A Scalable Approach to Fuzzy Rough Nearest Neighbour Classification with Ordered Weighted Averaging Operators

Oliver Urs Lenz[1][0000−0001−9925−9482], Daniel Peralta[1,2][0000−0002−7544−8411], and Chris Cornelis[1][0000−0002−7854−6025]

[1] Department of Applied Mathematics, Computer Science and Statistics, Ghent University {oliver.lenz,chris.cornelis}@ugent.be http://www.cwi.ugent.be
[2] Data Mining and Modelling for Biomedicine group, VIB Center for Inflammation Research, Ghent University daniel.peralta@irc.vib-ugent.be https://www.irc.ugent.be

Abstract. Fuzzy rough sets have been successfully applied in classification tasks, in particular in combination with OWA operators. There has been a lot of research into adapting algorithms for use with Big Data through parallelisation, but no concrete strategy exists to design a Big Data fuzzy rough sets based classifier. Existing Big Data approaches use fuzzy rough sets for feature and prototype selection, and have often not involved very large datasets. We fill this gap by presenting the first Big Data extension of an algorithm that uses fuzzy rough sets directly to classify test instances, a distributed implementation of FRNN-OWA in Apache Spark. Through a series of systematic tests involving generated datasets, we demonstrate that it can achieve a speedup effectively equal to the number of computing cores used, meaning that it can scale to arbitrarily large datasets.

Keywords: Fuzzy rough sets · OWA operators · Big Data · Apache Spark

1 Introduction

Fuzzy rough sets [7] encode two complementary types of uncertainty: degrees of membership, and the approximation of concepts. This expressiveness has led to their adoption in a variety of machine learning contexts. Fuzzy Rough Nearest Neighbours (FRNN), introduced in [9] (as FRNN-FRS), was an attempt to use fuzzy rough sets directly for classification and obtain better results than existing lazy learners like Fuzzy Nearest Neighbours (FNN) and k Nearest Neighbours (kNN). FRNN considers the lower and upper approximation of each class and classifies a test instance based on its membership in these.

Like other lazy learners, FRNN does not require training and so can be applied directly to classify test instances with a training set. FRNN is also conceptually attractive because its predictions are directly interpretable. Upper approximation membership encodes to what extent a test instance is similar to

the training instances of a class, and so possibly belongs to this class. Lower approximation membership encodes to what extent a test instance is not similar to the training instances of other classes and so necessarily belongs to this class.

However, it was pointed out by [17] that, as originally defined, FRNN makes predictions that are necessarily identical to those of traditional 1NN. This fact, and the more general observation already made in [9] that FRNN is sensitive to noise, motivated a number of revised proposals. In this paper we focus on FRNN-OWA, introduced in [15], which incorporates Ordered Weighted Averaging (OWA) operators into the definition of lower and upper approximation. This involves the application of weight vectors, and the choice of these weight vectors offers a great degree of flexibility. For example, because lower and upper approximations are calculated for each class, it is possible to use different types of weights for different classes. This idea has been applied successfully by [15] and subsequent studies [19] and [20] to imbalanced datasets, where a judicious choice of weights increases the signal of the minority class.

Over the course of the past two decades, ever larger quantities of data have become available as potential inputs for machine learning algorithms, to the point where the performance of machine learning algorithms is often no longer constrained by the availability of training data, but by the capability of the algorithms to handle training data. One popular tactic to increase data processing capacity is to break down the work of an algorithm into a series of parallel tasks, and to execute these tasks on a cluster of computing cores. A number of frameworks exist that automate many of the aspects of parallel cluster computing, including Apache Spark [11], which we use in this paper.

Handling large amounts of data is a particular challenge for lazy learners like FRNN-OWA, which have to process the entire training set when they receive a test instance. Since the application of fuzzy rough sets in machine learning problems is a relatively recent, ongoing endeavour, it is not surprising that while there exist distributed implementations of kNN [13] and Fuzzy kNN [12] classification, no Big Data implementation exists of a fuzzy rough set classifier. The few implementations that do try to extend the use of fuzzy rough sets to a Big Data context focus on preprocessing algorithms like Fuzzy Rough Feature and Prototype Selection, and only one has been applied to a real dataset with more than 1 million instances [8].

This paper seeks to address this absence by presenting the first Big Data implementation of an algorithm that uses fuzzy rough sets directly to classify test instances (FRNN-OWA). By effectively parallelising the FRNN-OWA algorithm, our implementation can be scaled to arbitrarily large datasets by adding additional computing cores. We demonstrate this through a series of systematic tests on generated datasets of up to 2^{24} instances. In addition, we show that our implementation can be used to classify test instances with real datasets containing over 10 million instances.

In Section 2 of this paper, we first define and explain the motivation for FRNN-OWA and give an overview of existing attempts at Big Data implementations of algorithms involving fuzzy rough sets. We then formulate our proposal

in Section 3, describe our experimental setup in Section 4 and present the results in Section 5. We conclude in Section 6 that our implementation demonstrates the viability of using large quantities of available data to classify unseen instances with fuzzy rough sets.

2 Background

2.1 Fuzzy Rough Nearest Neighbour Classification with OWA operators

Recall the following concepts from fuzzy rough set theory. An information system (X, A) consists of a set of instances X and a set A of attributes $a : X \longrightarrow V_a$. A t-norm $T : [0,1] \times [0,1] \longrightarrow [0,1]$ is an associative, commutative and monotonically increasing binary operation for which 1 is an identity element. An implication $I : [0,1] \times [0,1] \longrightarrow [0,1]$ is a binary operation that is monotonically decreasing in its first argument and monotonically increasing in its second argument, and for which $I(0,0) = I(0,1) = I(1,1) = 1$ and $I(1,0) = 0$. An indiscernibility relation $R : X \times X \longrightarrow [0,1]$ is a fuzzy tolerance relation (i.e. reflexive and symmetric) such that $(\forall a \in A : a(x) = a(y)) \implies R(x,y) = 1$.

Given an information system (X, A) and a choice of indiscernibility relation R on X, t-norm T and implication I, the upper and lower approximations of a fuzzy set C in X are defined as in (1).

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

In FRNN, C can be any of the crisp decision classes, and a test instance y is classified to the class C for which the average of $\overline{C}(y)$ and $\underline{C}(y)$ is highest. For crisp C and the minimum t-norm $\min(\cdot, \cdot)$ and Kleene-Dienes implication $\max(1 - \cdot, \cdot)$, which [9] uses, (1) simplifies to (2).

$$\begin{aligned} \overline{C}(y) &= \max_{x \in C}(R(y,x)) \\ \underline{C}(y) &= \min_{x \notin C}(1 - R(y,x)) \end{aligned} \tag{2}$$

It can be seen from (2) that a test instance necessarily has the highest membership degree in the lower and upper approximations of the class of the most indiscernible training instance. Since the indiscernibility relation R corresponds inversely to a generalised metric, the most indiscernible training instance is the nearest neighbour under this metric, meaning that FRNN is indistinguishable from 1NN classification.

To solve this, FRNN-OWA replaces max and min in (2) with Ordered Weighted Averaging (OWA) operators, which were first defined in [21]. For a given

k-dimensional weight vector w with values in $[0, 1]$ that sum to 1, the OWA operator F_w corresponding to w acts on any k-dimensional vector v by rearranging its coefficients such that they descend, and taking the inner product with w. With abuse of notation, we will also apply F_w to sets of size k.

For the special cases of the basis vectors $e_1 = \langle 1, 0, \ldots, 0 \rangle$ and $e_k = \langle 0, \ldots, 0, 1 \rangle$, we get $F_{e_1} = \max$ and $F_{e_k} = \min$. While the choice of weights is in principle open, the idea of FRNN-OWA is to use weights that approximate max and min, such that the contribution of training instances to the membership of a test instance to the lower and upper approximations of a class gradually vanishes as the training distances are ranked further away from the test instance.

Thus, FRNN-OWA changes (2) into (3).

$$\begin{aligned} \overline{C}(y) &= F_{w_1}(\{R(y,x) | x \in C\}) \\ \underline{C}(y) &= F_{w_2}(\{1 - R(y,x) | x \in X \setminus C\}) \end{aligned} \tag{3}$$

Note that the use of OWA operators becomes computationally costly as the number of instances in the training set increases, since we need to to sort all training instances for each test instance. The computational complexity of FRNN-OWA is $\mathcal{O}(dn + n \log(n))$ per test instance, for $n = |X|$ and $d = |A|$.

2.2 Big Data implementations of fuzzy rough sets

The existing literature on using fuzzy rough sets in a Big Data context is limited, and has focused on preprocessing algorithms, which reduce the size of training data, improve its quality, or both, by acting on its instances, its attributes, or both.

The first publication to explicitly adapt a fuzzy rough set algorithm for Big Data was by Asfoor et al [1]. The authors point out that for a given information system (X, A) and fuzzy set C in X, the time complexity of calculating the membership of each instance of X in the lower and upper approximations of C is $\mathcal{O}(dn^2)$. In addition, the resulting indiscernibility matrix has size $\mathcal{O}(dn^2)$, and storing it in memory becomes highly problematic as n grows. They solve these challenges with a distributed implementation in Message Passing Interface (MPI) that avoids calculating and storing the whole matrix. This work was continued by Vluymans et al [18], who present a distributed implementation in Apache Spark of Fuzzy Rough Prototype Selection (FRPS), a preprocessing algorithm for kNN classification developed in [16] and adapted in [18] for kNN regression. Asfoor [2] also adapts OWA-FRPS, a more robust version of FRPS with OWA operators, into a distributed implementation (POWA-FPRS) that approximates the ordered weighted average by partitioning the data and calculating the ordered weighted average of the ordered weighted averages within these partitions.

Jensen & Mac Parthaláin [10] point out that the calculation of fuzzy rough sets scales badly to large numbers of instances, and that this is further compounded if the feature space is also large. They propose three variants of Fuzzy Rough Feature Selection (FRFS). In nnFRFS and nnFDM (based on FRFS

with Fuzzy Discernability Matrices), the indiscernibility relation is modified to only consider the k nearest neighbours of each instance. Fuzzy Rough Feature Grouping (FRFG) introduces a preliminary step in which overlapping groups of correlated features are defined. For each pass, only the most decisive feature from each group is considered, and other features in the same group are then skipped, thus reducing the number of candidates that have to be evaluated.

A number of other authors have presented Big Data implementations of FRFS. Qian et al [14] propose to reduce the computational cost of FRFS by relaxing the calculations of the lower and upper approximations, potentially reducing the specificity of the resulting feature selection. Zeng et al [22, 23] present a mechanism to incrementally update fuzzy rough approximations in a hybrid information system (HIS) (in which a hybrid metric combines different types of attributes) and apply this to feature selection. Finally, Hu et al [8] present a distributed implementation of multi-kernel attribute reduction using kernelised fuzzy rough sets, and evaluate the results for Support Vector Machines (SVM) and Classification and Regression Trees (CART).

As can be seen in Table 1, half of these works only use datasets with up to a few thousand instances. The connected studies of [1, 18, 2] work with generated datasets of up to 10,000,000 instances and only [8] tests on real datasets with more than one million instances.

Table 1. Articles with Big Data implementations of fuzzy rough algorithms — largest numbers of training instances in generated and real datasets

Article		Generated	Real
[1]	Asfoor et al 2014	10,000,000	—
[18]	Vluymans et al 2015	10,000,000	320,395
[2]	Asfoor 2015	10,000,000	320,395
[10]	Jensen & Mac Parthaláin 2015	—	832
[14]	Qian et al 2015	—	2310
[23]	Zeng et al 2015	—	2800
[22]	Zeng et al 2017	—	2800
[8]	Hu et al 2018	—	4,898,431
	Present study	16,777,216	11,000,000

The studies mentioned above have demonstrated the usefulness of scalable implementations of fuzzy rough prototype and feature selection. However, the potential to apply fuzzy rough classification algorithms in a big data context remains untapped, which is what we wish to address.

3 A scalable version of FRNN-OWA

We propose a parallel implementation of FRNN-OWA that can classify test instances with arbitrary large datasets in a fixed amount of time if we add sufficient parallel computing power.

FRNN-OWA is a 'nearest neighbour' classifier in the sense that if we use suitable weights, the influence of training instances vanishes as the training distances are ranked further away from a given test instance. So while, as mentioned in Sec. 2.1, sorting the entire set of training instances for each test instances is computationally costly, the precise order among the more distant training instances is actually of little consequence. For this reason, we adapt an idea from [10] (discussed in Sec. 2.2) and restrict the application of OWA weights to the k nearest training instances of a test instance y, within a class C for the upper approximation and without for the lower approximation, for some value k. We denote these by $\mathrm{NN}(y, C)$ and $\mathrm{NN}(y, X \setminus C)$ respectively.

The definitions for the upper and lower approximation which we use are given in (4), and we classify a test instance y to the class C for which the average of $\overline{C}(y)$ and $\underline{C}(y)$ is highest.

$$\begin{aligned} \overline{C}(y) &= F_{w_1}(\{R(y,x) | x \in \mathrm{NN}(y, C)\}) \\ \underline{C}(y) &= F_{w_2}(\{1 - R(y,x) | x \in \mathrm{NN}(y, X \setminus C)\}) \end{aligned} \tag{4}$$

We have chosen to use additive weights in this paper, defined as $w_1 = (\frac{2(k+1-i)}{k(k+1)})_{1 \leq i \leq k}$ and $w_2 = (\frac{2i}{k(k+1)})_{1 \leq i \leq k}$, and to set $k = 20$, after initial testing with different types of weights and a range of values for k on datasets of various sizes convinced us that these generally produce good results.

The time complexity of sorting all distances for every class is $\mathcal{O}(n \log(n))$, whereas the time complexity of identifying the k closest distances per class is just $\mathcal{O}(n)$. Since we do need to sort the k smallest distances per class, our proposal reduces the overall time complexity per test instance from $\mathcal{O}(dn + n \log(n))$ to $\mathcal{O}(dn + n + 2ck \log(k))$, where c is the number of classes. Since k and c are kept constant, for large n this further reduces to $\mathcal{O}((d + 1)n)$. Thus, this variant of FRNN-OWA scales linearly with training set size.

There exist several different frameworks for parallel computing that provide different trade-offs between ease of use, automated performance optimisation and user control. Since our main objective is to demonstrate the conceptual viability of our approach, rather than to obtain the absolutely fastest run times possible, we have chosen to implement our algorithm in Spark, which offers a relatively straightforward path to parallelisation. We implement FRNN-OWA through the Python API of Spark, using high-level dataframe operations that allow us to express operations as SQL instructions which are automatically distributed across the nodes in the cluster.

Our implementation is structured as follows:

0. Initialise Spark.
1. Read the training set, combine all attributes into a feature vector. If the attributes are numerical, scale the features to $[0, 1]$.

2. Read the test set, combine all attributes into a feature vector. If the attributes are numerical, apply the same scaling as in step 1.
3. Optional: divide the training set from step 1 into a large number of small partitions.
4. Fill a dataframe of length k with additive weights.
5. Broadcast the test set from step 2 to all partitions, cross join with the training set from step 1, calculate the distance between each pair of test and training instances and select the k closest distances per class per test instance.
6. Cache the dataframe from step 5.
7. Join the weights from step 4 with the distances from step 5, multiply, and sum per class and test instance to get the upper approximations.
8. For every test instance and class, join the weights from step 5 with the k closest training instances from step 5 that do not belong to that class, multiply, and sum to get the lower approximations.
9. Join the upper and lower approximations from steps 7 and 8 and for every test instance, select the class for which the sum of the approximations is highest.
10. Divide the number of test instances from step 9 for which the predicted class matches the actual class by the total number of test instances and report the accuracy.

Step 3 was used only to prevent out-of-memory errors with the largest datasets when using multiple executors per node. Anecdotally, it seemed to increase run times, and so we did not include step 3 with our baseline measurements with only one core, so as not to obtain unduly positive speedups.

Step 5 is the costliest step, because it involves a cross join between training and test instances. Broadcasting the test set makes it available on all partitions, which means that the training set does not have to be replicated across partitions. Ordinarily, Spark would not preserve the resulting dataframe after its use in step 7, and would have to recalculate step 5 for step 8. To prevent this, we cache the dataframe in step 6.

4 Experimental setup

All experiments were performed on the Golett cluster of the Ghent University Tier-2 of the Flemish Supercomputer Centre (VCS). The computing nodes of the Golett cluster are equipped with 2 x 12-core Intel E5-2680v3 (Haswell-EP @ 2.5 GHz) processors, 64 GB memory and 500 GB hard drives, and connected by FDR-10 InfiniBand. The experiments were run in Spark clusters of up to 64 executors, 4 cores per executor and 16 GB memory per executor. These Spark clusters occupied up to 32 nodes of the Golett cluster, with 8 cores per node. The algorithm was implemented in Spark 2.4.0 and run with the Hadoop Yarn resource manager.

The shared nature of the Golett cluster and the general inavailability of fully free nodes necessitated the choice of using only 8 cores per node, while limiting

the number of cores per executor to 4 meant that two executors fit precisely onto one node. During initial testing, increasing the number of nodes per executor far above 4 led to diminishing returns. Of the 64 GB of memory per node, 8 GB was reserved for the operating system. Our cluster was limited to using one third of the remaining 56 GB on the basis of using one third of the number of cores. Thus, we chose 16 GB of memory per executor to maximise this resource, whereas in practice this amount was limited to 9.33 GB per executor.

The scaling of our implementation was tested on a series of generated datasets with varying training set sizes. Each training set had 20 real-valued attributes and 10 classes. Training set size varied from 2^{10} to 2^{24}.

The algorithm was also tested on four real datasets from the UCI Machine Learning Repository [6], summarised in Table 2. SUSY [4], HEPMASS [3] and HIGGS [4] are three large datasets of Monte Carlo simulations of particle physics collisions. The attributes are all real and indiscernibility was defined as the complement of the Manhattan distance, with both attributes and distance scaled to $[0, 1]$. *Poker hand* [5] is a slightly smaller dataset of possible hands of cards in the game of poker. It was included here because its attributes are categorical, necessitating a different indiscernibility relation. We chose the complement of the Hamming distance scaled to $[0, 1]$.

Table 2. Real datasets used in the present study, properties

Name	Number of instances	Attribute type	Number of attributes	Number of classes
Poker hand	1,025,010	categorical	10	10
SUSY	5,000,000	real	18	2
HEPMASS	10,500,000	real	28	2
HIGGS	11,000,000	real	28	2

Our primary performance measure is $T_{p,n}$, the time it takes using p cores to classify one test instance with n training instances. Time measurement starts with the initialisation of Spark and ends with the calculation of the accuracy. We report the average run time per test instance, derived from running the algorithm with a test set of 100 instances. These were, respectively, generated in addition to the generated training sets, and drawn and subtracted from the real training sets. For the generated training sets, we also report a speedup figure $S_{p,n}$ which is defined as $T_{1,n}/T_{p,n}$.

5 Results

Table 3 summarises the run times of our distributed implementation of FRNN-OWA for various generated training set sizes and various numbers of cores, and Table 4 the resultant speedups with respect to the baseline of using only one core. The speedups are also plotted in Fig. 1.

Table 3. Run times in seconds per test instance of FRNN-OWA applied to generated training sets of different sizes, for different numbers of cores

Cores	Training set size														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
1	0.83	0.83	1.3	1.3	1.9	3.1	6.1	11	21	50	104	201	428	858	1627
2	0.37	0.44	0.63	0.86	1.3	1.8	3.1	5.8	11	27	68	78	202	424	876
4	0.33	0.39	0.55	0.81	1.2	1.0	1.6	3.0	5.4	12	29	39	82	273	356
8	0.54	0.41	0.74	1.0	1.0	1.0	1.3	1.6	3.1	5.9	18	20	39	95	189
16	0.44	0.54	0.59	0.86	1.1	1.1	1.8	1.0	1.5	3.1	6.0	13	27	55	110
32	0.38	0.50	0.65	0.94	1.2	1.1	1.1	1.3	1.1	1.8	3.8	5.9	15	21	42
64	0.55	0.75	0.86	1.4	1.3	1.2	1.2	1.4	1.1	2.2	3.2	6.0	12	11	23
128	0.51	0.63	0.71	1.0	1.2	1.2	1.3	1.2	1.2	1.4	2.0	4.1	6.7	7.2	14
256	0.75	0.77	1.0	1.2	1.5	1.5	1.5	1.4	1.3	1.5	1.5	2.1	7.2	6.4	14

Values rounded for readability to two significant digits (< 100) or whole integers (≥ 100)

Table 4. Speedups of FRNN-OWA applied to generated training sets of different sizes, for different numbers of cores

Cores	Training set size														
	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2.2	1.9	2.1	1.5	1.4	1.7	2.0	2.0	1.9	1.9	1.5	2.6	2.1	2.0	1.9
4	2.5	2.1	2.4	1.6	1.6	3.0	3.8	3.7	4.0	4.3	3.6	5.2	5.2	3.1	4.6
8	1.5	2.0	1.8	1.3	1.8	2.9	4.8	6.9	7.0	8.5	5.9	10	11	9.1	8.6
16	1.9	1.5	2.2	1.5	1.7	2.9	3.5	11	14	16	17	15	16	16	15
32	2.2	1.6	2.0	1.3	1.6	2.8	5.6	9.1	20	28	27	34	28	41	38
64	1.5	1.1	1.5	0.89	1.5	2.5	5.1	8.3	19	23	32	33	35	79	72
128	1.6	1.3	1.9	1.3	1.5	2.5	4.8	9.2	18	35	52	49	65	120	118
256	1.1	1.1	1.3	1.0	1.3	2.1	4.2	8.1	16	34	68	95	59	133	115

Values rounded for readability to two significant digits (< 100) or whole integers (≥ 100)

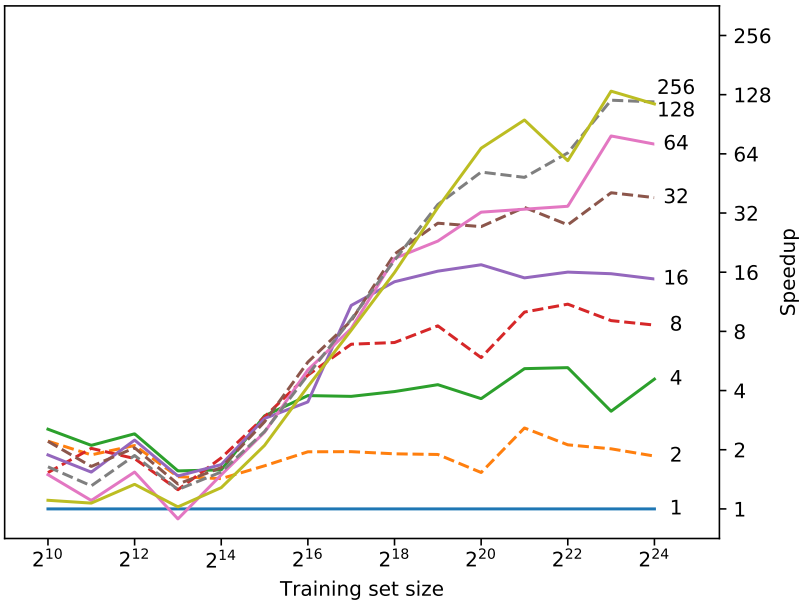

Fig. 1. Speedups for different numbers of cores, with FRNN-OWA applied to generated training sets of different sizes

The results show first of all that there is a certain amount of random fluctuation, which is to be expected on shared infrastructure. For training sets with fewer than 2^{11} instances, the overhead of the implementation is the dominating factor, and run time is effectively constant. For training sets with fewer than 2^{13} instances, overhead is still large enough that it negates the effect of adding more cores: speedup is constant. As training set size grows beyond 2^{13} instances, the speedup with p cores starts to climb more or less linearly until it reaches its theoretical maximum, p. This is reflected in the distinct diagonal cluster of lines in Fig. 1. Only the maximal configuration with 256 cores does not reach its full potential speedup within the space of these dataset sizes.

Table 5 shows the run times of our implementation of FRNN-OWA applied to the real datasets, which demonstrate that our implementation can be used to classify instances using FRNN-OWA with very large training sets.

Table 5. Run times per test instance of FRNN-OWA applied to real datasets, with 256 cores

Name	Time (s)
Poker hand	1.2
SUSY	4.3
HEPMASS	27
HIGGS	30

6 Conclusion and further work

In this paper we have argued that until now, classifiers based on fuzzy rough sets have not been fit to handle Big Data, and that other attempts to adapt fuzzy rough sets for use with Big Data have mostly involved demonstrations on not very large datasets. To address this, we have presented the first implementation of a classifier based on fuzzy rough sets that can be scaled to handle arbitrarily large datasets. We have proposed a parallelised version of FRNN-OWA that can divide execution time over as many computing cores as is required.

To evaluate the performance of our implementation, we divised a series of systematic experiments, measuring run time on generated datasets varying in size from 2^{10} to 2^{24} instances, and calculating the speedup obtained by using between 1 and 256 computing cores. The results of these experiments showed that with sufficiently large datasets, the execution time of our implementation is effectively reduced by a factor equal to the number of computing cores. We then demonstrated that our implementation can be used for classifying test instances with a number of large real datasets of up to 11,000,000 instances.

We believe that the work presented in this paper constitutes a necessary first step towards adapting fuzzy rough sets for Big Data, and that it enables both the application of fuzzy rough sets to concrete classification problems, as well as several types of further research.

Having restricted the application of OWA operators to the k nearest neighbours of a test instance, a natural question to ask is what value for k is sufficiently large. In the future we wish to determine whether it is necessary to tune k for each dataset or whether a certain value is always good enough. This question also has to take into account the choice of weights. In fact, restricting the application of OWA operators to the k nearest neighbours opens up for consideration new types of weights whose accuracy reaches a global maximum for value of k and decreases as k approaches the full training set size.

We also want to investigate whether we can further reduce the computational complexity of FRNN-OWA by approximating some of the calculations. It is easy to think of Big Data merely in terms of large datasets that pose computational challenges. However, as data becomes available ever more easily in ever greater quantities, the types of questions that we want to answer change. Traditionally, researchers have asked which machine learning model can produce the best classification results for a given training set. But in a context where the amount

of training data is essentially unlimited, it may be more relevant to ask which machine learning model can produce the best classification results in a given amount of time. If the accuracy loss from approximating parts of FRNN-OWA is less than the accuracy gain from the additional training data that can be processed in the same amount of time, this may be a worthwile trade-off.

Acknowledgement

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).

References

1. Asfoor, H., Srinivasan, R., Vasudevan, G., Verbiest, N., Cornelis, C., Tolentino, M., Teredesai, A., De Cock, M.: Computing fuzzy rough approximations in large scale information systems. In: Big Data (Big Data), 2014 IEEE International Conference on. pp. 9–16. IEEE (2014)
2. Asfoor, H.M.: Fuzzy Rough Set Approximations in Large Scale Information Systems. Master's thesis, University of Washington (2015)
3. Baldi, P., Cranmer, K., Faucett, T., Sadowski, P., Whiteson, D.: Parameterized machine learning for high-energy physics. arXiv preprint arXiv:1601.07913 (2016)
4. Baldi, P., Sadowski, P., Whiteson, D.: Searching for exotic particles in high-energy physics with deep learning. Nature communications **5**, 4308 (2014)
5. Cattral, R., Oppacher, F., Deugo, D.: Evolutionary data mining with automatic rule generalization. Recent Advances in Computers, Computing and Communications **1**(1), 296–300 (2002)
6. Dua, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
7. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. International Journal of General System **17**(2-3), 191–209 (1990)
8. Hu, Q., Zhang, L., Zhou, Y., Pedrycz, W.: Large-scale multimodality attribute reduction with multi-kernel fuzzy rough sets. IEEE Transactions on Fuzzy Systems **26**(1), 226–238 (2018)
9. Jensen, R., Cornelis, C.: A new approach to fuzzy-rough nearest neighbour classification. In: International Conference on Rough Sets and Current Trends in Computing. pp. 310–319. Springer (2008)
10. Jensen, R., Mac Parthaláin, N.: Towards scalable fuzzy–rough feature selection. Information Sciences **323**, 1–15 (2015)
11. Karau, H., Konwinski, A., Wendell, P., Zaharia, M.: Learning spark: lightning-fast big data analysis. O'Reilly Media, Inc. (2015)
12. Maillo, J., Luengo, J., García, S., Herrera, F., Triguero, I.: Exact fuzzy k-nearest neighbor classification for big datasets. In: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–6. IEEE (2017)
13. Maillo, J., Ramírez, S., Triguero, I., Herrera, F.: kNN-IS: An iterative spark-based design of the k-nearest neighbors classifier for big data. Knowledge-Based Systems **117**, 3–15 (2017)

14. Qian, Y., Wang, Q., Cheng, H., Liang, J., Dang, C.: Fuzzy-rough feature selection accelerator. Fuzzy Sets and Systems **258**, 61–78 (2015)
15. Ramentol, E., Vluymans, S., Verbiest, N., Caballero, Y., Bello, R., Cornelis, C., Herrera, F.: IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. IEEE Transactions on Fuzzy Systems **23**(5), 1622–1637 (2015)
16. Verbiest, N., Cornelis, C., Herrera, F.: OWA-FRPS: A prototype selection method based on ordered weighted average fuzzy rough set theory. In: International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing. pp. 180–190. Springer (2013)
17. Verbiest, N., Cornelis, C., Jensen, R.: Fuzzy rough positive region based nearest neighbour classification. In: Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on. pp. 1–7. IEEE (2012)
18. Vluymans, S., Asfoor, H., Saeys, Y., Cornelis, C., Tolentino, M., Teredesai, A., De Cock, M.: Distributed fuzzy rough prototype selection for big data regression. In: Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC), 2015 Annual Conference of the North American. pp. 1–6. IEEE (2015)
19. Vluymans, S., Fernández, A., Saeys, Y., Cornelis, C., Herrera, F.: Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach. Knowledge and Information Systems **56**(1), 55–84 (2018)
20. Vluymans, S., Sánchez Tarragó, D., Saeys, Y., Cornelis, C., Herrera, F.: Fuzzy rough classifiers for class imbalanced multi-instance data. Pattern Recognition **53**, 36–45 (2016)
21. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on systems, Man, and Cybernetics **18**(1), 183–190 (1988)
22. Zeng, A., Li, T., Hu, J., Chen, H., Luo, C.: Dynamical updating fuzzy rough approximations for hybrid data under the variation of attribute values. Information Sciences **378**, 363–388 (2017)
23. Zeng, A., Li, T., Liu, D., Zhang, J., Chen, H.: A fuzzy rough set approach for incremental feature selection on hybrid information systems. Fuzzy Sets and Systems **258**, 39–60 (2015)