Scalable approximate FRNN-OWA classification

Oliver Urs Lenz, Daniel Peralta, and Chris Cornelis

***Abstract*—Fuzzy Rough Nearest Neighbour classification with Ordered Weighted Averaging operators (FRNN-OWA) is an algorithm that classifies unseen instances according to their membership in the fuzzy upper and lower approximations of the decision classes. Previous research has shown that the use of OWA operators increases the robustness of this model. However, calculating membership in an approximation requires a nearest neighbour search. In practice, the query time complexity of exact nearest neighbour search algorithms in more than a handful of dimensions is near-linear, which limits the scalability of FRNN-OWA. Therefore, we propose approximate FRNN-OWA, a modified model that calculates upper and lower approximations of decision classes using the approximate nearest neighbours returned by Hierarchical Navigable Small Worlds (HNSW), a recent approximative nearest neighbour search algorithm with logarithmic query time complexity at constant near-100% accuracy. We demonstrate that approximate FRNN-OWA is sufficiently robust to match the classification accuracy of exact FRNN-OWA while scaling much more efficiently. We test four parameter configurations of HNSW, and evaluate their performance by measuring classification accuracy and construction and query times for samples of various sizes from three large datasets. We find that with two of the parameter configurations, approximate FRNN-OWA achieves near-identical accuracy to exact FRNN-OWA for most sample sizes within query times that are up to several orders of magnitude faster.**

***Index Terms*—Big data applications, classification algorithms, fuzzy rough sets, nearest neighbor searches, scalability.**

I. Introduction

OVER the course of the last few decades, an ever increasing amount of data has become available for machine learning. This compels us to scrutinise the scalability of existing algorithms, and where necessary to develop variants or alternatives that scale better. For machine learning problems where run time is a greater impediment to performance than the amount of available data, approximative algorithms with a slightly lower accuracy but a higher capacity may offer a worthwhile trade-off. If the computational complexity of an algorithm is superlinear, computation time may blow up as dataset size grows. But even if it is strictly linear, processing orders of magnitude more data requires orders of magnitude more computing time. Ideally, then, we would like to have access to algorithms with logarithmic computational complexity.

Fuzzy rough sets [1] generalise rough sets [2], [3] to information systems with a fuzzy indiscernibility relation. Fuzzy Rough Nearest Neighbour (FRNN) classification [4] uses this fuzzy indiscernibility relation to predict the class membership of unseen instances. FRNN has been made more flexible and robust through the addition of Ordered Weighted Averaging (OWA) operators [5], [6]. The resulting combination (FRNN-OWA) has been shown to be particularly effective for various types of imbalanced classification [6]–[8].

Computing FRNN-OWA classification requires the identification, in each decision class, of those training instances that are least discernible from a given test instance. This task is equivalent to a nearest neighbour search, and consequently, a straightforward implementation without preprocessing has a time complexity that is linear with respect to the number of training instances. This restricts the applicability of FRNN-OWA to large datasets.

In previous work, we have presented an implementation of FRNN-OWA in Apache Spark that reduces its run time through parallellisation accross a cluster of computers [9]. While this provides a method to perform classification with large datasets, it cannot speed up FRNN-OWA by a factor larger than the number of processor cores available, and it would be more effective if we could reduce the time complexity of FRNN-OWA itself.

There is a rich literature of nearest neighbour search algorithms [10]–[14]. One of the most popular exact approaches uses a so-called KD-tree [15]. It has a theoretical query time complexity that is logarithmic, but in practice this is hard to achieve, and its query time complexity for real datasets with more than a handful of attributes is much closer to linear [16]. A recent approximate nearest neighbour proposal, Hierarchical Navigable Small World (HNSW) [17], promises to achieve actual logarithmic query time complexity at a very low constant error rate.

In this article, we propose approximate FRNN-OWA, which we obtain by incorporating HNSW nearest neighbour identification into FRNN-OWA. We hypothesise that this is a particularly fortuitous combination because the use of OWA operators has been shown to make FRNN-OWA robust against noise [18], and this should translate into a certain amount of tolerance for the nearest neighbour misidentifications introduced by HNSW.

To test the performance of approximate FRNN-OWA, we explore four different parameter settings of HNSW, and show that it is possible to achieve logarithmic query time complexity while sacrificing a minimal amount of accuracy with respect to exact FRNN-OWA. This means that approximate FRNN-OWA can be applied to very large datasets, and we demonstrate this by performing cross-validation on three of the largest datasets of the UCI Machine Learning Repository [19].

We start by providing the necessary background information on FRNN-OWA (Section II) and HNSW (Section III). We then present approximate FRNN-OWA (Section IV), describe the experiments by which we evaluate its performance (Section V) and present and discuss the results (Section VI). Finally,

O. U. Lenz, D. Peralta and C. Cornelis are with the Department of Applied Mathematics, Computer Science and Statistics, Ghent University, e-mail: {oliver.lenz,chris.cornelis}@ugent.be.

D. Peralta is with the Data Mining and Modelling for Biomedicine group, VIB Center for Inflammation Research, Ghent University, e-mail: daniel.peralta@irc.vib-ugent.be

we summarise the article and discuss possible future research (Section VII).

II. Fuzzy Rough Nearest Neighbour classification with OWA weights

Rough sets [2], [3] approximate sets of instances by way of some attributes defined over those instances. A formal definition is given in Definition 1.

Definition 1 (Rough set)**.** Let X be a finite set of instances and A a finite set of attributes $a : X \longrightarrow V_a$. Then (X, A) is called an *information system*. The *indiscernibility* relation R in (X, A) is the equivalence relation $\{(x, y) \in X \times X | \forall a \in A : a(x) = a(y))\}$. The upper and lower approximations $\overline{C}$ and $\underline{C}$ of a subset $C \subseteq X$ are the subsets defined by:

$$\begin{aligned} \overline{C}(y) &= \max_{x \in X}(R(y, x) \wedge C(x)) \\ \underline{C}(y) &= \min_{x \in X}(R(y, x) \implies C(x)) \end{aligned} \tag{1}$$

A *rough set* in (X, A) is a pair $(\overline{C}, \underline{C})$ for some $C \subseteq X$.

The equivalence classes of R consist of all instances that have the same values for all attributes. The upper approximation of C contains all instances equivalent under R to some instance of C, while the lower approximation of C contains only those instances whose entire equivalent class is contained in C. They are the closure and interior of the quasi-discrete topology generated by R.

Fuzzy rough sets [1] represent a fuzzification of rough set theory. In order to be able to extend the definitions of lower and upper approximations to fuzzy sets, we have to replace the crisp indiscernibility relation R with a fuzzy tolerance relation, and the conjunction and implication with a t-norm $T : [0, 1] \times [0, 1] \longrightarrow [0, 1]$ and a fuzzy implication $I : [0, 1] \times [0, 1] \longrightarrow [0, 1]$ (Definition 2).

Definition 2 (Fuzzy rough set)**.** Let (X, A) be an information system. A *fuzzy indiscernibility* relation in (X, A) is a fuzzy tolerance relation $R : X \times X \longrightarrow [0, 1]$ such that $(\forall a \in A : a(x) = a(y)) \implies R(x, y) = 1$. For a choice of such a fuzzy indiscernibility relation R, a t-norm T and a fuzzy implication I, the upper and lower approximations $\overline{C}$ and $\underline{C}$ of a fuzzy subset C of X are defined by:

$$\begin{aligned} \overline{C}(y) &= \max_{x \in X}(T(R(y, x), C(x)) \\ \underline{C}(y) &= \min_{x \in X}(I(R(y, x), C(x)) \end{aligned} \tag{2}$$

A *fuzzy rough set* in (X, A) is a pair $(\overline{C}, \underline{C})$ for some fuzzy subset C of X.

For any $z \in [0, 1]$, since T is a t-norm, $T(z, 0) = 0$ and $T(z, 1) = z$, and since I is an implication, $I(z, 1) = 1$ and $I(z, 0) = N_I(z)$, where N_I is the negation induced by I. Therefore, if C is crisp, (2) reduces to (3).

$$\begin{aligned} \overline{C}(y) &= \max_{x \in C}(R(y, x)) \\ \underline{C}(y) &= \min_{x \in X \setminus C}(N_I(R(y, x))) \end{aligned} \tag{3}$$

In correspondence with previous work [4], [6], [20], we will assume in the rest of this article that N is the standard negation $z \longmapsto 1 - z$.

Fuzzy Rough Nearest Neighbour (FRNN) classification [4] uses the membership of a test instance y in the upper and lower approximation of a crisp decision class C as a measure of the extent that y possibly and necessarily belongs to C, and predicts that y belongs to the class for which these measures are highest.

In practice, FRNN classifies a test element y as the class of the single element of X least discernible from y. This means that FRNN makes the same predictions as traditional $1\,\mathrm{NN}$ classification (with a dissimilarity corresponding to R), and that it is similarly sensitive to noise. In the case of $k\,\mathrm{NN}$ this can be remedied by choosing $k \geq 3$. For FRNN, the solution has been to soften $\max$ and $\min$ by replacing them with Ordered Weighted Averaging (OWA) operators ([21], Definition 3) that also depend on the the next-largest indiscernibility values [5], [6]. The resulting model, FRNN-OWA has a greater tolerance for noise than strict FRNN and generally produces better results than a number of alternative proposals [18].

Definition 3 (OWA operator)**.** Let V be an m-dimensional vector space, and w an m-dimensional weight vector with values in $[0, 1]$ that sum to 1. The *OWA operator* F_w corresponding to w acts on any vector $v \in V$ by sorting its coefficients in descending order and taking the inner product with w. We say that two weight vectors are *dual* if they have inversely ordered matching coefficients.

The upper and lower approximations in FRNN-OWA are defined by (4), for some choice of weight vectors $\overline{w}$ and $\underline{w}$. Note that in order to apply the corresponding OWA operator, we create a vector in $\mathbb{R}^m$ through an implicit but arbitrary ordering of values.

$$\begin{aligned} \overline{C}(y) &= F_{\overline{w}}(\langle R(y, x) | x \in C \rangle) \\ \underline{C}(y) &= F_{\underline{w}}(\langle 1 - R(y, x) | x \in X \setminus C \rangle) \end{aligned} \tag{4}$$

As we noted in previous work [9], in practice it is desirable to work with weight vectors $\overline{w}$ and $\underline{w}$ of length $k \ll m$. Formally, these can be interpreted as full weight vectors where the last or first $m - k$ values are equal to 0. The first benefit is presentational, since this restricts the range of possible OWA operators to precisely those we are interested in: OWA operators that approximate $\max$ and $\min$ by emphasising the largest and smallest elements, respectively. And secondly, it fundamentally reduces the computational complexity of FRNN-OWA by replacing the sorting of the decision classes with nearest neighbour searches.

Some possible dual choices for $\overline{w}$ and $\underline{w}$ are listed in Table I. Strict weights represent the trivial choice, for which we recover $F_{\overline{w}} = \max$ and $F_{\underline{w}} = \min$. Trimmed and mean weights correspond, respectively, to the k-trimmed and k-mean maxima and minima of [22]. Additive and exponential weights are the two weight types presented in [6].

There has been a limited number of attempts to apply fuzzy rough sets in the context of Big Data. For a current overview,

TABLE I
EXAMPLES OF DUAL UPPER ($\overline{w}$) AND LOWER ($\underline{w}$) WEIGHT VECTORS OF LENGTH k USED WITH FRNN-OWA

Name	$\overline{w}$	$\underline{w}$
Strict	$\langle 1, 0, \ldots, 0 \rangle$	$\langle 0, \ldots, 0, 1 \rangle$
Trimmed	$\langle 0, \ldots, 0, 1 \rangle$	$\langle 1, 0, \ldots, 0 \rangle$
Mean	$\left\langle \frac{1}{k}, \ldots, \frac{1}{k} \right\rangle$	$\left\langle \frac{1}{k}, \ldots, \frac{1}{k} \right\rangle$
Additive	$\left\langle \frac{2(k+1-i)}{k(k+1)} \right\rangle_{1 \leq i \leq k}$	$\left\langle \frac{2i}{k(k+1)} \right\rangle_{1 \leq i \leq k}$
Exponential	$\left\langle \frac{2^{k-i}}{2^k - 1} \right\rangle_{1 \leq i \leq k}$	$\left\langle \frac{2^{i-1}}{2^k - 1} \right\rangle_{1 \leq i \leq k}$

TABLE II
EXISTING WORK USING FUZZY ROUGH SETS IN A BIG DATA CONTEXT — USE WITH PROTOTYPE SELECTION (PS), FEATURE SELECTION (FS) OR CLASSIFICATION (C) AND LARGEST GENERATED AND REAL DATASET SIZES

Article	Use	Generated	Real
[23] Asfoor et al. 2014	—	10 000 000	—
[24] Vluymans et al. 2015	PS	10 000 000	320 395
[25] Asfoor 2015	PS	10 000 000	320 395
[26] Jensen & Mac Parthaláin 2015	FS	—	832
[27] Qian et al. 2015	FS	—	2310
[28] Zeng et al. 2015	FS	—	2800
[29] Zeng et al. 2017	FS	—	2800
[30] Hu et al. 2018	FS	—	4 898 431
[9] Lenz et al. 2019	C	16 777 216	11 000 000

see [9] (summarised in Table II). All previous studies only used fuzzy rough sets for feature or prototype selection, and only one tested their implementations on real datasets with more than a million instances [30]. In [9], we presented an implementation of FRNN-OWA in Apache Spark that could in principle be used to distribute computation time over as many computing nodes as desired. While this is a useful tool, which enabled us to use training sets of more than 10 million instances to classify unseen instances, it requires a considerable amount of computational infrastructure. In particular, performing cross-validation on these large datasets proved to be impractical.

III. APPROXIMATE NEAREST NEIGHBOURS WITH HIERARCHICAL NAVIGABLE SMALL WORLD GRAPHS

Finding the nearest neighbours of an instance in a dataset is a classical computational problem. A brute force approach that compares the distances of a query instance to all training instances scales linearly with training set size, which makes nearest neighbour searches with large training sets impractical. It is possible to achieve better performance by using the distribution of the training set over the attribute space to limit explicit comparison with the query instance to certain training instances. This requires preprocessing the training set to abstract its spatial structure into some data structure. Since this abstraction is independent of any query instances, this introduces a construction stage, and with it, a trade-off between the fixed, one-time construction time and the reduction in the query time per query instance. It also blurs the traditional distinction between lazy and eager learners, since we are no longer comparing query instances directly to the training instances, and this construction stage can be seen as a training stage.

There is a distinction to be made between exact spatial representations of the training set that can faithfully identify the nearest neighbours of a query instance, and approximative representations, which offer a second trade-off between a limited number of incorrect predictions in exchange for even further reduced query times.

A classical example of an exact representation is the binary KD-tree [15], which iteratively divides the training set in two with a series of hyperplanes. The theoretical asymptotic average query time complexity of KD-trees is $\mathcal{O}(\log n)$ [31], but KD-trees suffer from the "curse of dimensionality", in the sense that with datasets with more than a handful of attributes, the time complexity in practice is much closer to linear [16].

One class of approximative approaches uses a search graph, whose nodes correspond to the training instances and the edges encode the spatial structure of the dataset (by connecting certain instances that are not necessarily nearest neighbours in the attribute space). To identify the k nearest neighbours of a query instance y, this graph is traversed iteratively by passing to the node that is nearest to y (in the attribute space) from among the neighbours of the current node. This requires the inspection of all neighbouring nodes of the current node, and a record is kept of the k training instances closest to y that have been inspected.

Both query time and the correctness of the result depend on the edges between the nodes. Everything else being equal, query time is reduced if there are fewer neighbours per node and if the graph can be traversed in fewer steps, whereas more edges generally improve the reachability of the actual k nearest neighbours of y. The Navigable Small Worlds (NSW) model [32] strives to strike a good balance between these tendencies through a mix of short and long distance connections. It adds training instances to the graph in random order, and inserts edges to the M nearest instances already present, for some value of M. As a consequence, connections that are established early generally cover larger distances than connections that are established later. By starting the nearest neighbour search at an instance that was inserted early, we gain immediate access to these long-distance connections and can traverse the dataset with large steps until we reach the general neighbourhood of our query instance. The search stops when the list of nearest neighbours is no longer updated between steps. It is possible to increase the accuracy to any desired level by repeating the search from different training instances. By keeping a record of instances that have been inspected across searches and excluding them from future consideration, unnecessary repetition is avoided.

The query time complexity of NSW is $\mathcal{O}(\log^2 n)$ for a constant accuracy of 0.999. In order to improve query time complexity further, the authors recently introduced a modified

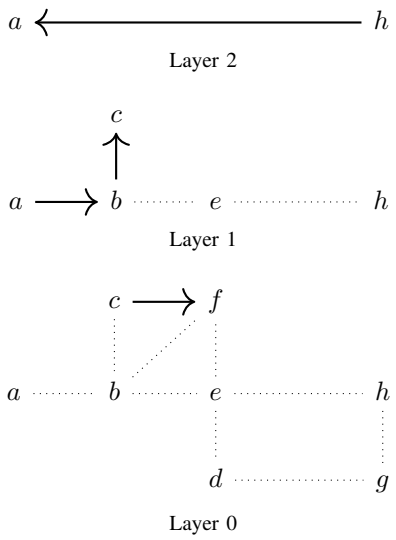


Fig. 1. Schematic depiction of a nearest neighbour search in HNSW. The search begins in the highest layer, and continues in each lower layer from the element that was previously found. The size of the layers follows an exponential distribution.

version of NSW called Hierarchical Navigable Small Worlds (HNSW) [17]. The main conceit of HNSW is that it employs a hierarchy of m_L search layers that can be viewed as a vertical stack. The bottom layer consists of a search graph of the whole training set, and each subsequent layer consists of a search graph of a subset of the subset below it.

To identify the k nearest neighbours of a query instance y, we start in the top layer and run a greedy search to select one nearest neighbour, like in NSW. This process is repeated in the lower layers, in each case using the previously selected neighbour as the new entry point. In the bottom layer, the greedy search is expanded to identify ef_q candidate neighbours. From these, the k instances nearest to y are returned. This process is depicted in Fig. 1.

A very similar algorithm is used to iteratively construct the hierarchy of search layers. Each new training instance x is assigned its highest layer l_x at random, following an exponential distribution. We then traverse the existing hierarchy from top to bottom by running a greedy search in each existing layer. In all layers higher than l_x, the search is restricted to identifying a single neighbour, which we use as the starting point in the next layer. In l_x and all lower layers, we expand the search to greedily identify ef_c candidate neighbours. We insert x as a new node, and connect it to up to M neighbours by repeatedly adding an edge to the closest remaining candidate neighbour that is closer to x than to any of the already selected neighbours. We then use the selected neighbours as multiple starting points in the next layer.

The HNSW model has a query time complexity of $\mathcal{O}(\log n)$ and a construction time complexity of $\mathcal{O}(n \log n)$ if accuracy is kept constant. The principal parameters that can be used to tune performance are summarised in Table III. Higher values for ef_c, M and ef_q increase accuracy in return for longer run times. However, the authors recommend to experimentally identify values for ef_c and M that produce reasonable results and then increase ef_q to attain the desired level of accuracy. In a recent empirical comparison of approximate nearest neighbour search algorithms, HNSW achieved the highest speed at all accuracy levels on all four datasets that were considered [33]. While the scope of this experiment was limited and its results are only indicative, it allows us to select HNSW as representative of the state of the art in approximate nearest neighbour search algorithms.

TABLE III
PARAMETRES OF HNSW

Name[1]	Description	Min	Default[2]
Construction parameters			
ef_c	Number of candidate neighbours to identify	M	200
M	Maximum number of edges to insert between candidate neighbours and a newly added training instance	5	16
Query parameters			
ef_q	Number of candidate neighbours to identify	k	20
k	Number of neighbours to return	1	—

[1] The parameters ef_c and ef_q are called $efConstruction$ and ef respectively in [17], they have been relabelled here for the sake of readability.
[2] These are the default values in the Non-Metric Space Library (NMSLIB) [34], the implementation used in this article.

IV. FRNN-OWA WITH APPROXIMATE NEAREST NEIGHBOUR SELECTION

To adapt FRNN-OWA for use with Big Data, we propose to loosen the strict ordering of training instances required by OWA operators and apply the weights to the approximate nearest neighbours of a test instance as returned by the HNSW model. Since this is the only step of the query phase of FRNN-OWA that is dependent on the training set size, our proposal will result in query times that scale logarithmically.

To facilitate the formal definition of approximate FRNN-OWA classification, we employ a slightly different but essentially equivalent formulation of fuzzy rough set theory. Under the traditional approach represented in Definitions 1 and 2, an information system revolves around the set of instances X. The set of attributes A consists of maps from X to various feature spaces, and the indiscernibility relation R and the upper and lower approximations $\overline{C}$ and $\underline{C}$) are also defined in X. However, this means that the membership of a test instance y in $\overline{C}$ and $\underline{C}$ is only formally defined if $y \in X$. But we want to exclude y from X for the purpose of calculating $\overline{C}$ and $\underline{C}$. Conceptually, y cannot be in X because we don't know whether $y \in C$. This cannot be resolved by simply stipulating $y \notin C$, because that implies $\underline{C}(y) = 0$ (using the strict $\min$ operator).

To solve this, we recentre fuzzy rough set theory around A, which is now a single (potentially multi-dimensional) feature space, and define X, R, $\overline{C}$, $\underline{C}$ and y within A (Definitions 4 and 5). This is arguably more elegant even if classification is not a concern, since it makes it clear that instances with identical attribute values are functionally equivalent in an information system, and it removes the need to stipulate that R respect attribute values. This construction also corresponds

to the familiar perspective of vectors as elements of a vector space.

Definition 4 (Fuzzy information system)**.** A *fuzzy information system* is a triple (A, X, R), consisting of a set A (the *attribute space*), a finite multisubset X of A (the *dataset*) and a fuzzy tolerance relation R in A (the *fuzzy indiscernibility relation*).

For the purpose of this article, we assume that attributes are real-valued, i.e. $A = \mathbb{R}^m$ for some $m \in \mathbb{N}$ and elements $x \in X$ are real vectors $\langle x_1, x_2, \ldots, x_m \rangle$. We follow previous work [6], [18], [20] and use the fuzzy indiscernibility relation corresponding to a scaled version of the Manhattan distance. Denote $r_i(X) = \max_{x \in X}(x_i) - \min_{x \in X}(x_i)$. Then we define R by (5).

$$R(y, x) = 1 - \sum_{i \leq m} \frac{|y_i - x_i|}{m \cdot r_i(X)} \tag{5}$$

Since our present concern is the upper and lower approximations of crisp decision classes, we restrict C to be crisp in Definition 5.

Definition 5 (Fuzzy rough set)**.** Let (A, X, R) be a fuzzy information system, $\overline{w}$ and $\underline{w}$ a choice of weight vectors of some length length k (the *upper* and *lower weight vectors*) and N a (not necessarily deterministic) process that takes an integer k, a multiset C in A, and an element y of A and returns a submultiset of C of size k (the *neighbour selector*). Then for any submultiset C of X, the *upper* and *lower approximations* $\overline{C}$ and $\underline{C}$ are the fuzzy subsets of A defined by:

$$\begin{aligned} \overline{C}(y) &= F_{\overline{w}}(N(k, C, y)) \\ \underline{C}(y) &= F_{\underline{w}}(\{1 - r | r \in N(k, X \setminus C, y\}) \end{aligned} \tag{6}$$

The *fuzzy rough set* generated by C is the pair $(\overline{C}, \underline{C})$.

Definition 6 (Fuzzy classification system)**.** A *fuzzy classification system* is a quadruple $(A, X, R, \mathcal{C})$ such that (A, X, R) is a fuzzy information system and $\mathcal{C}$ is a partition of X into decision classes. A *classifier* in $(A, X, R, \mathcal{C})$ is a partition $A \longrightarrow \mathcal{C}$.

Definition 7 (Approximate FRNN-OWA classification)**.** Let $(A, X, R, \mathcal{C})$ be a classification system, $\overline{w}$ and $\underline{w}$ a choice of upper and lower weight vectors and N a neighbour selector. Then the *upper* and *lower approximation classifiers* are the classifiers defined by $y \longmapsto \arg\max_{C \in \mathcal{C}}(\overline{C}(y))$ and $y \longmapsto \arg\max_{C \in \mathcal{C}}(\underline{C}(y))$ respectively.

As has been pointed out in [6], if there are just two decision classes C_1 and C_2 and $\overline{w}$ and $\underline{w}$ are dual, then $\underline{C_1}(y) = 1 - \overline{C_2}(y)$ (up to consistency of N), and so the upper and lower approximation classifiers are identical. During initial experimentation on datasets with two classes, we found that choosing dual additive weights of length $k = 40$ produces close to optimal results in terms of accuracy across our range of sample sizes, so we fix this choice to obtain a clear comparison, and use the upper approximation classifier.

Definition 7 allows us to equip FRNN-OWA classification with alternative nearest neighbour search algorithms N. In this article we will compare an exact KD-tree search (*exact* FRNN-OWA) with four different parameter configurations of HNSW (Table IV). The goal is to identify a combination of ef_c and M that produces a good baseline in terms of accuracy. For this purpose, we fix $ef_q = k$. Combination A1 represents the default values $ef_c = 200$ and $M = 16$ (cf. Table III). For combinations A2–4, we lower ef_c to $ef_q = 40$ and to its minimum value $M = 16$ and M to its minimum value of 5.

TABLE IV
APPROXIMATE FRNN-OWA, PARAMETER CONFIGURATIONS OF HNSW

Name	ef_c	M
A1	200	16
A2	40	16
A3	16	16
A4	40	5

Approximative models like HNSW may misidentify nearest neighbours, which will lower the general accuracy of approximate FRNN-OWA. Nonetheless, we hypothesise that approximate FRNN-OWA can still rival exact FRNN-OWA in terms of accuracy for two reasons. First, the authors of HNSW have demonstrated that it can operate at close to 100% accuracy. And second, the misidentification of a nearest neighbour of a test instance need not automatically lead to its misclassification. This ought to be true in particular for FRNN-OWA due to its high noise tolerance.

We will measure the accuracy deficits of FRNN-OWA-A1–4 experimentally, but we can already predict that since lower parameter values necessarily induce lower accuracy in HNSW, configurations A1, A2 and A3 will be decreasingly accurate, and A4 will be less accurate than A2, leaving only the relative order between A3 and A4 uncertain.

V. EXPERIMENTAL SETUP

The goal of our experiments is to compare the accuracy and run times of approximate and exact FRNN-OWA. In particular, we want to test whether the approximate variants FRNN-OWA-A1–4 can match the accuracy of exact FRNN-OWA within logarithmically scaling query times.

All experiments are carried out on a single laptop computer equipped with a 4-core i7-8550U (Kaby Lake Refresh @ 1.8 GHz) processor and 16 GB of memory. We use our own Python implementation of FRNN-OWA, which incorporates the Cython implementation of the Scikit-Learn library [35] for KD-Tree nearest neighbour searches (which is compiled to C) and the implementation in C++ provided by the Non-Metric Space Library (NMSLIB) [34] for HNSW nearest neighbour searches. To ensure a fair comparison, all experiments are single-threaded.

We have selected three of the largest datasets from the UCI Machine Learning Repository [19]: SUSY [36], HIGGS [36] and HEPMASS [37] (Table V). These are sufficiently large that performing cross-validation with exact FRNN-OWA is not practical. To measure time complexity and to detect the

TABLE V
DATASETS USED IN THE PRESENT STUDY, PROPERTIES

Name	Number of instances	Attribute type	Number of attributes	Number of classes
SUSY	5 000 000	real	18	2
HIGGS	11 000 000	real	28	2
HEPMASS	10 500 000	real	28	2

TABLE VI
ACCURACY DEFICIT OF APPROXIMATE FRNN-OWA FOR 5-FOLD CROSS-VALIDATION ON SAMPLES OF 2^{20} INSTANCES

Method	SUSY	HIGGS	HEPMASS
A1	-0.01%	-0.09%	-0.20%
A2	-0.09%	-0.14%	-1.55%
A3	-0.48%	-0.51%	-3.87%
A4	-0.72%	-2.06%	-6.54%

TABLE VII
5-FOLD CROSS-VALIDATION ACCURACY OF APPROXIMATE FRNN-OWA

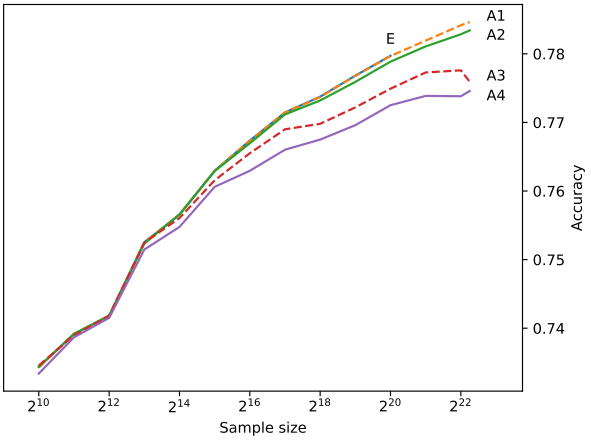
Method	SUSY	HIGGS	HEPMASS
A1	78.5%	67.9%	84.4%
A2	78.3%	67.6%	83.4%
A3	77.6%	65.4%	81.3%
A4	77.5%	63.7%	78.4%

effect of dataset size on accuracy, we take random samples of different sizes. This setup allows us to evaluate run time and accuracy on real data, while ensuring that the datasets of different sizes are in all other respects comparable to each other.

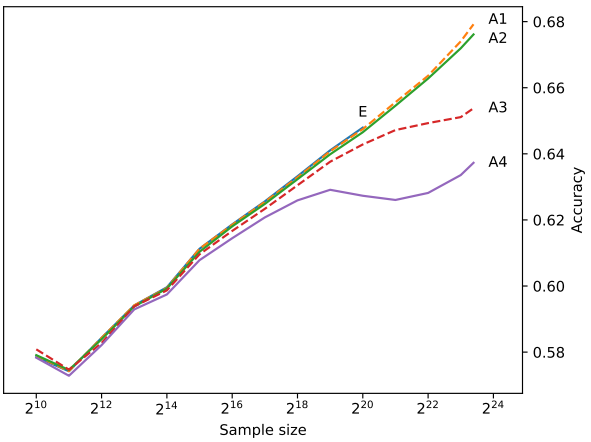
We perform two separate series of experiments. First, to see whether approximate FRNN-OWA is able to match the accuracy of exact FRNN-OWA, we perform 5-fold cross validation, starting with samples of 2^{10} instances and increasing in powers of 2 up to the full dataset size. In the case of exact FRNN-OWA, we stop at 2^{20} instances. Second, in order to get a clear picture of query time complexity, we perform simple holdout testing using test sets with a fixed size of 2^5 instances and training sets with 2^{10}–2^{20} instances. To limit the effect of chance, both series of experiments are repeated for five different samples per sample size and the average results are reported.

VI. RESULTS

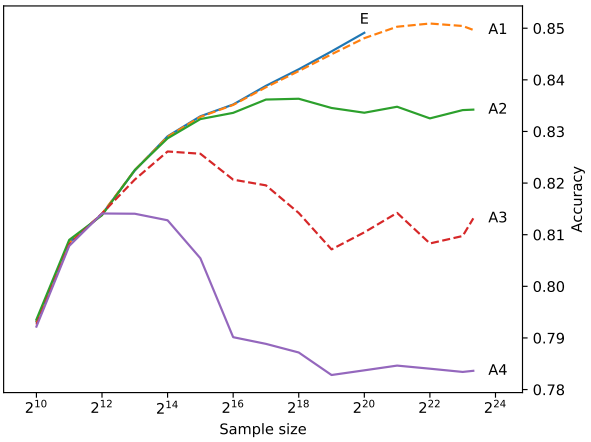
Fig. 2 shows the accuracy obtained with 5-fold cross-validation on samples of the SUSY, HIGGS and HEPMASS datasets. As discussed in Section III, the accuracy of HNSW decreases with sample size for a given parameter configuration. This is reflected in the approximate FRNN-OWA implementations — which incorporate HNSW — in the form of an accuracy deficit with respect to exact FRNN-OWA that eventually opens up as sample size grows large enough. The final accuracy deficits at sample sizes of 2^{20} instances —



(a) SUSY

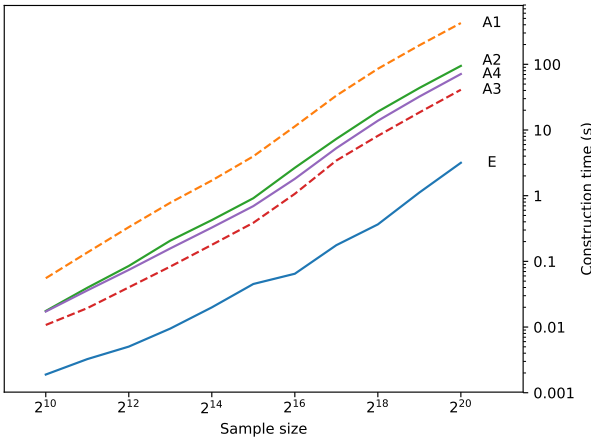


(b) HIGGS

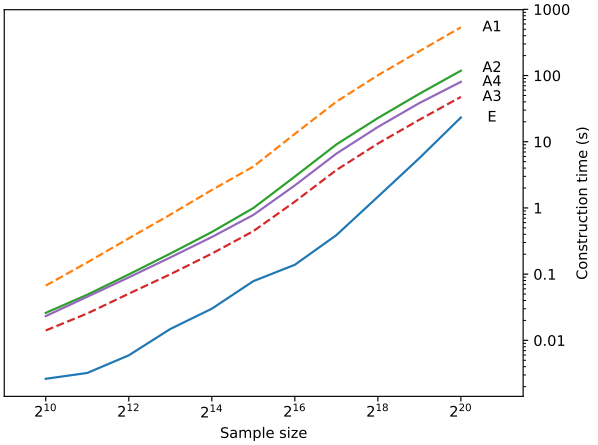


(c) HEPMASS

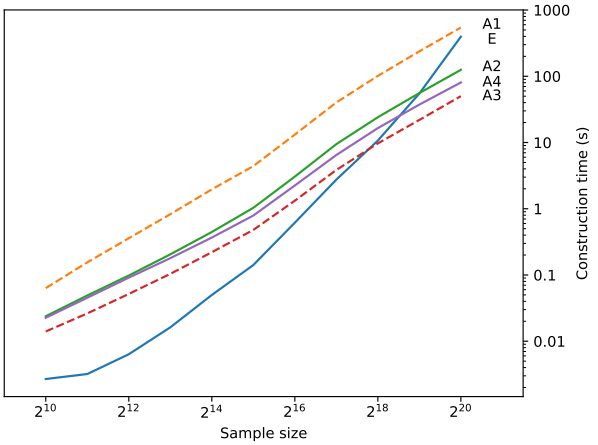
Fig. 2. 5-fold cross-validation accuracy of exact (E) and approximate (A1–4) FRNN-OWA on samples of SUSY, HIGGS and HEPMASS



(a) SUSY

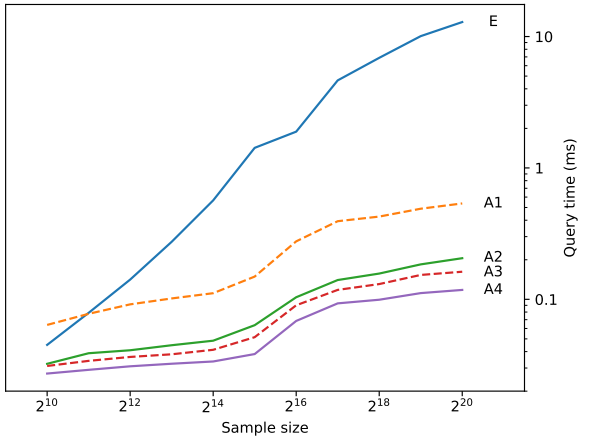


(b) HIGGS

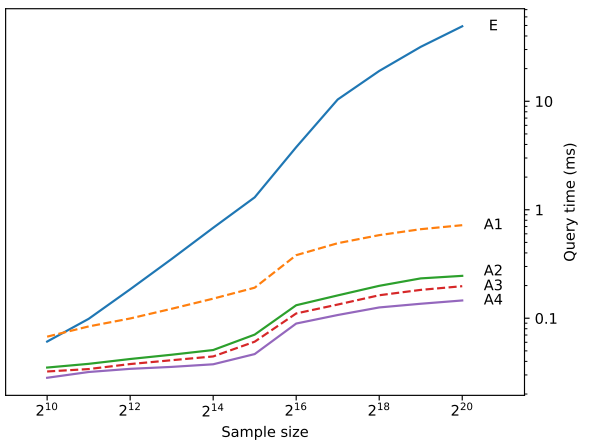


(c) HEPMASS

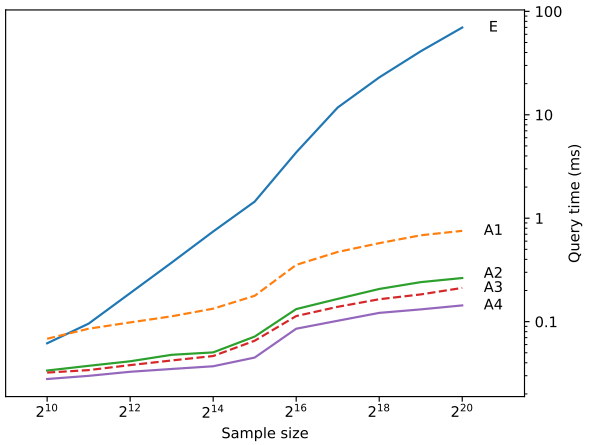
Fig. 3. Construction times of exact (E) and approximate (A1–4) FRNN-OWA on samples of SUSY, HIGGS and HEPMASS.



(a) SUSY

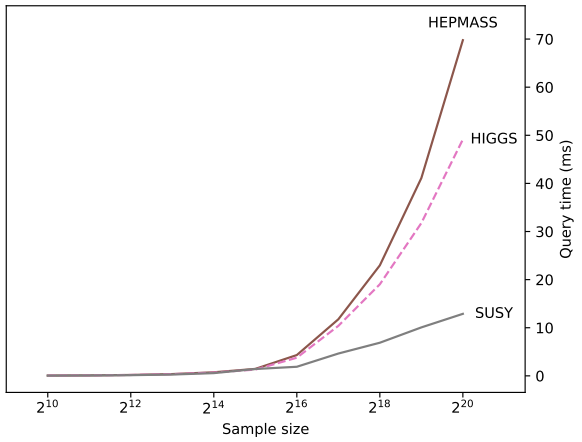


(b) HIGGS

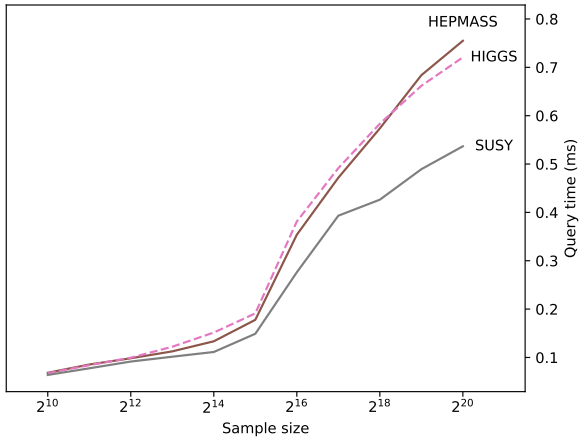


(c) HEPMASS

Fig. 4. Query times per test instance of exact (E) and approximate (A1–4) FRNN-OWA on samples of SUSY, HIGGS and HEPMASS.



(a) FRNN-OWA-E

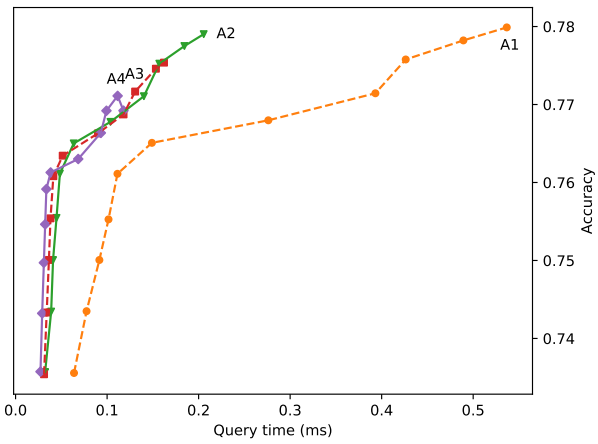


(b) FRNN-OWA-A1

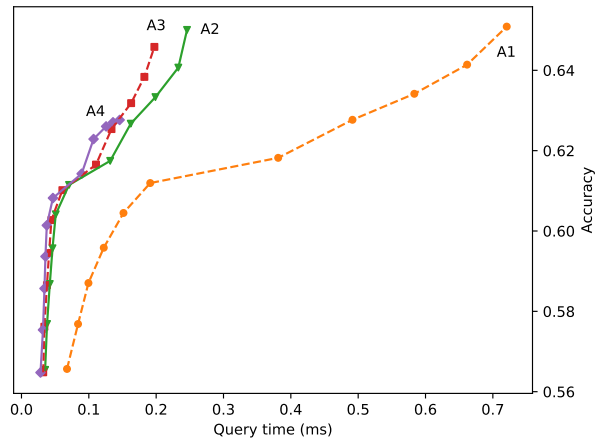
Fig. 5. Query times per test instance of FRNN-OWA-E and -A1 on samples of SUSY, HIGGS and HEPMASS.

the largest sample size for which cross-validation with exact FRNN-OWA proved feasible — are listed in Table VI. The final accuracy figures for approximate FRNN-OWA over the whole datasets are listed in Table VII.

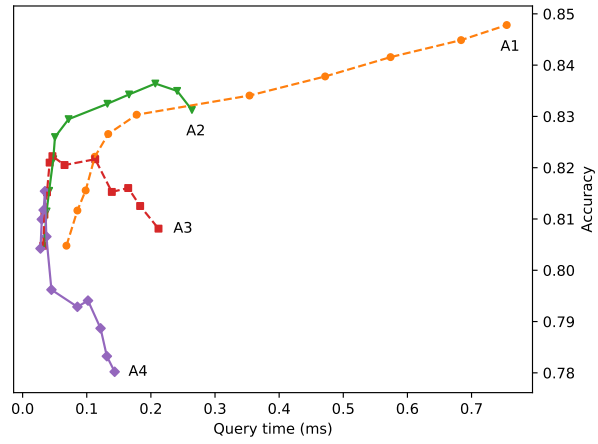
The results bear out our prediction that configurations A1, A2 and A3 produce decreasing levels of accuracy. They also show that A4 produces lower accuracy than A3 across the board. On SUSY and HIGGS, A1 and A2 perform extremely close to exact FRNN-OWA. A3 and A4 drop off as sample size grows, with A4 performing relatively worse on HIGGS. Despite having an identical number of attributes, HEPMASS poses a much greater challenge than HIGGS, with A2 performing markedly worse than exact FRNN-OWA. A1 stays close to exact FRNN-OWA up to 2^{20} instances, but its accuracy doesn't increase further for larger sample sizes. It is possible that this is due to a certain amount of saturation of HEPMASS and that the accuracy of exact FRNN-OWA levels off in a similar manner. However, if this is not the case, it would be possible to match the accuracy of exact FRNN-OWA by increasing ef_q.



(a) SUSY



(b) HIGGS



(c) HEPMASS

Fig. 6. Holdout accuracy as a function of query time per test instance of approximate (A1–4) FRNN-OWA with training sets of various sizes, on samples of SUSY, HIGGS and HEPMASS.

Fig. 3 shows the construction times of FRNN-OWA with various training set sizes. The construction time of approximate FRNN-OWA starts out by being longer than the construction time of exact FRNN-OWA, but with HIGGS and HEPMASS the difference becomes less pronounced as training set size grows. The observed time complexity of approximate FRNN-OWA is linear, supporting the claimed $\mathcal{O}(n \log n)$ construction time complexity of HNSW.

The log-log graphs in Fig. 4 show that the query time per test instance of approximate FRNN-OWA scales much better with training set size than the query time of exact FRNN-OWA. Fig. 5 displays the same query times in lin-log graphs for exact FRNN-OWA and FRNN-OWA-A1 (the graphs for FRNN-OWA-A2–4 are very similar). It appears that with SUSY, exact FRNN-OWA possibly achieves its theoretically predicted logarithmic time complexity as training set size grows beyond 2^{16}, but that with the higher dimensional HIGGS and HEPMASS, this does not happen within the range of the tested training set sizes. The query times of FRNN-OWA-A1 scale more unevenly, but seemingly better than logarithmically as sample size grows beyond 2^{15}

Finally, the query times of approximate FRNN-OWA are repeated in Fig. 6 together with the corresponding holdout test accuracy. These graphs illustrate the trade-off between accuracy and query time which FRNN-OWA-A1–4 offer when the quantity of available data is not a significant limiting factor. It can be seen that for the datasets tested, most levels of accuracy can be reached with A2 in less time than A1, while A3 and A4 offer no clear advantage. However, to achieve the highest levels of accuracy with HEPMASS we do need A1.

From these results we can conclude that approximate FRNN-OWA can produce accuracy figures that are extremely close to exact FRNN-OWA, while achieving a query time reduction that grows to several orders of magnitude for large sample sizes. For most sample sizes considered, this comes at a cost of a somewhat longer construction time. The construction and query times of the various configurations of approximate FRNN-OWA differ by a constant factor and scale equally well. The different accuracy results for HIGGS and HEPMASS show that it is difficult to generalise across different datasets and that achieving an optimal trade-off between accuracy and query time requires dataset-specific tuning. However, based on this limited overview, it seems that the default parameter values of HNSW, combination A1, are also a safe default choice for approximate FRNN-OWA. A2 may also be good enough if query time is a particular concern, whereas the relatively limited additional query time reduction of A3 and A4 does not seem to warrant the more significant loss in accuracy.

VII. Conclusion

The scalability of FRNN-OWA classification is restricted by the nearest neighbour searches that it requires. Existing exact nearest neighbour search algorithms struggle to achieve better than linear query time complexity in practice. In contrast, HNSW, a state of the art approximative algorithm, is able to identify nearest neighbours with near-100% accuracy and logarithmic query time complexity. In this article, we have presented approximate FRNN-OWA, a variant of FRNN-OWA that incorporates HNSW.

To compare the performance of exact and approximate FRNN-OWA, we defined four parameter configurations (A1–4) of HNSW and selected three very large datasets with up to 11 million instances (SUSY, HIGGS and HEPMASS). These datasets were then used to draw a series of random samples of different sizes. To evaluate classification accuracy, we performed cross-validation on these samples as well as the full datasets. Finally, we measured construction and query times through simple hold-out testing on the samples.

As a result of these experiments, we found that on SUSY and HIGGS, the parameter configurations A1 and A2 achieve near-identical accuracy as exact FRNN-OWA for all sample sizes, without any need for further tuning. For HEPMASS, this was still true for A1 up to the largest sample size for which cross-validation with exact FRNN-OWA was feasible. Crucially, the experimental query time complexity of all parameter configurations was sub-logarithmic, resulting in a speed-up with respect to exact FRNN-OWA that grew to several orders of magnitude as sample size increased.

Now that the usefulness of using HNSW as part of FRNN-OWA has been demonstrated, we propose that it is possible to achieve further improvements through deeper integration. At present, the HNSW algorithm concludes when it has identified the nearest neighbours of a test instance within a decision class, and this is repeated for each decision class. It may be possible to terminate this search early, as soon as a decision on the classification of the test instance can be reached.

It will also be necessary to modify HNSW if we want to extend its use to regression, since this will involve calculating the upper and lower approximations of a fuzzy rather than a crisp set, which does not fully reduce to identifying nearest neighbours within a crisp subset.

Finally, fuzzy rough sets have been applied successfully for prototype selection, and it may be possible to adapt this approach to improve the quality of the hierarchical search graph used by HNSW. An important challenge here will be to limit the resulting increase in construction time.

Acknowledgment

The research reported in this paper was conducted with the financial support of the Odysseus programme of the Research Foundation – Flanders (FWO). D. Peralta is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO, 170303/12X1619N).

References

[1] D. Dubois and H. Prade, "Rough fuzzy sets and fuzzy rough sets," *Int. J. Gen. Syst.*, vol. 17, no. 2-3, pp. 191–209, 1990.
[2] Z. Pawlak, "Rough sets," ICS PAS, Rep. 431, 1981.
[3] ——, "Rough sets," *Int. J. Comput. Inf. Sci.*, vol. 11, no. 5, pp. 341–356, 1982.
[4] R. Jensen and C. Cornelis, "A new approach to fuzzy-rough nearest neighbour classification," in *Proc. Int. Conf. Rough Sets and Current Trends Computing*, 2008, pp. 310–319.
[5] C. Cornelis, N. Verbiest, and R. Jensen, "Ordered weighted average based fuzzy rough sets," in *Proc. Int. Conf. Rough Sets and Knowledge Technology*, 2010, pp. 78–85.

[6] E. Ramentol, S. Vluymans, N. Verbiest, Y. Caballero, R. Bello, C. Cornelis, and F. Herrera, "IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1622–1637, 2015.
[7] S. Vluymans, D. Sánchez Tarragó, Y. Saeys, C. Cornelis, and F. Herrera, "Fuzzy rough classifiers for class imbalanced multi-instance data," *Pattern Recognit.*, vol. 53, pp. 36–45, 2016.
[8] S. Vluymans, A. Fernández, Y. Saeys, C. Cornelis, and F. Herrera, "Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach," *Knowl. Inf. Syst.*, vol. 56, no. 1, pp. 55–84, 2018.
[9] O. U. Lenz, D. Peralta, and C. Cornelis, "A scalable approach to fuzzy rough nearest neighbour classification with ordered weighted averaging operators," in *Proc. Int. Joint Conf. Rough Sets (IJCRS) 2019*, Debrecen, Hungary, Jun. 17–21 2019, pp. 197–209.
[10] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Computing*, 1998, pp. 604–613.
[11] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.
[12] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, 2014.
[13] C. D. Yu, J. Huang, W. Austin, B. Xiao, and G. Biros, "Performance optimization for the k-nearest neighbors kernel on x86 architectures," in *SC'15: Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis*, Austin, TX, Nov. 15–20 2015.
[14] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Trans. Big Data*, to be published, doi: 10.1109/TBDATA.2019.2921572.
[15] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
[16] A. Andoni and P. Indyk, "Nearest neighbors in high-dimensional spaces," in *Handbook of discrete and computational geometry*, 3rd ed., J. E. Goodman, J. O'Rourke, and C. D. Tóth, Eds. Boca Raton: Chapman and Hall/CRC, 2017, ch. 43, pp. 1135–1155.
[17] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2018.2889473.
[18] L. D'eer, N. Verbiest, C. Cornelis, and L. Godo, "A comprehensive study of implicator–conjunctor-based and noise-tolerant fuzzy rough sets: definitions, properties and robustness analysis," *Fuzzy Sets Syst.*, vol. 275, pp. 1–38, 2015.
[19] D. Dua and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[20] S. Vluymans, N. Mac Parthaláin, C. Cornelis, and Y. Saeys, "Weight selection strategies for ordered weighted average based fuzzy rough sets," *Inf. Sci.*, 2019.
[21] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 1, pp. 183–190, 1988.
[22] Q. Hu, L. Zhang, S. An, D. Zhang, and D. Yu, "On robust fuzzy rough set models," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 4, pp. 636–651, 2012.
[23] H. Asfoor, R. Srinivasan, G. Vasudevan, N. Verbiest, C. Cornelis, M. Tolentino, A. Teredesai, and M. De Cock, "Computing fuzzy rough approximations in large scale information systems," in *Proc. 2014 IEEE Int. Conf. Big Data (Big Data 2014)*, 2014, pp. 9–16.
[24] S. Vluymans, H. Asfoor, Y. Saeys, C. Cornelis, M. Tolentino, A. Teredesai, and M. De Cock, "Distributed fuzzy rough prototype selection for big data regression," in *Proc. 2015 Annu. Conf. North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conf. Soft Computing (WConSC)*, 2015, pp. 1–6.
[25] H. M. Asfoor, "Fuzzy rough set approximations in large scale information systems," Master's thesis, University of Washington, 2015.
[26] R. Jensen and N. Mac Parthaláin, "Towards scalable fuzzy–rough feature selection," *Inf. Sci.*, vol. 323, pp. 1–15, 2015.
[27] Y. Qian, Q. Wang, H. Cheng, J. Liang, and C. Dang, "Fuzzy-rough feature selection accelerator," *Fuzzy Sets Syst.*, vol. 258, pp. 61–78, 2015.
[28] A. Zeng, T. Li, D. Liu, J. Zhang, and H. Chen, "A fuzzy rough set approach for incremental feature selection on hybrid information systems," *Fuzzy Sets Syst.*, vol. 258, pp. 39–60, 2015.
[29] A. Zeng, T. Li, J. Hu, H. Chen, and C. Luo, "Dynamical updating fuzzy rough approximations for hybrid data under the variation of attribute values," *Inf. Sci.*, vol. 378, pp. 363–388, 2017.
[30] Q. Hu, L. Zhang, Y. Zhou, and W. Pedrycz, "Large-scale multimodality attribute reduction with multi-kernel fuzzy rough sets," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 1, pp. 226–238, 2018.
[31] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, Sep. 1977.
[32] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Approximate nearest neighbor algorithm based on navigable small world graphs," *Inf. Syst.*, vol. 45, pp. 61–68, 2014.
[33] E. Bernhardsson. (2018, Jun.) New approximate nearest neighbor benchmarks. Blog post. [Online]. Available: https://erikbern.com/2018/06/17/new-approximate-nearest-neighbor-benchmarks.html
[34] L. Boytsov and B. Naidan, "Engineering efficient and effective non-metric space library," in *Proc. Int. Conf. Similarity Search and Applications*, 2013, pp. 280–293.
[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[36] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nat. Commun.*, vol. 5, p. 4308, 2014.
[37] P. Baldi, K. Cranmer, T. Faucett, P. Sadowski, and D. Whiteson, "Parameterized neural networks for high-energy physics," *Eur. Phys. J. C*, vol. 76, no. 5, p. 235, May 2016.

Oliver Urs Lenz holds double MSc degrees in Mathematics from Leiden University and the University of Padova (2011), as well as an MA degree in Linguistics from Leiden University (2012). He has worked as a data scientist for three startups in Amsterdam and Oslo, the last of which he co-founded. He is currently a PhD student at Ghent University, funded by the Odysseus Programme of the Flemish Research Foundation (FWO), working mainly on the application of fuzzy rough sets in machine learning with large, imbalanced and/or incompletely labelled datasets.

Daniel Peralta received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2011 and 2016 respectively. He is currently a post-doctoral researcher at Ghent University and the Vlaams Instituut voor Biotechnologie (Ghent, Belgium), within the Data Mining and Modeling for Biomedicine research group. He has published 15 papers in international journals, and received the Foundation BBVA Award for Young Computer Science Researchers in 2018. His research interests include data mining, deep learning, biometrics and large-scale parallel and distributed computing.

Chris Cornelis received the M.Sc. and Ph.D. degrees in computer science from Ghent University, Ghent, Belgium.

He is currently a Postdoctoral Fellow with Ghent University, Belgium, supported by the Odysseus programme of the Science Foundation – Flanders. He has cosupervised nine Ph.D. theses and has authored more than 160 papers in international journals, edited volumes, and conference proceedings. His current research interests include fuzzy sets, rough sets, and machine learning.