (6) and (7). In particular, two sorting operations are required. In total, the second step of the classification procedure is linear in the number of instances and linearithmic in k.
3. Finally, the predicted labelset for x needs to be constructed. This requires a pass through set Y, for which the cost is linear in $|Y|$. This value is upper bounded by the total number of training instances. The presence of each class label needs to be evaluated. The third step in the classification of x is consequently linear in the number of training instances and the number of possible class labels.

In summary, if n is the number of training instances, m the number of possible class labels, d the number of features and k the number of neighbors, the computational cost of the training phase of FRONEC is $\mathcal{O}(n^2 \cdot m)$ and the cost to classify an instance is $\mathcal{O}(n \cdot d \cdot m \cdot k \log(k))$.

4. Experimental set-up

In this section, we establish the details of our experimental study, of which the results will be reported in Section 5. We describe the datasets (Section 4.1) on which we conduct our experiments and the metrics we use to evaluate the classification performance of the multi-label methods (Section 4.2). In Section 4.3, we specify how the different parameters of the algorithms are set. Section 4.4 describes the statistical tests used in our analysis.

4.1. Datasets

The majority of the publicly available multi-label datasets have a very high dimensionality, which may form an issue for nearest neighbor based classifiers [1]. To avoid this problem, we use the Mldatagen generator [36] to create synthetic multi-label datasets. We have used the HyperSpheres strategy and have fixed the number of features to 20, the number of labels to 10 and the number of instances to 5000. We have created a total number of 30 datasets, by varying the number of relevant, irrelevant and redundant features as well as the percentage of class noise. The names of the datasets reflect the characteristics of the attributes and the percentage of class noise. For example, dataset 'd-5-10-5-p30' has 5 relevant attributes, 10 irrelevant attributes, 5 redundant attributes and 30% class noise. We use 5-fold cross validation in all experiments. Aside from the synthetic datasets, we also include six real-world multi-label dataset with a relatively moderate

Table 1
Real-world multi-label datasets. We list the number of instances (nInst), features (nFeat) and number of possible labels (nLab).

Dataset	nInst	nFeat	nLab
Birds	645	260	19
Emotions	593	72	6
Flags	194	19	7
Music	592	71	6
Scene	2407	294	6
Yeast	2417	103	14

dimensionality. Their characteristics are listed in Table 1. These datasets are used in a final comparison between our proposal and state-of-the-art nearest neighbor multi-label classifiers in Section 5.3. All datasets and used partitions are available at http://www.cwi.ugent.be/sarah.php.

4.2. Evaluation metrics

There exists a wide spectrum of metrics to evaluate the performance of multi-label classifiers. We use the Hamming loss, *F*-measure, recall, precision and subset accuracy. These measures are example-based and expect a strict assignment of instances to classes (i.e. instead of a label ranking). This is appropriate in the context of nearest neighbor classification.

Let $x \in X$ be a test instance, L_x its true class vector and $\hat{L}_x$ the predicted class vector. The above evaluation metrics are defined as follows:

- Hamming loss:

$$hloss = \frac{1}{|X|}\frac{1}{m}\sum_{x \in X} |L_x \Delta \hat{L}_x|,$$

  with the Δ operator measures as defined in Section 3.2.2. The total number of prediction errors is divided by both the number of instances $|X|$ and number of labels m.
- *F*-measure:

$$F = \frac{2 \cdot p \cdot r}{p + r},$$

  where p and r are the precision and recall measures given by

$$p = \frac{1}{|X|}\sum_{x \in X} \frac{|L_x \cap \hat{L}_x|}{|\hat{L}_x|} \quad \text{and} \quad r = \frac{1}{|X|}\sum_{x \in X} \frac{|L_x \cap \hat{L}_x|}{|L_x|}.$$

  For each instance x, the recall compares the number of correctly predicted labels for x to all labels of x, while the precision compares the number of correctly predicted labels to all predicted labels.
- Subset accuracy:

$$SubAcc = \frac{1}{|X|}\sum_{x \in X} I(L_x = \hat{L}_x),$$

  where the indicator function $I(\cdot)$ evaluates to 1 if its argument is true and to 0 otherwise. This is the most stringent metric of the three, because it evaluates full equality of L_x and $\hat{L}_x$.

4.3. Methods and parameter settings

From the nearest neighbor based methods described in Section 2, we select the BRKNN-b, LPKNN, MLKNN, IBLR+ and MLDGC methods as representative methods. From a preliminary comparison, we observed that these methods have a good classification performance. Furthermore, their operation is easy to understand and they require only the k parameter to be set by the user. The MLKNN and IBLR methods are often included in experimental comparisons of newly proposed multi-label classifiers. MLDGC is a recently proposed method from [34]. Similar to our set-up, the authors compared this method within the family of nearest neighbor multi-label classifiers. MLDGC was shown to outperform the other methods in this study, but, as noted in the conclusion of Reyes et al. [34], it remains important to compare newly proposed nearest neighbor multi-label methods to both MLDGC and the other state-of-the-art nearest neighbor classifiers. We follow their guideline here.

Since these methods are all related to the nearest neighbor classification paradigm, they all depend on the parameter k, the number of nearest neighbors used in the prediction process. In other experimental studies, its value is usually set to 10. However, in this paper, we do not fix this value beforehand, but allow the classifier to set its own k value during the training phase, depending on the data at hand. A method does so by evaluating its classification performance, measured by

the leave-one-out subset accuracy on the training set, for values of k between 1 and 20. It sets the parameter to the value yielding the best performance. We have opted to use the subset accuracy as evaluation measure during this process, because it is the most exact one, that is, it counts predictions that are entirely correct. Using leave-one-out validation on the training set is a valid parameter validation procedure, as the test set is kept completely separate during the optimization step. The results reported in Section 5 are evaluations on the test set, that was not known during the parameter tuning process.

We evaluate and compare six alternatives of our FRONEC method. As described in Section 3.2.1, versions FRONEC-1, FRONEC-2 and FRONEC-3 differ from each other by their use of the instance quality measure. Each of these methods has a choice between the two label similarity relations described in Section 3.2.2. As part of our experimental study, we compare these six alternatives among each other. In expressions (6)–(8), we use the Łukasiewicz connectives, that is, $\mathcal{I}(a,b) = \min(1, 1-a+b)$ and $\mathcal{T}(a,b) = \max(a+b-1, 0)$.

The fuzzy rough operators within FRONEC depend on the definition of the feature similarity relation $R(\cdot, \cdot)$. For two instances x and y, we set the value of this relation to

$$R(x,y) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} R_f(x,y), \tag{10}$$

where $\mathcal{F}$ is the set of all features in the dataset. The feature-wise relation is defined as

$$R_f(x,y) = 1 - \frac{|f(x) - f(y)|}{range(f)}$$

when f is a numeric feature. In this expression, $range(f)$ evaluates to the range of the feature f and $f(x)$ and $f(y)$ are the values of x and y for this feature. When f is nominal, we set

$$R_f(x,y) = \begin{cases} 1 & \text{if } f(x) = f(y) \\ 0 & \text{if } f(x) \neq f(y). \end{cases}$$

For the sake of a sensible and fair comparison, we set the distance relation $d(\cdot, \cdot)$ used by all other methods to the complement of the similarity relation, that is, $d(x,y) = 1 - R(x,y)$. This implies that the k nearest neighbors of an element x are those that are most similar to this instance according to relation (10).

4.4. Statistical analysis

We complement the experimental results with a statistical analysis using non-parametric tests. To make a pairwise comparison between the performance of two methods, we rely on the Wilcoxon signed-ranks test [45]. To compare methods M1 and M2, the differences in their performance results (in absolute value) are ranked. The smallest difference is assigned rank 1, the largest rank D, with D the number of datasets in the study. The ranks of positive and negative differences are summed up to R^+ and R^- respectively. The former refers to wins for method M1, the latter to wins for M2. A test statistic and associated p-value are derived from R^+ and R^-. When the p-value is lower than the predefined significance level α, a significant difference in performance is concluded. In this study, we set the significance level to $\alpha = 0.05$. When reporting results of the Wilcoxon test, we list R^+, R^- and the p-value.

To assess whether significant differences exist within a group of methods, we use the Friedman test [13] and Holm post-hoc procedure [19]. The former ranks the performances of the methods and decides whether any significant differences are present among them. If the associated p-value is smaller than the significance level α, it is concluded that significant differences exist, but there is no indication as to where these may be found. The post-hoc procedure is used for this purpose. The lowest ranked method is used as control method to which all other methods are compared. When the p-value of such a comparison is smaller than α, we conclude that the control method significantly outperforms the method it was compared to.

5. Experimental evaluation

We now proceed with the empirical analysis of our proposal and a comparison of FRONEC to popular nearest neighbor multi-label classifiers. In Section 5.1, we compare the six versions of FRONEC among each other. Sections 5.2 and 5.3 compare our proposal to the five selected nearest neighbor multi-label classifiers. The former does so on the 30 synthetic datasets, while the latter uses the six real-world datasets listed in Table 1. As a reminder, the full experimental results are available on http://www.cwi.ugent.be/sarah.php.

5.1. FRONEC variants

In this section, we use the 30 synthetic datasets described in Section 4.1 to compare six versions of our proposal, namely the FRONEC-1, FRONEC-2 and FRONEC-3 methods with the two possible label similarity relations $R_d^{(1)}$ and $R_d^{(2)}$. In this stage, we fix the value of k to 20 and select the additive OWA weighting scheme within the instance quality calculations (6)–(8). We observed that these settings provide good results on average. In later sections, we use the procedure described in Section 4.3 to allow FRONEC to choose the OWA weighting scheme and k value itself during the training phase.

Table 2
Comparison of the different FRONEC methods using the additive weighting scheme for the OWA aggregations. The value of k was set to 20.

	hloss		*F*		*SubAcc*	
	$R_d^{(1)}$	$R_d^{(2)}$	$R_d^{(1)}$	$R_d^{(2)}$	$R_d^{(1)}$	$R_d^{(2)}$
FRONEC-1	0.2666	0.2776	0.4174	0.4396	0.0625	0.0603
FRONEC-2	0.2653	0.2787	0.4290	0.4489	0.0720	0.0679
FRONEC-3	0.2616	0.2722	0.4251	0.4476	0.0700	0.0685

Table 3
Results of the Wilcoxon test comparison of the quality measures (6)–(8) within FRONEC for both label similarity relations. *P*-values that imply significant differences at the 5% significance level are printed in boldface.

	$R_d^{(1)}$				$R_d^{(2)}$			
	Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
hloss	$Q^{(2)}$ vs $Q^{(1)}$	316.0	149.0	.080864	$Q^{(1)}$ vs $Q^{(2)}$	243.0	192.0	.565543
	$Q^{(3)}$ vs $Q^{(1)}$	465.0	0.0	**.000001**	$Q^{(3)}$ vs $Q^{(1)}$	425.5	9.5	**.000005**
	$Q^{(3)}$ vs $Q^{(2)}$	432.5	2.5	**.000002**	$Q^{(3)}$ vs $Q^{(2)}$	465.0	0.0	**.000001**
F	$Q^{(2)}$ vs $Q^{(1)}$	422.0	13.0	**.000009**	$Q^{(2)}$ vs $Q^{(1)}$	477.0	18.0	**.000009**
	$Q^{(3)}$ vs $Q^{(1)}$	407.0	58.0	**.000297**	$Q^{(3)}$ vs $Q^{(1)}$	422.0	43.0	**.000093**
	$Q^{(2)}$ vs $Q^{(3)}$	328.0	137.0	**.047743**	$Q^{(2)}$ vs $Q^{(3)}$	266.5	198.5	.475675
SubAcc	$Q^{(2)}$ vs $Q^{(1)}$	373.0	62.0	**.000672**	$Q^{(2)}$ vs $Q^{(1)}$	370.5	94.5	**.004289**
	$Q^{(3)}$ vs $Q^{(1)}$	453.5	11.5	**.000005**	$Q^{(3)}$ vs $Q^{(1)}$	455.0	10.0	**.000005**
	$Q^{(2)}$ vs $Q^{(3)}$	291.5	143.5	.103411	$Q^{(3)}$ vs $Q^{(2)}$	288.0	177.0	.246954

In Table 2, we present the average results of the six methods taken over the 30 synthetic datasets. We use the Hamming loss, *F*-measure and subset accuracy measures to evaluate their classification performance. Aside from these average results, we also take statistical comparisons by means of the Wilcoxon test into account.

5.1.1. Choice of label similarity relation

We observe that the choice of label similarity relation has the largest influence on the Hamming loss and *F*-measure metrics. This is reflected in the average values reported in the table, but also in the results per dataset available on our web page. For the Hamming loss, almost all datasets prefer the $R_d^{(1)}$ relation, while $R_d^{(2)}$ gives better *F*-measure values on all datasets. When comparing the results of the two label similarity relations with the Wilcoxon test, we observe that $R_d^{(1)}$ is always significantly better than $R_d^{(2)}$ for the Hamming loss ($R^+ = 453$, $R^- = 12$, $p = .000005$ for FRONEC-1; $R^+ = 465$, $R^- = 0$, $p = .000001$ for FRONEC-2; $R^+ = 435$, $R^- = 0$, $p = .000002$ for FRONEC-3), while the reverse holds for the *F*-measure ($R^+ = 465$, $R^- = 0$, $p = .000001$ for FRONEC-1; $R^+ = 464$, $R^- = 1$, $p = .000002$ for FRONEC-2; $R^+ = 465$, $R^- = 0$, $p = .000002$ for FRONEC-3). With respect to the subset accuracy, the results are less clear-cut, although the majority of the datasets prefer $R_d^{(1)}$, which is also expressed by the average values reported in Table 2 and the results of the Wilcoxon test comparing $R_d^{(1)}$ to $R_d^{(2)}$ ($R^+ = 379.5$, $R^- = 85.5$, $p = .002353$ for FRONEC-1; $R^+ = 433$, $R^- = 32$, $p = .000031$ for FRONEC-2; $R^+ = 306$, $R^- = 129$, $p = .051715$ for FRONEC-3). Since there does not seem to be a general inclination towards one of the two label similarity relations, we continue to use both. It would not be prudent to discard $R_d^{(2)}$, as it significantly outperforms $R_d^{(1)}$ with respect to the *F*-measure.

The difference in the relation preferences can be explained by the fact that more labels are generally predicted when $R_d^{(2)}$ is used, that is, the size of the predicted labelsets is larger on average when using $R_d^{(2)}$ compared to using $R_d^{(1)}$. This will be shown and discussed in further detail in the remainder of the paper. An explanation of this behavior lies with the comparison conducted in Section 3.2.2. Due to its definition and the fact that $R_d^{(1)}$ can take on only a limited set of values, less extreme differences in the values of this relation between large and small labelsets are observed. Small labelsets are not so easily dominated by larger ones, while, based on the example comparison in Section 3.2.2, $R_d^{(2)}$ seems to be more sensitive to the size of a labelset. Furthermore, when comparing the labelsets of two instances x and y, the situation $(L_x)_i \neq (L_y)_i$ is penalized more severely by $R_d^{(1)}$ than by $R_d^{(2)}$. As a result, the latter is naturally allowed to make more guesses in its predictions and will lead to more predicted labels. Its superior values for the *F*-measure show that this characteristic is not taken to the extreme (e.g. predicting every label) and a favorable balance between recall and precision is obtained. We come back to this point later in this paper.

5.1.2. Choice of quality measure

The results of the Wilcoxon tests comparing FRONEC-1, FRONEC-2 and FRONEC-3 can be found in Table 3. When comparing the three versions, we observe that the latter two provide better results than the former, both on average as well as based on the results per dataset and the statistical analysis. The difference between these methods lies with their use of the instance quality measure. The inclusion of the operator related to the fuzzy rough upper approximation is shown to be more beneficial than that of the one related to the lower approximation. The lower approximation relies on a fuzzy

Table 4
Summary of the classification results on the 30 synthetic datasets. We present the average (av) and standard deviation (stdev) across these datasets, as well as the number of times a method attained the best or worst result for a measure. The best average values are printed in bold.

		BRKNN-b	LPKNN	MLKNN	IBLR+
hloss	av ± stdev	0.2785 ± 0.0847	0.3093 ± 0.0998	0.2461 ± 0.0859	**0.2395** ± 0.0848
	nBest/nWorst	0/4	0/23	2/0	28/0
SubAcc	av ± stdev	0.0591 ± 0.0499	0.0571 ± 0.0600	0.0445 ± 0.0412	0.0514 ± 0.0433
	nBest/nWorst	1/0	0/17	2/11	10/0
F	av ± stdev	0.4438 ± 0.1202	0.4211 ± 0.1071	0.3398 ± 0.2016	0.3659 ± 0.2042
	nBest/nWorst	8/0	4/10	0/17	6/2
p	av ± stdev	0.4986 ± 0.1339	0.4343 ± 0.0918	0.6244 ± 0.0770	**0.6463** ± 0.0715
	nBest /nWorst	0/4	0/23	4/0	26/0
r	av ± stdev	0.4101 ± 0.1255	0.4125 ± 0.1225	0.2634 ± 0.1826	0.2870 ± 0.1922
	nBest/nWorst	7/0	8/0	0/27	0/3
		MLDGC	FRONEC-$R_d^{(1)}$	FRONEC-$R_d^{(2)}$	
hloss	av ± stdev	0.2710 ± 0.0792	0.2685 ± 0.0875	0.2823 ± 0.0929	
	nBest/nWorst	0/3	0/0	0/0	
SubAcc	av ± stdev	0.0557 ± 0.0474	**0.0715** ± 0.0621	0.0672 ± 0.0619	
	nBest/nWorst	1/2	15/0	2/2	
F	av ± stdev	0.4161 ± 0.1457	0.4293 ± 0.1337	**0.4471** ± 0.1250	
	nBest/nWorst	0/1	0/0	13/0	
p	av ± stdev	0.4994 ± 0.1238	0.5096 ± 0.1114	0.4789 ± 0.0987	
	nBest/nWorst	0/3	0/0	0/0	
r	av ± stdev	0.3738 ± 0.1652	0.3806 ± 0.1487	**0.4259** ± 0.1489	
	nBest/nWorst	0/0	0/0	15/0	

implicator $\mathcal{I}$, which models an implication. The quality $Q_x^{(1)}(y)$ in the prediction of x evaluates how strongly the similarity between x and its neighbors z implies the similarity of the labelsets L_y and L_z. The upper approximation on the other hand uses a t-norm $\mathcal{T}$, related to a conjunction. Since a t-norm is commutative, the similarity between x and its neighbors z and the similarity between L_y and L_z have equal importance. The value of $Q_x^{(2)}(y)$ aggregates the fuzzy conjunctions of these two types similarity values. In the search of a consensus labelset among those of the nearest neighbors of x, the second approach seems more intuitive. We conclude that the FRONEC-2 alternative can be preferred overall, since it obtains good classification results and is computationally more efficient than FRONEC-3 (Section 3.2.3), while not being significantly inferior to the latter for all evaluation measures. Nevertheless, Table 3 does show that FRONEC-3 significantly outperforms FRONEC-2 with respect to the Hamming loss. If a user is particularly interested in this metric, we advise the use of FRONEC-3 instead of FRONEC-2.

5.2. Comparison on synthetic datasets

In this section, we compare the performance of FRONEC to the selected nearest neighbor based multi-label classifiers on the 30 synthetic datasets described in Section 4.1. The comparison on the real-world datasets from Table 1 is postponed to Section 5.3. We use version FRONEC-2, which we denote as FRONEC in the remainder of this discussion. Our method depends on two internal parameters, the number of neighbors k and the weighting scheme for the OWA aggregation in (7). We use the optimization procedure in Section 4.3 to let FRONEC select its own setting during the training phase, depending on the data under consideration. For k, it evaluates all natural numbers between 1 and 20. The candidate weighting schemes are the additive, exponential and inverse additive weights listed in Section 3.1 as well as the weights that correspond to the traditional fuzzy rough model, by using the strict maximum in (7). The median k values selected by the internal optimization procedures are 18.5 (BRKNN-b), 19 (LPKNN), 17 (MLKNN), 15.5 (IBLR+), 15 (MLDGC), 19 (FRONEC-$R_d^{(1)}$) and 19 (FRONEC-$R_d^{(2)}$). The FRONEC methods most often select the additive OWA weight setting. Note that although FRONEC sets two parameters instead of one, this does not lead to an unfair comparison with the other methods. The reason is that BRKNN-b, LPKNN, MLKNN, IBLR+ and MLDGC have only one user-defined value, which is k. They do not require the specification of additional parameters. Furthermore, since the FRONEC methods usually select the additive OWA weighting scheme, fixing this choice would not greatly harm the performance either. This can also be taken into account when the parameter optimization step of FRONEC is deemed too time-consuming and the user would prefer to only optimize the k parameter.

The full results for the different performance metrics are again made available on our web page. A summary of these results is presented in Table 4. We provide the average value and standard deviation of the results across the 30 synthetic datasets. For each evaluation measure, the best average value is printed in bold. Note that for the F-measure, recall, precision and subset accuracy the best result is the highest value, while for the Hamming loss this is the lowest one. For each method, we also count the number of times it yielded the best (nBest) or worst (nWorst) result on these datasets.

To complement these results, we report the results of the statistical analysis in Tables 5–6. The former presents the analysis of the Friedman test and Holm post-hoc procedure, while the latter conducts a pairwise comparison of the FRONEC

Table 5
Statistical analysis of the results summarized in Table 4. We use the Friedman test and the Holm post-hoc procedure. *P*-values of the latter implying statistical significance are printed in bold.

Hamming loss ($p_{Friedman} \leq .000001$)			**Subset accuracy** ($p_{Friedman} = .000001$)		
Method	Friedman rank	p_{Holm}	Method	Friedman rank	p_{Holm}
BRKNN-b	4.5667	**≤.000001**	BRKNN-b	3.6333	**.046302**
LPKNN	6.7000	**≤.000001**	LPKNN	5.6000	**≤.000001**
MLKNN	1.9333	.120233	MLKNN	4.7167	**.000126**
IBLR+	1.0667	–	IBLR+	3.5833	**.046302**
MLDGC	4.3333	**≤.000001**	MLDGC	4.2167	**.003643**
FRONEC-$R_d^{(1)}$	3.9333	**.000001**	FRONEC-$R_d^{(1)}$	2.3667	–
FRONEC-$R_d^{(2)}$	5.4667	**≤.000001**	FRONEC-$R_d^{(2)}$	3.8833	**.019635**
***F*-measure** ($p_{Friedman} \leq .000001$)			**Precision** ($p_{Friedman} \leq .000001$)		
BRKNN-b	2.5833	.135166	BRKNN-b	4.5667	**≤.000001**
LPKNN	4.7000	**.000001**	LPKNN	6.7000	**≤.000001**
MLKNN	6.4333	**≤.000001**	MLKNN	1.8667	.188593
IBLR+	4.6167	**.000001**	IBLR+	1.1333	–
MLDGC	4.3167	**.000013**	MLDGC	4.5000	**≤.000001**
FRONEC-$R_d^{(1)}$	3.6000	**.001821**	FRONEC-$R_d^{(1)}$	3.8667	**.000002**
FRONEC-$R_d^{(2)}$	1.7500	–	FRONEC-$R_d^{(2)}$	5.3667	**≤.000001**
Recall ($p_{Friedman} \leq .000001$)					
BRKNN-b	2.8667	**.019769**			
LPKNN	3.0667	**.014322**			
MLKNN	6.9000	**≤.000001**			
IBLR+	5.9667	**≤.000001**			
MLDGC	3.8000	**.000193**			
FRONEC-$R_d^{(1)}$	3.8333	**.000193**			
FRONEC-$R_d^{(2)}$	1.5667	–			

methods to the other methods by means of the Wilcoxon test. We include the pairwise comparison to better evaluate the individual differences between our proposal and the state-of-the-art nearest neighbor based multi-label methods.

5.2.1. Summary of results

Based on the results in Table 4 and the statistical analysis in Tables 5–6, we can observe the following:

- *Hamming loss:* the dominance of the IBLR+ method is clear. It has the lowest average result and yields the best value on 28 out of the 30 datasets. The statistical analysis in Table 5 shows that IBLR+ significantly outperforms all others except MLKNN for this evaluation measure. The results of the Wilcoxon tests confirm that IBLR+ significantly outperforms our proposal with respect to the Hamming loss. The same holds for MLKNN. MLDGC also significantly outperforms FRONEC-$R_d^{(2)}$. However, FRONEC-$R_d^{(1)}$ is significantly better than FRONEC-$R_d^{(2)}$ as well as LPKNN for this measure. This version also outperforms BRKNN-b and MLDGC, albeit not significantly.
- *Subset accuracy:* the highest average result is obtained by our FRONEC method using relation $R_d^{(1)}$. It also attains the most wins, namely on 15 out of the 30 datasets. This method is assigned the lowest Friedman rank in Table 5 and significantly outperforms all others. This is remarkable, because, as described in Section 4.3, all methods optimize this measure internally, so we could have expected their results to be competitive with each other. This is not the case and our FRONEC-$R_d^{(1)}$ is the best general option when one is most interested in the subset accuracy measure. The dominance of FRONEC-$R_d^{(1)}$ is confirmed by the Wilcoxon tests in Table 6.
- *F-measure, recall and precision:* the highest average value for the *F*-measure is obtained by our FRONEC method using label similarity relation $R_d^{(2)}$. This method also obtains the most wins, namely on 13 out of the 30 datasets. It is closely followed by BRKNN-b. In the statistical analysis in Table 5, FRONEC-$R_d^{(2)}$ is assigned the lowest Friedman rank and is shown to significantly outperform all other methods except BRKNN-b. The same holds for the Wilcoxon test in Table 6. Although the same phenomenon can be observed in the experimental results of [34], it is remarkable that IBLR+ and MLKNN perform so poorly in terms of the *F*-measure, while they provided the best Hamming loss results. The explanation lies with their recall and precision values, which are not at all balanced. These methods yield very poor results for the former measure and very good for the latter. This means that most of their predicted labels are correct (high precision), but they fail to predict many relevant labels (low recall). The other methods, in particular FRONEC-$R_d^{(2)}$, obtain a better recall-precision balance, which is reflected in a superior *F*-measure.

In summary, we can conclude that FRONEC performs best for the subset accuracy, *F*-measure and recall, but that IBLR+ is preferred in the evaluation by the Hamming loss and precision. As should be clear from the descriptions provided in Section 4.2, the subset accuracy and recall are related measures, as are the Hamming loss and precision. Consequently, it is not surprising that a method that performs well for one measure in a pair, also attains good results for the other. The *F*-measure evaluates the trade-off between recall and precision. Since no method dominates all others for both measures,

Table 6
Pairwise comparison between FRONEC (FR) and other methods with the Wilcoxon test. *P*-values implying significant differences at the 5% significance level are printed in boldface (in favor of a FRONEC method) or underlined (in favor of some other method).

Hamming loss				Subset accuracy			
Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
FR-$R_d^{(1)}$ vs BRKNN-b	303.0	162.0	.143162	FR-$R_d^{(1)}$ vs BRKNN-b	416.5	48.5	**.000136**
FR-$R_d^{(1)}$ vs LPKNN	465.0	0.0	**.000002**	FR-$R_d^{(1)}$ vs LPKNN	465.0	0.0	**.000002**
MLKNN vs FR-$R_d^{(1)}$	465.0	0.0	.000002	FR-$R_d^{(1)}$ vs MLKNN	400.0	65.0	**.000531**
IBLR+ vs FR-$R_d^{(1)}$	465.0	0.0	.000002	FR-$R_d^{(1)}$ vs IBLR+	314.0	151.0	.090014
FR-$R_d^{(1)}$ vs MLDGC	270.0	195.0	.432080	FR-$R_d^{(1)}$ vs MLDGC	376.5	58.5	**.000544**
BRKNN-b vs FR-$R_d^{(2)}$	287.0	178.0	.256721	FR-$R_d^{(2)}$ vs BRKNN-b	288.0	177.0	.249392
FR-$R_d^{(2)}$ vs LPKNN	465.0	0.0	**.000002**	FR-$R_d^{(2)}$ vs LPKNN	454.5	10.5	**.000004**
MLKNN vs FR-$R_d^{(2)}$	465.0	0.0	.000002	FR-$R_d^{(2)}$ vs MLKNN	331.5	133.5	**.040187**
IBLR+ vs FR-$R_d^{(2)}$	465.0	0.0	.000001	FR-$R_d^{(2)}$ vs IBLR+	269.0	196.0	.445469
MLDGC vs FR-$R_d^{(2)}$	380.0	85.0	.002334	FR-$R_d^{(2)}$ vs MLDGC	318.0	147.0	.075401
FR-$R_d^{(1)}$ vs FR-$R_d^{(2)}$	465.0	0.0	**.000001**	FR-$R_d^{(1)}$ vs FR-$R_d^{(2)}$	402.0	33.0	**.000063**
***F*-measure**				**Precision**			
Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
BRKNN-b vs FR-$R_d^{(1)}$	407.5	57.5	.000296	FR-$R_d^{(1)}$ vs BRKNN-b	299.5	165.5	.162853
FR-$R_d^{(1)}$ vs LPKNN	314.0	151.0	.091680	FR-$R_d^{(1)}$ vs LPKNN	465.0	0.0	**.000002**
FR-$R_d^{(1)}$ vs MLKNN	464.0	1.0	**.000002**	MLKNN vs FR-$R_d^{(1)}$	465.0	0.0	.000002
FR-$R_d^{(1)}$ vs IBLR+	404.0	61.0	**.000390**	IBLR+ vs FR-$R_d^{(1)}$	465.0	0.0	.000002
FR-$R_d^{(1)}$ vs MLDGC	368.0	97.0	**.005039**	FR-$R_d^{(1)}$ vs MLDGC	309.0	156.0	.113248
FR-$R_d^{(2)}$ vs BRKNN-b	294.0	171.0	.202225	BRKNN-b vs FR-$R_d^{(2)}$	328.0	137.0	.048318
FR-$R_d^{(2)}$ vs LPKNN	412.5	52.5	**.000198**	FR-$R_d^{(2)}$ vs LPKNN	465.0	0.0	**.000002**
FR-$R_d^{(2)}$ vs MLKNN	465.0	0.0	**.000002**	MLKNN vs FR-$R_d^{(2)}$	465.0	0.0	.000002
FR-$R_d^{(2)}$ vs IBLR+	404.0	31.0	**.000053**	IBLR+ vs FR-$R_d^{(2)}$	465.0	0.0	.000002
FR-$R_d^{(2)}$ vs MLDGC	465.0	0.0	**.000002**	MLDGC vs FR-$R_d^{(2)}$	349.0	116.0	.016106
FR-$R_d^{(2)}$ vs FR-$R_d^{(1)}$	462.0	3.0	**.000002**	FR-$R_d^{(1)}$ vs FR-$R_d^{(2)}$	463.0	2.0	**.000002**
Recall							
Comparison	R^+	R^-	p				
BRKNN-b vs FR-$R_d^{(1)}$	394.0	71.0	.000810				
LPKNN vs FR-$R_d^{(1)}$	371.0	94.0	.004250				
FR-$R_d^{(1)}$ vs MLKNN	465.0	0.0	**.000002**				
FR-$R_d^{(1)}$ vs IBLR+	464.0	1.0	**.000002**				
FR-$R_d^{(1)}$ vs MLDGC	290.5	174.5	.227683				
FR-$R_d^{(2)}$ vs BRKNN-b	304.0	161.0	.137613				
FR-$R_d^{(2)}$ vs LPKNN	325.0	140.0	.055767				
FR-$R_d^{(2)}$ vs MLKNN	465.0	0.0	**.000002**				
FR-$R_d^{(2)}$ vs IBLR+	465.0	0.0	**.000002**				
FR-$R_d^{(2)}$ vs MLDGC	465.0	0.0	**.000002**				
FR-$R_d^{(2)}$ vs FR-$R_d^{(1)}$	465.0	0.0	**.000002**				

it is important to include the *F*-measure as summary metric to evaluate this trade-off. The results in Tables 4–6 show that FRONEC provides the best *F*-measure results, which can be interpreted as the most appropriate way to balance the different behavior measured by the recall and precision. From this observation, we can conclude that our method is the overall best one, taking all included evaluation measures into account.

The results in Table 6 also stress the large effect that the choice of label similarity relation has on the performance of FRONEC. For each evaluation measure, the difference between FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ is found to be significant, sometimes in favor of the former (Hamming loss, subset accuracy, precision) and sometimes in favor of the latter (*F*-measure, recall). The characteristics of the two label similarity relations have been explained in Section 5.1.

Interestingly, the MLDGC method, although recently proposed and shown in the original paper to outperform the other included methods, does not perform notably well on these datasets. The explanation lies with the characteristics of the datasets used in our study. First, we have opted to limit ourselves to relatively low dimensional datasets, as the suitability of a nearest neighbor approach decreases when the number of features increases and the locality property is lost. As we focus on nearest neighbor related methods, we feel it is appropriate to limit the number of features. When a high dimensional dataset needs to be processed with our proposal or one of the other included methods, we advise the application of a feature selection technique prior to the classification step. Considering the results in [34], it can be observed that MLDGC obtains the most wins and highest performance differences on relatively high dimensional datasets. This indicates that this method may be more robust against the high dimensionality than other nearest neighbor classifiers, but it does not take away from the fact that the intuition behind it is somehow lost in the process. A second dataset property that causes the inferior performance of MLDGC is the label density of our datasets, measured as the average number of labels per sample divided by the total number of possible classes. The authors of [34] acknowledged that their method performed best on

datasets with a low label density. The label density of the datasets in [34] ranges from 0.009 to 0.485, while it ranges between 0.119 and 0.429 for our synthetic datasets, with an average of 0.282. These values are relatively high compared to those in the MLDGC study, which explains the lesser performance of this method.

We also wish to stress the comparison between FRONEC on the one hand and BRKNN-b and LPKNN on the other. These methods are highly related. To classify an instance, they first determine its k nearest neighbors. Next, the labelsets of these neighbors are aggregated into a prediction. BRKNN-b and LPKNN do so by considering the labelsets of the neighbors themselves, while FRONEC searches the dataset for a labelset that forms an appropriate consensus. The results in Table 4 show that more accurate predictions are obtained by using the fuzzy rough approach incorporated in FRONEC.

5.2.2. Deeper discussion on FRONEC, IBLR+ and MLKNN

A pertinent question is why the precision (and consequently the Hamming loss) of FRONEC is relatively low compared to that of IBLR+ and MLKNN, which are the best performers for this measure. IBLR+ and MLKNN are popular methods used in comparative studies on multi-label classifiers, so a careful comparison of our proposal with these algorithms is warranted. The results above show that FRONEC outperforms IBLR+ and MLKNN with respect to the subset accuracy, F-measure and recall, but not for the precision and Hamming loss. As stated above, the superior F-measure result of FRONEC shows that it reaches the best trade-off between the two different prediction aspects represented by precision and recall. However, it remains crucial to understand why the precision of FRONEC is relatively low.

The answer to our question lies with the cardinality of the predicted labelsets. Based on our empirical evidence, the IBLR+ and MLKNN methods make consistently fewer predictions than FRONEC, that is, the number of labels predicted for an instance x by IBLR+ or MLKNN is lower than the number predicted by FRONEC. On our 30 synthetic datasets, for which the highest possible cardinality of a labelset is $m = 10$, the average difference in cardinality between the true and predicted labelsets is 1.5519 (IBLR+), 1.6238 (MLKNN), 0.7558 (FRONEC-$R_d^{(1)}$) and 0.3417 (FRONEC-$R_d^{(2)}$). The higher these values, the smaller the predicted labelsets are compared to the true ones. For completeness, the average values for BRKNN-b, LPKNN and MLDGC are 0.5132, 0.1306 and 0.7638 respectively. The differences in label cardinality per dataset can be found on the associated web page. On each dataset, MLKNN and IBLR+ yield the largest difference in true and predicted labelset cardinality and consequently the smallest predicted labelsets.

In the definition of the precision recalled in Section 4.2, the size of the predicted labelset appears in the denominator. When the denominator of a fraction decreases, the overall value increases. As the predicted labelset sizes are smaller for IBLR+ and MLKNN than they are for FRONEC and these values are used in the denominator of the precision definition, a higher result for the first two methods is a logical consequence. We also note the difference between the values for FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$, which relates back to a point made in Section 5.1. The only difference between these two methods is their label similarity relation. Using relation $R_d^{(2)}$ tends to result in more predictions, which is reflected in its smaller difference between the true and predicted labelset cardinalities, larger labelset sizes and, finally, its lower precision value.

One could argue that the predicted labelset cardinality is not the only component influencing the precision measure. Indeed, even when only a few labels are predicted, the precision will still be low when these predictions are incorrect. In order to verify whether the size of the predicted labelset is truly the most important factor influencing the precision difference between IBLR+ and MLKNN and FRONEC in our study, we have examined whether the correct predictions made by the former two methods are also discovered by FRONEC. This is the case. On average over the 30 datasets, FRONEC-$R_d^{(1)}$ finds 93.36% of the correct predictions made by IBLR+ and 96.90% of the correct predictions made by MLKNN. For FRONEC-$R_d^{(2)}$, these values are 93.54% and 96.19% respectively. This implies that our method very rarely misses a correct prediction of IBLR+ or MLKNN and corroborates our statement that, when taking the five evaluation measures into account, FRONEC can be preferred over IBLR+ and MLKNN.

The choice between FRONEC and IBLR+ and MLKNN depends on the relative importance or cost of false positive and false negative predictions. Only in applications where false positives are severely penalized should IBLR+ and MLKNN be used instead of FRONEC. This comes at the risk of possibly missing many correct classes (low recall).

5.2.3. Complexity comparison

When we compare the execution time of these methods, we observe average training times of 10.3279 s (LPKNN), 15.9900 s (MLDGC), 16.9344 s (BRKNN-b), 18.4833 s (MLKNN) and 69.3622 s (IBLR+) compared to 184.7964 s and 185.1450 s for FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ respectively. These values are taken as averages over ten runs of the algorithms and the results per dataset can be found on our web page. The higher training time of FRONEC is mainly due to its optimization of an additional parameter during the procedure described in Section 4.3. While its competing methods only need to set the value for k, FRONEC also makes a choice between four candidate OWA weighting schemes. In terms of the testing time, all methods can be considered fast with average times of 2.1851 s (BRKNN-b), 1.8784 s (LPKNN), 1.8891 s (MLKNN), 2.0514 s (IBLR+), 1.8982 s (MLDGC), 6.6611 s (FRONEC-$R_d^{(1)}$) and 6.9777 s (FRONEC-$R_d^{(2)}$) to classify a full test set. The test sets of the synthetic datasets all consist of 1000 instances. A more detailed discussion on execution times can be found in Section 5.3.2, but we discuss the theoretical complexity of the methods here. Recall that the theoretical complexity of our proposal has been derived in detail in Section 3.2.3. We disregard the parameter optimization step in this analysis.

Training. The simplest methods are BRKNN-b and LPKNN, since they have no real training phases and simply store all training instances for later use. MLKNN, MLDGC, IBLR+ and FRONEC do perform some calculations at training time.

For each training instance, MLKNN locates its nearest neighbors and counts the occurrences of the classes among these elements. The combined cost of these two procedures is $\mathcal{O}(n^2 \cdot d + n \cdot k \cdot m)$. It is also beneficial to compute and store the class counts of the training instances only once, which has a cost $\mathcal{O}(n \cdot m)$. Based on the precomputed values, MLKNN derives prior and posterior probabilities of all classes. The calculation of the former depends on the overall class counts and can be computed at $\mathcal{O}(m)$ total cost. The latter requires the construction of two frequency arrays, which takes up $\mathcal{O}(n + k)$ for each class. In total, the probability calculations can be obtained at $\mathcal{O}(m \cdot (n + k))$ cost and the complete training phase complexity of MLKNN is $\mathcal{O}(n^2 \cdot d + n \cdot k \cdot m)$.

In [34], the reported training cost of MLDGC is $\mathcal{O}(n^2 \cdot d)$. This corresponds to locating the nearest neighbors for each training instance. However, the additional cost of the remaining internal calculations is ignored. The label similarity values between all pairs of training instances are derived in $\mathcal{O}(n^2 \cdot m)$, while the neighborhood weight values can be determined at a total cost of $\mathcal{O}(n \cdot k)$. The total training cost of MLDGC is therefore $\mathcal{O}(n^2 \cdot (d + m) + n \cdot k)$.

The IBLR+ method constructs a binary logistic regression classifier for each class at training time. It first transforms the data such that the logistic regression method uses neighborhood information, represented in class confidence values, as well as original data features as predictors. For each class l, the class confidence feature is computed as the percentage of neighbors of the training instance that belong to class l. The construction of the new dataset requires the neighborhood calculation for all training instances ($\mathcal{O}(n^2 \cdot d)$) and the label confidence calculation for all training instances ($\mathcal{O}(n \cdot k \cdot m)$) and has a total cost of $\mathcal{O}(n^2 \cdot d + n \cdot k \cdot m)$. The training time of a logistic classifier is dominated by the cost of the internal optimization procedure, needed to determine the logistic regression coefficients. We denote this cost as $\mathcal{O}(Opt)$. Among other things, this incorporates the cost of computing the objective function and gradient (both $\mathcal{O}(n \cdot d)$) in each iteration. Since a classifier is constructed for each class, the total cost is $\mathcal{O}(m \cdot Opt)$. The total training cost of IBLR+ is $\mathcal{O}(n^2 \cdot d + n \cdot k \cdot m + m \cdot Opt)$.

Recall that FRONEC has a training cost of $\mathcal{O}(n^2 \cdot m)$. This cost is quadratic in n, as is the cost of MLKNN and MLDGC. The training cost of IBLR+ is at least quadratic in n. BRKNN-b and LPKNN have negligible training phases. These derivations are reflected in the runtime values listed above and in Section 5.3.2.

Classification. To classify an instance, BRKNN-b, LPKNN, MLKNN and MLDGC first locate its nearest neighbors in the training set, which can be achieved at $\mathcal{O}(n \cdot d)$ cost. In each of these methods, the class prediction procedure following the neighborhood calculation costs $\mathcal{O}(k \cdot m)$. Their total classification is therefore $\mathcal{O}(n \cdot d + k \cdot m)$. To classify a test instance, the IBLR+ method determines its nearest neighbors ($\mathcal{O}(n \cdot d)$) and derives the class confidence scores ($\mathcal{O}(m \cdot k)$). For each class, the corresponding logistic classifier is called to classify the instance ($\mathcal{O}(d)$). The total classification cost of IBLR+ is therefore $\mathcal{O}(n \cdot d + m \cdot (k + d))$. FRONEC has a classification cost of $\mathcal{O}(n \cdot d \cdot m \cdot k \log(k))$. The runtime comparisons show that FRONEC has the highest classification time, although it is close to that of the other methods.

5.3. Comparison on real-world datasets

In Section 5.2, we experimentally compared the performance of the BRKNN-b, LPKNN, MLKNN, IBLR+ and MLDGC methods to our FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ proposals. We now proceed with a comparison of these algorithms on the six real-world multi-label datasets described in Table 1. As before, we use them in a 5-fold cross validation setup. In Section 5.3.1, we list and discuss the complete prediction results. Section 5.3.2 compares the classifiers in terms of their execution times. We report average values of the timing experiments, but the full results, based on ten runs of the methods, can be obtained from our web page. We refer back to Section 5.2.3 for a comparison of the theoretical complexity of the classifiers.

5.3.1. Prediction performance

Tables 7 and 8 present the classification results. For each dataset, the best result is printed in bold and the worst result is underlined. The most remarkable observation is the poor performance of the IBLR+ method on these datasets, which is in contrast with its good results on the synthetic datasets in Section 5.2. Moreover, as we noted and explained in Section 5.2.1, the MLDGC method does not perform as well in our study as it did in its original proposal. It is outperformed by FRONEC for all evaluation measures.

As we observed in the analysis of the synthetic datasets, our FRONEC method makes the most accurate predictions based on the most stringent evaluation measure, the subset accuracy. FRONEC also retains the best balance between the recall and precision measures, as reflected in its value for the F-measure. On the real-world datasets, LPKNN achieves a good trade-off between precision and recall as well, such that a competitive F-measure value is obtained. Looking back at Table 4, the precision and recall values of LPKNN on the synthetic datasets were close together as well, although they were both lower than the results of FRONEC-$R_d^{(2)}$.

FRONEC-$R_d^{(2)}$ yields the best average result for the subset accuracy and F-measure. The highest average recall is obtained by BRKNN-b, which is mainly due to its outlying strong performance on the Birds dataset for this measure. However, for each other measure, BRKNN-b yields the worst result on this dataset, such that we can safely ignore it as a strong competitor. Its high recall is due to the high number of label predictions it makes for the elements in this dataset. The average difference between the true and predicted label cardinality on Birds is -6.5126, meaning that BRKNN-b predicts more than six labels

Table 7
Experimental results of the seven multi-label classifiers on the datasets in Table 1 for the Hamming loss and subset accuracy measures.

Hamming loss							
Dataset	BRKNN-b	LPKNN	MLKNN	IBLR+	MLDGC	FRONEC-$R_d^{(1)}$	FRONEC-$R_d^{(2)}$
Birds	0.3820	0.0450	**0.0410**	0.1030	0.0420	0.0422	0.0446
Emotions	**0.1902**	0.2094	0.1932	0.2136	0.1946	0.1964	0.1978
Flags	0.2860	0.2674	0.2670	0.2696	0.2876	**0.2664**	**0.2664**
Music	**0.1840**	0.2024	**0.1840**	0.2154	0.1954	0.1946	0.1962
Scene	0.0872	0.0876	**0.0814**	0.1264	0.0956	0.0824	0.0816
Yeast	0.2248	0.2100	**0.1928**	0.2024	0.2314	0.2030	0.2062
Mean	0.2257	0.1703	**0.1599**	0.1884	0.1744	0.1642	0.1655
Subset accuracy							
Dataset	BRKNN-b	LPKNN	MLKNN	IBLR+	MLDGC	FRONEC-$R_d^{(1)}$	FRONEC-$R_d^{(2)}$
Birds	0.1410	0.5036	**0.5454**	0.3700	0.5224	0.5284	0.5240
Emotions	0.3204	0.3372	0.3018	0.2560	0.3186	0.3454	**0.3574**
Flags	0.1652	0.2120	0.1494	0.1914	0.1712	0.1992	**0.2158**
Music	0.3428	0.3444	0.3128	0.2530	0.3158	0.3360	**0.3480**
Scene	0.6998	0.7072	0.6598	0.4870	0.6604	0.7256	**0.7272**
Yeast	0.2154	0.2592	0.1966	0.1826	0.2254	**0.2704**	0.2656
Mean	0.3141	0.3939	0.3610	0.2900	0.3690	0.4008	**0.4063**

Table 8
Experimental results of the seven multi-label classifiers on the datasets in Table 1 for the *F*-measure, precision and recall measures.

***F*-measure**							
Dataset	BRKNN-b	LPKNN	MLKNN	IBLR+	MLDGC	FRONEC-$R_d^{(1)}$	FRONEC-$R_d^{(2)}$
Birds	0.1512	**0.5506**	0.5024	0.3212	0.5322	0.4822	0.4960
Emotions	**0.6918**	0.6674	0.6698	0.6496	0.6782	0.6864	0.6882
Flags	0.7022	0.7170	0.7212	**0.7240**	0.6952	0.7224	0.7200
Music	**0.7028**	0.6826	0.6876	0.6466	0.6748	0.6916	0.6922
Scene	0.7536	0.7472	0.7566	0.6580	0.7196	0.7628	**0.7646**
Yeast	0.6240	0.6446	0.6510	0.6462	0.6194	**0.6562**	0.6520
Mean	0.6043	0.6682	0.6648	0.6076	0.6532	0.6669	**0.6688**
Precision							
Dataset	BRKNN-b	LPKNN	MLKNN	IBLR+	MLDGC	FRONEC-$R_d^{(1)}$	FRONEC-$R_d^{(2)}$
Birds	0.0858	0.5886	**0.7234**	0.2472	0.6646	0.6976	0.6222
Emotions	0.6972	0.6596	**0.7162**	0.6642	0.6986	0.6840	0.6764
Flags	0.7090	**0.7368**	0.7280	0.7168	0.7154	0.7312	0.7340
Music	0.7080	0.6680	**0.7300**	0.6618	0.7020	0.6836	0.6774
Scene	0.7622	0.7720	**0.8152**	0.6386	0.7576	0.7884	0.7902
Yeast	0.6344	0.6604	**0.7192**	0.6866	0.6176	0.6736	0.6666
Mean	0.5994	0.6809	**0.7387**	0.6025	0.6926	0.7097	0.6945
Recall							
Dataset	BRKNN-b	LPKNN	MLKNN	IBLR+	MLDGC	FRONEC-$R_d^{(1)}$	FRONEC-$R_d^{(2)}$
Birds	**0.6330**	0.5204	0.3858	0.4590	0.4516	0.3716	0.4144
Emotions	0.6874	0.6754	0.6288	0.6358	0.6594	0.6896	**0.7002**
Flags	0.6992	0.6994	0.7162	**0.7316**	0.6766	0.7142	0.7068
Music	0.6986	0.6986	0.6510	0.6324	0.6496	0.7002	**0.7076**
Scene	**0.7452**	0.7244	0.7062	0.6794	0.6854	0.7392	0.7406
Yeast	0.6150	0.6296	0.5948	0.6102	0.6214	**0.6398**	0.6380
Mean	**0.6797**	0.6580	0.6138	0.6247	0.6240	0.6424	0.6513

too many on average, which is about a third of the number of possible labels. The MLKNN method wins for the remaining two evaluation measures, the Hamming loss and precision, but has the lowest average result for the recall.

Linking this discussion back to the details provided in Section 5.2.2, the average difference between the cardinality of the true and predicted labelsets is −1.0465 (BRKNN-b), 0.0706 (LPKNN), 0.3045 (MLKNN), −0.0621 (IBLR+), 0.1364 (MLDGC), 0.1267 (FRONEC-$R_d^{(1)}$) and 0.0941 (FRONEC-$R_d^{(2)}$). As before, the high value of MLKNN explains its superior precision value. Among the labels correctly predicted by MLKNN, FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ also derive 93.35% and 93.17% respectively, which show that the correct predictions of MLKNN are almost always made by our method as well.

5.3.2. Timing comparison

To complement the prediction performance analysis, we provide an indication of the execution times of these methods, both in the training and testing phases. The former includes two components: (i) the time spent to select an appropriate

parameter setting and (ii) the remaining work required during training. Our FRONEC method needs to set both a value for k and an OWA weighting scheme, while the other methods only need to decide their k value. This is reflected in the relatively high parameter selection time of FRONEC-$R_d^{(1)}$ (13.1974 s) and FRONEC-$R_d^{(2)}$ (13.1185 s) compared to the values of BRKNN-b (0.8369 s), LPKNN (1.0060 s), MLKNN (0.8025 s) and MLDGC (0.8503 s). However, the time spent by IBLR+ to select a good k value is notably higher at 68.4973 s on average. The high computational expense of IBLR+ has been commented on in [34] as well.

Once the parameters have been set, a method may perform some additional operations during its training phase. As discussed in Section 3.2.3, FRONEC precomputes the label similarity between each pair of training instances. This is achieved in 0.0210 s and 0.0338 s by FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ respectively. The MLKNN and MLDGC methods also perform some additional calculations at an average cost of 0.7598 s and 0.7933 s. IBLR+ needs to construct a binary classifier for each class, which comes at the relatively high cost of 9.3378 s. The BRKNN-b and LPKNN do little else at training time apart from selecting their k value, such that their additional training time is negligible at 0.0023 s and 0.0010 s respectively.

The average total testing times of the BRKNN-b, LPKNN, MLKNN and MLDGC methods on these six datasets are close together, namely 0.3450 s, 0.3462 s, 0.3359 s and 0.3343 s respectively. This is the average time to predict the outcomes of the full test sets. The values of IBLR+ and our FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ methods are slightly higher at 0.4735 s, 1.0482 s and 1.0487 s respectively. Nevertheless, these values are still low and, in all, we can conclude that each of the seven included methods has a fast classification time.

In summary, we find that our proposal is also competitive with the state-of-the-art in terms of its execution time. It takes a slightly longer time to set its internal parameters compared to BRKNN-b, LPKNN, MLKNN and MLDGC, but this is due to the fact that an additional parameter (the OWA weighting scheme) needs to be chosen. If necessary, the total training time of FRONEC can be reduced by fixing the OWA weighting scheme or limiting the number of possibilities. Notwithstanding, compared to IBLR+, the training time of FRONEC remains very moderate. With respect to the testing time, all methods are able to derive their predictions acceptably fast.

6. Conclusion

In a multi-label classification dataset, each observation is associated with one or more classes. The challenge is to predict all classes at once, between which correlations may exist. Among the multi-label classifiers proposed in the literature, we have reviewed the family of methods based on the nearest neighbor classification principle. Generally put, the predicted labelset is derived based on the information contained in the k nearest neighbors of the instance to classify.

We have proposed a new multi-label nearest neighbor classifier based on fuzzy rough set theory, called FRONEC. It is related to the existing methods BRKNN-b and LPKNN and shares their intuitive nature. To classify an instance, FRONEC locates its k nearest neighbors in the training set. Instead of aggregating the classes of these neighbors to derive the prediction like BRKNN-b or LPKNN do, our method uses a fuzzy rough quality measure to locate the training instances whose labelsets represent a consensus of the labelsets of the neighbors. This consensus forms the predicted labelset for the test element.

Our instance quality measure depends on a fuzzy relation that measures the similarity in the labelsets of observations. We have proposed and evaluated two candidate label similarity relations. We have explained why the preference for a particular relation depends on the evaluation metric and included both alternatives in the comparison with other nearest neighbor based methods.

We have carefully compared our proposal to five popular nearest neighbor based multi-label classifiers, using five evaluation measures. Our comparison includes the MLKNN and IBLR+ methods, which are commonly used as state-of-the-art methods in comparative studies, and the recent proposal MLDGC. As advised in [34], we limited the comparison to the family of nearest neighbor based multi-label classifiers, which are competitive to each other with respect to their understandable character. More complex multi-label classifiers have been proposed in the literature, but a comparison with them is outside the scope of this paper. Our aim has been to provide the community with a new strong representative of the nearest neighbor classifier family. We have succeeded in this objective and have shown that our proposal could even replace MLKNN and IBLR+ in future studies. In the first part of our experimental study, conducted on 30 synthetic multi-label datasets, our method outperformed its competitors in terms of the F-measure and subset accuracy. With respect to the F-measure, we observed that we achieve an appropriate balance between the recall and precision, even outperforming the other methods in terms of the former. The MLKNN and IBLR+ methods obtained a high precision at the cost of a low recall. Secondly, we repeated the analysis on six real-world multi-label datasets, on which IBLR+ did not perform well at all. Our FRONEC method and MLKNN provided good results, although the latter did yield the lowest average recall among the compared algorithms. Taking both parts of the experimental evaluation into account, we can conclude that we have proposed a competitive classifier within the nearest neighbor family, that is easy to understand and implement and has a strong classification performance.

One possible line of future research is the combination of our FRONEC method with multi-instance classifiers for application in the area of multi-instance multi-label classification [62]. We can opt to use the method in full or only use its core characteristics, like the label similarity relation or the consensus derivation. Of particular interest would be the interaction of FRONEC with our earlier fuzzy and fuzzy rough multi-instance classifiers [41,42]. These methods classify multi-instance observations by computing their membership degree to the possible classes using notions from fuzzy or fuzzy rough set theory. If more than one class can be assigned to the same observation, the consensus approach used in FRONEC can be

integrated in the existing classifiers to accommodate for this possibility. From a different viewpoint, we can also generalize the full FRONEC method to handle multi-instance data. The crucial modification lies with the feature similarity relation, for which alternatives evaluated in [41] can be used.

A second interesting aspect for future study is related to the label similarity relation. In this paper, we have discussed (and experimentally demonstrated) the strong influence of the choice of the label similarity relation on the results of FRONEC. A more in-depth study of alternative relations may lead to additional insights into the characteristics of the method and can result in further performance enhancements.

Acknowledgments

The research of Sarah Vluymans is funded by the Special Research Fund (BOF) of Ghent University (Grant no. BOF.DOC.2014.0074). Yvan Saeys is an ISAC Marylou Ingram Scholar.

References

[1] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is nearest neighbor meaningful? in: Proceedings of the International Conference on Database Theory, Springer, 1999, pp. 217–235.
[2] P. Bhowmick, A. Basu, P. Mitra, A. Prasad, Sentence level news emotion analysis in fuzzy multi-label classification framework, Special issue: Natural Lang. Process. Appl. (2010) 143.
[3] M. Boutell, J. Luo, X. Shen, C. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771.
[4] K. Brinker, E. Hüllermeier, Case-based multilabel ranking, in: Proceedings of the 20th International joint conference on Artificial Intelligence, 2007, pp. 702–707.
[5] J. Chen, Y. Tang, C. Chen, B. Fang, Y. Lin, Z. Shang, Multi-label learning with fuzzy hypergraph regularization for protein subcellular location prediction, IEEE Trans. Nanobiosci. 13 (4) (2014) 438–447.
[6] W. Cheng, E. Hüllermeier, Combining instance-based learning and logistic regression for multilabel classification, Mach. Learn. 76 (2–3) (2009) 211–225.
[7] W. Cheng, E. Hüllermeier, A simple instance-based approach to multilabel classification using the mallows model, in: Working Notes of the First International Workshop on Learning from Multi-Label Data, 2009, pp. 28–38.
[8] T. Chiang, H. Lo, S. Lin, A ranking-based KNN approach for multi-label classification., ACML 25 (2012) 81–96.
[9] C. Cornelis, R. Jensen, G. Hurtado, D. Ślezak, Attribute selection with fuzzy decision reducts, Inf. Sci. (Ny) 180 (2) (2010) 209–224.
[10] C. Cornelis, N. Verbiest, R. Jensen, Ordered weighted average based fuzzy rough sets, in: Proceedings of the International Conference on Rough Sets and Knowledge Technology, Springer, 2010, pp. 78–85.
[11] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[12] D. Dubois, H. Prade, Rough fuzzy sets and fuzzy rough sets, Int. J. General Syst. 17 (2–3) (1990) 191–209.
[13] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Am. Stat. Assoc. 32 (200) (1937) 675–701.
[14] J. Fürnkranz, E. Hüllermeier, E. Mencía, K. Brinker, Multilabel classification via calibrated label ranking, Mach. Learn. 73 (2) (2008) 133–153.
[15] X. Geng, Y. Tang, Y. Zhu, G. Cheng, An improved multi-label classification algorithm BRkNN, J. Inf. Comput. Sci. 11 (16) (2014) 5927–5936.
[16] E. Gibaja, S. Ventura, Multi-label learning: a review of the state of the art and ongoing research, Wiley Interdiscip. Rev. 4 (6) (2014) 411–444.
[17] S. Godbole, S. Sarawagi, Discriminative methods for multi-labeled classification, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2004, pp. 22–30.
[18] F. Herrera, F. Charte, A. Rivera, M. Del Jesus, Multilabel Classification: Problem Analysis, Metrics and Techniques, Springer, 2016.
[19] S. Holm, A simple sequentially rejective multiple test procedure, Scand. J. Stat. (1979) 65–70.
[20] Q. Hu, D. Yu, Z. Xie, Neighborhood classifiers, Expert Syst. Appl. 34 (2) (2008) 866–876.
[21] Q. Hu, L. Zhang, S. An, D. Zhang, D. Yu, On robust fuzzy rough set models, IEEE Trans. Fuzzy Syst. 20 (4) (2012) 636–651.
[22] K. Huang, Z. Li, A multilabel text classification algorithm for labeling risk factors in SEC form 10-K, ACM Trans. Manage. Inf. Syst. (TMIS) 2 (3) (2011) 18.
[23] J. Jiang, S. Tsai, S. Lee, FSKNN: multi-label text categorization based on fuzzy similarity and k nearest neighbors, Expert Syst. Appl. 39 (3) (2012) 2813–2821.
[24] X. Jing, F. Wu, Z. Li, R. Hu, D. Zhang, Multi-label dictionary learning for image annotation, IEEE Trans. Image Process. 25 (6) (2016) 2712–2725.
[25] F. Kang, R. Jin, R. Sukthankar, Correlated label propagation with application to multi-label learning, in: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2, 2006, pp. 1719–1726.
[26] J. Keller, M. Gray, J. Givens, A fuzzy k-nearest neighbor algorithm, IEEE Trans. Syst. Man Cybern. SMC-15 (4) (1985) 580–585.
[27] X. Lin, X. Chen, Mr. KNN: soft relevance for multi-label classification, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ACM, 2010, pp. 349–358.
[28] H. Liu, X. Wu, S. Zhang, Neighbor selection for multilabel classification, Neurocomputing 182 (2016) 187–196.
[29] S. Liu, J. Chen, A multi-label classification based approach for sentiment classification, Expert Syst. Appl. 42 (3) (2015) 1083–1093.
[30] C. Mallows, Non-null ranking models. i, Biometrika 44 (1/2) (1957) 114–130.
[31] R. Nanculef, I. Flaounas, N. Cristianini, Efficient classification of multi-labeled text streams by clashing, Expert Syst. Appl. 41 (11) (2014) 5431–5450.
[32] Z. Pawlak, Rough sets, Int. J. Parallel Program 11 (5) (1982) 341–356.
[33] A. Radzikowska, E. Kerre, A comparative study of fuzzy rough sets, Fuzzy Sets Syst. 126 (2) (2002) 137–155.
[34] O. Reyes, C. Morell, S. Ventura, Effective lazy learning algorithm based on a data gravitation model for multi-label learning, Inf. Sci. (Ny) 340 (2016) 159–174.
[35] E. Spyromitros, G. Tsoumakas, I. Vlahavas, An empirical study of lazy multilabel classification algorithms, in: Proceedings of the Hellenic conference on Artificial Intelligence, Springer, 2008, pp. 401–406.
[36] J. Tomás, N. Spolaôr, E. Cherman, M. Monard, A framework to generate synthetic multi-label datasets, Electron Notes Theor. Comput. Sci. 302 (2014) 155–176.
[37] I. Triguero, C. Vens, Labelling strategies for hierarchical multi-label classification techniques, Pattern Recognit. 56 (2016) 170–183.
[38] N. Verbiest, Fuzzy Rough and Evolutionary Approaches to Instance Selection, Ph.D. thesis, Ghent University, 2014.
[39] N. Verbiest, C. Cornelis, R. Jensen, Fuzzy rough positive region based nearest neighbour classification, in: Proceedings of the 2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2012, pp. 1–7.
[40] S. Vluymans, L. D'eer, Y. Saeys, C. Cornelis, Applications of fuzzy rough set theory in machine learning: a survey, Fundam. Inform. 142 (1–4) (2015) 53–86.
[41] S. Vluymans, D. Sánchez Tarragó, Y. Saeys, C. Cornelis, F. Herrera, Fuzzy multi-instance classifiers, IEEE Trans. Fuzzy Syst. 24 (6) (2016) 1395–1409.
[42] S. Vluymans, D. Sánchez Tarragó, Y. Saeys, C. Cornelis, F. Herrera, Fuzzy rough classifiers for class imbalanced multi-instance data, Pattern Recognit 53 (2016) 36–45.

[43] H. Wang, C. Ding, H. Huang, Multi-label classification: Inconsistency and class balanced k-nearest neighbor, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, 2010, pp. 1264–1266.
[44] X. Wang, W. Zhang, Q. Zhang, G. Li, MultiP-SChlo: multi-label protein subchloroplast localization prediction with chouâ;;s pseudo amino acid composition and a novel multi-label classifier, Bioinformatics 31 (16) (2015) 2639–2645.
[45] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 1 (6) (1945) 80–83.
[46] X. Wu, V. Kumar, J. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. McLachlan, A. Ng, B. Liu, S. Philip, Z. Zhou, M. Steinbach, D. Hand, D. Steinberg, Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[47] X. Xia, X. Yang, S. Li, C. Wu, L. Zhou, RW.KNN: a proposed random walk knn algorithm for multi-label classification, in: Proceedings of the 4th Workshop on Workshop for Ph.D. Students in Information & Knowledge Management, ACM, 2011, pp. 87–90.
[48] J. Xu, An empirical comparison of weighting functions for multi-label distance-weighted k-nearest neighbour method, in: Proceedings of the First International Conference on Artificial Intelligence, Soft Computing and Applications (AIAA2011), 2011, pp. 13–20.
[49] J. Xu, V. Jagadeesh, B. Manjunath, Multi-label learning with fused multimodal bi-relational graph, IEEE Trans. Multimedia 16 (2) (2014) 403–412.
[50] R. Yager, On ordered weighted averaging aggregation operators in multicriteria decisionmaking, IEEE Trans. Syst. Man Cybern. 18 (1) (1988) 183–190.
[51] Y. Yao, Relational interpretations of neighborhood operators and rough set approximation operators, Inf. Sci. (Ny) 111 (1–4) (1998) 239–259.
[52] Z. Younes, F. Abdallah, T. Denœux, Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies, in: Proceedings of the 2008 16th European Signal Processing Conference, IEEE, 2008, pp. 1–5.
[53] Z. Younes, F. Abdallah, T. Denœux, An evidence-theoretic k-nearest neighbor rule for multi-label classification, in: Proceedings of the International Conference on Scalable Uncertainty Management, Springer, 2009, pp. 297–308.
[54] Z. Younes, F. Abdallah, T. Denœux, Evidential multi-label classification approach to learning from data with imprecise labels, in: Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 2010, pp. 119–128.
[55] Z. Younes, F. Abdallah, T. Denœux, Fuzzy multi-label learning under veristic variables, in: Proceedings of the 2010 IEEE International Conference on Fuzzy Systems (FUZZ), IEEE, 2010, pp. 1–8.
[56] Z. Younes, F. Abdallah, T. Denœux, H. Snoussi, A dependent multilabel classification method derived from the k-nearest neighbor rule, EURASIP J. Adv. Signal Process. (2011).
[57] Y. Yu, W. Pedrycz, D. Miao, Multi-label classification by exploiting label correlations, Expert Syst. Appl. 41 (6) (2014) 2989–3004.
[58] L. Zadeh, Fuzzy sets, Inf. Control 8 (3) (1965) 338–353.
[59] M. Zhang, Z. Zhou, ML-KNN: a lazy learning approach to multi-label learning, Pattern Recognit. 40 (7) (2007) 2038–2048.
[60] M. Zhang, Z. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (8) (2014) 1819–1837.
[61] S. Zhang, Shell-neighbor method and its application in missing data imputation, Appl. Intell. 35 (1) (2011) 123–133.
[62] Z. Zhou, M. Zhang, S. Huang, Y. Li, Multi-instance multi-label learning, Artif. Intell. 176 (1) (2012) 2291–2320.