fuzzy-rough-learn 0.1: a Python library for machine learning with fuzzy rough sets

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. We present *fuzzy-rough-learn*, the first Python library of fuzzy rough set machine learning algorithms. It contains three algorithms previously implemented in R and Java, as well as two new algorithms from the recent literature. We briefly discuss the use cases of *fuzzy-rough-learn* and the design philosophy guiding its development, before providing an overview of the included algorithms and their parameters.

Keywords: Fuzzy rough sets · OWA operators · Machine learning · Python package · Open-source software

1 Background

Since its conception in 1990, fuzzy rough set theory [2] has been applied as part of a growing number of machine learning algorithms [17]. Simultaneously, the distribution and communication of machine learning algorithms has spread beyond academic literature to a multitude of publicly available software implementations [7,10,19]. And also during the same period, Python has grown from its first release in 1991 [13] to become one of the world's most popular high-level programming languages.

Python has become especially popular in the field of data science, in part due to the self-reinforcing growth of its package ecosystem. This includes *scikit-learn* [11], which is currently the go-to general purpose Python machine learning library, and which contains a large collection of algorithms.

Only a limited number of fuzzy rough set machine learning algorithms have received publicly available software implementations. Variants of Fuzzy Rough Nearest Neighbours (FRNN) [5], Fuzzy Rough Rule Induction [6], Fuzzy Rough Feature Selection (FRFS) [1] and Fuzzy Rough Prototype Selection (FRPS) [15,14] are included in the R package *RoughSets* [12], and have also been released for use with the Java machine learning software suite WEKA [4,3].

So far, none of these algorithms seem to have been made available for Python in a systematic way. In this paper, we present an initial version of *fuzzy-rough-learn*, a Python library that fills this gap. At present, it includes FRNN, FRFS, FRPS, as well as FROVOCO [18] and FRONEC [16], two more recent algorithms designed for imbalanced and multilabel classification. These implementations all make use of a significant modification of classical fuzzy rough set theory: the incorporation of Ordered Weighted Averaging (OWA) operators in the calculation of upper and lower approximations for increased robustness [1].

We discuss the use cases and design philosophy of *fuzzy-rough-learn* in Section 2, and provide an overview of the included algorithms in Section 3.

2 Use cases and design philosophy

The primary goal of *fuzzy-rough-learn* is to provide implementations of fuzzy rough set algorithms. The target audience is researchers with some programming skills, in particular those who are familiar with *scikit-learn*. We envision two principal use cases:

- The application of fuzzy rough set algorithms to solve concrete machine learning problems.
- The creation of new or modified fuzzy rough set algorithms to handle new types of data or to achieve better performance.

A third use case falls somewhat in between these two: reproducing or benchmarking against results from existing fuzzy rough set algorithms.

To facilitate the first use case, *fuzzy-rough-learn* is available from the two main Python package repositories, pipy and conda-forge, making it easy to install with both pip and conda. *fuzzy-rough-learn* has an integrated test suite to limit the opportunities for bugs to be introduced. API documentation is integrated in the code and automatically updated online[1] whenever a new version is released, and includes references to the literature.

We believe that it is important to make fuzzy rough set algorithms available not just for use, but also for adaptation, since it is impossible to predict or accommodate all requirements of future researchers. Therefore, the source code for *fuzzy-rough-learn* is hosted on GitHub[2] and freely available under the MIT license. We have attempted to write accessible code, by striving for consistency and modularity. The coding style of *fuzzy-rough-learn* is a compromise between object-oriented and functional programming. It makes use of classes to model the different components of the classification algorithms, but as a rule, functions and methods have no side-effects. Finally, subject to these design principles, *fuzzy-rough-learn* generally follows the conventions of *scikit-learn* and the terminology of the cited literature.

3 Contents

fuzzy-rough-learn implements three of the fuzzy rough set algorithms mentioned in Section 1: FRFS, FRPS and FRNN, making them available in Python for the

[1] https://fuzzy-rough-learn.readthedocs.io
[2] https://github.com/oulenz/fuzzy-rough-learn

first time. In addition, we have included two recent, more specialised classifiers: the ensemble classifier FROVOCO, designed to handle imbalanced data, and the multi-label classifier FRONEC.

Together, these five algorithms form a representative cross-section of fuzzy rough set algorithms in the literature. In the future, we intend to build upon this basis by adding more algorithms.

3.1 Fuzzy Rough Feature Selection (FRFS)

Table 1. Parameters of FRFS in *fuzzy-rough-learn*

Name	Default value	Description
`n_features`	`None`	Number of features to select. If None, will continue to add features until positive region size becomes maximal.
`owa_weights`	`deltaquadsigmoid (0.2, 1)`	OWA weights to use for calculation of soft minimum in lower approximations.
`t_norm`	`'lukasiewicz'`	T-norm used to aggregate the similarity relation R from per-attribute similarities.

Fuzzy Rough Feature Selection (FRFS) [1] greedily selects features that induce the greatest increase in the size of the positive region, until it matches the size of the positive region with all features, or until the required number of features is selected.

The positive region is defined as the union of the lower approximations of the decision classes in X. Its size is the sum of its membership values.

The similarity relation R_B for a given subset of attributes B is obtained by aggregating with a t-norm the per-attribute similarities R_a associated with the attributes a in B. These are in turn defined, for any $x, y \in X$, as the complement of the difference between the attribute values x_a and y_a after rescaling by the sample standard deviation σ_a (1).

$$R_a(x, y) = \max(1 - \frac{|x_a - y_a|}{\sigma_a}, 0) \tag{1}$$

3.2 Fuzzy Rough Prototype Selection (FRPS)

Fuzzy Rough Prototype Selection (FRPS) [15,14] uses upper and/or lower approximation membership as a quality measure to select instances. It follows the following steps:

1. Calculate the quality of each training instance. The resulting values are the potential thresholds for selecting instances.

Table 2. Parameters of FRPS in *fuzzy-rough-learn*

Name	Default value	Description
`quality_measure`	`'lower'`	Quality measure to use for calculating thresholds. Either the upper approximation of the decision class of each attribute, the lower approximation, or the mean value of both.
`aggr_R`	`np.mean`	Function used to aggregate the similarity relation R from per-attribute similarities.
`owa_weights`	`invadd()`	OWA weights to use for calculation of soft maximum and/or minimum in quality measure.
`nn_search`	`KDTree()`	Nearest neighbour search algorithm to use.

2. For each potential threshold and corresponding candidate instance set, count the number of instances in the overall dataset that have the same decision class as their nearest neighbour within the candidate instance set (excluding itself).
3. Return the candidate instance set with the highest number of matches. In case of a tie, return the largest such set.

There are a number of differences between the implementations in [15] and [14]. In each case, the present implementation follows [14]:

– While [15] uses instances of all decision classes to calculate upper and lower approximations, [14] calculates the upper approximation membership of an instance using only instances of the same decision class, and its lower approximation membership using only instances of the other decision classes. This choice affects over what length the weight vector is 'stretched'.
– In addition, [14] excludes each instance from the calculation of its own upper approximation membership, while [15] does not.
– [15] uses additive weights, while [14] uses inverse additive weights.
– [15] defines the similarity relation R by aggregating the per-attribute similarities R_a using the Łukasiewicz t-norm, whereas [14] recommends using the mean.
– In case of a tie between several best-scoring candidate prototype sets, [15] returns the set corresponding to the median of the corresponding thresholds, while [14] returns the largest set (corresponding to the smallest threshold).

In addition, there are two implementation issues not addressed in [15] or [14]:

– It is unclear what metric the nearest neighbour search should use. It seems reasonable that it should either correspond to the similarity relation R (and therefore incorporate the same aggregation strategy from per-attribute similarities), or that it should match whatever metric is used by nearest neighbour classification subsequent to FRPS. By default, the present implementation uses Manhattan distance on the scaled attribute values.

– When the largest quality measure value corresponds to a singleton candidate instance set, it cannot be evaluated (because the single instance in that set has no nearest neighbour). Since this is an edge case that would not score highly anyway, it is simply excluded from consideration.

3.3 Fuzzy Rough Nearest Neighbour (FRNN) multiclass classification

Table 3. Parameters of FRNN in *fuzzy-rough-learn*

Name	Default value	Description
`upper_weights`	`additive()`	OWA weights to use in calculation of upper approximation of decision classes.
`upper_k`	`20`	Effective length of upper weights vector (number of nearest neighbours to consider).
`lower_weights`	`additive()`	OWA weights to use in calculation of lower approximation of decision classes.
`lower_k`	`20`	Effective length of lower weights vector (number of nearest neighbours to consider).
`nn_search`	`KDTree()`	Nearest neighbour search algorithm to use.

Fuzzy Rough Nearest Neighbours (FRNN) [5] provides a straightforward way to apply fuzzy rough sets for classification. Given a new instance y, we obtain class scores by calculating the membership degree of y in the upper and lower approximations of each decision class and taking the mean. This implementation uses OWA weights, but limits their application to the k nearest neighbours of each class, as suggested by [8].

3.4 Fuzzy Rough OVO Combination (FROVOCO) multiclass classification

Table 4. Parameters of FROVOCO in *fuzzy-rough-learn*

Name	Default value	Description
`nn_search`	`KDTree()`	Nearest neighbour search algorithm to use.

Fuzzy Rough OVO COmbination (FROVOCO) [18] is an ensemble classifier specifically designed for, but not restricted to, imbalanced data, which adapts itself to the Imbalance Ratio (IR) between classes. It balances one-versus-one decomposition with two global class afinity measures.

In a binary classification setting, the lower approximation of one class corresponds to the upper approximation of the other class, so when using OWA weights, the effective number of weight vectors to be chosen is 2. FROVOCO uses the IR-weighting scheme, which depends on the IR between the classes. If the IR is less than 9, both classes are approximated with exponential weights. If the IR is 9 or more, the smaller class is approximated with exponential weights, while the larger class is approximated with a reduced additive weight vector of effective length k equal to 10% of the number of instances.

Provided with a training set X, and a new instance y, FROVOCO calculates the class score of y for a class C from the following components:

$V(C, y)$ **weighted vote** For each other class $C' \neq C$, calculate the upper approximation memberships of y in C and C', using the IR-weighting scheme. Rescale each pair of values so they sum to 1, then sum the resulting scores.

$mem(C, y)$ **positive affinity** Calculate the average of the membership degrees of y in the upper and lower approximations of C, using the IR-weighting scheme.

$mse_n(C, y)$ **negative affinity** For each class C', calculate the average positive affinity of the members of C in C'. Combine these average values to obtain the signature vector S_C. Calculate the mean squared error of the positive affinities of y for each class and S_C, and divide it by the sum of the mean squared errors for all classes.

The final class score is calculated from these components in (2).

$$AV(C, y) = \frac{V(C, y) + mem(C, y)}{2} - \frac{1}{m} mse_n(C, y). \tag{2}$$

3.5 Fuzzy Rough Neighbourhood Consensus (FRONEC) multilabel classification

Table 5. Parameters of FRONEC in *fuzzy-rough-learn*

Name	Default value	Description
`Q_type`	`2`	Quality measure to use for identifying most relevant instances: based on lower (1), upper (2) or both approximations (3).
`R_d_type`	`1`	Label similarity relation to use: Hamming similarity (1) or based on prior probabilities (2).
`k`	`20`	Number of neighbours to consider for neighbourhood consensus.
`weights`	`additive()`	OWA weights to use for calculation of soft maximum and/or minimum.
`nn_search`	`KDTree()`	Nearest neighbour search algorithm to use.

Fuzzy Rough Neighbourhood Consensus (FRONEC) [16] is a multilabel classifier. It combines the instance similarity R, based on the instance attributes, with label similarity R_d, based on the label sets of instances. It offers two possible definitions for R_d. The first, $R_d^{(1)}$, is simply Hamming similarity scaled to $[0, 1]$. The second label similarity, $R_d^{(2)}$, takes into account the prior probability p_l of a label l in the training set. Let L the set of possible labels, and L_1, L_2 two particular label sets. Then $R_d^{(2)}$ is defined as follows:

$$\begin{aligned} a &= \sum_{l \in L_1 \cap L_2} (1 - p_l) \\ b &= \sum_{l \in L \setminus (L_1 \cup L_2)} p_l \\ R_d^{(2)} &= \frac{a + b}{a + b + \frac{1}{2} |L_1 \Delta L_2|} \end{aligned} \tag{3}$$

Provided with a training set X, and a new instance y, FRONEC predicts the label set of y by identifying the training instance with the highest 'quality' in relation to y. There are three possible quality measures, based on the upper and lower approximations.

$$\begin{aligned} Q_1(y, x) &= OWA_{w_l}(\{I(R(z, y), R_d(x, z)) | z \in N(y)\}) \\ Q_2(y, x) &= OWA_{w_u}(\{T(R(z, y), R_d(x, z)) | z \in N(y)\}) \\ Q_3(y, x) &= \frac{Q_1(y, x) + Q_2(y, x)}{2} \end{aligned} \tag{4}$$

Where R_d is a choice of label similarity, T the Łukasiewicz t-norm, I the Łukasiewicz implication, and $N(y)$ the k nearest neighbours of y in X, for a choice of k.

For a choice of quality measure Q, FRONEC predicts the labels of the training instance with the highest quality. If there are several such training instances, it predicts all labels that appear with at least half.

3.6 OWA operators and nearest neighbour searches

Each of the algorithms in *fuzzy-rough-learn* uses OWA operators [20] to calculate upper and lower approximations. OWA operators take the weighted average of an ordered collection of real values. By choosing suitably skewed weight vectors, OWA operators can thus act as soft maxima and minima. The advantage of defining upper and lower approximations with soft rather than strict maxima and minima is that the result is more robust, since it no longer depends completely on a single value.

To allow experimentation with other weights, we have included a range of pre-defined weight types, as well as a general `OWAOperator` class that can be extended and instantiated by users and passed as a parameter to the various classes.

Similarly, users may customise the nearest neighbour search algorithm that is used in all classes except FRFS by defining their own subclass of `NNSearch`. For example, by choosing an approximative nearest neighbour search like Hierarchical Navigable Small World [9], we obtain Approximate FRNN [8].

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, 170303/12X1619N).

References

1. Cornelis, C., Verbiest, N., Jensen, R.: Ordered weighted average based fuzzy rough sets. In: Proceedings of the 5th International Conference on Rough Set and Knowledge Technology (RSKT 2010). Lecture Notes in Artificial Intelligence, vol. 6401, pp. 78–85. Springer (2010)
2. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. International Journal of General Systems **17**(2–3), 191–209 (1990)
3. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD explorations newsletter **11**(1), 10–18 (2009)
4. Jensen, R.: Fuzzy-rough data mining with Weka (2010), http://users.aber.ac.uk/rkj/Weka.pdf
5. Jensen, R., Cornelis, C.: A new approach to fuzzy-rough nearest neighbour classification. In: Proceedings of the 6th International Conference on Rough Sets and Current Trends in Computing (RSCTC 2008). Lecture Notes in Artificial Intelligence, vol. 5306, pp. 310–319. Springer (2008)
6. Jensen, R., Cornelis, C., Shen, Q.: Hybrid fuzzy-rough rule induction and feature selection. In: Proceedings of the 2009 IEEE International Conference on Fuzzy Systems. pp. 1151–1156. IEEE (2009)
7. Jović, A., Brkić, K., Bogunović, N.: An overview of free software tools for general data mining. In: Proceedings of the 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2014). pp. 1112–1117. IEEE (2014)
8. Lenz, O.U., Peralta, D., Cornelis, C.: Scalable approximate FRNN-OWA classification. IEEE Transactions on Fuzzy Systems (to be published). https://doi.org/10.1109/TFUZZ.2019.2949769
9. Malkov, Y.A., Yashunin, D.A.: Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(4), 824–836 (2020)
10. Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., García, Á.L., Heredia, I., Malík, P., Hluchỳ, L.: Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. Artificial Intelligence Review **52**(1), 77–124 (2019)

11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**(85), 2825–2830 (2011)
12. Riza, L.S., Janusz, A., Bergmeir, C., Cornelis, C., Herrera, F., Ślęzak, D., Benítez, J.M.: Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets". Information Sciences **287**, 68–89 (2014)
13. van Rossum, G., de Boer, J.: Interactively testing remote servers using the Python programming language. CWI Quarterly **4**(4), 283–303 (1991)
14. Verbiest, N.: Fuzzy rough and evolutionary approaches to instance selection. Ph.D. thesis, Ghent University (2014)
15. Verbiest, N., Cornelis, C., Herrera, F.: OWA-FRPS: A prototype selection method based on ordered weighted average fuzzy rough set theory. In: Proceedings of the 14th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing (RSFDGrC 2013). Lecture Notes in Artificial Intelligence, vol. 8170, pp. 180–190. Springer (2013)
16. Vluymans, S., Cornelis, C., Herrera, F., Saeys, Y.: Multi-label classification using a fuzzy rough neighborhood consensus. Information Sciences **433**, 96–114 (2018)
17. Vluymans, S., D'eer, L., Saeys, Y., Cornelis, C.: Applications of fuzzy rough set theory in machine learning: a survey. Fundamenta Informaticae **142**(1–4), 53–86 (2015)
18. 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)
19. Wang, Z., Liu, K., Li, J., Zhu, Y., Zhang, Y.: Various frameworks and libraries of machine learning and deep learning: a survey. Archives of Computational Methods in Engineering pp. 1–24 (to be published). https://doi.org/10.1007/s11831-018-09312-w
20. Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on systems, Man, and Cybernetics **18**(1), 183–190 (1988)